Electronic version of an article published as
Discrete Mathematics, Algorithms and Applications Vol. 6, No. 2 (2014) 1450022 (16 pages),
DOI: 10.1142/S1793830914500220 (c) 2014 Copyright World Scientific Publishing Company.

Priority-based task reassignments in hierarchical 2D mesh-connected systems using tableaux

Dohan Kim*

A.I. Research Co., 2537-1 Kyungwon Plaza 201, Sinheung-dong, Sujeong-gu,
Seongnam-si, Kyunggi-do, 461-811, South Korea

Abstract

Task reassignments in 2D mesh-connected systems (2D-MSs) have been researched for several decades. We propose a hierarchical 2D mesh-connected system (2D-HMS) in order to exploit the regular nature of a 2D-MS. In our approach priority-based task assignments and reassignments in a 2D-HMS are represented by tableaux and their algorithms. We show how task relocations for a priority-based task reassignment in a 2D-HMS are reduced to a jeu de taquin slide.

Keywords: Task relocation; Task reassignment; Young tableau; 2D mesh; Jeu de taquin

1 Introduction

A distributed system is a collection of processing nodes connected by an interconnection network [21, 31]. Among various interconnection networks for distributed systems, a two-dimensional (2D) mesh has received extensive study due to its simplicity, efficiency, and structural regularity [6, 26, 35, 36]. Some data structures, such as matrices and arrays, naturally fit into a 2D mesh-connected system (2D-MS) [26]. Since tasks are often assigned to a submesh in a 2D-MS, continuous submesh allocations and deallocations of different sizes may cause *fragmentation* [35, 36] in a 2D-MS. Task relocation is an approach to decrease fragmentation by reassigning a running task to an idle processing node, which involves capturing and transferring the state of the running task to the idle node in a 2D-MS [36].

Although the construction of a 2D-MS using heterogeneous nodes is considered in [6], most of the traditional approaches [1, 24, 26, 35, 36] are based on the assumptions that nodes in a 2D-MS are homogeneous. Further, the topology of the assigned tasks on a 2D-MS is restricted (e.g., rectangular or square-mesh shape) and priority-based task assignments and reassignments have not often been considered. Therefore, there is a lack of systematic mechanisms of task relocations in a heterogeneous 2D-MS.

*E-mail: dkim@airesearch.kr

For decades, a wide variety of ways to tackle task assignment and reassignment problems in a distributed system have been researched, such as graph-theoretic [3, 15, 20, 25, 34], mathematical programming [8], and heuristics [10]. One of the common methods to represent and solve a task assignment problem is a graph-theoretic method using a graph-matching algorithm [15, 25, 34]. To complement the graph-theoretic method, our previous work [16] presented a *Young tableaux* [13, 22, 29, 37] approach to representing task assignments.

We use tableaux and their algorithms for priority-based task reassignments in a hierarchical 2D-MS, where a hierarchical 2D-MS (2D-HMS) is defined as a 2D-MS consisting of heterogeneous nodes whose priorities (or execution rates) of rows and columns are sorted in descending order. In this paper we convert a 2D-HMS into a *Young diagram* [22, 29] and represent a task assignment of a 2D-HMS using a tableau. Our greedy task relocation policy is based on a 2D-HMS in which task relocations are performed systematically by using tableau algorithms.

The remainder of this paper is organized as follows. We provide an introduction to tableaux and their algorithms in Section 2. Section 3 presents a representation of a 2D-HMS using a Young diagram. In this section we define a hierarchical 2D mesh tableau in order to represent a priority-based task assignment and its reassignments in a 2D-HMS. Section 4 shows how task relocations in a hierarchical 2D mesh tableau under the greedy task relocation policy are reduced to a jeu de taquin slide, and how they are applied to a 2D-HMS. Finally, we conclude in Section 5.

2 Preliminaries

This section provides necessary definitions and terminology used in this paper. Definitions and results in this section are found in [1, 7, 11–14, 16, 22, 24, 27–30, 33, 36, 37].

A *heterogeneous system* N is a set of heterogeneous nodes $N = \{n_1, n_2, \ldots, n_m\}$ whose communications are described by a network topology. By a *node* we mean a processor (or agent) that carries out a task. A heterogeneous system N is said to be *consistent* if node $n_a \in N$ executes a task d times faster than node $n_b \in N$, then it executes all other tasks d times faster than node n_b. In a consistent system the computation cost of task v_s on node n_t is defined by $\omega(v_s, n_t) = r(v_s)/e(n_t)$, where $r(v_s)$ is the computation (or resource) requirement of task v_s, and $e(n_t)$ is the execution rate of node n_t.

Let $T = \{t_1, t_2, \ldots, t_a\}$ be a set of a tasks with or without precedence constraints and $N = \{n_1, n_2, \ldots, n_b\}$ be a set of b nodes. Let $A : T \to N$ be a task assignment function between T and N. Let $t_n^e(A)$ denote the total execution time of node n for the task assignment A and let $t_n^i(A)$ denote the total idle time of node n for the task assignment A. The *turnaround time* of node n for the task assignment A is the total time spent in the node n for the task assignment A. Let $t_n(A) := t_n^i(A) + t_n^e(A)$ and $t(A) := \max_n t_n(A)$. We call $t(A)$ the *task turnaround time* of the task assignment A.

A *2D mesh-connected system* (or *2D-MS for short*) is a set of $m \times n$ nodes structured as a rectangular grid of height m and width n. Each node is addressed by its coordinate

(i, j) for $1 \leq i \leq m$ and $1 \leq j \leq n$. An *internal node* (x, y), where $1 < x < m$ and $1 < y < n$, is directly connected to its four adjacent nodes $(x-1, y)$, $(x+1, y)$, $(x, y-1)$, $(x, y+1)$. A node in the four corners has two adjacent nodes, while a node in the remaining boundary has three adjacent nodes, respectively (see Figure 1(d) in Section 3). An $m' \times n'$ *submesh* of an $m \times n$ 2D-MS is a grid of nodes belonging to the 2D-MS with height m' and width n' such that $1 \leq m' \leq m$ and $1 \leq n' \leq n$. A submesh is called *free* if every node in the submesh is idle. We assume that each incoming job (i.e., a set of tasks) requests a submesh of a certain size and that every node in the allocated submesh cannot be used for an incoming job until it is deallocated. We say that *internal fragmentation* occurs if more nodes in a 2D-MS are allocated to a job than required. We say that *external fragmentation* occurs if a large enough submesh cannot be found for an incoming job although there are a sufficient number of nodes in a 2D-MS are available.

A *partition* of n is defined as a sequence $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_i)$, where the λ_k are weakly decreasing and $\sum_{k=1}^{i} \lambda_k = n$. If λ is a partition of n, then we write $\lambda \vdash n$.

Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_i) \vdash n$. A *Young diagram* (or *Ferrers diagram*) of shape λ is a left-justified, finite collection of cells, with row j containing λ_j cells for $1 \leq j \leq i$. Each cell in a Young diagram in row i and column j has a coordinate (i, j), as in a 2D-MS.

Let λ be a partition. An *inner corner* of the Young diagram of shape λ is a cell $(i, j) \in \lambda$ whose removal leaves the Young diagram of a partition.

Let $\lambda \vdash n$. A *tableau* t of shape λ is a Young diagram of shape λ filled with a set of elements, often positive integers. An entry of a cell having a coordinate (i, j) in tableau t is denoted by $t_{i,j}$.

A *Young tableau* T of shape λ is a tableau of shape λ whose entries are the numbers from 1 to n, each occurring once.

A *standard Young tableau* is a Young tableau whose entries are strictly increasing in rows and columns. Let $\lambda \vdash n$.

A *partial tableau* p of shape λ is a tableau whose entries are strictly increasing in rows and columns. Note that a partial tableau is the standard Young tableau if the entries of p are exactly $\{1, 2, \ldots, n\}$. For instance, the following t_1 is a standard tableau, but t_2 is not.

$$t_1 = \begin{array}{|c|c|c|}\hline 1 & 3 & 5 \\ \hline 2 & 4 \\ \cline{1-2} 6 \\ \cline{1-1} \end{array}, \quad t_2 = \begin{array}{|c|c|c|}\hline 1 & 3 & 5 \\ \hline 4 & 2 \\ \cline{1-2} 6 \\ \cline{1-1} \end{array}.$$

The number of standard Young tableaux of a given shape λ is obtained from the *hook formula*.

If $\nu = (i, j)$ is a cell in the Young diagram of shape λ, then the *hook* of ν, denoted by H_ν, is the set of all cells directly to the right of ν or directly below ν including ν itself, that is

$$H_\nu = H_{i,j} = \{(i,\, j') : j' \geq j\} \cup \{(i',\, j) : i' \geq i\}.$$

The *hook length* of $\nu = (i, j)$, denoted by $h_{i,j}$, is the number of cells in its hook, i.e., $h_{i,j} = |H_{i,j}|$.

The number of standard Young tableaux of a given shape $\lambda \vdash n$ is obtained by the following theorem.

Theorem 2.1 ([12]). *If $\lambda \vdash n$, then the number f^λ of standard Young tableaux of shape λ is*

$$f^\lambda = \frac{n!}{\prod_{(i,j)\in\lambda} h_{i,j}}.$$

For instance, labeling each cell with its hook length for the Young diagram of shape $(3,2,1)$ and $(4,4,4,4)$ are given by

$$\begin{array}{|c|c|c|}\hline 5&3&1\\\hline 3&1\\\cline{1-2} 1\\\cline{1-1}\end{array} \quad , \quad \begin{array}{|c|c|c|c|}\hline 7&6&5&4\\\hline 6&5&4&3\\\hline 5&4&3&2\\\hline 4&3&2&1\\\hline\end{array} \quad .$$

If the shape is $\lambda = (3,2,1)$, then $f^{(3,2,1)} = 6!/(5 \cdot 3^2 \cdot 1^3) = 16$. If the shape is $\lambda = (4,4,4,4)$, then $f^{(4,4,4,4)} = 16!/(7 \cdot 6^2 \cdot 5^3 \cdot 4^4 \cdot 3^3 \cdot 2^2 \cdot 1) = 24024$.

Let $\mu = (\mu_1, \mu_2, \ldots, \mu_i)$ and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_j)$ be partitions with $\mu \subseteq \lambda$ (i.e., $i \le j$ and $\mu_k \le \lambda_k$ for $1 \le k \le i$). Then, a *skew shape* of λ/μ is the set of cells $\lambda/\mu = \{c : c \in \lambda \text{ and } c \notin \mu\}$. A skew shape of λ/μ is *normal* if $\mu = \emptyset$. A tableau of skew shape λ/μ is called a *skew tableau* of shape λ/μ. A *partial skew tableau* of shape λ/μ (or a partial tableau of skew shape λ/μ) is a skew tableau of shape λ/μ whose entries are strictly increasing in rows and columns.

A partial skew tableau is called the *standard skew tableau* if its entries are precisely $\{1, 2, \ldots, n\}$. For instance, consider skew tableaux of shape $\lambda/\mu = (4,3,3,2)/(2,2)$ given by

$$t_1 = \begin{array}{cc|c|c}\multicolumn{2}{c}{}&1&7\\\cline{3-4}\multicolumn{2}{c|}{}&3\\\hline 2&4&5\\\hline 6&9\\\cline{1-2}\end{array} \quad , \quad t_2 = \begin{array}{cc|c|c}\multicolumn{2}{c}{}&1&7\\\cline{3-4}\multicolumn{2}{c|}{}&5\\\hline 2&4&3\\\hline 6&9\\\cline{1-2}\end{array} \quad .$$

We see that t_1 is a partial skew tableau, but t_2 is not.

A group $(G, \cdot)$ is a nonempty set G, closed under a binary operation $\cdot$, such that the following axioms are satisfied: (i) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in G$, (ii) there is an identity element $e \in G$ such that for all $x \in G$, $e \cdot x = x \cdot e = x$, (iii) for each element $a \in G$, there is an element $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

The group of all bijections $I_n \to I_n$, whose binary operation is function composition, is called the *symmetric group on n letters* and denoted $\mathfrak{S}_n$. Since $\mathfrak{S}_n$ is the group of all permutations of a set $I_n = \{1, 2, \ldots, n\}$, the order of $\mathfrak{S}_n$, i.e., $|\mathfrak{S}_n|$, is $n!$.

Let $i_1, i_2, \ldots, i_n$ be distinct elements of $I_n = \{1, 2, \ldots, n\}$. Then, $[i_1\, i_2 \cdots i_n] \in \mathfrak{S}_n$ denotes the permutation that maps $1 \mapsto i_1, 2 \mapsto i_2, \ldots, n \mapsto i_n$.

Suppose $x < y < z$. A *Knuth transformation* of a permutation $\pi \in \mathfrak{S}_n$ is a transformation of $\pi \in \mathfrak{S}_n$ into another permutation $\tau \in \mathfrak{S}_n$ that has one of the following forms:

(1) $\pi = [x_1 \cdots y\, x\, z \cdots x_n] \in \mathfrak{S}_n \Longrightarrow \tau = [x_1 \cdots y\, z\, x \cdots x_n] \in \mathfrak{S}_n$,
(2) $\pi = [x_1 \cdots y\, z\, x \cdots x_n] \in \mathfrak{S}_n \Longrightarrow \tau = [x_1 \cdots y\, x\, z \cdots x_n] \in \mathfrak{S}_n$,
(3) $\pi = [x_1 \cdots x\, z\, y \cdots x_n] \in \mathfrak{S}_n \Longrightarrow \tau = [x_1 \cdots z\, x\, y \cdots x_n] \in \mathfrak{S}_n$,
(4) $\pi = [x_1 \cdots z\, x\, y \cdots x_n] \in \mathfrak{S}_n \Longrightarrow \tau = [x_1 \cdots x\, z\, y \cdots x_n] \in \mathfrak{S}_n$.

