Optimal Scheduling for UET-UCT Grids Into Fixed Number of Processors

Theodore Andronikos, Nectarios Koziris,
George Papakonstantinou and Panayiotis Tsanakas
National Technical University of Athens
Dept. of Electrical and Computer Engineering
Computer Science Division
Zografou Campus, Zografou 15773, Greece
e-mail: {tedandr, nkoziris, papakon}@cslab.ece.ntua.gr

Abstract

The n-dimensional grid is one of the most representative patterns of data flow in parallel computation. Many scientific algorithms, which require nearest neighbor communication in a lattice space, are modeled by a task graph with the properties of a simple or enhanced grid. In this paper we consider an enhanced model of the n-dimensional grid by adding extra diagonal edges and allowing unequal boundaries for each dimension. First, we calculate the optimal makespan for the generalized UET-UCT (Unit Execution Time – Unit Communication Time) grid topology and, then, we establish the minimum number of processors required, to achieve the optimal makespan. We present the optimal time schedule, using unbounded and bounded number of processors, without allowing task duplication. This paper proves that UET-UCT scheduling of generalized n-dimensional grids into fixed number of processors is low complexity tractable.

1 Introduction

Task scheduling is one of the most important and difficult problems in parallel systems. Since the general scheduling problem is known to be NP-complete (see Ullman [15]), researchers have given attention to other methods such as heuristics, approximation algorithms etc. In their paper Papadimitriou and Yannakakis [13] proved the intractability of the general scheduling problem of a task graph with arbitrary communication and computation times and proposed a clever heuristic with guaranteed worst performance twice the optimum makespan. In addition to this, Gerasoulis and Yang have proposed in [10], [17] the Dominant Sequence Clustering, a low complexity heuristic for general task graph scheduling, which is based on the critical path of tasks. On the other hand, by restricting the general scheduling problem to instances with simple properties, we may come up with tractable solutions. For example, Jung et al. in [9] have presented a polynomial algorithm that finds the optimal makespan when the communication cost is constant and task duplication is allowed.

When both computation and communication times are restricted to have unit time length, it is known that scheduling general UET-UCT graphs with bounded number of processors is NP-complete, as Rayward-Smith proved in [14], or Picouleau in [13] by reduction from the unbounded UET-UCT instance. Even the case of unlimited processors, when no task duplication is allowed, is in general polynomially intractable [13]. On the other hand, using task duplication, Colin et Chretienne in [8] have presented a polynomial optimal schedule for arbitrary task graphs with UET and SCT (Small Communication Times, thus including UCT). Since the arbitrary task graph scheduling with UET-UCT and no duplication with unlimited processors is NP-complete, researchers have focused on special cases of DAGs. In [6] Chretienne presented an algorithm linear in the cardinality of the vertices of the graph, for optimal makespan on SCT in-trees and out-trees (thus covering UCT). In addition to this, there exist polynomial optimal solutions for Series-Parallel digraphs, bipartite graphs and trees with UCT as surveyed in [7].

This paper solves the problem of UET-UCT scheduling for task graphs having the form of a grid, on bounded number of processors, assuming no duplication. Grids and particularly generalized grids are typical task graphs, which model most of the signal processing algorithms and linear algebra methods such as matrix multiplication, LU decomposition etc. Although the general UET-UCT task graph scheduling problem into fixed number of processors is NP-complete, we restrict ourselves to grids which, as we prove, have low complexity tractable schedules. We extend the simple grid model of [4] by considering generalized n-dimensional grids. In [1], [2] we have proved that the problem of time and space scheduling for generalized grids is has a polynomial time closed formula solution, when unbounded number of processors are available. We have calculated the *optimal makespan* for UET-UCT grids and *optimal number of processors*, i.e., the minimum number of processors required to achieve the optimal makespan. In this paper we extend the scheduling strategy presented in [1],[2] by considering bounded number of processors. We present an optimal time and space scheduling policy for UET-UCT grids. Our schedule partitions the vertices of the grid into disjoint sets that lie on a family of parallel hyperplanes. Each hyperplane contains vertices, which are executed on different processors at the same time if unlimited processors are available. Since the number of available processors is less than the cardinality of vertices in most of the hyperplanes ("large hyperplanes"), we use a folding strategy for space scheduling. If m processors are available, we use shifts of m vertices for each execution step. Our method is based on the fact that for every "large" hyperplane, there are m vertices available for execution, which are ancestors of m already executed points of a previous hyperplane.

