Stony Brook University

A Novel Analysis Method for Parallel Processing

A Dissertation Presented

by **Zhemin Zhang**

to

The Graduate School

in Partial Fulfillment of the requirements

for the Degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Stony Brook University

December 2014

Stony Brook University

The Graduate School

Zhemin Zhang

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Thomas G. Robertazzi

Professor in department of Electrical and Computer Engineering

Sangjin Hong

Professor in department of Electrical and Computer Engineering

Peter Milder

Assistant Professor in department of Electrical and Computer Engineering

Hussein Badr

Associate Professor in department of Computer Science

This dissertation is accepted by the Graduate School

Charles Taber

Dean of the Graduate School

Abstract of the Dissertation

A Novel Analysis Method for Parallel Processing

by

Zhemin Zhang

Doctor of Philosophy

in

Electrical and Computer Engineering

Stony Brook University

2014

Though the divisible load scheduling problem has been studied for over two decades, the optimal solution for this problem can be obtained only in a few network topologies by the approach proposed in [6], which recursively equates multiple processing nodes in the network to a super processing node. The limitation of this equivalent approach lies in the fact that a node is required to finishing receiving load from its neighboring nodes simultaneously under this approach, such that linear equations can be established for all nodes in the network. However, the requirement is satisfied in very few network topologies. In this thesis, we address the problem of divisible

load scheduling in network topologies where the requirement is not satisfied, and propose a novel analysis method, which formulates divisible load scheduling in these network topologies as a *Maximum Finish Time Minimization (MFTM)* problem. The MFTM problem minimizes the maximum finish time of all nodes in the network, and further analysis reveals that the MFTM problem can be transformed into the *Finish Time Minimization (FTM)* problem, which resembles the linear optimization problem, indicating an optimal solution by linear programming. With the novel analysis method, we study divisible load scheduling in a variety of network topologies, including mesh, torus, Gaussian network, the ring and multi-root tree, and propose the corresponding optimal algorithms. Besides, considering the high time complexity of the optimal algorithm in the mesh, torus and Gaussian network, we also propose a heuristic algorithm to reduce the time complexity. Extensive comparison results show that the heuristic algorithm achieves performance extremely close to the optimal algorithm algorithm, and both the optimal algorithm and heuristic algorithm significantly outperform previously proposed divisible scheduling algorithms in the studied network topologies.

Contents

List of Figures

List of Tables

Acknowledgement

It would not be possible to write this doctoral thesis without the help of so many people. Here I can only mention a small part of them.

First and foremost, I would like give my earnest thanks to my advisor, Professor Thomas G. Robertazzi, for supporting me both financially and intellectually. Without his help, I would not be able to pursue my PhD degree, not to mention continuing my academic career. He is the most easygoing and one of the smartest people I know. As an advisor, he is a role model that teaches me how to do research with patience, carefulness and hard work. He not only gave me freedom to manage my time and do research without pressure but also provided instructive suggestions when I faced difficulties. He expands my understanding and ability to do this research which is a tremendous help for my future career.

I thank Professor Yuanyuan Yang, who lead me into the academic world. Her advice is the important guideline for my future career.

I thank my friends, Kai Wang, Yang Liu, Li Shi, Yunlong Wang, Li Geng, Zhe Shen, who are members in the COSINE Laboratory, (Communication, Signal Processing, and Networking), Stony Brook University. Their invaluable suggestions and encouragement helped me work on this dissertation. I am very grateful to each of them. In addition, I thank Professor Petar M. Djuric for his encouragement and for providing facilities and the needs for the Cosine Laboratory. This invaluable support makes this laboratory much more convenient and pleasant for research work.

I am grateful to my parents: Jiuqi Zhang and Xiaomei Xiong. They give me unconditional support throughout my life and have cherished with me every achievement I have made. Without their encouragement and support, my study would never have been done in Stony Brook University. I thank my younger brother, Weimin Zhang, for his taking care of our parents during my 5-year study abroad. He is a good boy, and I feel so lucky to have a sibling like him.

Chapter 1

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 processing node and the transmission time of divisible load from one processing node 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 processing node 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 pro-

cessing time by distributing divisible load among multiple processing nodes which are interconnected by a specific network topology. The processing speed of these processing nodes 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 processing nodes, i.e., processing nodes that finish receiving load, and can start distributing load to the other processing nodes, 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 processing nodes 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 processing node to start processing and distributing load earlier. The scenario of mul-

tiple divisible load sources was studied in [23]-[25], where load distribution originates from multiple processing nodes in the network.

Till now, the optimal solution for divisible load scheduling can be obtained only by the method proposed in [6], which calculates the equivalent processing node of all the processing nodes in the network. However, the method is limited to certain specific network topologies, and the reason is that the method calculates the equivalent processing node from a group of linear equations, which requires either that a node in the network receives load from at most one of its neighbors, such as linear daisy chain, tree and multi-level tree with the root as source node, or that if a node in the network receives load from multiple neighboring nodes, these neighboring nodes have to finish transmitting load at the same time, for example, the hypercube with homogenous processing nodes and links, and the symmetry of the hypercube implies that the node must finish receiving load from multiple neighbors simultaneously. Otherwise, the method in [6] cannot be adopted, and that is why only heuristic algorithms are proposed for divisible load scheduling in mesh, torus and many other network topologies.

In this thesis, we propose a novel analysis method for divisible load scheduling, by which we are able to study the divisible load scheduling in the network where a node could receive load from multiple neighbors. With the novel analysis method, we formulate the divisible load scheduling problem as a *Maximum Finish Time Minimization (MFTM)* problem, which minimizes the maximum finish time of all nodes in the network, and discover that the nonlinear equalities in the MFTM problem can be relaxed and substituted

with linear inequalities, yielding the *relaxed MFTM* problem. By introducing the equal finish time for all nodes in the network, the relaxed MFTM problem can be further transformed into the *Finish Time Minimization (FTM)* problem, which resembles the linear optimization problem, implying an optimal solution based on linear programming. We then study the divisible load scheduling in the Gaussian network, mesh, torus, ring and multi-root tree to demonstrate the application of our proposed analysis method.

The following of the thesis proceeds as follows. Chapter 2 generally describes the novel analysis method. In Chapter 3, we study the divisible load scheduling in the Gaussian network, mesh and torus networks. Chapter 4 and 5 discuss the divisible load scheduling in the ring and multi-root tree, respectively. We conclude our work in Chapter 6.

Chapter 2

General Description of the Novel Analysis Method

In this chapter, we explain in detail why the equivalent processing node method proposed in [6] is inapplicable to the analysis of divisible load scheduling in networks where nodes might receive load from multiple neighbors, and how we address this issue with our novel analysis method.

First of all, we introduce the notations to be used in the description of the equivalent processing node method and our novel analysis method. Given an arbitrary network $G(V, E)$, where V and E are the sets of nodes and links in the network, nodes are denoted by integers ranging from 0 to $|V|-1$, and we name nodes with nonzero initial load as *source nodes*. The set of all source nodes is denoted as S, and the initial load of node $i \in S$ is denoted as $L(i)$. The distance between two nodes, say, i and j, in $G(V, E)$ is denoted as $d(i, j)$. The time to process a unit load by node i is denoted as $w(i)T_{cp}$, where T_{cp} is

the time that a standard processing node consumes to process one unit load. Similarly, the time to transmit a unit load from node i to its neighbor j is denoted as $z(i,j)T_{cm}$, and T_{cm} is the time to transmit a unit load through a standard link. We denote the load that node i processes by itself as $\alpha(i)$, and the load transmitted from node i to node j as $\beta(i,j)$. Note that negative $\beta(i,j)$ means that node i is actually receiving load from j, by which we have that $\beta(i,j) = -\beta(j,i)$. The time that node i starts and finishes processing load are denoted as $T_s(i)$ and $T_f(i)$, respectively.

As in [6], we assume that node i can transmit load to j only if j is i's neighbor, i.e., $d(i,j) = 1$, and that a node in the network can start processing load only after it finishes receiving loads from its neighbors, and an optimal divisible load scheduling should minimize the maximum finish time of all nodes in the network, i.e., minimize $\max\{T_f(i)|i \in G(V,E)\}$.

In the equivalent processing node method, if node i receives load from only one of its neighbors, say, j, we can obtain the following equation for node i, meaning that node i starts processing and transmitting load when it finishes receiving load from its neighbor.

$$T_s(i) = T_s(j) + \beta(j,i)z(j,i)T_{cm} \tag{2.1}$$

However, if node i also receives load from another neighboring node, say, j', the equivalent processing node method fails to obtain Eq. (2.1) since j

and j' might not finish transmitting load to i simultaneously, i.e.,

$$T_s(j) + \beta(j,i)z(j,i)T_{cm} \neq T_s(j') + \beta(j',i)z(j',i)T_{cm}$$

and we can not determine the start time of i.

To address this issue, we modify Eq. (2.1) as follows, which sets $T_s(i)$ as the time when node i finishes receiving load from all its neighbors, i.e.,

$$T_s(i) = \max\{T_s(j) + \beta(j,i)z(j,i)T_{cm} | \beta(j,i) > 0\} \tag{2.2}$$

and formulate the divisible load scheduling in $G(V,E)$ as an *Maximum Finish Time Minimization (MFTM)* problem in Table 2.1.

The object function of the MFTM problem is to minimize the maximum finish time of all node in $G(V,E)$. Constraint (2.3) and (2.4) state that source nodes start at time 0, and if a node is not the source node, and receives no load from the other nodes, its start time is set as 0 as well. Constraint (2.5) reflects Eq. (2.2), and constraint (2.6) means that it takes $\alpha(i)w(i)T_{cp}$ time for node i to process $\alpha(i)$ load. If i is the source node, $\alpha(i)$ equals $L(i)$ subtracting the load i sends out, otherwise, $\alpha(i)$ is the summation of received load from i's neighbors, as stated by the following two constraints. As discussed above, node i can only transmit load to its neighbors, and $\beta(i,j)$ and $\beta(j,i)$ are opposite numbers of each other, from which we have constraint (2.9) and (2.10). The last two constraints are true due to that source nodes only send out load, and a node cannot process negative load.

Table 2.1: Maximum Finish Time Minimization (MFTM) Problem Formulation for Divisible Load Scheduling in An Arbitrary Network $G(V, E)$

Minimize: $\max\{T_f(i)|i \in G(V, E)\}$

Subject to:

$$T_s(i) = 0, \text{ if } i \in S \tag{2.3}$$

$$T_s(i) = 0, \text{ if } i \notin S, \ \beta(j, i) = 0 \ \forall \ j \in G(V, E) \tag{2.4}$$

$$T_s(i) = \max\{T_s(j) + \beta(j, i)z(j, i)T_{cm}|\beta(j, i) > 0\}, \text{ if } i \notin S \tag{2.5}$$

$$T_f(i) = T_s(i) + \alpha(i)w(i)T_{cp} \tag{2.6}$$

$$\alpha(i) = L(i) - \sum_{j \in G(V,E)} \beta(i, j), \text{ if } i \in S \tag{2.7}$$

$$\alpha(i) = \sum_{j \in G(V,E)} \beta(j, i), \text{ if } i \notin S \tag{2.8}$$

$$\beta(i, j) = 0, \text{ if } d(i, j) \neq 1 \tag{2.9}$$

$$\beta(i, j) = -\beta(j, i) \tag{2.10}$$

$$\beta(i, j) \geq 0, \text{ if } i \in S \tag{2.11}$$

$$\alpha(i) \geq 0 \tag{2.12}$$

Clearly, constraint (2.5) and the object function of the MFTM problem are nonlinear, to obtain the optimal solution of the MFTM problem, we first linearly relax constraint (2.5) and the object function as follows.

Constraint (2.5) is relaxed into the following linear inequality.

$$T_s(i) \geq T_s(j) + \beta(j,i)z(j,i)T_{cm}, \text{ if } \beta(j,i) > 0 \tag{2.13}$$

In (2.13), $T_s(i)$ is no earlier than the latest time when the neighboring nodes of node i finish load transmission. In other words, node i does not have to start processing and transmitting load immediately when it finishes receiving load from all its neighbors, and we name the problem after linearly relaxing constraint (2.5) as the *relaxed MFTM* problem. For presentational convenience, we define

$$U = \{\alpha(i), \beta(i,j), T_s(i), T_f(i) | i,j \in G(V,E)\}$$

and we say that U is a feasible solution of the MFTM (or relaxed MFTM) problem if the elements in U satisfy all the constraints of the problem. The optimal solutions of the MFTM and relaxed MFTM problems are denoted as U^* and U_r^*, respectively. Since constraint (2.13) is relaxed from constraint (2.5), U^* must be a feasible solution of the relaxed MFTM problem, whereas, U_r^* might not be feasible for the MFTM problem as constraint (2.13) allows

$$T_s(i) > \max\{T_s(j) + \beta(j,i)z(j,i)T_{cm} | \beta(j,i) > 0\}$$

Nevertheless, we are able to prove that these two problems have equal optimal solutions, as stated by Theorem 2.1.

Theorem 2.1. *U^* is the optimal solution of the relaxed MFTM problem.*

Proof. We prove the theorem by contradiction. Since U^* is feasible for the relaxed MFTM problem, we assume that U_r^* is a better solution than U^*.

To show that the assumption does not hold, we construct another optimal solution of the MFTM problem, denoted as $U^{*\prime}$, from U_r^* as follows.

$$U^{*\prime} = \{\alpha(i), \beta(i,j), T_s(i), T_f(i) | \beta(i,j) \in U_r^*\}$$

In other words, $\beta(i,j)$ in $U^{*\prime}$ and U_r^* are equal. According to constraint (2.5) and (2.13), the start time of i in $U^{*\prime}$ will not be later than that in U_r^*, and node i process equal load in $U^{*\prime}$ and U_r^*, therefore, node i will not have later finish time in $U^{*\prime}$, and we have that

$$\max\{T_f(i) | T_f(i) \in U^{*\prime}\} \leq \max\{T_f(i) | T_f(i) \in U_r^*\}$$

On the other hand,

$$\max\{T_f(i) | T_f(i) \in U^*\} > \max\{T_f(i) | T_f(i) \in U_r^*\}$$

by the assumption, resulting that

$$\max\{T_f(i) | T_f(i) \in U^{*\prime}\} < \max\{T_f(i) | T_f(i) \in U^*\}$$

which contradicts that U^* is the optimal solution of the MFTM problem. Hence, the assumption is not true, and the theorem holds. □

By Theorem 2.1, U^* can be constructed using elements in U_r^* as follows.

$$U^* = \{\alpha(i), \beta(i,j), T_s(i), T_f(i) | \beta(i,j) \in U_r^*\}$$

Next, we introduce an equal finish time, denoted as T_f, for all nodes in the network to further linearize the object function of the relaxed MFTM problem, and transform it into the *Finish Time Minimization (FTM)* problem in Table 2.2.

In the FTM problem, we add constraint (2.24), which means that T_f must be no earlier than the finish time of any node in the network, and the object function becomes minimizing T_f. The following theorem tells that the object function of the MFTM problem and FTM problem have equal optimal value.

Theorem 2.2. *When the FTM problem is optimized,* $T_f = \max\{T_f(i) | T_f(i) \in U^*\}$.

