SIAM J. COMPUT.
Vol. 18, No. 6, pp. 1226-1244, December 1989

ON RELAXED SQUASHED EMBEDDING OF GRAPHS INTO A HYPERCUBE*

MING-SYAN CHEN† AND KANG G. SHIN‡

Abstract. Task allocation in an n-dimensional hypercube (or an n-cube) multicomputer consists of two sequential steps: (i) determination of the size of the cube required to accommodate an incoming task composed of a set of interacting modules, and (ii) allocation of the task to a cube of the dimension determined from (i). Step (i) is usually done manually by the users, which is often difficult and leads to the underutilization of processors in an n-cube system. The main objective here is to automate step (i). Step (ii) has already been addressed in [*IEEE Trans. Comput.*, 36 (1987), pp. 1396-1407].

Each incoming task is represented by a graph in which each node denotes a module of the task and each link represents the need of intermodule communication. Each module must be assigned to a subcube in such a way that node adjacencies in the associated task graph are preserved. This assignment problem is called the *relaxed squashed* (RS) *embedding* of a graph, and the minimal dimension of a cube required for a given graph is termed the *weak cubical dimension* of the graph. Some mathematical properties of the RS embedding are derived first. In light of these mathematical properties, fast algorithms are developed to RS embed task graphs. A heuristic function for the A^* search algorithm is also derived to determine the weak cubical dimension of a graph.

Key words. n-cube, loop switching addressing scheme, squashed embedding, weak cubical dimension, heuristic search

AMS(MOS) subject classifications. 05C10, 06E15, 14E25

1. Introduction.

Recently, hypercube multicomputers are beginning to spread widely in the research and development community as well as in commercial markets [Cor85], [Sei85], [Val82], [Wil87]. To execute a task in an n-dimensional hypercube (or n-cube) multicomputer, the task is usually decomposed into a set of interacting modules that are then assigned to a subcube. Thus, task allocation in an n-cube multicomputer system consists of two sequential steps: (i) determination of the dimension of the subcube required to accommodate all the modules of each incoming task, and (ii) allocation of each task to a subcube of the dimension determined from (i) in the hypercube multicomputer. As an efficient solution to (ii), we propose a first-fit linear search for required subcubes whose addresses are represented by the binary reflected Gray code [ChS87]. Conventionally, (i) is determined manually by the users, which is often very difficult and results in the underutilization of processors and degradation of system performance. The automation of step (i) is thus very important and will be the focus of this paper.

Each incoming task is described by a graph (called *task graph*), in which each node denotes a module of the task and each link represents the need of intermodule communication. We want to determine a subcube in the n-cube system that can accommodate the incoming task subject to some constraints. Note that different computing systems and user environments may require different criteria to be used for

* Received by the editors June 13, 1988; accepted for publication (in revised form) January 13, 1989. This work was partially supported by the Office of Naval Research under contract N00014-85-K-0122. This research was performed when M.-S. Chen was at the Real-Time Computing Laboratory, University of Michigan, Ann Arbor, Michigan.

† IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598.

‡ Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan 48109-2122.

the determination of the subcube size required for each task. Several important results in various types of embedding have been reported [Fir65], [GaG75], [GrP72], [Har86], [Har80], [HaM72]. Basically, *isomorphic embedding* is a node-to-node adjacency-preserving mapping [GaG75], [HaM72], whereas *isometric embedding* is a node-to-node distance-preserving mapping [Fir65]. In *homeomorphic embedding*, additional nodes are allowed to be inserted into edges so as to make the graph isomorphically embeddable into a cube [Har86]. *Squashed embedding* is a node-to-subcube distance-preserving mapping [GrP72].

In this paper, we propose and investigate a new type of embedding, called *relaxed squashed* (RS) *embedding*, a node-to-subcube adjacency-preserving mapping. In other words, adjacent modules in the task graph are assigned to adjacent subcubes. The dimension of the minimal cube required for the RS embedding of a given graph will henceforth be called the *weak cubical dimension* of the graph. Clearly, the problem of determining the weak cubical dimension of a task graph is similar to the squashed embedding problem [BGK72], [GrP71], [GrP72], [Yao78], [Win83] in the sense that each node in the source graph is mapped into a subcube. But, it differs from the squashed embedding problem in that only adjacency, rather than internode distance, must be preserved under the mapping. For example, the embedding from a path P_4 into a Q_2 in Fig. 1 preserves adjacency, but not distance.