The paper is organized as follows: In Section 2 we give the notation and some definitions and in Section 3 we present the optimal UET-UCT makespan for n-dimensional generalized grids and the minimum number of processors adequate for the optimal makespan. In Section 4 we establish a tight lower bound for optimal makespan for UET-UCT grids when the number of processors is fixed and a polynomial scheduling strategy that realizes this lower bound, and, thus, is optimal. Finally, in Section 5 we present an illustrative example of a 2-D and a 3-D grid optimally scheduled on unbounded and fixed number of processors.

2 Preliminary Concepts

2.1. Notation

In the rest of the paper the following notation is used:

- N is the set of non negative integers (naturals).
- n is the dimension of the grid.
- G_U is the n-dimensional grid with terminal point **U**=(u_1, …, u_n).

2.2. Basic Concepts

The directed edges of a grid represent **precedence constraints** that have to be satisfied in order to correctly complete the tasks represented by the vertices. The formal definition of the schedule must reflect our intuition that a vertex **j** correctly begins its execution at instant k iff *all the vertices* **i**+ IN(**j**) *have completed their execution and communicated their results (if needed) to* **j** *by that moment.*

- The **initial segment** of N^n with **terminal point U**=(u_1, …, u_n)+ N^n, denoted $N_{\mathbf{U}}$, is the set {(k_1, …, k_n)+ N^n | 0+k_i+u_i, 1+i+n}.
- In case n=1, the initial segment {k+ N | 0+k+u} is denoted N_u.
- Let $\mathbf{e}_i$ be $,, \underbrace{;\cdots;,,,}_{,-,},,, \;,, \underbrace{;\cdots;,,,}_{,\,-,})$, 1+i+n. The (n-dimensional) grid vector set, denoted GVS, is the set {**d**=(d_1, ..., d_n)+ N^n | **d**=$\lambda_1\mathbf{e}_1$+ … +$\lambda_n\mathbf{e}_n$, where λ_i+ {0, 1} + $\underset{i+1}{\overset{n}{+}}$ λ_i+0}.

Definition 2.1. The generalized n-dimensional grid with terminal point **U**, denoted $G_{\mathbf{U}}$, is the DAG with vertices the set $N_{\mathbf{U}}$ and directed edges the set {(**i**, **j**)+$N_{\mathbf{U}}^2$ | **j**=**i**+**d**, **d**+ GVS}. ■

Definition 2.2. We impose a **linear** ordering among the points of $N_{\mathbf{U}}$, which we call **lexicographic ordering**. Let **i**=(m_1, …, m_n) and **j**=(k_1, …, k_n) be two points of $N_{\mathbf{U}}$. We say that **i** is less than **j** and we write **i**+**j** iff m_r+k_r for some r, 1+r+n, and, if r+1, m_i=k_i, 1+i+r-1. ■

In the rest of the paper we use the lexicographic ordering of the vertices, i.e., when we write **i**+**j**, we mean that **i** is lexicographically less than **j**.

Definition 2.3. For every vertex **j** of a grid $G_{\mathbf{U}}$, we define the following sets:

(1) IN(**j**) = {**i**+ $N_{\mathbf{U}}$ | **j**=**i**+**d**, where **d**+ GVS}, and

(2) OUT(**j**) = {**i**+ $N_{\mathbf{U}}$ | **i**=**j**+**d**, where **d**+ GVS}. ■

Definition 2.4. A **schedule** for the grid $G_{\mathbf{U}}$, denoted S($G_{\mathbf{U}}$), is an ordered couple (TIME, PROC), where TIME and PROC are the **time** and **processor schedules**, respectively, defined as follows:

(1) TIME is a function from $N_{\mathbf{U}}$ onto N_k, for some k+ N, such that:

TIME(**j**)=t + the execution of task **j** begins at moment t, and

(2) PROC is a function from $N_{\mathbf{U}}$ onto N_m, for some m+ N, such that:

PROC(**j**)=r + task **j** is assigned to processor r, with the additional requirement

(3) +**i**+ IN(**j**) TIME(**j**)-TIME(**i**)+ $\begin{cases} +p, & ,\ ,, \quad \mathbf{i}),\ , \quad \mathbf{j}) \\ + & \\ +p+c, & ,\ ,, \quad \mathbf{i})+,\quad \mathbf{j}) \end{cases}$, where:

p is the processing time and *c* the communication delay. ■

Remark 2.1.

(1) In UET-UCT grids we assume *p*=1 and *c*=1.

(2) Condition (3) is necessary in order to ensure the **validity** of the schedule, i.e., that the precedence constraints are respected. ❒

Definition 2.5. Let TIME and PROC be functions from $N_{\mathbf{U}}$ onto N_k and N_m, respectively, for some k, m∈N.

Π The **makespan** of the time schedule TIME, denoted M_{TIME}, is k+*p*, where *p* is the processing time.