Proof. Let U_f^* denote the optimal solution of the FTM problem. Firstly, $\{T_f = \max\{T_f(i) | T_f(i) \in U^*\}\} \cup U^*$ is a feasible solution of the FTM problem, therefore, we have that $T_f \in U_f^* \leq \max\{T_f(i) | T_f(i) \in U^*\}$. On the other hand, by constraint 2.24, we have that T_f can not be smaller than $\max\{T_f(i) | T_f(i) \in U^*\}$ since otherwise we can construct a feasible solution for the relaxed MFTM problem, $U^{*\prime} = U_f^* - \{T_f\}$, which is better than U^*, and contradicts Theorem 2.1. □

Table 2.2: Finish Time Minimization (FTM) Problem Formulation for Divisible Load Scheduling in An Arbitrary Network $G(V, E)$

Minimize: T_f

Subject to:

$$T_s(i) = 0, \text{ if } i \in S \tag{2.14}$$
$$T_s(i) = 0, \text{ if } i \notin S, \ \beta(j,i) = 0 \ \forall \ j \in G(V,E) \tag{2.15}$$
$$T_s(i) \geq T_s(j) + \beta(j,i)z(j,i)T_{cm}, \text{ if } \beta(j,i) > 0 \tag{2.16}$$
$$T_f(i) = T_s(i) + \alpha(i)w(i)T_{cp} \tag{2.17}$$
$$\alpha(i) = L(i) - \sum_{j \in G(V,E)} \beta(i,j), \text{ if } i \in S \tag{2.18}$$
$$\alpha(i) = \sum_{j \in G(V,E)} \beta(j,i), \text{ if } i \notin S \tag{2.19}$$
$$\beta(i,j) = 0, \text{ if } d(i,j) \neq 1 \tag{2.20}$$
$$\beta(i,j) = -\beta(j,i) \tag{2.21}$$
$$\beta(i,j) \geq 0, \text{ if } i \in S \tag{2.22}$$
$$\alpha(i) \geq 0 \tag{2.23}$$
$$T_f \geq T_f(i), \forall i \in G(V,E) \tag{2.24}$$

With these conclusions for an arbitrary network, we now use our proposed novel analysis method to study the divisible load scheduling in the Gaussian network, mesh, torus, ring and multi-root tree in the next three chapters.

Chapter 3

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

3.1 Introduction

In this chapter, we study the divisible load scheduling in the Gaussian network, mesh and torus networks of processors. The Gaussian network is a new type of network topology proposed in [26], which 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. More-

over, 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 chapter 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 3.4.

For presentational convenience, the terminologies of processor and node will be used interchangeably in the rest of this chapter. We assume homogeneous processing speed and data rate for all the nodes and links, respectively, in the network, and startup time is ignored in this chapter. 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 chapter proceeds as follows. Section 3.2.2 introduces the Gaussian network, and some of its related properties to be used in the following parts of the chapter. In Section 3.3, 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 3.4. In

Section 3.5, 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 chapter in Section 3.6.

3.2 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.

3.2.1 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.

3.2.2 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

chapter.

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

$$\omega_1 - \omega_2 \equiv \mathbf{i}^j \bmod a + b\mathbf{i} \tag{3.1}$$

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

According to Eq. (3.1), 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. (3.1) 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. Fig. 3.1 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 chapter.

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.

$$\mathbf{D}(\omega_1, \omega_2) = \min\{|x| + |y|, x + y\mathbf{i} \equiv (\omega_1 - \omega_2) \bmod a + b\mathbf{i}\} \tag{3.2}$$

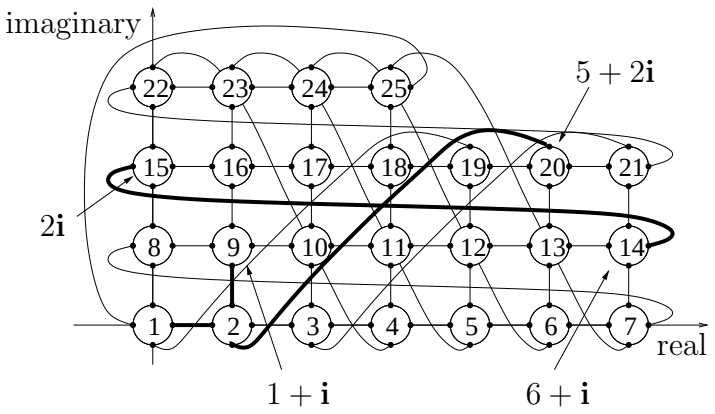

Figure 3.1: Gaussian network $G_{4+3\mathbf{i}}$ with nodes placed in two adjacent meshes.

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. 3.1, $\mathbf{D}(0, 1+\mathbf{i}) = |1|+|1| = 2$, 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. (3.2).

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 3.1 [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 3.1 and 3.2, 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

Table 3.1: 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 3.2: 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

average hop distance and network diameter than mesh and torus under equal network size.

Lemma 3.1. *The average hop distance of Gaussian network G_{a+bi} is*

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

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.

3.2.3 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. (3.1) [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

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

as shown in Fig. 3.2, 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. 3.3 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. 3.1.

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. 3.2, 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. (3.3), where ω_s and ω_p are the coordinates

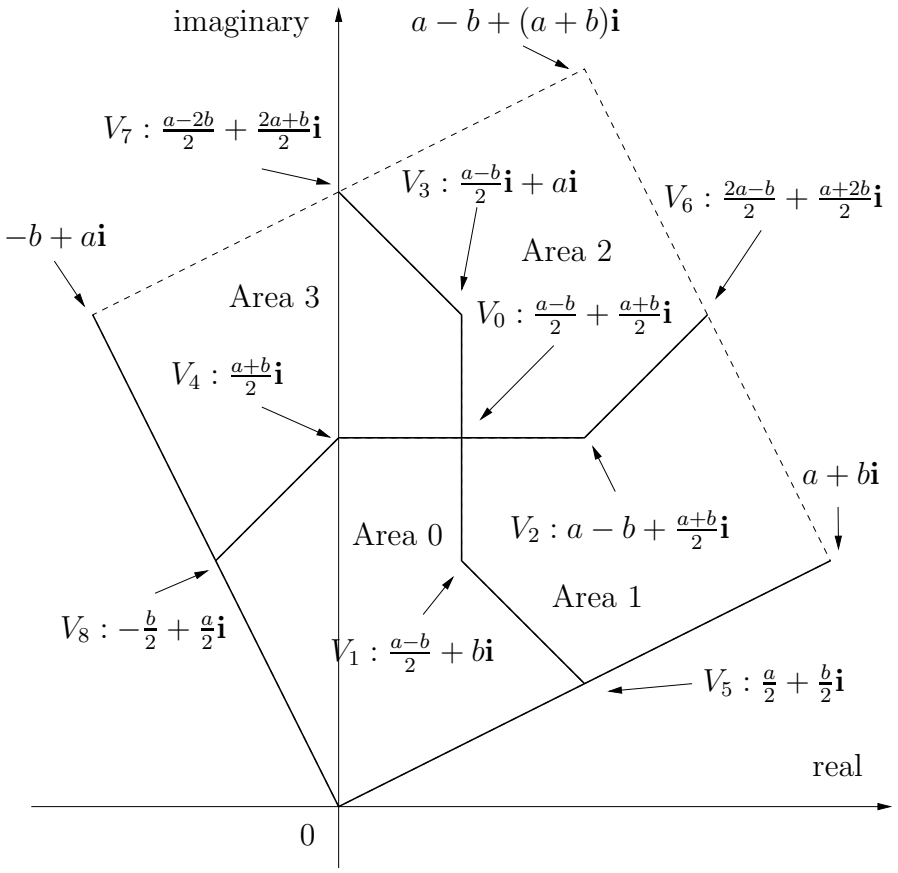

Figure 3.2: Half-open square $S_{a+b\mathbf{i}}$, which excludes dash-lined boundary, is decomposed into four areas. The vertex coordinates of each area are labeled.

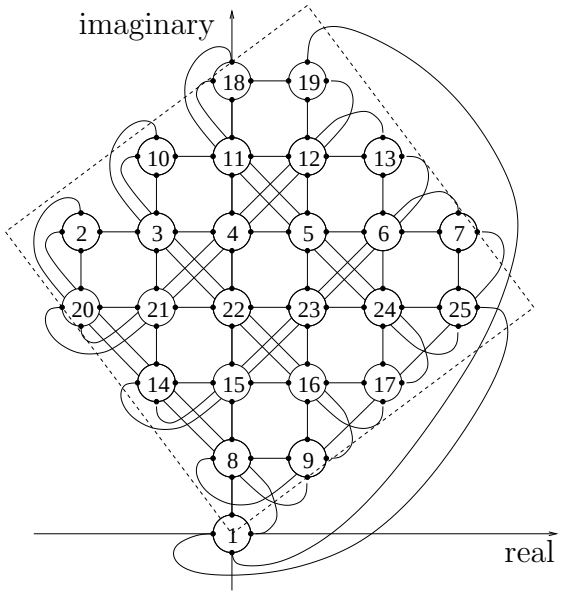

Figure 3.3: Nodes in Gaussian network $\mathbf{G}_{4+3\mathbf{i}}$ placed on half-open square $S_{4+3\mathbf{i}}$.

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. (3.3). 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 $G_{a+b\mathbf{i}}$.

$$\omega_p = \begin{cases} \omega_s & \text{if } \omega_s \text{ is in Area 0} \\ \omega_s - (a+b\mathbf{i}) & \text{if } \omega_s \text{ is in Area 1} \\ \omega_s - (1+\mathbf{i})(a+b\mathbf{i}) & \text{if } \omega_s \text{ is in Area 2} \\ \omega_s - \mathbf{i}(a+b\mathbf{i}) & \text{if } \omega_s \text{ is in Area 3} \end{cases} \tag{3.3}$$

As shown in Fig. 3.4, 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. 3.4, 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. 3.4, 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 when $a+b$ is even, Gaussian integers on D_0 and D_2 (or D_1 and D_3) belong to the same congruence class modulo $a+b\mathbf{i}$, which is true for Gaussian integers on B_0, B_1, B_2 and B_3 as well when a and b are both even. Hence, either one of Gaussian integers on D_0 and D_2 (or D_1 and D_3), and any one of Gaussian integers on B_0, B_1, B_2 and B_3 can be used to represent the corresponding nodes in $G_{a+b\mathbf{i}}$. Fig. 3.5 is an example of 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. 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 chapter, we also place nodes of $G_{a+b\mathbf{i}}$ in $P_{a+b\mathbf{i}}$. Fig. 3.6 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.

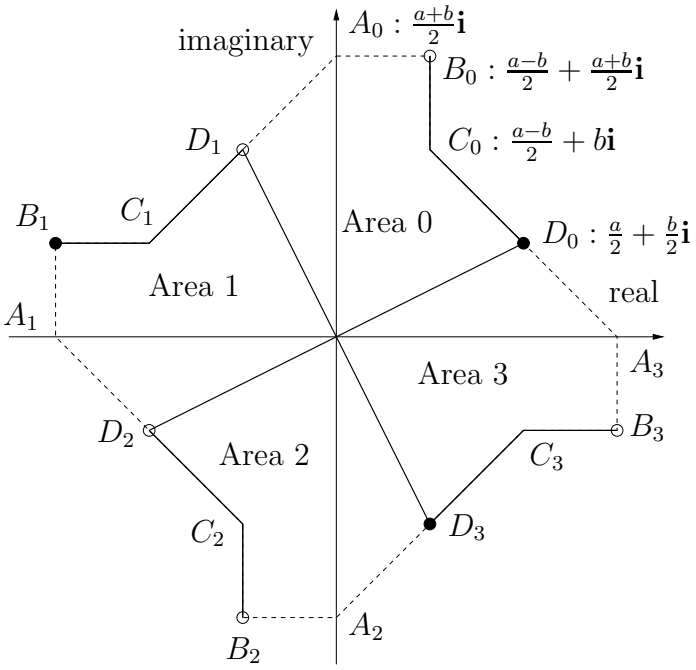

Figure 3.4: Half-open polygon for node placement of $G_{a+b\mathbf{i}}$, which excludes dash-lined boundary and points depicted by hollow dots on the boundary, and is denoted as $P_{a+b\mathbf{i}}$. Without the solid points on the boundary, i.e., B_1, D_0 and D_3, $P_{a+b\mathbf{i}}$ is 4-fold rotational 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 $P_{a+b\mathbf{i}}$.

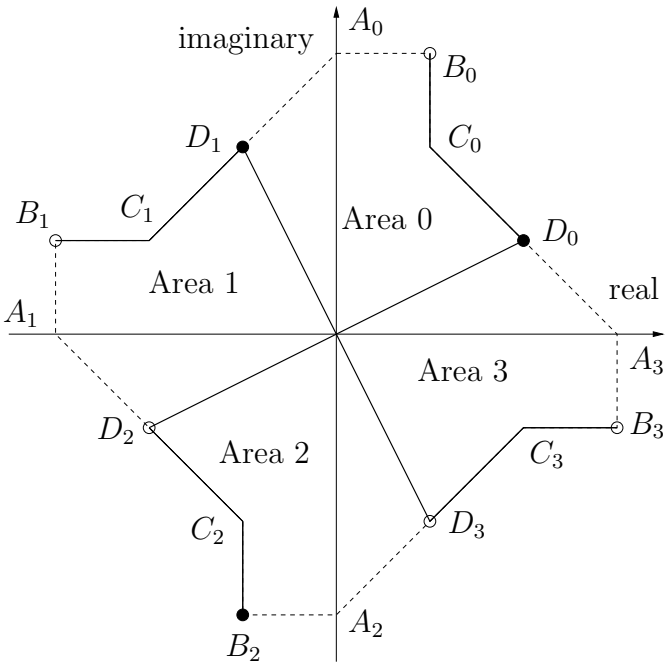

Figure 3.5: Another half-open polygon, denoted as $P'_{a+b\mathbf{i}}$, for node placement of $G_{a+b\mathbf{i}}$, where B_2, D_0 and D_3 on the boundary are solid points.

Corollary 3.1. *The nodes of an optimal Gaussian network $G_{b+1+b\boldsymbol{i}}$ can be placed in the interior of a square, i.e., excluding its boundary, with $\pm\frac{2b+1}{2}$ and $\pm\frac{2b+1}{2}\boldsymbol{i}$ 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 $|x| \leq \frac{a-b}{2}$ or $|y| \leq \frac{a-b}{2}$. 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 an optimal Gaussian network $G_{b+1+b\mathbf{i}}$. □

Since $G_{4+3\mathbf{i}}$ is an optimal Gaussian network, we can see from Fig. 3.6 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.

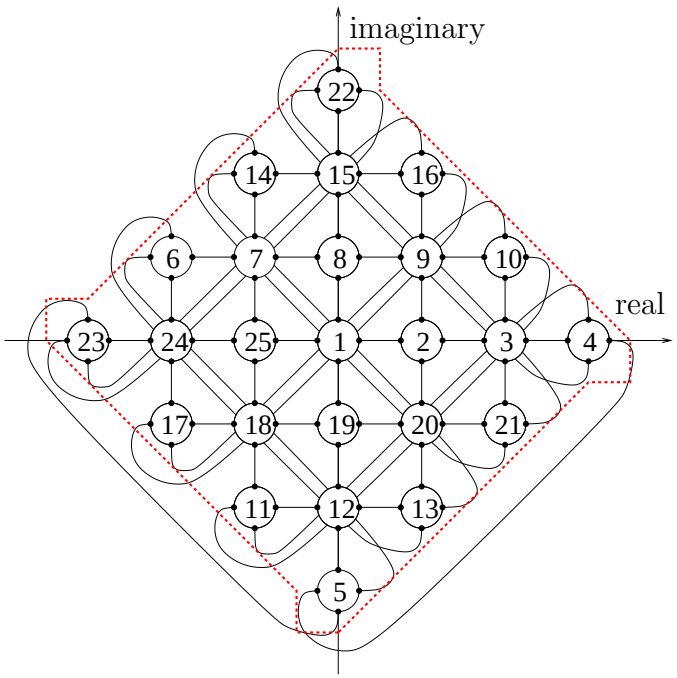

Figure 3.6: Node placement of $G_{4+3\mathbf{i}}$ in $P_{4+3\mathbf{i}}$, the boundary of which is plotted in dash line.