Two permutations $\pi, \tau \in \mathfrak{S}_n$ are called *Knuth-equivalent* if one of them can be obtained from the other by a sequence of Knuth transformations, denoted $\pi \cong_K \tau$.

For instance, we see that $[2\,1\,3] \in \mathfrak{S}_3$ and $[2\,3\,1] \in \mathfrak{S}_3$ are Knuth-equivalent by the above (1) and (2), written $[2\,1\,3] \cong_K [2\,3\,1]$. Similarly, $[1\,3\,2] \in \mathfrak{S}_3$ and $[3\,1\,2] \in \mathfrak{S}_3$ are Knuth-equivalent by the above (3) and (4), written $[1\,3\,2] \cong_K [3\,1\,2]$.

Let t be a tableau. The *reading word* or *row word* of t, denoted $r(t)$, is the permutation of entries of t obtained by concatenating the rows of t from bottom to top, i.e., $r(t) = R_k R_{k-1} \ldots R_1$, where $R_1, \ldots, R_k$ are the rows of t.

Algorithm 1: A forward jeu de taquin slide [22, 23]

Input: A partial tableau P of skew shape λ/μ; an inner corner of μ
Output: A partial tableau P'
begin
  Pick x to be an inner corner of μ;
  while *x is not an inner corner of λ* **do**
    if $x = (i, j)$ **then**
      Let x' be the cell of $\min\{P_{i+1,j}\,,\, P_{i,j+1}\}$;
      (If only one of $P_{i+1,j}$ and $P_{i,j+1}$ exists, then choose that value as a minimum.)
    end
    Slide $P_{x'}$ into cell x and set $x := x'$;
  end
  return *The resulting partial tableau P'*;
end

The *jeu de taquin* of Scützenberger [23,29] consists of a set of rules for transforming *partial tableaux*, while some properties of partial tableaux are preserved during transformations. A forward jeu de taquin slide is described in Algorithm 1. Note that the resulting tableau of a forward jeu de taquin slide is still a partial tableau. We say that partial tableaux P and P' are *jeu de taquin equivalent*, written $P \cong_{jdt} P'$, if P' can be obtained from P by some sequence of jeu de taquin slides, or vice versa. (The reader is encouraged to verify that $\cong_{jdt}$ is an equivalence relation on the set of partial tableaux.)

Lemma 2.1 ([29]). *Each jeu de taquin slide converts the reading word of a standard skew tableau into a Knuth-equivalent one.*

Theorem 2.2 ([22, 23]). *The jeu de taquin equivalence class of a given partial skew tableau P contains exactly one partial tableau of normal shape.*

Theorem 2.3 ([23, 29])**.** *Let P and P' be standard skew tableaux. They are jeu de taquin equivalent, i.e., $P \cong_{jdt} P'$, if and only if their reading words are Knuth-equivalent, i.e., $r(P) \cong_K r(P')$.*

3 Representations of a 2D-HMS

A graph-theoretic approach to Young diagrams or tableaux has already been researched in [19]. It focuses on graphs having the shape of a Young diagram or a tableau, while this paper focuses on converting a 2D-MS into a Young diagram. This section presents how a 2D-HMS is represented by a Young diagram with additional properties. In this section we define a hierarchical 2D mesh tableau in order to represent a task assignment and its reassignments in a 2D-HMS.

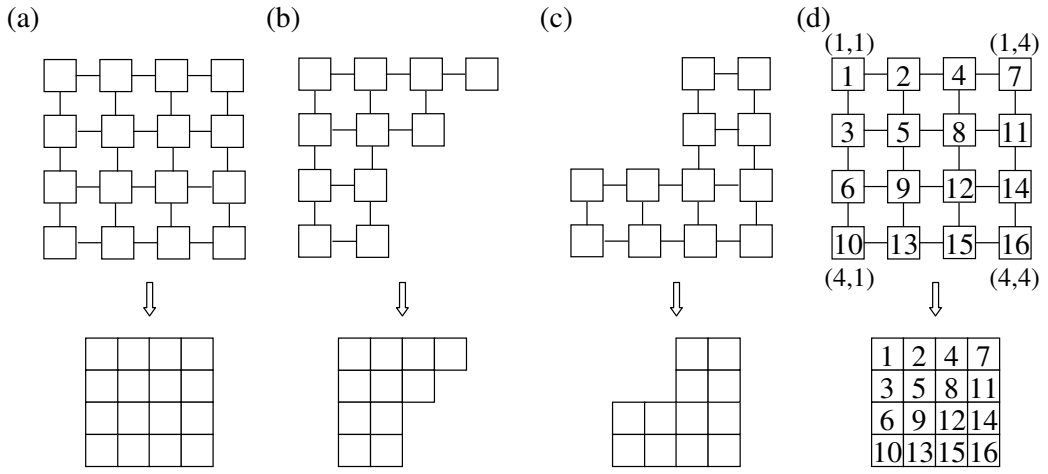

Figure 1: The conversion of each 2D-MS (or its variant) into a Young diagram or a tableau.

Figure 1(a) shows our approach to convert a 2D-MS into a Young diagram in a compact manner. Similarly to Figure 1(a), Figure 1(b) and (c) convert the variants of a 2D-MS into their corresponding Young diagrams. Meanwhile, each label (except labels involving coordinates $(1,1)$, $(1,4)$, $(4,1)$, and $(4,4)$) in Figure 1(d) denotes a task ID in order to represent a task assignment in a 2D-MS.

To exploit the regular nature of a 2D mesh topology, we consider a hierarchical 2D mesh diagram of canonical shape, where the rows and columns of heterogeneous nodes are sorted in descending order by their priorities (or execution rates). We first define a 2D-HMS consisting of $m \times n$ heterogeneous nodes. Then, we define a hierarchical 2D mesh diagram to represent a 2D-HMS.

Definition 3.1. A *hierarchical 2D mesh-connected system* (2D-HMS) of $m \times n$ heterogeneous nodes is a heterogeneous 2D-MS with the following partial order

$$N(i-1,j) \prec_p N(i,j), \;\; N(i,j-1) \prec_p N(i,j), \quad 1 < i \leq m, \; 1 < j \leq n,$$

where $N(a,b) \prec_p N(c,d)$ means that a node addressed by (a,b) has a higher priority (or execution rate) than a node addressed by (c,d).