Π The **processor-span** of the processor schedule PROC, denoted P_{PROC}, is m+1. ■

Clearly, the makespan gives the completion time of the last task and, hence, the time required for the completion of the whole grid, and the processor-span gives the number of processors required by the specific schedule.

Definition 2.6. We now give the following definitions:

Π Assuming unbounded number of processors, a time schedule is optimal, denoted $TIME_{OPT}$, iff its makespan is minimal.

Π In this case a processor schedule is optimal, denoted $PROC_{OPT}$, iff it realizes $TIME_{OPT}$ using the minimal processor-span.

Π Assuming m processors, a time schedule is optimal, denoted $TIME_{m\text{-}OPT}$, iff its makespan is minimal among all time schedules that use m processors.

Π Assuming m processors, a processor schedule is optimal, denoted $PROC_{m\text{-}OPT}$, iff it realizes $TIME_{m\text{-}OPT}$. ■

In [2] and [3] the following result was proved:

Fact 1. Let $G_{\mathbf{U}}$ be an n-dimensional generalized grid with terminal point **U** and let TIME be a time schedule for $G_{\mathbf{U}}$. Then: M_{TIME} = TIME(**U**)+*p*, where *p* is the processing time. ❒

3 Properties of UET-UCT Grids

In order to achieve the execution of a grid in optimal time, we partition its vertices into parallel hyperplanes, taking into account the presence of communication delays. These hyperplanes contain the vertices that can be executed in parallel; if two points belong to the same hyperplane, they can be computed at the same moment and if they belong to successive hyperplanes they can be computed at successive moments.

Definition 3.1. Given the n-dimensional UET-UCT grid $G_{\mathbf{U}}$, with **U**=(u_1, …, u_n), we give the following definitions:

Π The **length** of $G_{\mathbf{U}}$ along the i-th dimension, 1ΠiΠn, denoted $L^i_{\mathbf{U}}$, is 2(u_1+ … +u_{i-1}+u_{i+1}+ … +u_n)+u_i.

Π Π^i_k = {(k_1, …, k_n)∈$N_{\mathbf{U}}$ | 2(x_1+ … +x_{i-1}+x_{i+1}+ … +x_n)+x_i=k, k∈N}, 1ΠiΠn.

Π Π^i_1 = *max*{ $\left|\Pi^i_k\right|$ | 0ΠkΠ$L^i_{\mathbf{U}}$ }, 1ΠiΠn. ■

Geometrically, Π^i_k consists of the common points of the grid $G_{\mathbf{U}}$ and the n-1 dimensional hyperplane 2(x_1+ … +x_{i-1}+x_{i+1}+ … +x_n)+x_i=k.

Every vertex **j**=(k_1, …, k_n) of the grid has a **maximal** coordinate, i.e., a coordinate k_i for which k_iΠk_r, 1ΠrΠn. In [1] and [2] several important results for UET-UCT grids were shown. We state these results here, without proof:

Fact 2. Let $G_{\mathbf{U}}$ be an n-dimensional generalized grid with terminal point **U** and let **j**=(k_1, …, k_n) be a point of $N_{\mathbf{U}}$ with maximal coordinate k_i. The earliest computation time of **j** is $1\prod_{k\Pi 1, k\Pi i}^{k} k_k \Pi k_i$.

This result demonstrates that in UET-UCT grids not all the n dimensions of the grid are equivalent. If our aim is to minimize the execution time of a vertex **j**, then we must pay special attention to the "maximal coordinate" of **j**, in the sense that the same processor must execute all points along this direction. This scheduling strategy has the effect of eliminating the communication overhead along this direction. Hence, in order to achieve the optimal parallel time of a UET-UCT grid, we have to give precedence to the maximal coordinate of the terminal point.

Fact 3. Let $G_{\mathbf{U}}$ be a UET-UCT grid with terminal point **U**=(u_1, …, u_n) and let $TIME_{OPT}$ be an optimal time schedule for $G_{\mathbf{U}}$. Then, we have $M_{TIME_{OPT}}$ = 2(u_1+ … +u_{i-1}+u_{i+1}+ … +u_n)+u_i+1 = $L^i_{\mathbf{U}}$ +1.