3.3 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.

3.3.1 Problem Formulation

As mentioned in Section 3.1, we assume homogeneous processing speed and link data rate in our chapter. 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) 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

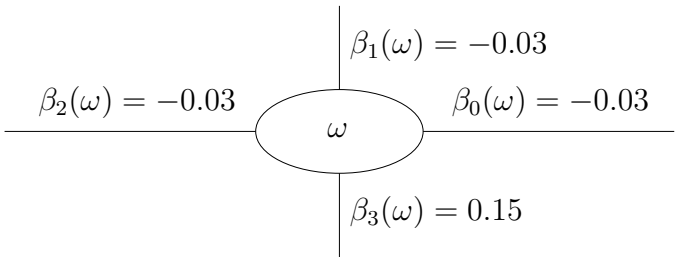

Figure 3.7: Node ω receives 0.15 load from its neighbor 3, and sends 0.03 load to each of its rest 3 neighbors.

$\alpha(\omega)$, 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 3.2.2, 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. 3.7 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 3.3, 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 (3.4) means that it takes $\alpha(\omega)T_{cp}$ time for node ω to process its load. Constraint (3.5) and (3.6) 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 (3.7). The next 4 constraints originate from our rules of $\beta_j(\omega)$, as discussed above. Constraint (3.12) and (3.13) 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.

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 iden-

Table 3.3: 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:

$$T_f(\omega) = T_s(\omega) + \alpha(\omega)T_{cp} \tag{3.4}$$

$$T_s(0) = 0 \tag{3.5}$$

$$T_s(\omega) = \max\{T_s(n_j(\omega)) + \beta_j(\omega)T_{cm} | \beta_j(\omega) > 0\} \tag{3.6}$$

$$T_s(\omega) = 0, \text{ if } \forall j \in \{0,1,2,3\}, \ \beta_j(\omega) = 0 \tag{3.7}$$

$$\beta_j(\omega) \geq 0, \text{ if } \beta_j(\omega) \in U_\beta^+ \tag{3.8}$$

$$\beta_j(\omega) \leq 0, \text{ if } \beta_j(\omega) \in U_\beta^- \tag{3.9}$$

$$\beta_j(\omega) = 0, \text{ if } \beta_j(\omega) \in U_\beta^0 \tag{3.10}$$

$$\beta_j(\omega) = -\beta_k(n_j(\omega)), \text{ if } k = \mathrm{mod}_4(j+2) \tag{3.11}$$

$$\alpha(\omega) = \sum_{j=0}^{3} \beta_j(\omega), \text{ if } \omega \neq 0 \tag{3.12}$$

$$\alpha(0) = 1 + \sum_{j=0}^{3} \beta_j(0) \tag{3.13}$$

$$\alpha(\omega) \geq 0 \tag{3.14}$$

tify 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 3.3, as will be seen in Section 3.4.

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 3.2. *In U^*, if ω_1 and $\omega_2 = \omega_1 \cdot \boldsymbol{i}$ are both in $P_{a+b\boldsymbol{i}}$, $\beta_j(\omega_1) = \beta_k(\omega_2)$, where $k = mod_4(j+1)$.*

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 3.2. *In U^*, $\beta_j(-\frac{a+b}{2} + \frac{a-b}{2}\boldsymbol{i}) = \beta_k(-\frac{a+b}{2} + \frac{a-b}{2}\boldsymbol{i})$, where $k = mod_4(j+1)$, when $a+b$ is even.*

Proof. By the proof of Lemma 3.2, 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 3.3. *In U^*, $\beta_j(\frac{a}{2} + \frac{b}{2}\boldsymbol{i}) = \beta_k(\frac{a}{2} + \frac{b}{2}\boldsymbol{i})$ and $\beta_j(\frac{b}{2} - \frac{a}{2}\boldsymbol{i}) = \beta_k(\frac{b}{2} - \frac{a}{2}\boldsymbol{i})$, where $k = mod_4(j+2)$, 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 3.4. *In U^*, $\beta_0(x+y\boldsymbol{i}) = \beta_0(x-y\boldsymbol{i}) = \beta_1(-y+x\boldsymbol{i}) = \beta_1(y+x\boldsymbol{i}) = \beta_2(-x-y\boldsymbol{i}) = \beta_2(-x+y\boldsymbol{i}) = \beta_3(y-x\boldsymbol{i}) = \beta_3(-y-x\boldsymbol{i})$ if the Gaussian network is optimal.*

Proof. Corollary 3.1 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. 3.8. Therefore, we have $\beta_0(x+y\mathbf{i}) = \beta_2(-x+y\mathbf{i}) = \beta_0(x-y\mathbf{i}) = \beta_1(y+x\mathbf{i}) = \beta_3(-y-x\mathbf{i})$.

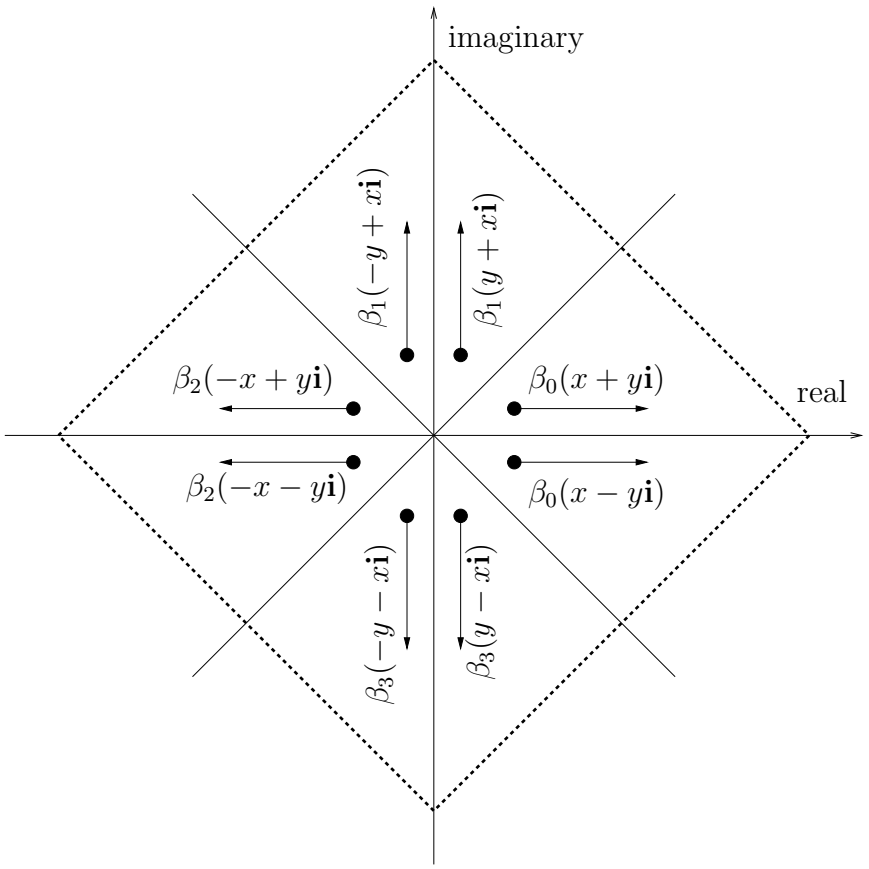

Figure 3.8: The symmetry of the square renders that $\beta_0(x+y\mathbf{i}) = \beta_0(x-y\mathbf{i}) = \beta_1(-y+x\mathbf{i}) = \beta_1(y+x\mathbf{i}) = \beta_2(-x-y\mathbf{i}) = \beta_2(-x+y\mathbf{i}) = \beta_3(y-x\mathbf{i}) = \beta_3(-y-x\mathbf{i})$ in U^* when the Gaussian network is optimal.

Moreover, since the square is also 4-fold rotational symmetric, by Lemma 3.2, 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 3.3.4, 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.

3.3.2 Linear Relaxation of the MFTM Problem

In order to solve the MFTM problem, we first relax constraint (3.6) of the MFTM problem into a linear constraint as follows, and denote the new optimization problem as the *relaxed MFTM* problem.

$$T_s(\omega) \geq T_s(n_j(\omega)) + \beta_j(\omega)T_{cm}, \text{if } \beta_j(\omega) > 0 \tag{3.15}$$

Since constraint (3.15) is relaxed from constraint (3.6), 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 3.5. *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 (3.6) and (3.7), 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.,

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

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. 3.9, where node ω_k $(1 \leq k \leq h)$ 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 $1 \leq k \leq h$. 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 $1 \leq k \leq h$, as shown in Fig. 3.9. After the load redistribution, we have that $T'_f(0) = T_f(0) + \delta\beta T_{cp}$ by

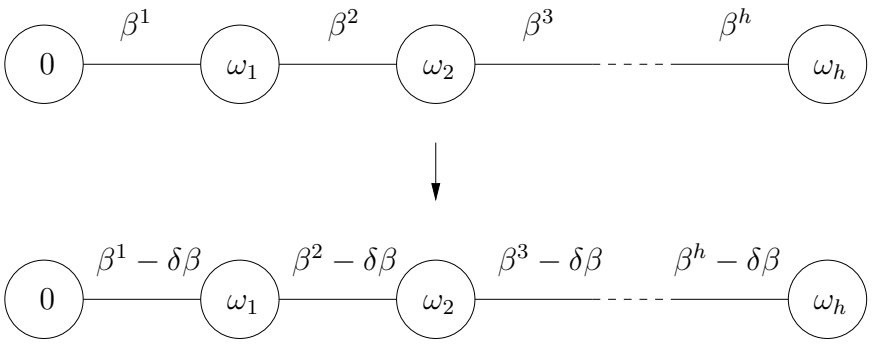

Figure 3.9: Load redistribution when $T_f(0) < T_f(\omega_h) = \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, where $\beta^k > 0$, and β^k is reduced by $\delta\beta$ $(0 < \delta\beta < \beta^k)$ for all $1 \leq k \leq h$.

constraint (3.13) in Table 3.3. 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 (3.12) in Table 3.3, we have that $\alpha'(\omega_k) = \alpha(\omega_k)$ for $1 \leq k < h$, 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 $1 \leq k \leq h$, i.e., $T'_s(\omega_k) = T_s(\omega_k)$, and constraint (3.15) of the relaxed MFTM problem is still satisfied. Therefore, we have that $T'_f(\omega_k) = T_f(\omega_k)$ when $1 \leq k < h$, 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.

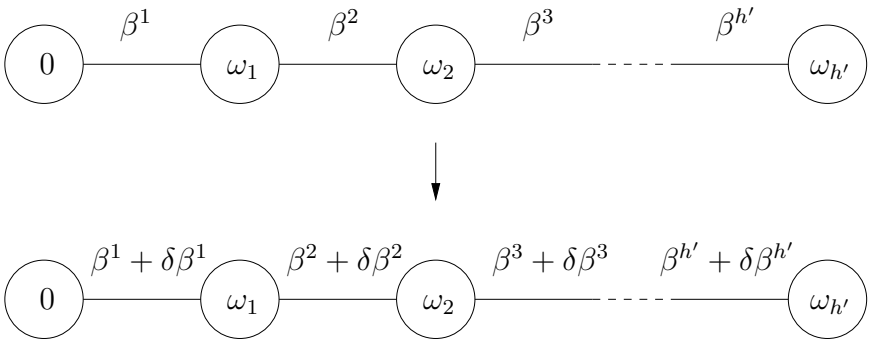

Figure 3.10: Load redistribution when $T_f(\omega_{h'}) < T_f(0) = \max\{T_f(\omega)|\omega \in G_{a+b\mathbf{i}}\}$, where $\beta^k > 0$, and β^k is increased by $\delta\beta^k$ ($\delta\beta^k > 0$) for all $1 \leq k \leq h'$.

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. 3.10, we denote the load received by node ω_k as β^k, where $1 \leq k \leq h'$. Next, we firstly reduce the finish time of node 0 without prolonging the maximum finish time by redistributing load as follows.

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 $1 \leq k \leq h'$. 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 (3.15) 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 $1 < k \leq h'$, we have that

$$\Delta T_s(\omega_k) = \sum_{l=1}^{k} \delta\beta^l T_{cm} \tag{3.16}$$

for $1 \leq k \leq h'$. To avoid increasing the finish time of ω_k when $1 \leq k < h'$, 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 (3.4).

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 $1 \leq k < h'$, as shown in Fig. 3.11, 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. 3.11. 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 $1 \leq k < h'$.

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

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

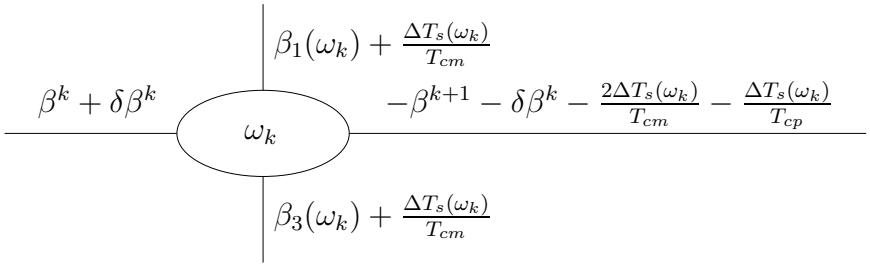

Figure 3.11: After the load redistribution, $T'_s(\omega_k) = \Delta T_s(\omega_k) + T_s(\omega_k)$, to avoid delaying the start time of its neighbor 1 and 3, the load sent to these two neighbors, i.e., $\beta_1(\omega_k)$ and $\beta_3(\omega_k)$, are each reduced by $\frac{\Delta T_s(\omega_k)}{T_{cm}}$. 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.

Hence, we have that

$$\delta\beta^k + \frac{\Delta T_s(\omega_k)}{T_{cp}} \leq \delta\beta^{k+1} \leq \delta\beta^k + \frac{2\Delta T_s(\omega_k)}{T_{cm}} + \frac{\Delta T_s(\omega_k)}{T_{cp}} \tag{3.17}$$

in general.