Definition 3.2. A *hierarchical 2D mesh diagram of canonical shape* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k) \vdash n$ for $\lambda_1 = \lambda_2 = \cdots = \lambda_k$ is a Young diagram of shape $\lambda \vdash n$, where each cell (i,j) represents each node (i,j) in a 2D-HMS. Therefore, it has the following partial order

$$N(i-1,j) \prec_p N(i,j), \;\; N(i,j-1) \prec_p N(i,j), \quad 1 < i \leq k, \; 1 < j \leq \lambda_1,$$

where $N(a,b) \prec_p N(c,d)$ means that a node represented by cell (a,b) has a higher priority (or execution rate) than a node represented by cell (c,d).

We say "a node represented by cell (a,b)" in Definition 3.2 and "a node addressed by (a,b)" interchangeably for a hierarchical 2D mesh diagram. The following proposition involves in a counting aspect of organizing a hierarchical 2D mesh diagram of canonical shape $\lambda \vdash n$ using n heterogeneous nodes in a distributed system.

Proposition 3.1. *Let* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k) \vdash n$ *for* $\lambda_1 = \lambda_2 = \cdots = \lambda_k$ *and let* $N = \{1, 2, \ldots, n\}$ *be a set of* $k \times \lambda_1$ *heterogeneous nodes having priorities*[1] *represented by node IDs from 1 to* n*. A total order relation* $<$ *is defined naturally on* N *such that for any two nodes* $u \in N$ *and* $v \in N$, $u < v$ *implies that node* u *has a higher priority (or execution rate) than node* v*. Then, the number of ways to organize a hierarchical 2D mesh diagram of shape* $\lambda \vdash n$ *by arranging* n *nodes in* N *is* f^λ *(i.e., the number of standard Young tableaux of shape* λ*).*

Proof. It immediately follows from the definition of a standard tableau and Definition 3.2. □

We next define a *hierarchical 2D mesh tableau* whose main usage is to represent a priority-based task assignment and its reassignments in a 2D-HMS. A priority-based task assignment and its reassignments in a 2D-HMS are discussed in the next section.

Definition 3.3. A *hierarchical 2D mesh tableau of shape* $\lambda \vdash n$ is a tableau of shape λ whose underlying Young diagram is a hierarchical 2D mesh diagram of canonical shape $\lambda \vdash n$. It is denoted by $(\mathrm{HMT}_{i,j})$ of shape λ or λ-$(\mathrm{HMT}_{i,j})$ for short. Entries of λ-$(\mathrm{HMT}_{i,j})$ denote task IDs from the set $\{1, 2, \ldots, m\}$ for $m \leq n$. The empty entry of a cell is allowed in λ-$(\mathrm{HMT}_{i,j})$, while all non-empty entries along with their cells must form a tableau of normal or skew shape, called a *maximally embedded tableau* of λ-$(\mathrm{HMT}_{i,j})$.

Definition 3.4. Let λ-$(\mathrm{HMT}_{i,j})$ be a hierarchical 2D mesh tableau of shape $\lambda \vdash n$. If the maximally embedded tableau of λ-$(\mathrm{HMT}_{i,j})$ is a tableau of normal shape, we say that λ-$(\mathrm{HMT}_{i,j})$ is of *normal shape*. Meanwhile, if the maximally embedded tableau of λ-$(\mathrm{HMT}_{i,j})$ is a tableau of skew shape, we say that λ-$(\mathrm{HMT}_{i,j})$ is of *skew shape*. If the maximally embedded tableau of λ-$(\mathrm{HMT}_{i,j})$ is a partial tableau (i.e., a tableau whose entries are strictly increasing in rows and columns), we say that λ-$(\mathrm{HMT}_{i,j})$ is *standard*. Otherwise, we say that λ-$(\mathrm{HMT}_{i,j})$ is *generalized*.

[1] In some priority schemes [18, 28] a higher number indicates a higher priority. Throughout this paper it is assumed that a lower number indicates a higher priority.

4 Priority-based task reassignments in a 2D-HMS

Let $T_m = \{1, 2, \ldots, m\}$ be a set of m tasks having priorities represented by task IDs from 1 to m, where a lower task ID indicates a higher priority. A total order relation $\prec_t$ is defined on T_m as follows. $t_1 \prec_t t_2$ for any two tasks $t_1 \in T_m$ and $t_2 \in T_m$ means that t_1 has a higher priority than t_2. Let R_n be a hierarchical 2D-MS (2D-HMS) consisting of $n\,(n \geq m)$ heterogeneous nodes whose priorities (or execution rates) of rows and columns are sorted in descending order. Let $A : T_m \to R_n$ be an injective task assignment function between T_m and R_n whose constraints are defined as follows:

1. Priorities of the assigned tasks on nodes strictly decrease in rows and columns, i.e., $t_1 \prec_t t_2$ whenever $A(t_1) \prec_p A(t_2)$ for any two tasks $t_1 \in T_m$ and $t_2 \in T_m$.
2. Nodes of $A(T_m)$ in R_n is left-justified, where the row sizes of $A(T_m)$ are weakly decreasing.

The first constraint ensures that a node with a higher priority executes a task with a higher priority. The second constraint ensures that if one node is idle and the other node is busy for two adjacent nodes in a 2D-HMS, the node with the lower priority is chosen to be idle. We say that a task assignment or reassignment A in a 2D-HMS is *priority-based* if it satisfies the above constraints in a 2D-HMS.

The *priority-based task reassignment problem* in a 2D-HMS is defined as follows: We are given an initial task assignment $A = A_0$ along with a task completion sequence $(b_i)_{i=1}^{m}$ provided in run-time, where $b_i \in T_m$ for $1 \leq i \leq m$. Find a priority-based task (re)assignment sequence $(A_k)_{k=0}^{m-1}$.

The constraints and assumptions that we have made are:

1. Both tasks and nodes are heterogeneous.
2. Each node can process at most one task at a time.
3. Each priority-based task reassignment is achieved by an iterative sequence of task relocations, where each task relocation is allowed between two adjacent nodes in a 2D-HMS if one node is in idle state and the other node is in busy state.
4. Two task relocations do not occur simultaneously in a 2D-HMS.
5. The number of tasks are less than or equal to the number of the available nodes in a 2D-HMS.

Recall that the underlying Young diagram of λ-$(\mathrm{HMT}_{i,j})$ is a hierarchical 2D mesh diagram of canonical shape (see Definition 3.3). Therefore, each cell of λ-$(\mathrm{HMT}_{i,j})$ represents each node in a 2D-HMS. By assigning each entry (i.e., task) in a standard λ-$(\mathrm{HMT}_{i,j})$ of normal shape to its underlying cell representing a node in a 2D-HMS, we see that a standard λ-$(\mathrm{HMT}_{i,j})$ of normal shape represents a priority-based task assignment in a 2D-HMS. We define a *descent pair* of a generalized λ-$(\mathrm{HMT}_{i,j})$, which does not allow a generalized λ-$(\mathrm{HMT}_{i,j})$ to represent a priority-based task assignment in a 2D-HMS.