Fact 4. Let $G_{\mathbf{U}}$ be a UET-UCT grid with terminal point **U**=(u_1, …, u_n). Then, the following hold:

$$\left|\Pi^i_k\right| \Pi \begin{pmatrix} n \Pi \frac{k}{2}\Pi 1 \\ n\Pi 1 \end{pmatrix} \Pi \prod_{r\Pi 1}^{n}(\Pi 1)\Pi \begin{pmatrix} n \Pi \frac{k}{2}\Pi(u_1\Pi 1)\Pi\ldots\Pi(u_r\Pi 1)\Pi 1 \\ n\text{-}1 \end{pmatrix}$$

, 0ΠkΠ$L^i_{\mathbf{U}}$ and:

$$k_i \Pi \Pi \left\lceil \frac{k_i}{1} \right\rceil 1 \quad 1\,1 \quad 11 \quad , \text{ and} \quad \Pi\left\lceil \frac{k_i \Pi 1}{1} \right\rceil 1 \quad 1\,1 \quad 11$$

$k_k \Pi k_k$, 1ΠrΠiΠn.

Fact 5. Let $G_{\mathbf{U}}$ be a UET-UCT grid with terminal point **U**=(u_1, …, u_n). Then, we have:

Π $\left|\Pi^i_1\right| \Pi \left|\Pi^i_1\right| \Pi\ldots\Pi \left|\Pi^i_{1k_k}\right| \Pi \left|\Pi^i_{11k_k\Pi 11}\right| \Pi\ldots\Pi \left|\Pi^i_{1\Pi\frac{k^i_{\mathbf{U}}}{1}\Pi}\right|$,

where m_e = $\Pi\frac{k^i_{\mathbf{U}}}{1}\Pi$ and

Π $\left|\Pi^i_1\right| \Pi \left|\Pi^i_1\right| \Pi\ldots\Pi \left|\Pi^i_{1k_k\Pi 1}\right| \Pi \left|\Pi^i_{11k_k\Pi 1\,1\Pi 1}\right| \Pi\ldots\Pi \left|\Pi^i_{1\Pi\frac{k^i_{\mathbf{U}}\Pi 1}{1}\Pi 1}\right|$,

$$\text{where } m_o = \left| \frac{\le\le \frac{\le\le k^{i}_{\mathbf{U}} \le 1}{\le\le\ 2} \le}{\le\ 2} \right| .$$
$$\le \qquad\qquad \le$$

Fact 6. Let $G_{\mathbf{U}}$ be a UET-UCT grid with terminal point $\mathbf{U}=(u_1, \ldots, u_n)$. Then, we have:

$$\le^{i}_{1} = \left| \le^{i}_{2\le \frac{\le k^{i}_{\mathbf{U}}}{\le\ 1} \le} \right| .$$

Fact 7. Let $G_{\mathbf{U}}$ be a UET-UCT grid with terminal point $\mathbf{U}=(u_1, \ldots, u_n)$, let u_i be a maximal coordinate of $\mathbf{U}$ and let $PROC_{OPT}$ be an optimal processor schedule for $G_{\mathbf{U}}$. Then, we have:

$$P_{PROC_{OPT}} = \le^{i}_{1} = \left| \le^{i}_{2\le \frac{\le k^{i}_{\mathbf{U}}}{\le\ 1} \le} \right| .$$

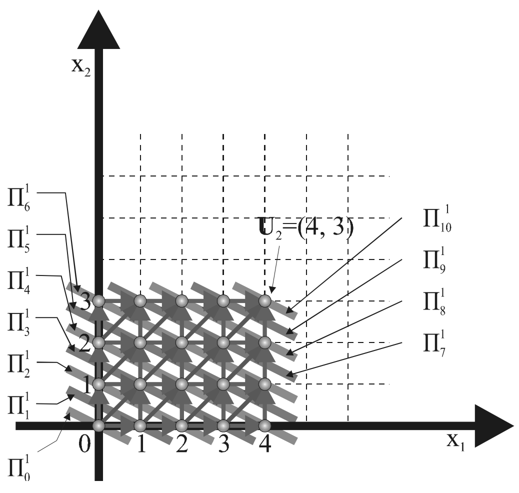

Figure 3.1. An optimal schedule for $G_{\mathbf{U}_2}$.

Example 3.1. In this example we examine the issues involved in scheduling 2-dimensional grids using unbounded number of processors. Given an arbitrary **j** which is executed at time k, consider the set OUT(**j**). Under any optimal scheduling strategy, *at most one* **i**≤ OUT(**j**) will be executed at time k+1 and all others at later time instants. Obviously, the issue here is the selection of the **i** that will lead to the optimal makespan. As we have pointed out the maximal coordinate of the grid determines this selection.