The above inequality indicates that $\delta\beta^{l-1} \leq \delta\beta^l$ for $1 \leq l \leq k$, applying Eq. (3.16), we have that

$$\Delta T_s(\omega_k) \leq k\delta\beta^k T_{cm} \tag{3.18}$$

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

$$\delta\beta^{k+1} \leq \delta\beta^k(1 + 2k + \frac{T_{cm}}{T_{cp}}k) \tag{3.19}$$

therefore,

$$\delta\beta^k \leq \delta\beta^1 \prod_{l=0}^{k-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) \tag{3.20}$$

and

$$\Delta T_s(\omega_k) \leq k\delta\beta^1 T_{cm} \prod_{l=0}^{k-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) \tag{3.21}$$

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. 3.12, 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

$$T'_f(\omega_{h'}) = T_f(\omega_{h'}) + \Delta T_s(\omega_{h'}) + (\delta\beta^{h'} + \frac{3\Delta T_s(\omega_{h'})}{T_{cm}})T_{cp} \tag{3.22}$$

By Eq. (3.20), (3.21) and (3.22), we have that

$$T'_f(\omega_{h'}) \leq T_f(\omega_{h'}) + \delta\beta^1 T_{cp} \prod_{l=0}^{h'-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) + (1 + 3\frac{T_{cp}}{T_{cm}})h'\delta\beta^1 T_{cm} \prod_{l=0}^{h'-1}(1 + 2l + \frac{T_{cm}}{T_{cp}}l) \tag{3.23}$$

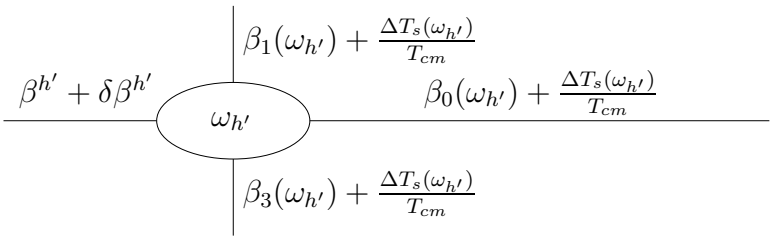

Figure 3.12: After the load redistribution, $T'_s(\omega_{h'}) = \Delta T_s(\omega_{h'}) + T_s(\omega_{h'})$, to avoid delaying the start time of its neighbor 0, 1 and 3, the load sent to these 3 neighbors, i.e., $\beta_0(\omega_{h'})$, $\beta_1(\omega_{h'})$ and $\beta_3(\omega_{h'})$, are each reduced 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.

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 3.5 indicates the following corollary.

Corollary 3.6. *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 3.5, $\forall \omega \neq 0$, $T_f(\omega)$ 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 3.5. □

Theorem 3.5 and Corollary 3.6 in conjunction prove Theorem 3.7.

Theorem 3.7. *When the relaxed MFTM problem is optimized,* $T_f(\omega)$ *must be equal for all* $\omega \in G_{a+b\mathbf{i}}$.

With Theorem 3.7, we show that the MFTM problem and relaxed MFTM problem share identical optimal solution in the next theorem.

Theorem 3.8. U^* *is the optimal solution of the relaxed MFTM problem.*

Proof. Since constraint (3.15) is relaxed from constraint (3.6), 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 3.7, 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 3.7 and 3.8, 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.

3.3.3 Finish Time Minimization (FTM) Problem

Since all nodes have to finish processing load simultaneously by Theorem 3.7 when the relaxed MFTM problem is optimized, constraint (3.4) 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 (3.15) 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 3.4.

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 (3.29). 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 3.9. *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

Table 3.4: Finish Time Minimization (FTM) Problem Formulation for Divisible Load Scheduling in A Gaussian Network

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

Subject to:

$$T_f(S_\beta^+) = T_s(\omega) + \alpha(\omega)T_{cp} \tag{3.24}$$

$$T_s(0) = 0 \tag{3.25}$$

$$T_s(\omega) \geq T_s(n_j(\omega)) + \beta_j(\omega)T_{cm}, \text{ if } \beta_j(\omega) \in S_\beta^+ \tag{3.26}$$

$$\beta_j(\omega) \geq 0, \text{ if } \beta_j(\omega) \in U_\beta^+ \tag{3.27}$$

$$\beta_j(\omega) \leq 0, \text{ if } \beta_j(\omega) \in U_\beta^- \tag{3.28}$$

$$\beta_j(\omega) = 0, \text{ if } \beta_j(\omega) \in U_\beta^0 \cup (U_\beta^+ - S_\beta^+) \tag{3.29}$$

$$\beta_j(\omega) = -\beta_k(n_j(\omega)), \text{ if } k = \text{mod}_4(j+2) \tag{3.30}$$

$$\alpha(\omega) = \sum_{j=0}^{3} \beta_j(\omega), \text{ if } \omega \neq 0 \tag{3.31}$$

$$\alpha(0) = 1 + \sum_{j=0}^{3} \beta_j(0) \tag{3.32}$$

$$\alpha(\omega) \geq 0 \tag{3.33}$$

$$S_\beta^+ \subseteq U_\beta^+ \tag{3.34}$$

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 (3.26) 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 3.3. *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 3.9, U^* is the optimal solution of the FTM problem, thus, if $S_\beta^+ = S_\beta^*$, constraint (3.26) and (3.29) 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 3.3.

We notice that constraint (3.24)-(3.33) 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 (3.24)-(3.33) in Table 3.4. By Lemma 3.3, 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 $\omega_2 = \omega_1 \cdot \mathbf{i}$, $k = \mathrm{mod}_4(j+1)$.
- 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 $k = \mathrm{mod}_4(j+2)$.
- 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.

Table 3.5: 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 (3.24)-(3.33) in Table 3.4;
 end if;
end for;
The FTM problem and $LP(S_\beta^+)$ with minimum object function value have equal optimal solution.
End

By Lemma 3.2, $\beta_j(\omega_1) = \beta_k(\omega_2)$ when $\omega_2 = \omega_1 \cdot \mathbf{i}$ and $k = \mathrm{mod}_4(j+1)$ in U^*, thus, they are in or not in S_β^* concurrently. Similarly, S_β^* has the next two features by Corollary 3.2 and 3.3.

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 3.4, 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 3.5.

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 3.5 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 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 3.5, 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.

3.3.4 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 [32]. In other words, L is the number of bits encoding $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$.

To convert $LP(S_\beta^+)$ to the above standard form, we eliminate all $\beta_j(\omega)$s

that are zero by constraint (3.29), substitute $\beta_j(\omega) \in U_\beta^-$ with $-\beta_k(n_j(\omega))$, where $\beta_k(n_j(\omega)) \in U_\beta^+,\ k = \mathrm{mod}_4(j+2)$, and add nonnegative slack variables, denoted as $T_j(\omega)$, in constraint (3.26) 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^+.

$$|S_\beta^+| \geq \begin{cases} a^2 + b^2 + 2,\ a, b \text{ are both odd} \\ a^2 + b^2 + 4,\ a, b \text{ are both even} \\ a^2 + b^2 - 1,\ a + b \text{ is odd, and } a > b + 1 \\ a^2 + b^2 - 1 + 4\lfloor \frac{b}{2} \rfloor,\ a = b + 1 \end{cases} \tag{3.35}$$

In addition,

$$|U_\beta^+| = \begin{cases} 2(a^2 + b^2),\ a + b \text{ is even} \\ 2(a^2 + b^2 - 2a + 1),\ a + b \text{ is odd} \end{cases} \tag{3.36}$$

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 vari-

ables by Lemma 3.2 and Corollary 3.2-3.4 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 3.5 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 chapter.

As mentioned in Section 3.1, our proposed MFTM problem formulation of divisible load scheduling in Gaussian network can be readily extended to

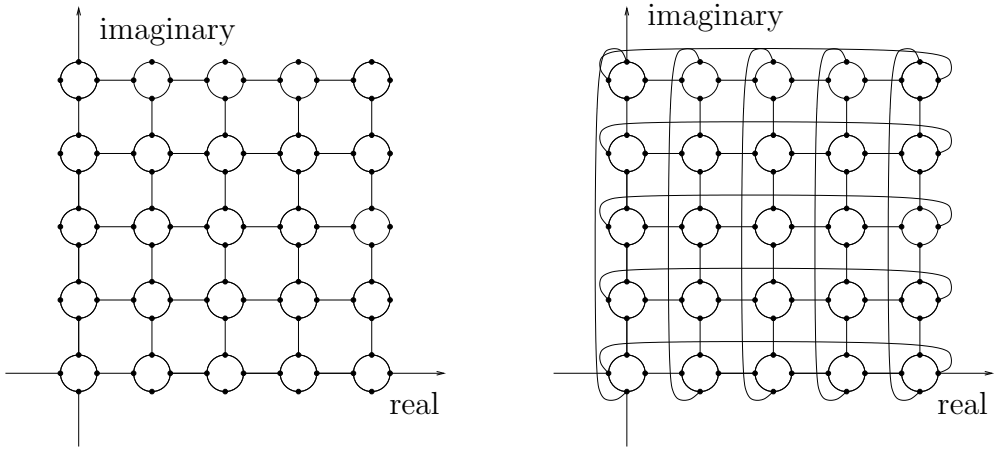

Figure 3.13: Node placement of 5×5 mesh and 5×5 torus in a square with 0, 4, $4\mathbf{i}$ and $4 + 4\mathbf{i}$ as the vertices in the complex plane.

mesh and torus, which will be discussed in the next section.

3.4 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. 3.13 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.,

$$n_j(\omega) = \omega + \mathbf{i}^j, \text{ if } n_j(\omega) \text{ is in the mesh} \tag{3.37}$$

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

$$n_j(x + y\mathbf{i}) = \begin{cases} \mathrm{mod}_a(x+1) + y\mathbf{i} & \text{if } j = 0 \\ x + \mathrm{mod}_b(y+1)\mathbf{i} & \text{if } j = 1 \\ \mathrm{mod}_a(x-1) + y\mathbf{i} & \text{if } j = 2 \\ x + \mathrm{mod}_b(y-1)\mathbf{i} & \text{if } j = 3 \end{cases} \tag{3.38}$$

By Eq. (3.37) and (3.38), 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 $\min\{\mathrm{mod}_a(x_2 - x_1), \mathrm{mod}_a(x_1 - x_2)\} + \min\{\mathrm{mod}_b(y_2 - y_1), \mathrm{mod}_b(y_1 - y_2)\}$ 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. (3.37) and (3.38), ω is still neighbor $\mathrm{mod}_4(j+2)$ of $n_j(\omega)$ in mesh and torus, hence, $\beta_j(\omega) = -\beta_k(n_j(\omega))$, where $k = \mathrm{mod}_4(j + 2)$. 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 3.3 as well, which can be transformed into the FTM problem in Table 3.4 by Theorem 3.7, 3.8 and 3.9, 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.

3.5 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 3.3, 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. 3.13 (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.

3.5.1 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. 3.14, 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

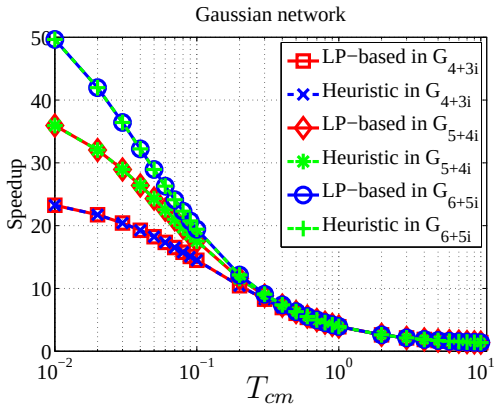

Figure 3.14: Speedup comparison between LP-based algorithm and heuristic algorithm with respect to different network sizes and inverse data rates in Gaussian networks.

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.

3.5.2 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. 3.15. 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 dis-

tributing 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. 3.16 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. 3.16 (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.

3.5.3 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 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. 3.17. 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

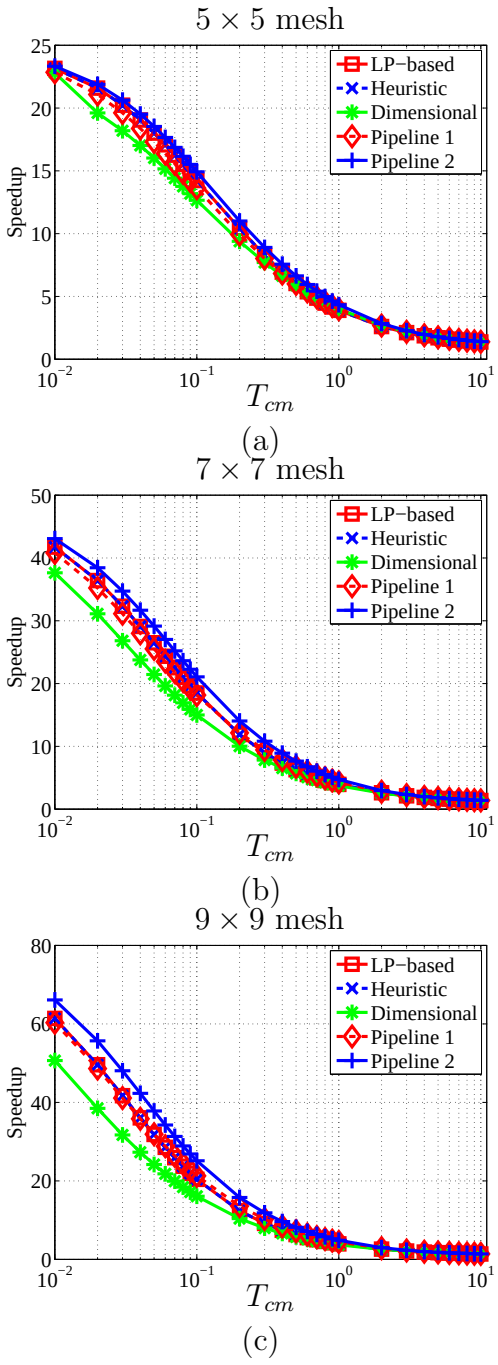

Figure 3.15: Speedup comparison among LP-based algorithm, heuristic algorithm and dimensional algorithm with respect to different network sizes and inverse data rates in meshes. (a) 5×5 mesh. (b) 7×7 mesh. (c) 9×9 mesh.

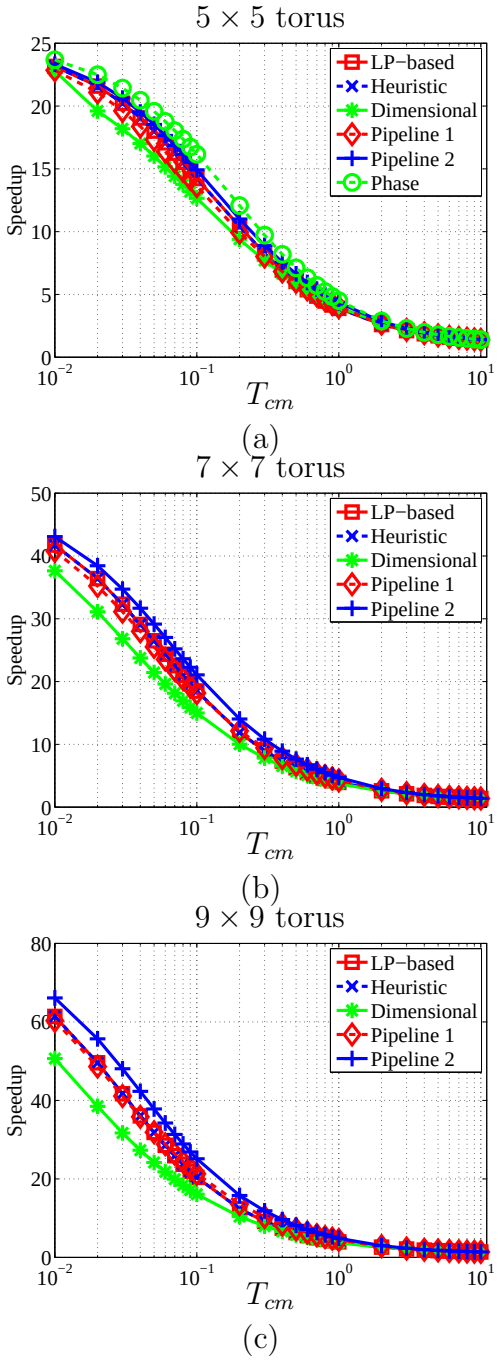

Figure 3.16: Speedup comparison among LP-based algorithm, heuristic algorithm, dimensional algorithm and phase algorithm with respect to different network sizes and inverse data rates in tori. (a) 5×5 torus. (b) 7×7 torus. (c) 9×9 torus.