Definition 4.1. Let λ-($\mathrm{HMT}_{i,j}$) be a generalized hierarchical 2D mesh tableau. If $\mathrm{HMT}_{i,j} \prec_t \mathrm{HMT}_{i-1,j}$ (respectively, $\mathrm{HMT}_{i,j} \prec_t \mathrm{HMT}_{i,j-1}$), then, $\{(i-1,j),(i,j)\}$ (respectively, $\{(i,j-1),(i,j)\}$) is called a *descent pair* of a generalized λ-($\mathrm{HMT}_{i,j}$).

If a descent pair occurs in a task assignment represented by a generalized λ-($\mathrm{HMT}_{i,j}$), it is not a priority-based task assignment. In this paper λ-($\mathrm{HMT}_{i,j}$) is referred to as a standard λ-($\mathrm{HMT}_{i,j}$) of normal shape unless otherwise stated.

Now, consider a priority-based task assignment in Figure 2(a) represented by ($\mathrm{HMT}_{i,j}$) of shape $\lambda = (3,3,3) \vdash n$ for $n = 9$.

(a) $A_0 =$ $\begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&5&7\\\hline 6&8&9\\\hline\end{array}$, (b) $\begin{array}{|c|c|c|}\hline \bullet&2&4\\\hline 3&5&7\\\hline 6&8&9\\\hline\end{array}$ $\Rightarrow$ $\begin{array}{|c|c|c|}\hline 2&\bullet&4\\\hline 3&5&7\\\hline 6&8&9\\\hline\end{array}$ $\Rightarrow$ $\begin{array}{|c|c|c|}\hline 2&4&\bullet\\\hline 3&5&7\\\hline 6&8&9\\\hline\end{array}$ $\Rightarrow$ $\begin{array}{|c|c|c|}\hline 2&4&7\\\hline 3&5&\bullet\\\hline 6&8&9\\\hline\end{array}$ $\Rightarrow$ $\begin{array}{|c|c|c|}\hline 2&4&7\\\hline 3&5&9\\\hline 6&8&\bullet\\\hline\end{array}$.

(c) $A_0 =$ $\begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&5&7\\\hline 6&8&9\\\hline\end{array}$, $A_1 =$ $\begin{array}{|c|c|c|}\hline 2&4&7\\\hline 3&5&9\\\hline 6&8&\\\hline\end{array}$, $A_2 =$ $\begin{array}{|c|c|c|}\hline 2&4&7\\\hline 5&8&9\\\hline 6&&\\\hline\end{array}$, $A_3 =$ $\begin{array}{|c|c|c|}\hline 4&7&9\\\hline 5&8&\\\hline 6&&\\\hline\end{array}$, $A_4 =$ $\begin{array}{|c|c|c|}\hline 4&7&9\\\hline 6&8&\\\hline &&\\\hline\end{array}$,

$A_5 =$ $\begin{array}{|c|c|c|}\hline 4&7&9\\\hline 6&&\\\hline &&\\\hline\end{array}$, $A_6 =$ $\begin{array}{|c|c|c|}\hline 6&7&9\\\hline &&\\\hline &&\\\hline\end{array}$, $A_7 =$ $\begin{array}{|c|c|c|}\hline 7&9&\\\hline &&\\\hline &&\\\hline\end{array}$, $A_8 =$ $\begin{array}{|c|c|c|}\hline 9&&\\\hline &&\\\hline &&\\\hline\end{array}$.

Figure 2: A sequence of task reassignments for a task completion sequence $(1,3,2,5,8,4,6,7,9)$.

If task 1 in Figure 2(a) is completed first, the node addressed by $(1,1)$ becomes idle. We mark the cell $(1,1)$ as $\bullet$ to show that the node addressed by $(1,1)$ is now in the idle state. Once a node is in the idle state, it seeks the right and below node to perform task relocation. Recall that our task relocation is only allowed between two adjacent nodes if one is in the idle state and the other is in the busy state. If a node is in the idle state, it does not check the left and above node to perform task relocation. It is because the underlying 2D mesh Young diagram of λ-($\mathrm{HMT}_{i,j}$) is hierarchical, it is not an optimal choice if a task is to run on a node with the lower execution rate. Therefore, a node in the idle state always seeks both the right and below node in order to compare task priorities and to relocate a task. We see that task $\mathrm{HMT}_{1,2}$ has a higher priority than task $\mathrm{HMT}_{2,1}$, i.e., $2 \prec_t 3$. Therefore, task relocation involves the node addressed by $(1,1)$ and the node addressed by $(1,2)$. Now, task 2 has been relocated and the node addressed by $(1,2)$ becomes idle. If a node becomes idle, the choice for task relocation between the right and below node is always *greedy* [9], allowing the task with the higher priority to occupy the idle node (see Figure 2(b)). This process continues until no task relocation is possible, which means that the right and below node of $\bullet$ are both idle or both not available. The final state of Figure 2(b) is the task reassignment A_1 for the completion of task 1. Given an initial task assignment A_0 in Figure 2(a), Figure 2(c) shows the task (re)assignment sequence $(A_k)_{k=0}^{m-1}$ for $m = 9$ corresponding to the task completion sequence (1, 3, 2, 5, 8, 4, 6, 7, 9). Algorithm 2 describes the

Algorithm 2: Task reassignments: λ-($\mathrm{HMT}_{i,j}$) of normal shape

Input: An initial task assignment A_0 represented by λ-($\mathrm{HMT}_{i,j}$) of normal shape; a task completion sequence $(b_1, b_2, \ldots, b_m)$, where $m \geq 2$, provided in runtime

Output: A task (re)assignment sequence $(A_k)_{k=0}^{m-1}$

begin

Set λ-($\mathrm{HMT}_{i,j}^{(1)}$) :=λ-($\mathrm{HMT}_{i,j}$);

for $k \leftarrow 1$ **to** $m-1$ **do**

Let t be the maximally embedded tableau of λ-($\mathrm{HMT}_{i,j}^{(k)}$) and μ be the shape of t; Wait for a completion of task b_k in λ-($\mathrm{HMT}_{i,j}^{(k)}$); If task b_k is completed, then the underlying cell of task b_k in λ-($\mathrm{HMT}_{i,j}^{(k)}$) becomes vacated; Set x as the corresponding cell in λ-($\mathrm{HMT}_{i,j}^{(k)}$);

while x *is not an inner corner of* μ **do**

if $x = (i,\ j)$ **then**

Let x' be the cell of $\min\{\mathrm{HMT}_{i+1,j}^{(k)},\ \mathrm{HMT}_{i,j+1}^{(k)}\}$. (If only one non-idle cell exists in $(i+1, j)$ and $(i, j+1)$, then choose that cell.)