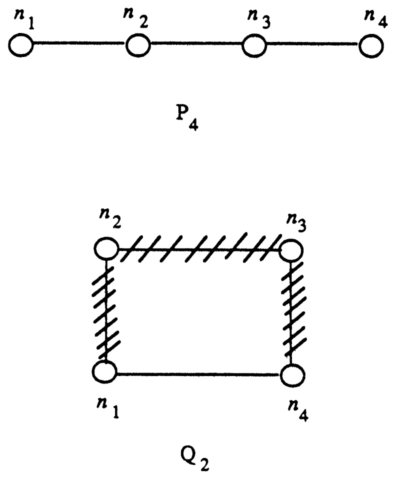


FIG. 1. *A mapping that preserves adjacency but not distance.*

From the result of the squashed embedding problem [Win83], we know that every graph has its weak cubical dimension, although the cubical dimension is defined only for cubical graphs [Har69]. Similarly to the determination of the cubical dimension of cubical graphs [KVC85] [CKV87], we shall prove that the problem of determining the existence of an RS embedding from a graph to a cube of a given dimension is NP-complete. This proof justifies the need of our heuristic approaches to the RS embedding problem. Some mathematical properties for the RS embedding problem will be derived first. Then, using these results, we shall develop (a) fast algorithms for the RS embedding of a given task graph and (b) a heuristic function for the A^* search algorithm to determine if a graph can be RS embedded into a cube of a given dimension.

By applying this search algorithm for different dimensions repeatedly, we can determine the weak cubical dimension of a graph.

This paper is organized as follows. The definitions and notation necessary for our discussion are given in § 2 where related topics and results are also reviewed. Section 3 deals with the mathematical properties of the RS embedding. Using these properties, fast algorithms for the RS embedding are then developed in § 4. A heuristic search algorithm to determine if a graph can be RS embedded into a cube of a given dimension is proposed. Illustrative examples are presented in § 5, and the paper concludes with § 6.

2. Preliminaries.

2.1. Notation and definitions.

Denote an undirected graph by $G_A = (V_A, E_A)$, where V_A and E_A are the set of nodes and the set of links in G_A, respectively, and use $\bar{G}$ to denote the *complement* of a graph G [Har69]. For two graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$, G_B is a *subgraph* of G_A if $V_B \subseteq V_A$ and $E_B \subseteq E_A$. An *induced subgraph* of G_A with a node set $V_S \subseteq V_A$ is the maximal subgraph of G_A with the node set V_S.

An edge in a connected graph is called a *bridge* if its removal disconnects the graph. Clearly, the removal of an edge from a tree will result in two trees, called the *attached trees* of the edge. The number of nodes in the larger[1] of the two attached trees of an edge is called the *weight* of the edge. The *centroid edge* of a tree is defined as the edge with the minimal weight. Besides, the graph operations, $\times$ (product), $\cup$ (union) and $+$ (join) [Har69] will be used to facilitate our presentation. Note that while the union operation may be applied on two graphs that are not disjoint, the join operation is applied only on two disjoint graphs. An illustrative example of the above operations is given in Fig. 2. An n-cube can now be defined as $Q_n = K_2 \times Q_{n-1}$, for all $n \geqq 1$, where K_2 is the complete graph with two nodes and Q_0 is a trivial graph with one node.

Let Σ be the ternary symbol set $\{0, 1, *\}$, where $*$ is the *don't care* symbol. Then, every subcube of an n-cube can be uniquely represented by a sequence of ternary symbols, called the *address* of the subcube. Also, let $|q|$ denote the dimension of the subcube q. The distance between two subcubes is then defined as follows.