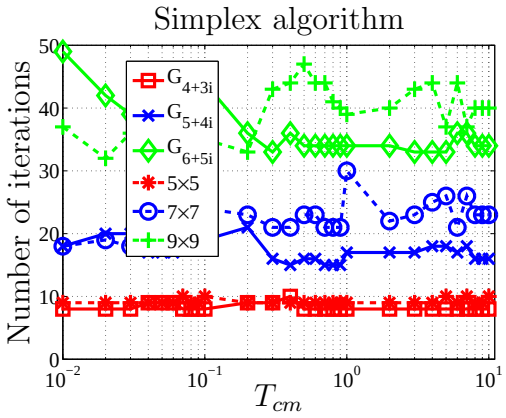

Figure 3.17: Number of iterations by the simplex method in solving $LP(U_\beta^+)$ with respect to different network sizes and topologies.

Fig. 3.17 to represent both networks. From Fig. 3.17, 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 $LP(U_\beta^+)$.

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

3.6 Conclusions

In this chapter, 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.

Chapter 4

Divisible Load Scheduling in A Ring

4.1 Introduction

As a basic network topology, ring has been widely adopted in many real-world network interconnections, ranging from the small on-chip networks to the large optical networks [33]-[37]. In this chapter, we study the divisible load scheduling in the ring, and propose the optimal algorithm.

The rest of the chapter proceeds as follows. In Section 4.2, we formulate the divisible load in the ring as the *Maximum Finish Time Minimization (MFTM)* problem, propose the optimal solution for it by linearly relaxing the MFTM problem, and discusses the time complexity of the proposed optimal algorithm. In Section 4.3, we compare our proposed optimal algorithms with previously proposed heuristic algorithm. We conclude the chapter in Section

4.4.

4.2 Scheduling A Divisible Loads in A Ring

In this section, we discuss the divisible load scheduling in the ring with one source node.

As shown in Fig. 4.1, starting from the source node, nodes in the ring are labeled from 0 to $N-1$ in the clockwise direction. For presentational convenience, we denote a segment in the ring consisting of all nodes and links from node i to node j in the clockwise direction by a tuple (i, j), which is also called *chain* (i, j). The MFTM problem formulation of divisible load scheduling in the ring with one source node is in Table 4.1, where the source node is 0, and the initial load of node 0 is normalized as 1 in constraint (4.6). To solve the MFTM problem, we firstly linearly relax constraint 4.3 as follows to obtain the relaxed MFTM problem.

$$T_s(i) \geq T_s(j) + \beta(j,i)z(j,i)T_{cm}\}, \text{ if } \beta(j,i) > 0 \tag{4.1}$$

We define

$$U = \{\alpha(i), \beta(i,j), T_s(i), T_f(i) | 0 \leq i, j \leq N-1\}$$

is a feasible solution of the MFTM (or relaxed MFTM) problem if it satisfy all the constraints of its constraints, and the optimal solution for the MFTM (or relaxed MFTM) problem is denoted as U^* (or U_r^*). As will be seen, the MFTM problem and the relaxed MFTM problem have identical optimal

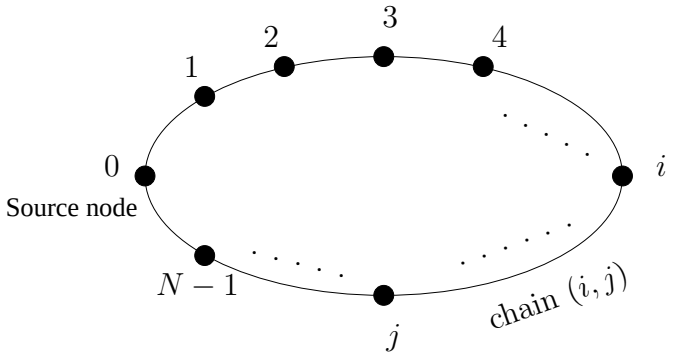

Figure 4.1: Starting from the source node, nodes in the ring are labeled from 0 to $N-1$ in the clockwise direction, and a segment in the ring consisting of all nodes and links from node i to node j in the clockwise direction is denoted by a tuple (i, j), which is named as *chain* (i, j).

solution, and to solve the MFTM problem in Table 4.1, we firstly analyze the properties of the optimal solution of the relaxed MFTM problem, i.e., U_r^*.

4.2.1 Properties of U_r^*

In this subsection, we prove several properties of U_r^*, based on which we propose the optimal algorithm for the relaxed MFTM problem of divisible load scheduling in a ring.

Theorem 4.1. *In* U_r^*, $T_f(i) = T_f(0)$ *if* $T_s(i) > 0$.

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 (4.3) and (4.4), we know that if $T_s(i) > 0$, node i must receive nonzero load from at least one of its neighbors, and that $T_f(i) = 0$ if $i \neq 0$ and $T_s(i) = 0$, therefore, $\max\{T_f(i) | 0 \leq i \leq N-1\}$ is either $T_f(0)$ or $\max\{T_f(i) | T_s(i) > 0\}$.

Table 4.1: Maximum Finish Time Minimization (MFTM) Problem Formulation for Divisible Load Scheduling in A Ring with One Source Node

Minimize: $\max\{T_f(i)|0 \leq i \leq N-1\}$

Subject to:

$$T_s(0) = 0 \tag{4.2}$$

$$T_s(i) = \max\{T_s(j) + \beta(j,i)z(j,i)T_{cm}|\beta(j,i) > 0\} \tag{4.3}$$

$$T_s(i) = 0, \text{ if } i \neq 0, \ \beta(j,i) = 0 \ \forall \ 0 \leq j \leq N-1 \tag{4.4}$$

$$T_f(i) = T_s(i) + \alpha(i)w(i)T_{cp} \tag{4.5}$$

$$\alpha(0) = 1 - \sum_{j=0}^{N-1} \beta(0,j) \tag{4.6}$$

$$\alpha(i) = \sum_{j=0}^{N-1} \beta(j,i), \text{ if } i \neq 0 \tag{4.7}$$

$$\beta(i,j) = 0, \text{ if } d(i,j) \neq 1 \tag{4.8}$$

$$\beta(i,j) = -\beta(j,i) \tag{4.9}$$

$$\beta(i,j) \geq 0, \text{ if } i = 0 \tag{4.10}$$

$$\alpha(i) \geq 0 \tag{4.11}$$

Next, we will reduce $\max\{T_f(i)|0 \leq i \leq N-1\}$ by redistributing load among nodes in the ring 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 i after the load redistribution as $T'_s(i)$ and $T'_f(i)$, respectively, and the load i kept for itself after the load redistribution is denoted as $\alpha'(i)$.

Two cases are considered in the proof.

Case 1: $T_f(0) < \max\{T_f(i)|0 \leq i \leq N-1\}$.

In this case,

$$\max\{T_f(i)|0 \leq i \leq N-1\} = \max\{T_f(i), T_s(i) > 0\}$$

and we suppose that a node with maximum finish time receives load from node 0 along an h-hop long path, as shown in Fig. 4.2, where the path is h-hop long, and node i_h has the maximum finish time. For convenience, the load received by node i_k from its previous hop node in the path is denoted as β_k, and $\beta_k > 0$ for all $1 \leq k \leq h$. Next, we decrease the finish time of i_h by redistributing load as follows.

We reduce β_k by $\delta\beta$, and keep $0 < \delta\beta < \beta_k$ for all $1 \leq k \leq h$, as shown in Fig. 4.2. After the load redistribution, we have that $T'_f(0) = T_f(0) + \delta\beta w(0) T_{cp}$ by constraint (4.6) in Table 4.1. As $T_f(0) < \max\{T_f(i)|0 \leq i \leq N-1\}$, we choose sufficiently small $\delta\beta$, such that $T'_f(0)$ is still earlier than $\max\{T_f(i)|0 \leq i \leq N-1\}$.

By constraint (4.7) in Table 4.1, we have that $\alpha'(i_k) = \alpha(i_k)$ for $1 \leq k < h$,

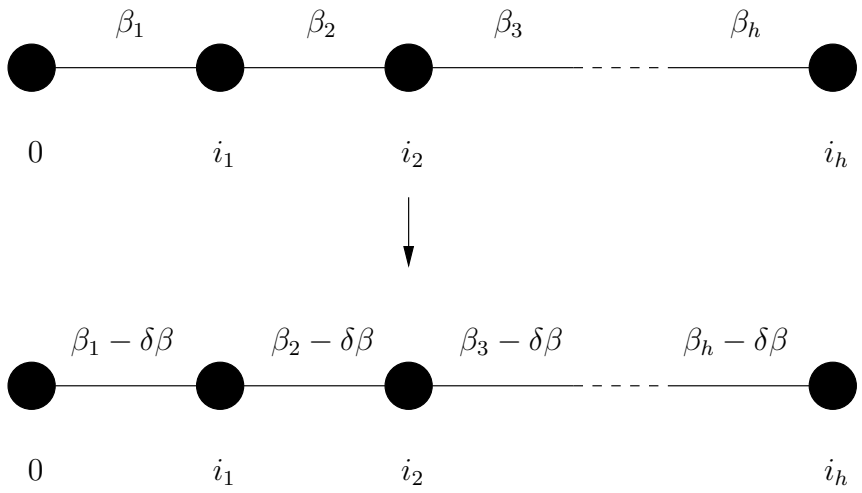

Figure 4.2: Load redistribution when $T_f(0) < T_f(i_h) = \max\{T_f(i)|0 \leq i \leq N-1\}$, where $\beta_k > 0$, and β_k is reduced by $\delta\beta$ ($0 < \delta\beta < \beta_k$ for all $1 \leq k \leq h$).

and $\alpha'(i_h) = \alpha(i_h) - \delta\beta$ after the load redistribution. Since i_k receives $\delta\beta$ less load from its previous hop node in the path, we can keep the start time of i_k unchanged after the load redistribution for $1 \leq k \leq h$, i.e., $T'_s(i_k) = T_s(i_k)$, and constraint (4.3) of the relaxed MFTM problem is still satisfied. Therefore, we have that $T'_f(i_k) = T_f(i_k)$ when $1 \leq k < h$, and

$$T'_f(i_h) = T_f(i_h) - \delta\beta w T_{cp} < \max\{T_f(i)|0 \leq i \leq N-1\}$$

Since the finish time of node 0 is still smaller than $\max\{T_f(i)|0 \leq i \leq N-1\}$ 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 $\max\{T_f(i)|0 \leq i \leq N-1\}$, which contradicts the fact that $\max\{T_f(i)|0 \leq i \leq N-1\}$ is already

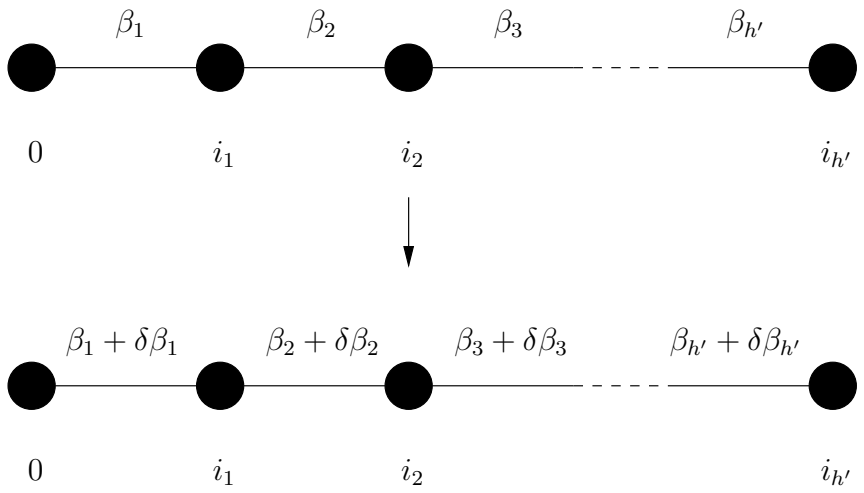

Figure 4.3: Load redistribution when $T_f(i_{h'}) < T_f(0) = \max\{T_f(i)|0 \leq i \leq N-1\}$, where $\beta_k > 0$, and β_k is increased by $\delta\beta_k$ ($\delta\beta_k > 0$) for all $1 \leq k \leq h'$.

minimized.

Case 2: $T_f(0) = \max\{T_f(i)|0 \leq i \leq N-1\}$.

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 receives load from node 0 along an h'-hop long path, and denote it as $i_{h'}$, as shown in Fig. 4.3. We denote the load received by node i_k in the path as β^k, where $1 \leq k \leq h'$. Next, we firstly reduce the finish time of node 0 without prolonging the maximum finish time by redistributing load as follows.

We increase β_k by $\delta\beta_k$, and have that $T'_f(0) = T_f(0) - \delta\beta_1 w(0) T_{cp}$.

After the load redistribution, we denote $T'_s(i_k) - T_s(i_k)$ as $\Delta T_s(i_k)$, the time when i_k finishes receiving load from i_{k-1} is then delayed by $\Delta T_s(i_{k-1}) + \delta\beta_k z(i_{k-1}, i_k) T_{cm}$, where $1 \leq k \leq h'$. Note that when $k = 1$, $i_{k-1} = 0$, and

$T'_s(0) = T_s(0) = 0$. We let $T'_s(i_k) = T_s(i_k) + \Delta T_s(i_k)$ such that constraint (4.3) is still satisfied after the load redistribution. Since $\Delta T_s(i_1) = T'_s(0) - T_s(0) + \delta\beta_1 z(0, i_1)T_{cm} = \delta\beta_1 z(0, i_1)T_{cm}$ and $\Delta T_s(i_k) = \Delta T_s(i_{k-1}) + \delta\beta_k z(i_{k-1}, i_k)T_{cm}$, where $1 < k \leq h'$, we have that

$$\Delta T_s(i_k) = \sum_{l=1}^{k} \delta\beta_l z(i_{l-1}, i_l)T_{cm} \tag{4.12}$$

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

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

and have that $T'_f(i_k) = T_f(i_k)$ by constraint (4.5).