end

Relocate the task on cell x' into cell x and set $x := x'$;

end

Set A_k:=λ-($\mathrm{HMT}_{i,j}^{(k)}$), where λ-($\mathrm{HMT}_{i,j}^{(k)}$) is of normal shape;

Set λ-($\mathrm{HMT}_{i,j}^{(k+1)}$):=$\lambda$-($\mathrm{HMT}_{i,j}^{(k)}$);

end

return *The task (re)assignment sequence* $(A_k)_{k=0}^{m-1}$;

end

procedure in Figure 2. Task reassignments take place in Algorithm 2 when a task completion sequence is provided in run time. It turns out that the iterative greedy task relocation mechanism in Algorithm 2 corresponds to a forward jeu de taquin slide discussed in Algorithm 1, except that task relocation starts with the cell that is indicated by the task completion sequence. Note that each task assignment in a task (re)assignment sequence $(A_k)_{k=0}^{m-1}$ in Algorithm 2 is a priority-based task assignment. We see that each task (re)assignment in $(A_k)_{k=0}^{m-1}$ in Algorithm 2 is represented by a hierarchical 2D mesh tableau of normal shape, where task IDs are increasing in rows and columns on the underlying hierarchical 2D mesh diagram.

Thus far, we have examined the case where an initial task assignment is represented by λ-($\mathrm{HMT}_{i,j}$) of normal shape. We now consider the case where an initial task assignment is represented by λ-($\mathrm{HMT}_{i,j}$) of skew shape.

Consider a top-left corner of a hierarchical 2D mesh tableau t_1 or t_2 in Figure 3, where the node with the highest priority is idle. Therefore, it is a natural choice to

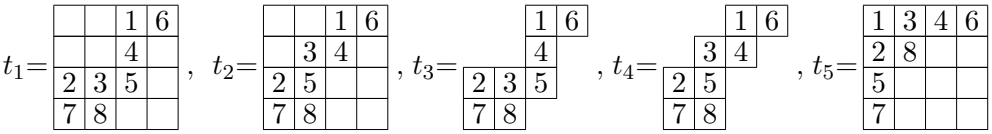

Figure 3: Task reassignments by using forward jeu de taquin slides.

Algorithm 3: Task reassignments: λ-(HMT$_{i,j}$) of skew shape

Input: An initial task assignment A_0 represented by λ-(HMT$_{i,j}$) of skew shape
Output: A task (re)assignment sequence $(A_k)_{k=0}^{m}$
begin
  If β is a partition of a positive integer n for the maximally embedded tableau of λ-(HMT$_{i,j}$) of skew shape α/β, then set m as the value of n; Set λ-(HMT$_{i,j}^{(1)}$) :=λ-(HMT$_{i,j}$); Set $k := 1$;
  while *the maximally embedded tableau of* λ-($HMT_{i,j}^{(k)}$) *is not of normal shape* **do**
    If the maximally embedded tableau of λ-(HMT$_{i,j}^{(k)}$) is of (skew) shape μ/ν, pick x to be an inner corner of ν;
    while x *is not an inner corner of* μ **do**
      if $x = (i,\, j)$ **then**
        Let x' be the cell of min$\{$HMT$_{i+1,j}^{(k)}$, HMT$_{i,j+1}^{(k)}\}$. (If only one non-idle cell exists in $(i+1, j)$ and $(i, j+1)$, then choose that cell.)
      end
      Relocate the task on cell x' into cell x and set $x := x'$;
    end
    Set A_k:=λ-(HMT$_{i,j}^{(k)}$); Set λ-(HMT$_{i,j}^{(k+1)}$):=λ-(HMT$_{i,j}^{(k)}$); $k := k + 1$;
  end
  return *The task (re)assignment sequence* $(A_k)_{k=0}^{m}$, where A_m is the task reassignment represented by λ-(HMT$_{i,j}^{(m)}$) of normal shape;
end

relocate a task from a node with the lower execution rate to a node with the higher execution rate if task relocation is necessary. Algorithm 3 describes the procedure, where an initial task assignment represented by a hierarchical 2D mesh tableau of skew shape is converted into the task reassignment represented by a hierarchical 2D mesh tableau of normal shape. As shown in Figure 3, t_3 is the maximally embedded tableau of t_1, and t_4 is the maximally embedded tableau of t_2, respectively. By performing task relocations discussed in Algorithm 3 iteratively, the task reassignment t_5 is obtained from the task assignment t_1 in Figure 3 by Algorithm 3. Similarly, the task reassignment t_5 is also obtained from the task assignment t_2 in Figure 3. If t_2 represents an initial task assignment A_0, then t_5 corresponds to the task reassignment A_3 in Algorithm 3.

An initial choice of task relocation for A_1 must target for either the cell $(1,2)$ or the cell $(2,1)$ in t_2. Note that the task (re)assignment sequence (A_0, A_1, A_2, A_3) is not uniquely determined, depending on the choice of an initial task relocation. However, A_3 is uniquely determined by Theorem 2.2 because the greedy-based task relocations on λ-$(\mathrm{HMT}_{i,j})$ follow the forward jeu de taquin slide rules. Unlike a task reassignment represented by λ-$(\mathrm{HMT}_{i,j})$ of normal shape, a task reassignment represented by λ-$(\mathrm{HMT}_{i,j})$ of skew shape do not always satisfy the constraints of a priority-based task assignment. For instance, non-idle nodes of A_0, A_1, and A_2 are not left-justified, which implies that the node with the higher execution rate is idle for some pairs of adjacent nodes in a 2D-HMS. However, the resulting task reassignment A_3 is a priority-based task assignment in a 2D-HMS. We next define an equivalence class of task reassignments up to task relocations under the greedy task relocation policy.

Definition 4.2. Two task assignments t_1 and t_2, represented by a hierarchical 2D mesh tableau λ-$(\mathrm{HMT}^1_{i,j})$ of skew shape and λ-$(\mathrm{HMT}^2_{i,j})$ of skew shape, respectively, are called *task reassignment equivalent* up to task relocations (under the greedy task relocation policy described in Algorithm 3), denoted by $t_1 \cong_t t_2$, if they have the same resulting task reassignment represented by the same λ-$(\mathrm{HMT}_{i,j})$ of normal shape.