Further, notice that in the UET-UCT case, due to the communication delays, there is no *unique* family of optimal hyperplanes for every grid. Instead, the family of optimal hyperplanes depends on the maximal coordinate of the terminal point. For $G_{\mathbf{U}_2}$ (see Figure 3.1) the optimal hyperplanes are $x_1+2x_2=k$, whereas for $G_{\mathbf{U}_{2\le}}$ (see Figure 3.2) the optimal hyperplanes are $2x_1+x_2=k$, $0\le k\le 10$. ❐

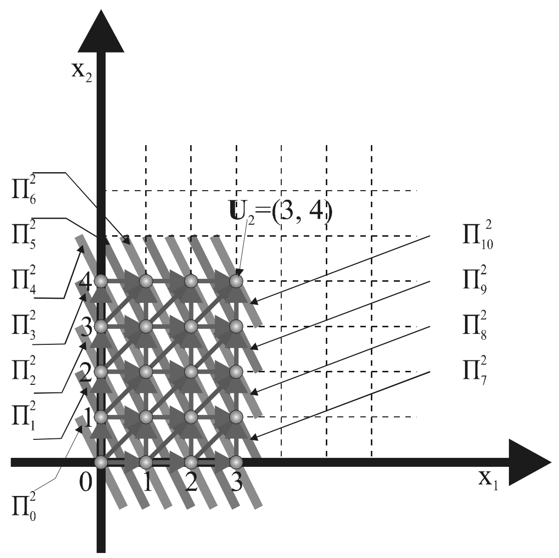

Figure 3.2. An optimal schedule for $G_{\mathbf{U}_{2\le}}$

Example 3.2. The optimal schedule for the grid $G_{\mathbf{U}_3}$ is depicted in Figure 3.3 and Figure 3.4. We must stress the fact that in order to achieve the optimal time schedule for the terminal point, we are forced to execute a specific set of points *at a later time*. In the case of $G_{\mathbf{U}_3}$ these points are of the form $k_2\mathbf{e}_2+k_3\mathbf{e}_3$, where $0\le k_2\le 2$ and $0\le k_3\le 1$ (see Figure 3.3).

Table 1. The optimal time schedule for $G_{\mathbf{U}_3}$.

	P_0	**P_1**	**P_2**	**P_3**	**P_4**	**P_5**
0	(0,0,0)					
1	(1,0,0)					
2	(2,0,0)	(0,0,1)	(0,1,0)			
3	(3,0,0)	(1,0,1)	(1,1,0)			
4		(2,0,1)	(2,1,0)	(0,1,1)	(0,2,0)	
5		(3,0,1)	(3,1,0)	(1,1,1)	(1,2,0)	
6				(2,1,1)	(2,2,0)	(0,2,1)
7				(3,1,1)	(3,2,0)	(1,2,1)
8						(2,2,1)
9						(3,2,1)

Notice that the vertices with the same y and z coordinate must be executed on the same processor, which means that the optimal time schedule must execute the vertices of $\le^{1}_{k}$ at instant k, $0\le k\le 9$ (see Figure 3.4). Consequently, we have $S_{TIME_{OPT}}(\mathbf{U}_3)=9$. The optimal time schedule is given in Table 1. It is important to note that this schedule is not optimal in terms of processors. ❐

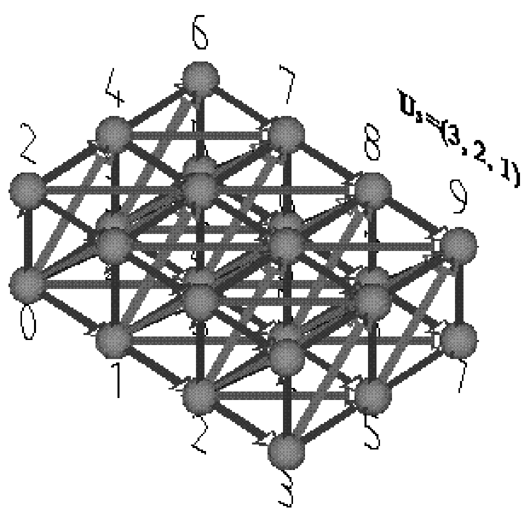

Figure 3.3. Execution Time for G_{U_3}.

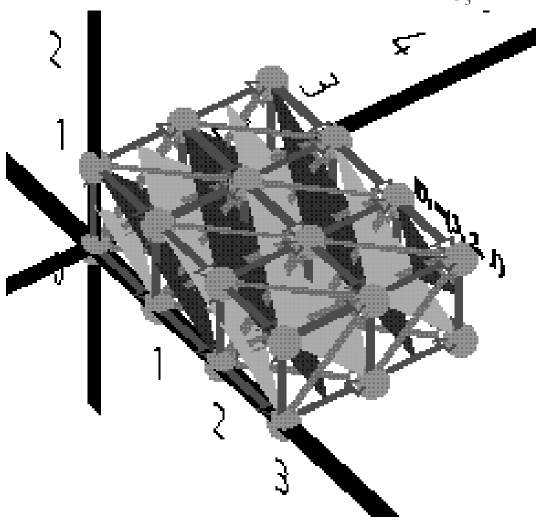

Figure 3.4. An optimal scheduling of G_{U_3}.