Therefore, we have that

$$\delta\beta_{k+1} = \delta\beta_k + \frac{\Delta T_s(i_k)}{w(i_k)T_{cp}}$$

for $1 \leq k < h'$. Let $w_{min} = \min\{w(i_k), 0 \leq k \leq h'\}$, the following equation can be obtained

$$\delta\beta_k \leq \delta\beta_{k+1} \leq \delta\beta_k + \frac{\Delta T_s(i_k)}{w_{min}T_{cp}} \tag{4.13}$$

By Eq. (4.12) and (4.13), we have that

$$\Delta T_s(i_k) \leq \delta\beta_k \sum_{l=1}^{k} z(i_{l-1}, i_l)T_{cm} \tag{4.14}$$

By Eq. (4.13) and (4.14), we further deduct that

$$\delta\beta_{k+1} \leq \delta\beta_k(1 + \frac{\sum_{l=1}^{k} z(i_{l-1}, i_l)T_{cm}}{w_{min}T_{cp}}) \tag{4.15}$$

which indicates that

$$\delta\beta_{h'} \leq \delta\beta_1 \prod_{k=1}^{h'}(1 + \frac{\sum_{l=1}^{k} z(i_{l-1}, i_l)T_{cm}}{w_{min}T_{cp}}) \tag{4.16}$$

Therefore, by Eq. (4.14) and (4.16), we obtain that

$$\Delta T_s(i_{h'}) \leq \delta\beta_1 \prod_{k=1}^{h'}(1 + \frac{\sum_{l=1}^{k} z(i_{l-1}, i_l)T_{cm}}{w_{min}T_{cp}} \cdot (\sum_{l=1}^{h'} z(i_{l-1}, i_l)T_{cm}) \tag{4.17}$$

Since $T'_s(i_{h'}) = T_s(i_{h'}) + \Delta T_s(i_{h'})$, and node $i_{h'}$ may send load to its neighbor other than $i_{h'-1}$, as shown in Fig. 4.4, where we assume that $i_{h'}$ sends $\beta_{h'+1}$ ($\beta_{h'+1} > 0$) load to its neighbor $i_{h'+1}$. To avoid delaying the start time of $i_{h'+1}$, in the worst case, $\beta_{h'+1}$ have to be decreased by $\frac{\Delta T_s(i_{h'})}{z(i_{h'}, i_{h'+1})T_{cm}}$. Therefore, we have that

$$\alpha'(i_{h'}) = \alpha(i_{h'}) + \delta\beta_{h'} + \frac{\Delta T_s(i_{h'})}{z(i_{h'}, i_{h'+1})T_{cm}}$$

in the worst case, and

$$T'_f(i_{h'}) = T_f(i_{h'}) + \Delta T_s(i_{h'}) + (\delta\beta_{h'} + \frac{\Delta T_s(i_{h'})}{z(i_{h'}, i_{h'+1})T_{cm}})w(h')T_{cp} \tag{4.18}$$

Since $T_f(i_{h'}) < \max\{T_f(i)|0 \leq i \leq N-1\}$ before the load redistribution,

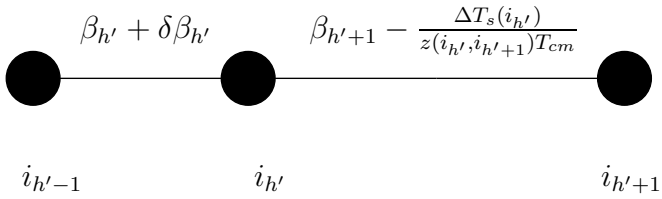

Figure 4.4: After the load redistribution, $T'_s(i_{h'}) = T_s(i_{h'}) + \Delta T_s(i_{h'})$, to avoid delaying the start time of its neighbor $i_{h'+1}$, the load sent to it, i.e., $\beta_{h'}$ is reduced by $\frac{\Delta T_s(i_{h'})}{z(i_{h'}, i_{h'+1})T_{cm}}$. Since $i_{h'}$ receives $\delta\beta_{h'}$ more load from its neighbor $i_{h'-1}$, we have that $\alpha'(i_{h'}) = \alpha(i_{h'}) + \delta\beta_{h'} + \frac{\Delta T_s(i_{h'})}{z(i_{h'}, i_{h'+1})T_{cm}}$ after the load redistribution.

by Eq. (4.16), (4.17) and (4.18), we can choose sufficiently small $\delta\beta_1 > 0$, such that $T'_f(i_{h'})$ is still earlier than $\max\{T_f(i)|0 \leq i \leq N-1\}$ after the load redistribution.

In summary, after the load redistribution, we have that

$$T'_f(0) = T_f(0) - \delta\beta_1 w(0) T_{cp} < \max\{T_f(i)|0 \leq i \leq N-1\}$$

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

Hence, if $T_s(i) > 0$, $T_f(i)$ must equal to $T_f(0)$ in U_r^*. □

Theorem 4.1 indicates the following corollary.

Corollary 4.2. *In* U_r^*, $\forall\omega \neq 0,\ T_s(\omega) > 0$.

Proof. We prove the corollary by contradiction, and assume that there exist $i \neq 0$, such that $T_s(i) = 0$. By Theorem 4.1, $\forall i \neq 0$, $T_f(i)$ equals to either $T_f(0)$ or 0, therefore, we can always find a pair of neighboring nodes, say, i_1 and i_2, such that $T_f(i_1) = T_f(0)$, $T_s(i_2) = T_f(i_2) = 0$. Next, we redistribute load by letting i_1 send a sufficiently small fraction of load to i_2, such that the finish time of i_2 is nonzero, but smaller than $\max\{T_f(i)|0 \leq i \leq N-1\}$. In addition, i_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 the ring, but now i_2 receives nonzero load from i_1, and its finish time is smaller than $\max\{T_f(i)|0 \leq i \leq N-1\}$, which contradicts Theorem 4.1. □

Theorem 4.1 and Corollary 4.2 in conjunction prove Theorem 4.3.

Theorem 4.3. *In U_r^*, $T_f(i)$ must be equal for all $0 \leq i \leq N-1$.*

By Theorem 4.3, constraint (4.5) in Table 4.1 can be simplified as follows:

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

where all nodes share equal finish time T_f, and the object function of the problem becomes minimizing T_f. Theorem 4.3 also indicates that the MFTM problem and relaxed MFTM problem share identical optimal solution, as stated by the next theorem.

Theorem 4.4. $U^* = U_r^*$.

Proof. Since constraint (4.1) is relaxed from constraint (4.3), 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, i, such that

$$T_s(i) > \max\{T_s(j) + \beta(j,i)z(j,i)T_{cm} | \beta(j,i) > 0\}$$

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

$$T'_s(i) = \max\{T_s(j) + \beta(j,i)z(j,i)T_{cm} | \beta(j,i) > 0\}$$

The new solution is still optimal for the relaxed MFTM problem, but the finish time of i is now earlier than $\max\{T_f(i) | 0 \leq i \leq N-1\}$, which contradicts Theorem 4.3, 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. □

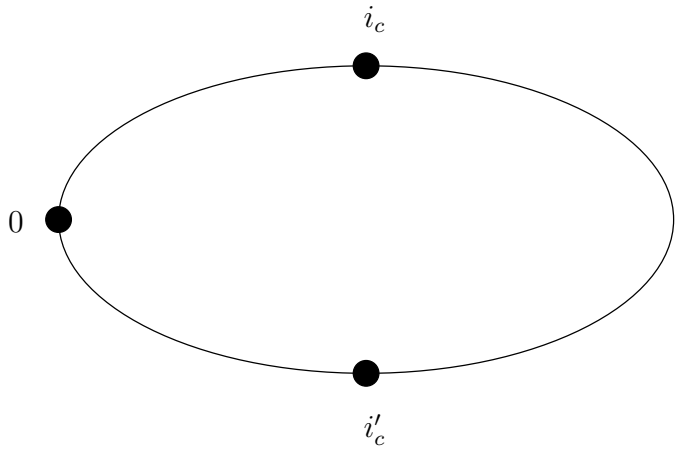

Figure 4.5: Existence of two convergence nodes, i_c and i'_c, in a ring.

Theorem 4.4 indicates that any node will start immediately after it finishes receiving load from its neighbors. For convenience, we say a node in the ring is the *converge node* if it receives nonzero load from both of its neighbors, and have the following lemma.

Lemma 4.1. *When the relaxed MFTM problem in Table 4.1 is optimized, there exists at most one converge node.*

Proof. We prove the lemma by contradiction and assume that there exist two converge nodes, denoted as i_c and i'_c, respectively, as shown in Fig. 4.5. Since there is only one source node in the ring, nodes residing at the chain (i_c, i'_c) would have negative load to process, which is unrealistic. Hence, the assumption is not true and the lemma holds. □

By Lemma 4.1, we discuss the relaxed MFTM problem in two situations, depending on whether converge node exists in its optimal solution in the next subsection.

4.2.2 Optimal Solution for the Relaxed MFTM Problem

In this subsection, we propose the optimal algorithm for the relaxed MFTM problem.

Existence of One Converge Node in U_r^*

We firstly discuss the situation with the existence of converge node. The converge node can be any node in the ring except for the source node 0, which only sends out load. Since there exists only one source node, given the converge node, say, i_c, load can only be sent to i_c along chain $(0, i_c)$ and chain $(i_c, 0)$, and we have that $\beta(i, i+1) > 0$ when $0 \leq i < i_c$, and $\beta(\text{mod}_N(i+1), i) > 0$ when $i_c \leq i \leq N-1$, where $\text{mod}_N(i)$ is i modulo N, as shown in Fig. 4.6. In addition, since $\beta(i,j) = 0$ if $d(i,j) \neq 1$ and $\beta(i,j) = -\beta(j,i)$ according to constraint (4.8) and (4.9) in Table 4.1, respectively, we can eliminate all $\beta(i,j)$s that are always zero, and substitute all negative $\beta(i,j)$s with positive ones. Moreover, only node i_c receives load from both of its neighbors, and a node starts processing and transmitting load immediately when it finishes receiving load from its neighbors by Theorem 4.4. Therefore, constraint (4.3) in Table 4.1 can be replaced by the following two constraints.

$$T_s(i_c) \geq T_s(j) + \beta(j, i_c) z(j, i_c) T_{cm} j = i_c - 1,\ \text{mod}_N(i_c + 1) \tag{4.19}$$

$$T_s(i) = T_s(j) + \beta(j,i)z(j,i)T_{cm},\ j = \begin{cases} i-1 \text{ if } 1 \leq i < i_c, \\ j = \mathrm{mod}_N(i+1) \text{ if } i_c < i \leq N-1 \end{cases} \tag{4.20}$$

Finally, all nodes must share equal finish time by Theorem 4.1, and we denote the equal finish time as $T_f(i_c)$ under a given converge node i_c. By the above discussion, we present an equivalent form of the relaxed MFTM problem of divisible load scheduling in a ring in Table 4.2, where the object function is to find out the i_c minimizing $T_f(i_c)$, and all $\beta(i,j)$s are nonnegative, reflected by constraint (4.25)-(4.28).

We notice that for a given converge node i_c, the problem in Table 4.2 becomes a linear optimization problem, which we denote as $LP(i_c)$. Since i_c can be $N-1$ different nodes in the ring according to constraint (4.29) in Table 4.2, we can find the optimal solution by solving these $N-1$ linear optimization problems, among which the one with minimum object function has equal optimal solution to the problem in Table 4.2.

No Existence of Converge Node

Now we discuss the situation that no converge node exists when the relaxed MFTM problem is optimized, and have the following corollary.

Corollary 4.5. *If there is no converge node when the relaxed MFTM problem is optimized, there exists one and only one pair of neighboring nodes, which do not send load to each other.*

Proof. We first show that there cannot be two pair of such nodes. As depicted

Table 4.2: Equivalent Form of the Relaxed MFTM Problem for Divisible Load Scheduling in A Ring with Converge Node

Minimize: $T_f(i_c)$

Subject to:

$$T_s(0) = 0 \tag{4.21}$$

$$T_s(i_c) \geq T_s(j) + \beta(j, i_c) z(j, i_c) T_{cm},\ j = i_c - 1,\ \mathrm{mod}_N(i_c + 1) \tag{4.22}$$

$$T_s(i) = T_s(j) + \beta(j, i) z(j, i) T_{cm},\ j = \begin{cases} i - 1 \text{ if } 1 \leq i < i_c \\ j = \mathrm{mod}_N(i + 1) \text{ if } i_c < i \leq N - 1 \end{cases} \tag{4.23}$$

$$T_f(i_c) = T_s(i) + \alpha(i) w(i) T_{cp} \tag{4.24}$$

$$\alpha(0) = 1 - \beta(0, 1) - \beta(0, N - 1) \tag{4.25}$$

$$\alpha(i_c) = \beta(i_c - 1, i_c) + \beta(\mathrm{mod}_N(i_c + 1), i_c) \tag{4.26}$$

$$\alpha(i) = \begin{cases} \beta(i - 1, i) - \beta(i, i + 1),\ \text{if } 1 \leq i < i_c \\ \beta(\mathrm{mod}_N(i + 1), i) - \beta(i, i - 1),\ \text{if } i_c < i \leq N - 1 \end{cases} \tag{4.27}$$

$$\alpha(i),\ \beta(i, j) \geq 0 \tag{4.28}$$

$$1 \leq i_c \leq N - 1 \tag{4.29}$$

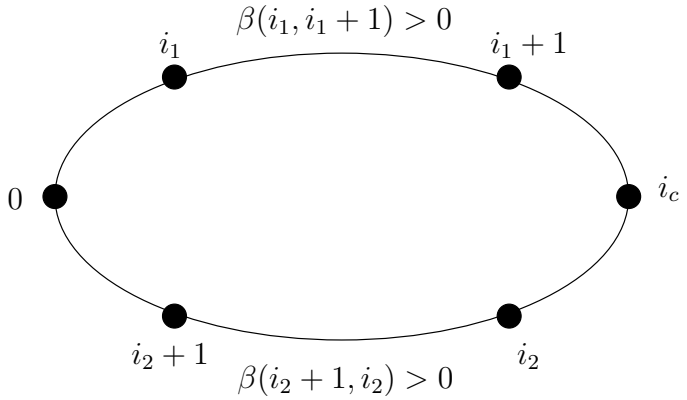

Figure 4.6: Load delivered to convergence node i_c along chain $(0, i_c)$ and chain $(i_c, 0)$, therefore, $\beta(i, i+1) > 0$ when $0 \leq i < i_c$ and $\beta(\text{mod}_N(i+1), i) > 0$ when $i_c \leq i \leq N-1$.

in Fig. 4.7, if there are two pair of such nodes, the ring is divided into two disjoint chains, and the one without the source node will not receive any load, the start and finish time of which, therefore, should be zero by constraint (4.4) and (4.5), which contradicts Theorem 4.1.

On the other hand, since there is only one source node, which only sends out load, and all nodes receive load from at most one of its neighbors, there must be one pair of neighboring nodes stopping sending load to each other.

□

By Corollary 4.5, we suppose that nodes i_s and $\text{mod}_N(i_s+1)$ stop sending loading to each other, and have that $\beta(i, i+1) > 0$ when $0 \leq i < i_s$ and $\beta(\text{mod}_N(i+1), i) > 0$ when $i_s < i \leq N-1$ as load is sent to i_s and $\text{mod}_N(i_s+1)$ along chain $(0, i_s)$ and $(\text{mod}_N(i_s+1), 0)$, respectively. As every node except for the source node receives load from one of its neighbors,

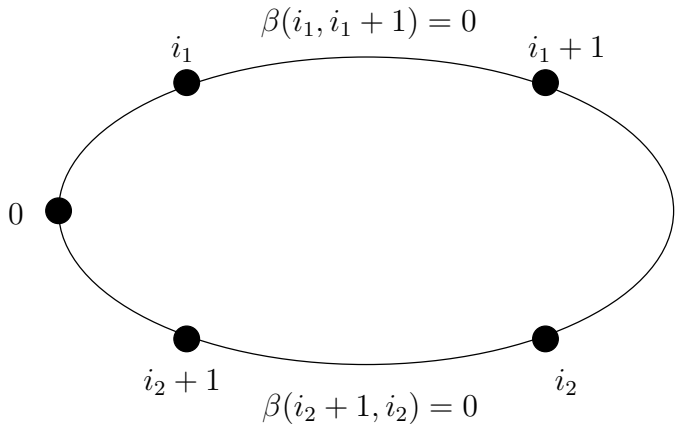

Figure 4.7: Node i_1 and i_2 stop sending load to their neighbors $i_1 + 1$ and $i_2 + 1$, respectively.

constraint (4.3) in Table 4.1 can be rewritten as

$$T_s(i) = T_s(j) + \beta(j,i)z(j,i)T_{cm},\ j = \begin{cases} i-1 \text{ if } 1 \le i \le i_s \\ j = \text{mod}_N(i+1) \text{ if } i_s < i \le N-1 \end{cases} \tag{4.30}$$

by Theorem 4.4. Similarly, we denote the equal finish time of all nodes as $T_f(i_s)$ by Theorem 4.1, and the equivalent form of the relaxed MFTM problem is given in Table 4.3, where all $\beta(i,j)$s being zero are eliminated, and all negative $\beta(i,j)$s are substituted with positive ones in constraint (4.34)-(4.36). The object function is to find out i_s that minimizes $T_f(i_s)$, and i_s can be any node in the ring, reflected by constraint (4.38).