Proposition 4.1. *Let λ-$(HMT^1_{i,j})$ and λ-$(HMT^2_{i,j})$ be hierarchical 2D mesh tableaux of skew shape whose maximally embedded tableaux are both standard skew tableaux. If two task assignments t_1 and t_2, represented by λ-$(HMT^1_{i,j})$ and λ-$(HMT^2_{i,j})$, respectively, are task reassignment equivalent, then the reading words of their maximally embedded tableaux of λ-$(HMT^1_{i,j})$ and λ-$(HMT^2_{i,j})$ are Knuth-equivalent.*

Proof. Let P be the maximally embedded tableau of λ-$(\mathrm{HMT}^1_{i,j})$ for the task assignment t_1, and Q be the maximally embedded tableau of λ-$(\mathrm{HMT}^2_{i,j})$ for the task assignment t_2. Since $t_1 \cong_t t_2$ by hypothesis, t_1 and t_2 have the same resulting task reassignment represented by the same λ-$(\mathrm{HMT}_{i,j})$ of normal shape by Definition 4.2. Each task relocation under the greedy task relocation policy described in Algorithm 3 follows the forward jeu de taquin slide rules described in Algorithm 1. Thus, $P \cong_{jdt} Q$ by Theorem 2.2. Since P and Q are both standard skew tableaux by hypothesis satisfying $P \cong_{jdt} Q$, we conclude that the reading words of P and Q are Knuth-equivalent by Theorem 2.3. □

For instance, the reading word of t_3 (i.e., the maximally embedded tableau of t_1) in Figure 3 is $\pi = [7\,8\,2\,3\,5\,4\,1\,6] \in \mathfrak{S}_8$. Similarly, the reading word of t_4 (i.e., the maximally embedded tableau of t_2) in Figure 3 is $\tau = [7\,8\,2\,5\,3\,4\,1\,6] \in \mathfrak{S}_8$. We leave it to the reader to verify that π and τ are Knuth-equivalent.

We say that an idle node in a 2D-HMS represented by λ-$(\mathrm{HMT}_{i,j})$ is *locally fragmented* if it is surrounded by busy nodes, where a busy node in a 2D-HMS corresponds to a cell having an non-empty entry in λ-$(\mathrm{HMT}_{i,j})$. If we do not apply a task reassignment at all, a locally fragmented node can be generated in a 2D-HMS when an internal node completes its task and becomes idle while other nodes are busy. It is easy to see that locally fragmented nodes increase the chance of occurring external fragmentation.

For instance, four locally fragmented nodes can serve only four 1×1 meshes of tasks, but cannot serve one 2×2 mesh of tasks or two 1×2 meshes of tasks. As discussed in [36], task relocation is an approach for alleviating the fragmentation problem in a 2D-MS. In our approach we use task relocations to avoid the generation of locally fragmented nodes while taking the priorities of tasks and nodes into account. The following proposition says that our priority-based task reassignment procedure does not generate any locally fragmented node in a 2D-HMS.

Proposition 4.2. *Let A_0 be an initial task assignment in a 2D-HMS represented by λ-($HMT_{i,j}$) of normal shape and let $(b_1, b_2, \ldots, b_m)$ for $m \geq 2$ be a task completion sequence in Algorithm 2. Each task reassignment A_k ($k = 1, \ldots, m-1$) in Algorithm 2 does not generate any locally fragmented node in the 2D-HMS.*

Proof. For each $k = 0, \ldots, m-1$, let t_k be the maximally embedded tableau of λ-($\mathrm{HMT}^k_{i,j}$) representing task (re)assignment A_k in a 2D-HMS and let i_k be the inner corner in t_k that becomes vacated by A_{k+1} by Algorithm 2. We see that t_k is a tableau of normal shape by Algorithm 2. By the definition of an inner corner, each inner corner in t_k is adjacent to a cell with the empty entry (i.e., an idle node in the 2D-HMS) in λ-($\mathrm{HMT}^k_{i,j}$). Since the node addressed by i_k does not become locally fragmented by A_{k+1}, we see that each task reassignment A_k ($k = 1, \ldots, m-1$) in Algorithm 2 does not generate any locally fragmented node in the 2D-HMS. □

As a simple example of Algorithm 2, consider a sequential task assignment in a 2D-HMS for priority-based task reassignments, which is described as follows. A set of m tasks $T_m = \{1, 2, \ldots, m\}$ with the precedence relationship $1 \to 2 \to \cdots \to m$ are to be assigned to a set of m heterogeneous nodes bijectively in a 2D-HMS of n heterogeneous nodes ($m \leq n$) and executed sequentially without gaps. We assume that a 2D-HMS is consistent for a sequential task assignment. Tasks are heterogeneous, and their priorities are assigned by their computation requirements in which the higher task priority indicates the larger computation requirement. If a descent pair occurs in a sequential task assignment represented by a generalized λ-($\mathrm{HMT}_{i,j}$), there is a node with a higher priority than the other node but it executes a task with a lower priority than the other node. Therefore, it always has the better task assignment in terms of task turnaround time. For instance, swapping tasks on a descent pair reduces task turnaround time for a sequential task assignment in a consistent 2D-HMS. Now, consider priority-based task reassignments in Algorithm 2 for a priority-based sequential task assignment of m tasks in a 2D-HMS, in which the task completion sequence is simply $(1, 2, \ldots, m)$. If we assume that every 2D-HMS is consistent in which task relocations are cost-free, we have Proposition 4.3.

Proposition 4.3. *Let A_0 be a priority-based sequential task assignment for m ($m \geq 2$) tasks represented by λ-($HMT_{i,j}$) of normal shape. Let T_1 be the task turnaround time for A_0 without task relocation, and T_2 be the task turnaround time with the task (re)assignment sequence $(A_k)_{k=0}^{m-1}$ (see Algorithm 2). Then, T_2 is less than T_1, i.e., $T_2 < T_1$.*

Proof. It suffices to show that each task relocation allows each task to reduce its task execution time for a sequential task assignment in a 2D-HMS. Let $t_{i,x}$ be the task execution time of task i on cell x in λ-(HMT$_{i,j}$) before task relocation. After task relocation, task i is relocated from cell x to its adjacent cell x' such that $N(x') \prec_p N(x)$, where $N(x') \prec_p N(x)$ means that an execution rate of node addressed by x' is higher than that of node addressed by x. Thus, $t_{i,x'} < t_{i,x}$. Since each task reassignment consists of iterative task relocations by Algorithm 2, we conclude that $T_2 < T_1$. □

However, when we consider a non-trivial task relocation cost in a consistent 2D-HMS, the sufficient condition for task i in a priority-based sequential task assignment to reduce its task turnaround time by means of task relocation from node n to its adjacent node n' in Algorithm 2 is that the task relocation cost of task i from node n to n' is less than the difference of task execution times caused by task relocation of task i from node n to n'. Note that a greedy task relocation policy in Algorithm 2 keeps λ-(HMT$_{i,j}$) from occurring any descent pair for each task reassignment.

Given a priority-based sequential task assignment in a 2D-HMS without any descent pair, we have discussed priority-based task reassignments in a 2D-HMS using Algorithm 2. We leave it as an open question to apply the priority-based task reassignment procedure in a 2D-HMS to other application areas, such as sorting on a mesh-connected computing environment [32].