DEFINITION 1. The Hamming distance, $H: \sum^n \times \sum^n \rightarrow I^+$, between two subcubes with addresses $\alpha = a_n a_{n-1} \cdots a_1$ and $\beta = b_n b_{n-1} \cdots b_1$ in a Q_n is defined as $H(\alpha, \beta) = \sum_{i=1}^n h(a_i, b_i)$, where

$$h(a, b) = \begin{cases} 1 & \text{if } [a=0 \text{ and } b=1] \text{ or } [a=1 \text{ and } b=0], \\ 0 & \text{otherwise.} \end{cases}$$

A subcube $\alpha = a_n a_{n-1} \cdots a_1$ is said to *contain* another subcube $\beta = b_n b_{n-1} \cdots b_1$, denoted by $\beta \subseteq \alpha$, if and only if all the nodes in β belong to α. The notation $\beta \subset \alpha$ is used to denote the case when $\beta \subseteq \alpha$ and $\beta \neq \alpha$. The *minimal upper subcube* of two subcubes α and β, denoted by $\operatorname{lcm}(\alpha, \beta)$, is then defined as the smallest subcube among all those subcubes which contain both α and β. Similarly, the *maximal lower subcube* of two subcubes α and β, denoted by $\gcd(\alpha, \beta)$, is the largest subcube among all those subcubes contained in both α and β. For notational convenience, we let $\gcd(\alpha, \beta) = \emptyset$ if $H(\alpha, \beta) \geqq 1$. For example, $H(00{*}1{*}, 1000{*}) = 2$, $\operatorname{lcm}({*}100, 0110) = {*}1{*}0$, and $\gcd(01{*}{*}, {*}10{*}) = 010{*}$. Also, let $D(n_i)$ denote the address of the subcube assigned to module n_i and $B(n_i)$ denote the set of nodes adjacent to n_i in the graph. For the graph of Fig. 3, we get $B(n_1) = \{n_2, n_3, n_4, n_6\}$.

[1] One tree is said to be *larger* than the other if it contains more nodes.