4 UET-UCT Grids & Fixed Number of Processors

4.1. Lower and Upper Bounds for the Optimal Makespan

In this section we establish lower and upper bounds for the optimal time schedule under the assumption that the number of processors is fixed. In the rest of this section we denote m the number of processors.

In the rest of the paper we assume that for the terminal point **U**=(u$_1$, …, u$_n$) of the n-dimensional UET-UCT grid G$_\mathbf{U}$ we have u$_1$≥u$_2$≥ … ≥u$_n$. We can do that without any loss of generality because for every grid G$_\mathbf{U}$, with terminal point **U**=(u$_1$, …, u$_n$), there exists an isomorphic grid G$_{\mathbf{U}+}$in normal form with terminal point **U**+=(u$_1$+, …, u$_n$+), where (u$_1$+, …, u$_n$+) is a permutation of (u$_1$, …, u$_n$). This assumption will greatly facilitate the statement of the subsequent results of this paper, regarding the UET-UCT case.

Lemma 4.1. Let G$_\mathbf{U}$ be a UET-UCT grid with terminal point **U**=(u$_1$, …, u$_n$) such that u$_1$≥u$_2$≥ … ≥u$_n$. Let $\mathbf{j}_1 < \ldots < \mathbf{j}_p < \ldots < \mathbf{j}_{|+^1_k|+1}$ and $\mathbf{j}_1 < \ldots < \mathbf{j}_q < \mathbf{j}_{q+1} < \ldots < \mathbf{j}_{|+^1_{k+1}|+1}$ be the index points of $+^1_k$ and $+^1_{k+1}$, respectively, 0+k+ $1^1_\mathbf{U}$, where $\mathbf{j}_q$=$\mathbf{j}_p$ +$\mathbf{e}_1$ and $\mathbf{j}_p$ +$\mathbf{e}_1$+ N$_\mathbf{U}$. Then, the following hold:

(1) $q = \left|+^1_{k+1}\right| + 1 \geq p$,

(2) $1 \quad \bigcup_{k+1|+^1_{k+1}|+1 1+p}^{|+^1_{k+1}|+1} 1 \quad 1\mathbf{j}_k 11 \bigcap + _k 1 + 1\mathbf{j}_1 111 \quad 1\mathbf{j}_p 11$, and

(3) $1 \quad \bigcup_{k+1}^{1|+^1_{k+1}|+1 1+p 1+1} 1 \quad 1\mathbf{j}_k 11 \bigcap + _k 1 + + 1.$ ❐

The above Lemma says that any point **j** of $+^1_{k+1}$ either depends on the point **i**=**j**-**e**$_1$ of $+^1_k$ or it does not depend on any point of $+^1_k$ at all. *Hence, the execution of* **j** *can begin at moment* t+1, *if the execution of* **i**=**j**-**e**$_1$ *can begin at moment* t, *regardless of whether any point of* $+^1_k$ *other than* **i** *began its execution at* t (see Figure 4.1).

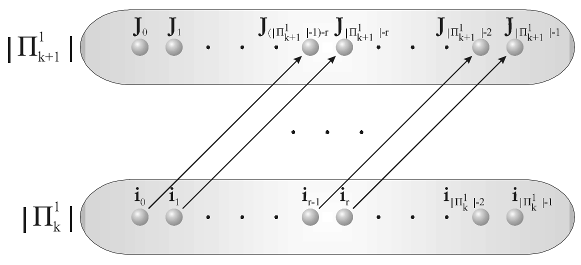

Figure 4.1. The Dependency Relation of $+^k_k$ and $+^k_{k+1}$.

4.2. The Scheduling Strategy

What we have shown so far suggests the following optimal scheduling strategy:

Let $\mathbf{j}_1 < \ldots < \mathbf{j}_{p+1} < \ldots < \mathbf{j}_{|+^1_k|+1}$ and $\mathbf{j}_1 < \ldots < \mathbf{j}_{|+^1_{k+1}|+1}$ be the index points of $+^1_k$ and $+^1_{k+1}$, respectively, 0+k+ $1^1_\mathbf{U}$, and let $\mathbf{j}_{|+^1_{k+1}|+1} = \mathbf{i}_r$ +**e**$_1$. Suppose that at moment t the lexicographically first p points of Π_k^1 $\mathbf{i}_{\square} < \ldots < \mathbf{i}_{i>1}$ are computed. We examine the following cases:

> p-1>r, where $\mathbf{j}_{|\Pi^1_{k>1}|>1} = \mathbf{j}^1_{\square} \; \square \mathbf{i}_i \square$ and $\mathbf{j}^1_{\square} \; \square \mathbf{i}_{i>1} \square$ is undefined. Then **Lemma 4.1** asserts that at moment t+1 the first $(((|\Pi^1_{k>1}|$-1)-r)+p)>p (obviously r>$|\Pi^1_{k>1}|$) points $\mathbf{j}_{\square} < \ldots < \mathbf{j}_{\square |\Pi^1_{k>1}|>1 \square > i \square > i>1}$ of $\Pi^1_{k>1}$ are available for execution.