Given i_s in Table 4.3, we notice that there are $3N + 1$ variables and the same number of linearly independent equations, and $T_f(i_s)$ can be obtained by solving a linear equation array with $3N + 1$ variables. As $0 \le i_s \le N - 1$,

Table 4.3: Equivalent Form of the Relaxed MFTM Problem for Divisible Load Scheduling in A Ring without Converge Node

Minimize: $T_f(i_s)$

Subject to:

$$T_s(0) = 0 \tag{4.31}$$

$$T_s(i) = T_s(j) + \beta(j,i)z(j,i)T_{cm},\ j = \begin{cases} i-1 \text{ if } 1 \le i \le i_s \\ j = \mathrm{mod}_N(i+1) \text{ if } i_s < i \le N-1 \end{cases} \tag{4.32}$$

$$T_f(i_s) = T_s(i) + \alpha(i)w(i)T_{cp} \tag{4.33}$$

$$\alpha(0) = 1 - \beta(0,1) - \beta(0,N-1) \tag{4.34}$$

$$\alpha(i) = \begin{cases} \beta(i-1,i) - \beta(i,\mathrm{mod}_N(i+1)),\ \text{if } 1 \le i \le i_s \\ \beta(\mathrm{mod}_N(i+1),i) - \beta(i,i-1),\ \text{if } i_s < i \le N-1 \end{cases} \tag{4.35}$$

$$\alpha(i),\ \beta(i,j) \ge 0 \tag{4.36}$$

$$\beta(i_s,\mathrm{mod}_N(i_s+1)) = \beta(\mathrm{mod}_N(i_s+1),i_s) = 0 \tag{4.37}$$

$$0 \le i_s \le N-1 \tag{4.38}$$

Table 4.4: High-Level Description of The Optimal Algorithm for the relaxed MFMT Problem of Divisible Load Scheduling in A Ring

for each $1 \leq i_c \leq N-1$
  calculate finish time $T_f(i_c)$ in Table 4.2 by solving $LP(i_c)$;
end for;
for each $0 \leq i_s \leq N-1$
  calculate finish time $T_f(i_s)$ in Table 4.3 by solving the
  corresponding linear equation array;
end for;
The solution is optimal when the finish time is minimum.
End

we can find the minimum $T_f(i_s)$ by solving N such linear equation arrays.

By the above discussion, we give the high-level description of the optimal algorithm for the relaxed MFTM problem of divisible load scheduling in a ring in Table 4.4.

4.2.3 Time Complexity of the Optimal Algorithm

In this subsection, we analyze the time complexity of the optimal algorithm relative to the number of nodes in the ring. Since the optimal algorithm needs to solve $N-1$ linear optimization problems, i.e., $LP(i_c)$ for $1 \leq i_c \leq N-1$ and N linear equation arrays, we begin with the analysis of the time complexity of solving $LP(i_c)$.

As discussed in the previous chapter, given the standard form of a linear optimization problem

$$\text{maximize } \mathbf{c}^T\mathbf{x}$$

$$\text{subject to } \begin{cases} \mathbf{Ax} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{cases} \tag{4.39}$$

where $\mathbf{c}$ is the n-dimensional coefficient vector and $\mathbf{A}$ is an $m \times n$ matrix, the linear optimization problem can be solved in polynomial time relative to the length of the binary encoding $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ by Khachiyan's result.

To convert $LP(i_c)$ to the above standard form, we add nonnegative slack variables, denoted as $T_j(i_c)$, in constraint (4.22) to transform the inequality to equality, i.e.,

$$T_s(i_c) = T_s(j) + \beta(j, i_c)T_{cm} + T_j(i_c)$$

After these operations, there are $3N + 3$ nonnegative variables and $3N + 2$ equalities in $LP(S_\beta^+)$, therefore, L in $LP(i_c)$ is polynomial to N, and the time complexity of solving $LP(i_c)$ is also polynomial to N by Khachiyan's result.

On the other hand, as solving a linear equation array with n variables equals to calculating the inverse of an $n \times n$ matrix, of which the time complexity is $O(n^{2.373})$ by Williams algorithm [38]. As mentioned above, we can calculate $T_f(i_s)$ by solving a linear equation array with $3N + 1$ variables, therefore, the time complexity of calculating $T_f(i_s)$ by Williams algorithm is $O(N^{2.373})$. Since $1 \leq i_c \leq N - 1$ and $0 \leq i_s \leq N - 1$, the time complexity of the optimal algorithm in Table 4.4 is polynomial to N.

4.3 Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm in terms of speedup, which is the ratio between the processing time of total load by a single standard processing node and the maximum finish time of all nodes in the ring.

As there is no specific algorithm for divisible load scheduling in the ring, we only compare our proposed optimal algorithm with the heuristic algorithm in [24], which schedules divisible load in arbitrary network with any number of source nodes. The heuristic algorithm regards the transmission time of a unit load, i.e., $z(i,j)T_{cm}$, as the weight of the link connecting node i to node j, and a node in the network only receives load from its closest source node in the sense that the source node has the minimum-weight path to it among all source nodes.

For the purpose of comprehensive comparison, we use 4 rings, each with 8 nodes. The unit load processing time, i.e., $w(i)T_{cp}$, of each node in the ring is a uniformly distributed random number in the interval $[0.5, 1.5]$, while the unit load transmission time, i.e., $z(i,j)T_{cm}$, of each link in the network is a uniformly distributed random number in the interval $[0.05, 0.15]$ considering that the transmission speed is usually much faster than the processing speed. The network parameters of the 4 rings in our comparison are given in Table 4.5. The total load is normalized to 1.

The speedup comparison results in the ring are given in Table 4.6, where we can see that the optimal algorithm and heuristic algorithm have very close

Table 4.5: Network Parameters of Four Rings with 8 Nodes

	Ring 1	Ring 2	Ring 3	Ring 4
$w(0)T_{cp}$	0.735	0.955	1.235	0.945
$w(1)T_{cp}$	0.565	0.615	0.935	0.635
$w(2)T_{cp}$	0.705	0.985	1.095	1.055
$w(3)T_{cp}$	0.915	1.495	1.125	1.395
$w(4)T_{cp}$	0.685	1.225	1.305	0.525
$w(5)T_{cp}$	0.515	0.835	1.205	0.505
$w(6)T_{cp}$	1.015	1.105	1.285	0.665
$w(7)T_{cp}$	1.195	0.595	0.765	1.155
$z(0,1)T_{cm}$	0.1075	0.1255	0.0625	0.1025
$z(1,2)T_{cm}$	0.1075	0.1125	0.0745	0.1155
$z(2,3)T_{cm}$	0.1035	0.0695	0.1105	0.0625
$z(3,4)T_{cm}$	0.1285	0.1205	0.0805	0.0995
$z(4,5)T_{cm}$	0.1245	0.1435	0.1165	0.1415
$z(5,6)T_{cm}$	0.0815	0.0515	0.0715	0.0735
$z(6,7)T_{cm}$	0.1035	0.1015	0.0595	0.0855
$z(7,0)T_{cm}$	0.0945	0.1275	0.1155	0.1365
$z(0,7)T_{cm}$	0.0835	0.1385	0.0885	0.1415
$z(7,6)T_{cm}$	0.0875	0.0745	0.0705	0.1275
$z(6,5)T_{cm}$	0.1095	0.1305	0.1285	0.1345
$z(5,4)T_{cm}$	0.1395	0.0925	0.1275	0.0925
$z(4,3)T_{cm}$	0.1275	0.0825	0.1255	0.0675
$z(3,2)T_{cm}$	0.0545	0.0695	0.0865	0.1435
$z(2,1)T_{cm}$	0.0615	0.0955	0.0685	0.1055
$z(1,0)T_{cm}$	0.0845	0.0835	0.0825	0.0725

Table 4.6: Comparison of Speedup between Optimal Algorithm and Heuristic Algorithm

	Ring 1	Ring 2	Ring 3	Ring 4
Optimal Algorithm	6.693	5.763	5.534	5.927
Heuristic Algorithm	6.657	5.760	5.521	5.927

performance in the 4 rings. This is because that the minimum-weight path chosen by the heuristic algorithm introduces the least communication delay, and is generally the best path for load distribution in the ring. Nevertheless, the optimal algorithm allows the convergence node to receive load from two neighbors while the each node receives load from at most one of its neighbors in the heuristic algorithm, therefore, the optimal algorithm still outperforms the heuristic algorithm in 3 of the 4 rings.

4.4 Conclusions

In this chapter, we studied divisible load scheduling in the ring by our proposed novel analysis method, and propose the optimal algorithm. We prove that the time complexity of the optimal algorithm is poly-nominal relative to the ring size, and comparison with previously proposed heuristic algorithm in terms of maximum finish time and speedup demonstrates the superiority of the optimal algorithms, which convinces us the merit of the proposed novel analysis method.

Chapter 5

Divisible Load Scheduling in A Multi-Root Tree

5.1 Introduction

A multi-root tree is a complete bipartite graph, where one set of disjoint nodes are the roots, and the other set of disjoint nodes are the leaves. Divisible load scheduling in the multi-root tree with two roots was studied in [23], where the roots are source nodes with nonzero initial load, and distribute load to the leaves. A leaf can start processing load when it finishes receiving load from the first root, but the second root is required to finish sending the load to the leaf before the leaf finishes processing the load from the first root. The requirement is unreasonable, and in this chapter, we study the divisible load scheduling in the multi-root tree with our proposed novel analysis method, and propose the optimal algorithm.

The rest of the chapter proceeds as follows. In Section 5.2, we formulate the divisible load in the ring as the *Maximum Finish Time Minimization (MFTM)* problem, propose the optimal solution for it by firstly relaxing the MFTM problem, and then transforming the relaxed MFTM problem into the *Finish Time Minimization (FTM)* problem, and discusses the time complexity of the proposed optimal algorithm. In Section 5.3, we compare our proposed optimal algorithms with previously proposed heuristic algorithm. We conclude the chapter in Section 5.4.

5.2 Scheduling A Divisible Load in A Multi-Root Tree

In this section, we study the divisible load scheduling in a multi-root tree, which is also formulated as the MFTM problem, which undergoes a linear relaxation to yield the relaxed MFTM problem. The relaxed MFTM problem is then is transformed into linear optimization problem, which we denote as *Finish Time Minimization (FTM)* problem.

5.2.1 Problem Formulation

As shown in Fig. 5.1, the multi-root tree consists of M roots and N leaves, and any root is connected to all the leaves. The roots are labelled from 0 to $M-1$, and leaves are labelled from M to $M+N-1$. All the roots are source nodes, and have positive load initially, the leaf can start processing

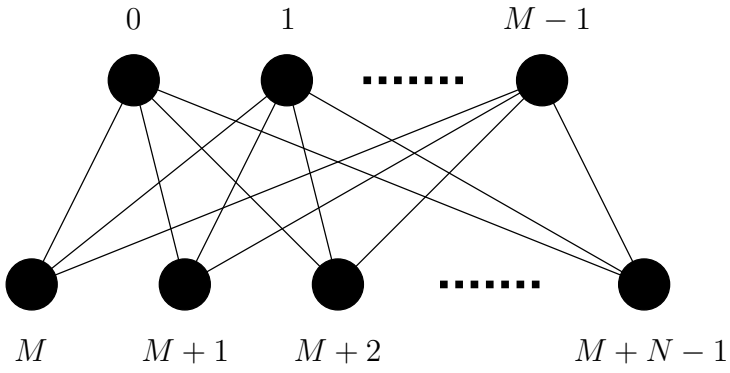

Figure 5.1: A multi-root tree with M roots and N leaves, where M roots are labeled by integers from 0 to $M-1$, and N roots are labeled by integers from M to $M+N-1$.

load when it finishes receiving load from all the roots. The MFTM problem formulation of divisible load scheduling in a multi-root tree is in Table 5.1, where the object is still to minimize the maximum finish time of all nodes in the multi-root tree, and constraint (5.1) reflects that i and j are used to denote roots and leaves, respectively.

Similar to the previous chapters, constraint (5.4) is relaxed to the linear inequality, by which we obtain the relaxed MFTM problem.

$$T_s(j) \geq T_s(i) + \beta(i,j)z(i,j)T_{cm}, \text{ if } \beta(i,j) > 0 \tag{5.10}$$

We define

$$U = \{\alpha(i), \alpha(j), \beta(i,j),$$
$$T_s(i), T_s(j), T_f(i), T_f(j) | 0 \leq i \leq M-1,\ M \leq j \leq M+N-1\}$$

Table 5.1: MFTM Problem Formulation of Divisible Load Scheduling in A Multi-Root Tree

Minimize: $\max\{T_f(i), T_f(j)\}$

Subject to:

$$0 \leq i \leq M-1,\ M \leq j \leq M+N-1 \tag{5.1}$$

$$T_s(i) = 0 \tag{5.2}$$

$$T_s(j) = 0,\ \text{if } \beta(i,j) = 0\ \forall\ 0 \leq i \leq M-1 \tag{5.3}$$

$$T_s(j) = \max\{T_s(i) + \beta(i,j)z(i,j)T_{cm} | \beta(i,j) > 0\} \tag{5.4}$$

$$T_f(i) = T_s(i) + \alpha(i)w(i)T_{cp} \tag{5.5}$$

$$T_f(j) = T_s(j) + \alpha(j)w(j)T_{cp} \tag{5.6}$$

$$\alpha(i) = L(i) - \sum_{j=M}^{M+N-1} \beta(i,j) \tag{5.7}$$

$$\alpha(j) = \sum_{i=0}^{M-1} \beta(i,j) \tag{5.8}$$

$$\alpha(i),\ \alpha(j),\ \beta(i,j) \geq 0 \tag{5.9}$$

is a feasible solution of the MFTM (or relaxed MFTM) problem if it satisfy all the constraints of its constraints, and the optimal solution for the MFTM (or relaxed MFTM) problem is denoted as U^* (or U_r^*). As will be seen, the MFTM problem and the relaxed MFTM problem have identical optimal solution, and to solve the MFTM problem in Table 5.1, we firstly analyze the properties of the optimal solution of the relaxed MFTM problem, i.e., U_r^*.

5.2.2 Properties of U_r^*

To solve the relaxed MFTM problem in Table 5.1, we firstly analyze the properties of its optimal solution, U_r^*, and have the following lemma and theorem.

Lemma 5.1. *When the relaxed MFTM problem in Table 5.1 is optimized,*

$$\max\{T_f(i)\} \leq \max\{T_f(j)\}$$

where $0 \leq i \leq M-1$, $M \leq j \leq M+N-1$.

Proof. We prove the lemma by contradiction, and assume that

$$\max\{T_f(i)\} > \max\{T_f(j)\}$$

Therefore, the maximum finish time of the multi-root tree should be $\max\{T_f(i)\}$. Next, we redistribute load among roots and leaves to obtain a smaller maximum finish time of the multi-root tree.

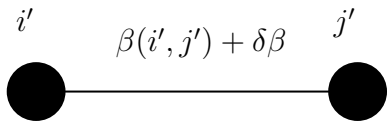

Figure 5.2: Root i' sends $\delta\beta$ more load to leaf j'.