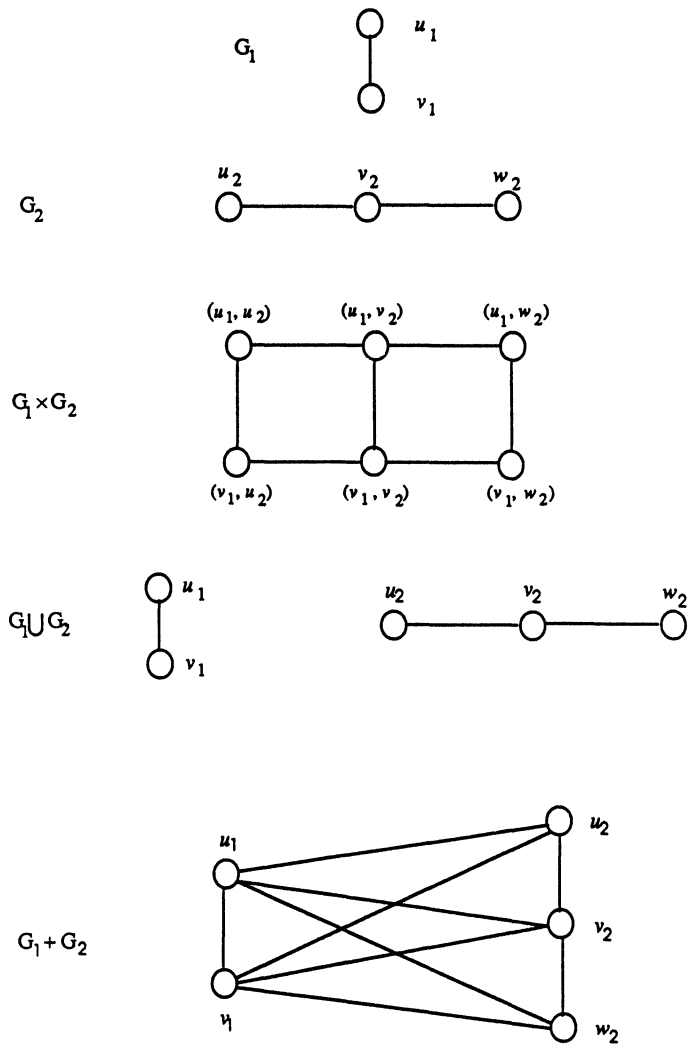


FIG. 2. *The product, union, and join operations on graphs.*

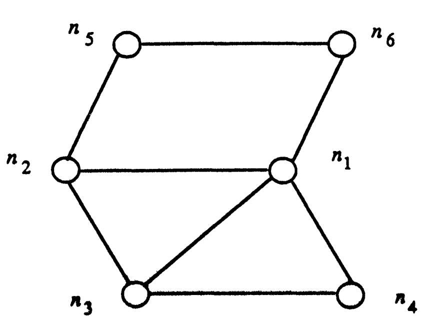


FIG. 3. *An example task graph.*

2.2. Previous related results. In [GrP71], an interesting addressing scheme for loop switching networks [Pie72] has been proposed. In this scheme, a loop switching network is represented by a graph in which nodes and links represent the loops and the contact points between loops, respectively. The problem is to find an addressing scheme in which each node is assigned a sequence of ternary symbols to correctly represent distances between nodes in the graph. More formally, this problem can be stated as follows. Given a connected graph G with n nodes, find the least integer $N(G)$ with which it is possible to assign each node v in G an address $D(v) \in \Sigma^{N(G)}$ such that $d_G(v_1, v_2) = H(D(v_1), D(v_2))$, for all $v_1, v_2 \in V_G$, where $d_G(v_1, v_2)$ is the distance between v_1 and v_2 in G, and V_G is the set of nodes in G. Naturally, the following two questions arise. (1) Does there always exist such an addressing scheme for an arbitrary network G with n nodes? (2) If the answer to (1) is yes, what is the least number $N(G)$ of ternary symbols that suffices to implement the addressing scheme for G? This problem was studied for more than a decade [BGK72], [GrP71], [GrP72], [Yao78] until an important conjecture $N(G) \leqq n-1$ was proved in [Win83]. Thus, questions (1) and (2) have been answered.

As pointed out in [GrP72], this problem is equivalent to the *squashed embedding* problem. Embed a task graph into a cube in such a way that each node of the graph is assigned to a subcube while preserving internode distances. Fig. 4 shows an example of the squashed embedding, where $D(v_1) = {*}11$, $D(v_2) = 110$, $D(v_3) = 010$, and $D(v_4) = 000$.

When task allocation in a hypercube multicomputer is considered, it is more important to preserve node adjacencies than internode distances, since node adjacencies are directly related to intermodule communication delays. Based on this observation, we shall consider the problem of embedding a given task graph into a hypercube in such a way that each task module must be assigned to a subcube while preserving task module adjacencies. This problem can be viewed as a relaxed version of the squashed

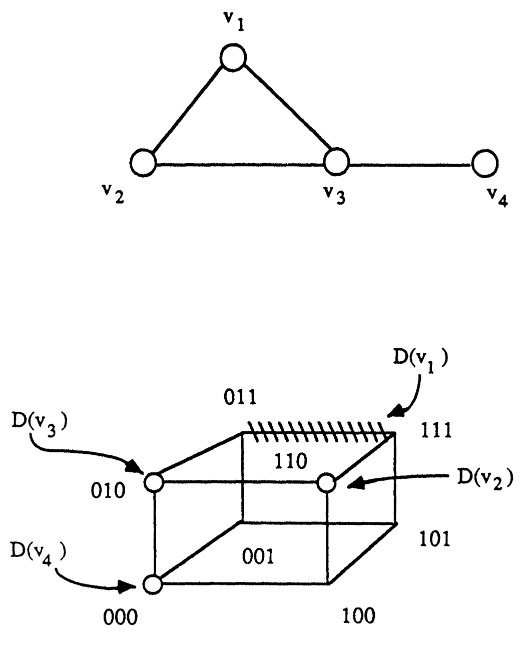


FIG. 4. *An example of squashed embedding.*

embedding problem since it preserves node adjacencies instead of internode distances. (Henceforth it will be called the *relaxed squashed* (RS) *embedding.*) Obviously, a graph can be RS embedded in any cube of size greater than or equal to its weak cubical dimension. Insofar as system utilization is concerned, however, we want to find a minimal cube for the RS embedding of each task graph. Several important properties of the RS embedding problem will be derived in the following section. We shall prove first that the problem of determining the existence of an RS-embedding for a given task graph is NP-complete, and then derive some mathematical properties of the RS embedding. These properties will be applied to develop our heuristic solutions.