> p-1>r, where $\mathbf{j}_{|\Pi^1_{k>1}|>1} = \mathbf{j}^1_{\square} \; \square \mathbf{i}_i \square$ and $\mathbf{j}^1_{\square} \; \square \mathbf{i}_{i>1} \square$ is undefined. Then **Lemma 4.1** asserts that at moment t+1 all the points of $\Pi^1_{k>1}$ are available for execution.

The conclusion is that either **at least** p points are available for execution, or **all** the points of $\Pi^1_{k>1}$ are available for execution at moment t+1.

Utilizing the aforementioned optimal scheduling strategy, we derive the optimal makespan for any UET-UCT grid, as stated in the following Theorem:

Theorem 4.1. Let $G_{\mathbf{U}}$ be a UET-UCT grid with terminal point $\mathbf{U}=(u_1, \ldots, u_n)$ such that $u_1>u_2> \ldots >u_n$, let m be the number of processors and let $\mathrm{TIME}_{m\text{-}OPT}$ be an optimal time schedule for $G_{\mathbf{U}}$. Then the following hold:

> $\mathrm{TIME}_{m>OPT}(\mathbf{U}) > r > 1 > \frac{>>_{k>r}^{L^1_U>r} |\Pi^1_k|}{m} > r > 2r > \frac{>>_{k>r}^{L^1_U>r} |\Pi^1_k|}{m} > 1,$

> $M_{\mathrm{TIME}_{m>OPT}} > r > 1 > \frac{>>_{k>r}^{L^1_U>r} |\Pi^1_k|}{m} > r > 1 > 2r > \frac{>>_{k>r}^{L^1_U>r} |\Pi^1_k|}{m},$

where r is the least natural such that $|\Pi^1_i|$>m. ❐

5 Case Study

Consider the grids $G_{\mathbf{U}_2}$ and $G_{\mathbf{U}_3}$, of **Examples 3.1** and **3.2**, with terminal points $\mathbf{U}_2$=(4, 3) and $\mathbf{U}_3$=(3, 2, 1), respectively. The maximal coordinate of both terminal points is the first. The grids $G_{\mathbf{U}_2}$ and $G_{\mathbf{U}_3}$ are partitioned into the hyperplanes Π^i_k, whose cardinality is depicted in Table 2 and Table 3, respectively. Incidentally, one can also see that their optimal makespan is 10 and 9, respectively, when unbounded number of processors is available.

Table 2. The cardinality of Π^1_k for $G_{\mathbf{U}_2}$.

Hyperplane	Cardinality
0	1
1	1
2	2
3	2
4	3
5	2
6	3
7	2
8	2
9	1
10	1

Table 3. The cardinality of Π^1_k for $G_{\mathbf{U}_3}$.

Hyperplane	Cardinality
0	1
1	1
2	3
3	3
4	4
5	4
6	3
7	3
8	1
9	1

If we have only 2 processors available, then the following Table 4 and Table 5 depict the optimal time schedule for the grids $G_{\mathbf{U}_2}$ and $G_{\mathbf{U}_3}$, respectively, based on the analysis of Section 4.

Table 4. The optimal UET-UCT schedule for $G_{\mathbf{U}_2}$.

	P_0	P_1
0	(0, 0)	
1	(1, 0)	
2	(2, 0)	(0, 1)
3	(3, 0)	(1, 1)
4	(4, 0)	(2, 1)
5	(0, 2)	(3, 1)
6	(1, 2)	(4, 1)
7	(2, 2)	(0, 3)
8	(3, 2)	(1, 3)
9	(4, 2)	(2, 3)
10		(3, 3)
11		(4, 3)

6 Conclusion

In this paper we have proved that UET-UCT scheduling of task graphs having the form of n-dimensional generalized grids is not only low complexity tractable but it also has a general solution expressed with a closed formula for both time and processors. We have presented an optimal strategy for both time and processor scheduling of these grids with unit communication delays, having bounded number of processors. The proposed scheduling strategy for these UET-UCT graphs is proved to achieve the optimal makespan.