Let $T_f(i') = \max\{T_f(i)|0 \leq i \leq M-1\}$, in the load redistribution, root i' sends another small fraction of load, $\delta\beta$, to leave j', as shown in Fig. 5.2, after which, we set the start time of j' as $T_s(j')+\delta\beta z(i', j')T_{cm}$ such that constraint (5.10) is still satisfied. Therefore, the finish time of i' and j' become $T_f(i') - \delta\beta w(i')T_{cp}$ and $T_f(j') + \delta\beta z(i', j')T_{cm} + \delta\beta w(j')T_{cp}$, respectively. Since $T_f(j')$ is smaller than $\max\{T_f(i)\}$, we can choose sufficiently small $\delta\beta$ such that the finish time of j' is still smaller than $\max\{T_f(i)\}$ after the load redistribution. In addition, the finish time of i' is also smaller than $\max\{T_f(i)\}$ after the load redistribution.

Hence, we can repeat the load redistribution for the rest roots with the maximum finish time to reduce their finish time, by which a smaller maximum finish time of the multi-root tree can be obtained, and this contradicts that the original maximum finish time is minimized already. Therefore, the assumption is not true, and the lemma holds. □

Theorem 5.1. *When the relaxed MFTM problem in Table 5.1 is optimized,* $T_f = T_f(j)\ \forall\ M \leq j \leq M + N - 1$.

Proof. We prove the theorem by contradiction, and assume that the leaves do not share equal finish time.

Let $T_f(j_1) = \max\{T_f(j)\}$ and $T_f(j_2) = \min\{T_f(j)\}$, since the leaves do not finishing processing load simultaneously, we have that $T_f(j_1) > T_f(j_2)$. By Lemma 5.1, j_1 should have the maximum finish time of the multi-root tree, therefore, there must exist at least one root, say, i, sending nonzero load to it, as shown in Fig. 5.3. Next, we redistribute load to reduce the finish time of j_1, and obtain a smaller maximum finish time of the multi-root tree.

In the load redistribution, i sends $\delta\beta$ less load to j_1, and $\delta\beta$ more load to j_2. After the load redistribution, the start time of j_2 is set as $T_s(j_2) + \delta\beta z(i, j_2)T_{cm}$ such that constraint (5.10) is not violated, and the start time of j_1 is kept unchanged. Therefore, we have that the finish time of j_1 and j_2 become $T_f(j_1) - \delta\beta w(j_1)T_{cp}$ and $T_f(j_2) + \delta\beta z(i, j_2)T_{cm} + \delta\beta w(j_2)T_{cp}$, respectively, and the finish time of i is still $T_f(i)$ by constraint (5.5). Since $T_f(j_1) > T_f(j_2)$, we can choose sufficiently small $\delta\beta$ to keep the finish time of j_2 smaller than the maximum finish time of the multi-root tree after the load redistribution.

By repeating the above steps, all leaves with maximum finish time can have earlier finish time, resulting a smaller maximum finish time, which contradicts that the solution is optimal. Therefore, the assumption is not true, and the theorem holds. □

Theorem 5.1 also indicates that the MFTM problem and relaxed MFTM problem share identical optimal solution, as stated by the next theorem.

Theorem 5.2. $U^* = U_r^*$.

Proof. Since constraint (5.10) is relaxed from constraint (5.4), U^* is feasible

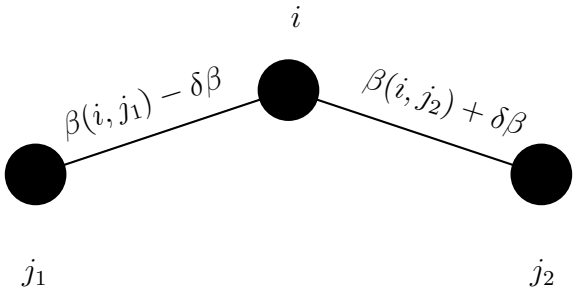

Figure 5.3: Root i sends $\delta\beta$ less load and $\delta\beta$ more load to leaf j_1 and j_2, respectively.

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 leaf, say, j, such that

$$T_s(j) > \max\{T_s(i) + \beta(i,j)z(j,i)T_{cm} | \beta(i,j) > 0\}$$

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

$$T'_s(j) = \max\{T_s(i) + \beta(i,j)z(j,i)T_{cm} | \beta(i,j) > 0\}$$

The new solution is still optimal for the relaxed MFTM problem, but the finish time of j is now earlier than $\max\{T_f(j)|M \leq j \leq M+N-1\}$, which contradicts Theorem 5.1, i.e., all leaves 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. □

Theorem 5.2 indicates that any node will start immediately after it finishes receiving load from the roots.

5.2.3 Optimal Solution for the Relaxed MFTM Problem

By the above lemma and theorems, we have that when the relaxed MFTM problem in Table 5.1 is optimized, all the leaves should have equal finish time, denoted as T_f, which is also the maximum finish time of all nodes in the multi-root tree. Based on the conclusion, we transform the relaxed MFTM problem in Table 5.1 to a *Finish Time Minimization (FTM)* problem in Table 5.2, where the object function is to minimize the equal finish time of all leaves, i.e., T_f, and the finish time of any root is no greater than T_f, reflected by constraint (5.16).

Clearly, the relaxed MFTM problem and FTM problem have equal optimal solution. In addition, as $T_s(i) = 0$, constraint (5.13) in Table 5.2 can be simplified as

$$T_s(j) \geq T_s(i) + \beta(i,j)z(i,j)T_{cm} \tag{5.20}$$

Table 5.2: Finish Time Minimization (FTM) Problem Formulation of Divisible Load Scheduling in A Multi-Root Tree

Minimize: T_f

Subject to:

$$0 \leq i \leq M-1,\ M \leq j \leq M+N-1 \tag{5.11}$$

$$T_s(i) = 0 \tag{5.12}$$

$$T_s(j) \geq T_s(i) + \beta(i,j)z(i,j)T_{cm},\ \text{if } \beta(i,j) > 0 \tag{5.13}$$

$$T_f(i) = T_s(i) + \alpha(i)w(i)T_{cp} \tag{5.14}$$

$$T_f = T_s(j) + \alpha(j)w(j)T_{cp} \tag{5.15}$$

$$T_f \geq T_f(i) \tag{5.16}$$

$$\alpha(i) = L(i) - \sum_{j=M}^{M+N-1} \beta(i,j) \tag{5.17}$$

$$\alpha(j) = \sum_{i=0}^{M-1} \beta(i,j) \tag{5.18}$$

$$\alpha(i),\ \alpha(j),\ \beta(i,j) \geq 0 \tag{5.19}$$

which holds for $\beta(i,j) \geq 0$. Therefore, the FTM problem in Table 5.2 is actually a linear optimization problem. Next, we analyze the time complexity of solving the FTM problem.

5.2.4 Time Complexity of the Optimal Algorithm

In this subsection, we analyze the time complexity of the optimal algorithm for solving the FTM problem in Table 5.2 relative to the number of roots and leaves in the ring.

To convert the FTM problem into the following standard form of a linear optimization program,

$$\text{maximize } \mathbf{c}^T\mathbf{x}$$
$$\text{subject to } \begin{cases} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{cases} \tag{5.21}$$

where $\mathbf{c}$ is the n-dimensional coefficient vector and $\mathbf{A}$ is an $m \times n$ matrix, we add nonnegative slack variables $T_1(i,j)$ and $T_2(i)$ in constraint (5.20) and (5.16), respectively. Then we have the slack form of these two constraints as follows.

$$T_s(j) = T_s(i) + \beta(i,j)z(i,j)T_{cm} + T_1(i,j) \tag{5.22}$$

$$T_f = T_f(i) + T_2(i) \tag{5.23}$$

After these operations, we have $2MN + 4M + 2N$ nonnegative variables and $MN + 4M + 2N$ linear equations in the FTM problem, therefore, according to Khachiyan's result, the time complexity of solving the FTM problem

should be polynomial relative to MN.

5.3 Performance Comparison

In this section, we evaluate the performance of our proposed algorithm in terms of speedup, which is the ratio between the processing time of total load by a single standard processing node and the maximum finish time of all nodes in the network.

As the algorithm for divisible load scheduling in the multi-root tree in [23] allows the node to process load before it finishes receiving load from all its neighbors, which is quite different from the assumptions of our proposed algorithm, we only compare our proposed optimal algorithm with the heuristic algorithm in [24], which schedules divisible load in arbitrary network with any number of source nodes. The heuristic algorithm regards the transmission time of a unit load, i.e., $z(i,j)T_{cm}$, as the weight of the link connecting node i to node j, and a node in the network only receives load from its closest source node in the sense that the source node has the minimum-weight path to it among all source nodes.

For the purpose of comprehensive comparison, we use 4 multi-root trees, each with 4 roots and 4 leaves. The unit load processing time, i.e., $w(i)T_{cp}$, of each node in the network is a uniformly distributed random number in the interval $[0.5, 1.5]$, while the unit load transmission time, i.e., $z(i,j)T_{cm}$, of each link in the network is a uniformly distributed random number in the interval $[0.05, 0.15]$ considering that the transmission speed is usually much

faster than the processing speed. The network parameters of the 4 multi-root trees in our comparison are given in Table 5.3. The total load is normalized to 1, and 4 roots of the multi-root tree each have 0.25 load initially.

Fig. 5.4 plots the speedup comparison results in the 4 multi-root trees, where the optimal algorithm greatly surpasses the heuristic algorithm. The speedup under the optimal algorithm is triple and even quadruple of that under the heuristic algorithm, and the reason is two-fold. First, the heuristic algorithm allows each leaf to receive load from only one of the 4 roots in the multi-root tree, while the leaves can receive from 4 roots simultaneously in the optimal algorithm, which is more efficient in utilizing links to distribute load, and results in much shorter load transmission time. Second, as the heuristic algorithm only focuses on the transmission time in load distribution, while ignores the processing time of each node in the network, it can easily cause unevenness in load distribution, resulting in large maximum finish time. In fact, under the heuristic algorithm, there is always one source node which has to process all its load in these 4 multi-root trees, and such source node has the maximum finish time. On the other hand, the optimal algorithm takes processing time, transmission time, and initial load of each source node into consideration, which distributes load evenly in the network, thus has much smaller maximum finish time.

Table 5.3: Network Parameters of Four Multi-Root Trees with 4 Roots and 4 Leaves

	Tree 1	Tree 2	Tree 3	Tree 4
$w(0)T_{cp}$	1.225	1.455	0.995	0.865
$w(1)T_{cp}$	0.505	1.055	1.295	1.115
$w(2)T_{cp}$	0.545	0.715	0.515	0.795
$w(3)T_{cp}$	1.235	0.735	0.545	1.145
$w(4)T_{cp}$	1.305	0.915	0.755	0.895
$w(5)T_{cp}$	0.705	0.955	0.845	0.705
$w(6)T_{cp}$	1.485	1.155	1.275	1.405
$w(7)T_{cp}$	1.495	0.725	1.375	1.385
$z(0,4)T_{cm}$	0.0655	0.1315	0.0865	0.0805
$z(0,5)T_{cm}$	0.1065	0.1235	0.1405	0.0765
$z(0,6)T_{cm}$	0.1105	0.0775	0.1035	0.0595
$z(0,7)T_{cm}$	0.0515	0.1125	0.0855	0.0595
$z(1,4)T_{cm}$	0.0995	0.1085	0.1435	0.1185
$z(1,5)T_{cm}$	0.0535	0.0905	0.1305	0.1125
$z(1,6)T_{cm}$	0.0825	0.0695	0.1035	0.0875
$z(1,7)T_{cm}$	0.1285	0.1045	0.0895	0.0585
$z(2,4)T_{cm}$	0.1125	0.0715	0.1245	0.1175
$z(2,5)T_{cm}$	0.0755	0.0805	0.1145	0.0545
$z(2,6)T_{cm}$	0.1285	0.0575	0.0515	0.1485
$z(2,7)T_{cm}$	0.1435	0.1105	0.1235	0.0895
$z(3,4)T_{cm}$	0.1185	0.0795	0.1475	0.1125
$z(3,5)T_{cm}$	0.1355	0.1455	0.0515	0.0915
$z(3,6)T_{cm}$	0.1095	0.0885	0.1375	0.1325
$z(3,7)T_{cm}$	0.0585	0.0645	0.1275	0.1195

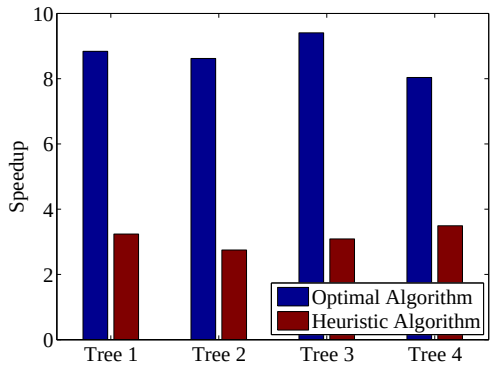

Figure 5.4: Speedup comparison between the optimal algorithm and heuristic algorithm in the multi-root tree.

5.4 Conclusions

In this chapter, we studied divisible load scheduling in the multi-root tree by our proposed novel analysis method, and propose the optimal algorithm. We prove that the time complexity of the optimal algorithm is poly-nominal relative to MN, where M and N are the number of roots and leaves, respectively, and comparison with previously proposed heuristic algorithm in terms of maximum finish time and speedup demonstrates the superiority of the optimal algorithms, which convinces us the merit of the proposed novel analysis method.

Chapter 6

Conclusions

In this thesis, we propose a novel analysis method for divisible load scheduling in networks where nodes are allowed to receive load from more than one of their neighbors, which can not be solved by the equivalent processing node method in [6]. We discover that the nonlinear equations introduced by the novel analysis method can be replaced by linear inequalities, and the object function can be linearized by introducing an equal finish time, after which the problem resembles a linear optimization problem, and can be solved by linear programming. Following the conclusion, we studied divisible load scheduling in the Gaussian network, mesh, torus, ring and multi-root tree by the novel analysis method, and propose the optimal algorithms for these network topologies. We prove that the time complexity of the optimal algorithms for the former three network topologies are exponential relative to the network size, and a heuristic algorithm is proposed to reduce the time complexity. Due to its high link utilization, the heuristic algorithm achieves

almost equally good performance to the optimal algorithm, and outperforms previously proposed heuristic algorithms in these three networks, which is demonstrated by our extensive simulation results. The time complexity of the optimal algorithms for a ring is polynomial relative to the ring size. The time complexity of the optimal algorithm for the multi-root tree is polynomial relative to MN, where M and N are the number of roots and leaves in the multi-root tree, respectively. The comparison with previously proposed heuristic algorithm for the ring and multi-root tree demonstrates the superiority of the optimal algorithms.

Bibliography

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

[33] Milo, et al. (2013). On-chip ring network designs for hard-real time systems. Proceedings of the 21st International conference on Real-Time Networks and Systems. Sophia Antipolis, France, ACM: 23-32.

[34] Le Beux, S., et al. (2011). Optical Ring Network-on-Chip (ORNoC): Architecture and design methodology. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2011.

[35] Chao, I. F. and M. C. Yuang (2013). "Toward Wireless Backhaul Using Circuit Emulation Over Optical Packet-Switched Metro WDM Ring Network." Lightwave Technology, Journal of 31(18): 3032-3042.

[36] Furukawa, H., et al. (2012). "Optical packet and circuit integrated node for ring network testbed." Photonics in Switching (PS), 2012 International Conference on.

[37] Jinno, M., et al. (2010). "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network [Topics in Optical Communications]." Communications Magazine, IEEE 48(8): 138-145.

[38] V. V. Williams, "Multiplying matrices faster than coppersmith-winograd," *Proceedings of the 44th symposium on Theory of Computing,* May 19-22, 2012.