5 Conclusions

In this paper we have presented a novel approach to representing priority-based task reassignments in a heterogeneous 2D mesh-connected system. To the best of our knowledge, this paper is the first attempt to study task assignments and reassignments in a 2D mesh-connected system using tableaux. We have proposed a hierarchical 2D mesh-connected system (2D-HMS) that is a heterogeneous 2D-MS in a distributed system with additional priority constraints on nodes. A priority-based task reassignment in a 2D-HMS is represented by a hierarchical 2D mesh tableau λ-(HMT$_{i,j}$) in which task relocations under the greedy task relocation policy are reduced to a jeu de taquin slide on λ-(HMT$_{i,j}$). Given an initial priority-based task assignment in a 2D-HMS represented by λ-(HMT$_{i,j}$), we have shown that our task reassignment procedure does not generate any locally fragmented node in the 2D-HMS while taking priorities of tasks and nodes into account.

References

[1] I. Ababneh. An efficient free-list submesh allocation scheme for two-dimensional mesh-connected multicomputers. *The Journal of Systems and Software*, 79:1168–1179, 2006.

[2] N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, Cambridge, UK, 1974.

[3] S.H. Bokhari. Dual Processor Scheduling with Dynamic Reassignment. *IEEE Transactions on Software Engineering*, 5:341–349, 1979.

[4] B. Bollobás. *Modern Graph Theory*. Springer, New York, NY, 1998.

[5] T.L. Casavant and J.G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14:141–154, 1988.

[6] U-R. Chen, C-C. Wu, and W. Lin. Meshlization of Irregular Grid Resource Topologies by Heuristic Square-Packing Methods. *International Journal of Grid and Distributed Computing*, 2:9–16, 2009.

[7] G-M. Chiu and S-K. Chen. An Efficient Submesh Allocation Scheme for Two-Dimensional Meshes with Little Overhead. *IEEE Transactions on Parallel and Distributed Systems*, 10:471–486, 1999.

[8] W.W. Chu, L.J. Holloway, M-T. Lan, and K. Efe. Task Allocation in Distributed Data Processing. *Computer*, 13:57–69, 1980.

[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

[10] K. Efe. Heuristic models of task assignment scheduling in distributed systems. *Computer*, 15:50–56, 1982.

[11] J.B. Fraleigh. *A First Course in Abstract Algebra*. Addison-Wesley, Reading, MA, 1998.

[12] J.S. Frame, G. de B. Robinson, and R.M. Thrall. The hook graphs of the symmetric group. *Canadian Journal of Mathematics*, 6:316–324, 1954.

[13] W. Fulton. *Young Tableaux: With application to Representation Theory and Geometry*. Cambridge University Press, Cambridge, UK, 1997.

[14] T. Hungerford. *Algebra*. Springer, New York, NY, 1980.

[15] M. Kafil and I. Ahmad. Optimal Task Assignment in Heterogeneous Distributed Computing Systems. *IEEE Concurrency*, 3:42–51, 1998.

[16] D. Kim. Representations of task assignments in distributed systems using young tableaux and symmetric groups. *arXiv.org*, arXiv:1012.1288 [cs.DC], 2010.

[17] D.E. Knuth. *The art of computer programming. Vol. 3: Sorting and searching*. Addison-Wesley Publishing Company, Reading, MA, 1973.

[18] Y-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31:406–471, 1999.

[19] S-M. Lee. Every Young Tableau Graph Is d-Graceful. *Annals of the New York Academy of Sciences*, 555:296–302, 1989.

[20] P. Manneback, E.M. Daoudi, and J. Qin. Optimal Scheduling and Granularity for a 2D-grid precedence graph on a MIMD computer. In L. Dekker, W. Smit, and J.C. Zuidervaart, editors, *EUROSIM*, Massively Parallel Processing Applications and Development, pages 879–886. Elsevier, Delft, The Netherlands, June 21–23 1994.

[21] S. Ramakrishnan, I-H. Cho, and L.A. Dunning. A Close Look at Task Assignment in Distributed Systems. In *INFOCOM '91, IEEE*, pages 806–812, Bal Harbour, FL, April 7–11 1991.

[22] B.E. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Springer, New York, NY, second edition, 2001.

[23] M. P. Scützenberger. Quelques remarques sur une construction de Schensted. *Math. Scand.*, 12:117–128, 1963.

[24] K-H. Seo. Fragmentation-Efficient Node Allocation Algorithm in 2D Mesh-Connected Systems. In *ISPAN '05: Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks*, pages 318–323, Las Vegas, NV, December 7–9 2005.

[25] C.C. Shen and W.H. Tsai. A Graph Matching Approach to Optimal Task Assignment in Distributed Computing System Using a Minimax Criterion. *IEEE Transactions on Computers*, pages 197–203, 1985.

[26] X. Shen, W. Liang, and Q. Hu. On Embedding Between 2D Meshes of the Same Size. *IEEE Transactions on Computers*, 46:880–889, 1997.

[27] Z. Shi, E. Jeannot, and J.J. Dongarra. Robust task scheduling in non-deterministic heterogeneous computing systems. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 1–10, Barcelona, Spain, September 25–28 2006.

[28] O. Sinnen. *Task Scheduling for Parallel Systems*. Wiley-Interscience, Hoboken, NJ, 2007.

[29] R.P. Stanley. *Enumerative Combinatorics, Vol. 2*. Cambridge University Press, Cambridge, UK, 1997.

[30] F. Suter, F. Desprez, and H. Casanova. From Heterogeneous Task Scheduling to Heterogeneous Mixed Parallel Scheduling. In M. Danelutto, M. Vanneschi, and D. Laforenza, editors, *Euro-Par 2004 Parallel Processing*, LNCS 3149, pages 230–237. Springer, Pisa, Italy, August 31–September 3 2004.

[31] A.S. Tanenbaum. *Distributed Operating Systems*. Prentice Hall, Upper Saddle River, NJ, 1995.

[32] C.D. Thompson and H.T. Kung. Sorting on a mesh-connected parallel computer. *Communications of the ACM*, 20:263–271, 1977.

[33] R. Vessenes. Generalized Foulkes' Conjecture and tableaux construction. *Journal of Algebra*, 277:579–614, 2004.

[34] L-L. Wang. Optimal assignment of task modules with precedence for distributed processing by graph matching and state-space search. *BIT*, 28:54–68, 1988.

[35] B.S. Yoo and C.R. Das. A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers. *IEEE Transactions on Computers*, 51:46–60, 2002.

[36] S-M. Yoo, H. Choo, H.Y. Youn, C. Yu, and Y. Lee. On task relocation in two-dimensional meshes. *Journal of Parallel and Distributed Computing*, 60:616–638, 2000.

[37] Y. Zhao. Young tableaux and the representations of the symmetric group. *The Harvard College Mathematics Review*, 2:33–45, 2008.