Table 5. The optimal UET-UCT schedule for G_{U_3}.

	P_0	P_1
0	(0, 0, 0)	
1	(1, 0, 0)	
2	(2, 0, 0)	(0, 1, 0)
3	(0, 0, 1)	(1, 1, 0)
4	(1, 0, 1)	(3, 0, 0)
5	(2, 0, 1)	(2, 1, 0)
6	(0, 1, 1)	(0, 2, 0)
7	(3, 0, 1)	(3, 1, 0)
8	(1, 1, 1)	(1, 2, 0)
9	(2, 1, 1)	(2, 2, 0)
10	(0. 2, 1)	(3, 2, 0)
11	(1, 2, 1)	(3, 1, 1)
12	(2, 2, 1)	
13	(3, 2, 1)	

References

1. Andronikos, T., Koziris, N., Papakonstantinou, G. and Tsanakas, P. Optimal Scheduling for UET-UCT Generalized n-Dimensional Grid Task Graphs, *Proceedings of the 11th IEEE International Parallel Processing Symposium (IPPS97),* pp. 146-151, Geneva, Switzerland, 1997.
2. Andronikos, T., Koziris, N., Papakonstantinou, G. and Tsanakas, P. Optimal Scheduling for UET/UET-UCT Generalized n-Dimensional Grid Task Graphs, accepted, *to appear in Journal of Parallel and Distributed Computing, 1999.*
3. Andronikos, T., Koziris, N., Papakonstantinou, G. and Tsanakas, P. UET/UET-UCT Scheduling for Grids, *Technical Report CS-CSLAB-TR-7-96, Computing Systems Laboratory, Jul. 1996.*
4. Bampis, E., Delorme, C., and Konig, J.C. Optimal Schedules for d-D Grid Graphs with Communication Delays. *Symposium on Theoretical Aspects of Computer Science (STACS96*). Grenoble France 1996.
5. Berman, G., and Fryer, K.D. *Introduction to Combinatorics*. Academic Press, New York, 1972.
6. Chretienne, P. A Polynomial algorithm to Optimally Schedule Tasks over a Virtual Distributed System under Tree-like Precedence Constraints. *Eur. J. Oper. Res.* 43, pp. 225-230, 1989.
7. Chretienne, P. and Picouleau C. (Ed.) Scheduling with Communication Delays: A Survey, *in* Scheduling Theory and its Applications *pp 65-90. John Wiley & Sons 1995.*
8. Colin, J. Y., and Chretienne, P. CPM Scheduling with Small Communication Delays and Task Duplication. *Oper. Res.* 39, pp. 680-684.
9. Jung, H., Kirousis, L., and Spirakis, P. Lower Bounds and Efficient Algorithms for multiprocessor Scheduling of DAGS with communication Delays. *Proceedings of 1st SPAA 1989, pp. 254-264,* and *Information and Computation 105, pp 94-104, 1993.*
10. Gerasoulis, A., and Yang, T. On the Granularity and Clustering of Directed Acyclic Task Graphs. *IEEE Trans. Parallel Distrib. Syst*. 4, 6, pp. 686-701, 1993.
11. Koziris, N., Andronikos, T., Economakos, G., Papakonstantinou, G., and Tsanakas, P. Automatic Hardware Synthesis of Nested Loops using UET Grids and VHDL, *Proceedings of the International Conference on High Performance Computing and Networking (HPCN97), Lecture Notes in Computer Science, Springer-Verlag,* pp. 888-897, Vienna, Austria 1997.
12. Papadimitriou, C., and Yannakakis, M. Toward an Architecture-Independent Analysis of Parallel Algorithms. *SIAM J. Comput*. 19, pp. 322-328, 1990. *Ext. Abstract in STOC 1988*.
13. Picouleau, C. Etude de Problems d' Optimization dans les Systemes Distribues. These, Universite Pierre et Marie Curie, 1992.
14. Rayward-Smith, V.J. UET Scheduling with Unit Interprocessor Communication Delays and Unlimited Number of Processors. *Discrete Applied Mathematics*. 18, pp. 55-71, 1987.
15. Ullman, J. NP-Complete Scheduling problems. *Journal of Computer and Syst. Sciences*. 10, pp. 384-393, 1975.
16. Varvarigou, T., Roychowdhury, V., and Kailath, T. Scheduling in and out Forests in the Presence of Communication Delays. *Proceedings of 7th International Parallel Processing Symposium, pp. 222-229, 1993*.
17. Yang, T. and Gerasoulis, A. DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors. *IEEE Trans. Parallel Distrib. Syst*. 5, 9, pp. 951-967, 1994.