3. Mathematical properties of RS-embedding.

THEOREM 1. *The problem of determining if a graph can be* RS *embedded into a cube of a given dimension is* NP-*complete.*

Proof. Suppose k is the dimension of the cube into which a source graph is to be RS embedded. Consider the instance that the source graph contains 2^k nodes. Clearly, the source graph can be RS embedded into a Q_k if and only if it can be isomorphically embedded into a Q_k. However, the problem of determining whether a graph of 2^k nodes can be isomorphically embedded into a Q_k has already been proved to be NP-complete in [CKV87], meaning that the problem of determining whether a graph of 2^k nodes can be RS embedded into a Q_k is also NP-complete. This theorem is thus proved by restriction [GaJ79]. $\square$

Theorem 1 justifies the need of heuristic solution approaches to the RS embedding problem. It is necessary to develop some mathematical properties of the RS embedding problem, on which these heuristic approaches will be based. The following theorem about the squashed embedding has been proved in [GrP72].

THEOREM 2 [GrP72]. $N(K_n) = n-1$, *where* K_n *is a complete graph with* n *nodes.*

Note that when the graph to be embedded is a complete graph, the requirement of preserving distance is the same as the adjacency requirement. This fact is described by the following corollary.

COROLLARY 2.1. *Let wd* (G) *be the weak cubical dimension of* G. *Then, wd* $(K_n) = n-1$.

Consider the case when G_1 is a subgraph of G_2. Clearly, we have less restirction in the RS embedding of G_1 than that of G_2. This leads to the following proposition.

PROPOSITION 1. *If* G_1 *is a subgraph of* G_2, *then* $\text{wd}(G_1) \leqq \text{wd}(G_2)$.

Since the number of nodes in the n-cube must be greater than or equal to that of the task graph to be embedded, we have the following corollary.

COROLLARY 2.2. *Let* G *be a graph with* n *nodes. Then,* $\lceil \log_2 n \rceil \leqq \text{wd}(G) \leqq n-1$.

Note that Corollary 2.2 provides loose bounds for the weak cubical dimension of a graph with n nodes.

THEOREM 3. *Let* $G=(V, E)$ *be a connected graph and let* $G_S=(V_S, E_S)$ *be a subgraph of* G. *Suppose the induced subgraph of* G *with the node set* V_S, *denoted by* $\text{ind}_G(V_S)$, *can be* RS *embedded into a* Q_m, *and the removal of all edges in* E_S *from* G *results in* $|V_S|$ *disjoint graphs,* $G_i=(V_i, E_i)$, $1 \leqq i \leqq |V_S|$. *Then,* $\text{wd}(G) \leqq \max_{1 \leqq i \leqq |V_S|}\{\text{wd}(G_i)\} + m$.

Proof. Let u_i, $1 \leqq i \leqq |V_S| = k$, be the nodes in $G_S \cap G_i$. Since $\text{ind}_G(V_S)$ can be RS embedded into a Q_m and $G_S \subseteq \text{ind}_G(V_S)$, there exists an addressing scheme for the RS embedding of G_S into the Q_m. Let $D_{G_S}(u_i)$ denote the address of $u_i \in V_S \cap V_i$ in the addressing scheme.

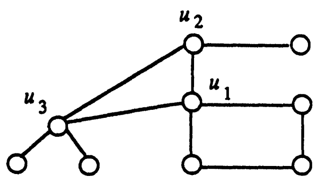
Partition V into k disjoint node sets V_i, $1 \leqq i \leqq k$. For $1 \leqq i \leqq k$, let $D_{G_i}(v)$ denote the address of $v \in V_i$ for the RS embedding of G_i into a $Q_{\text{wd}(G_i)}$ and let $D^k_{G_i}(v)$ denote the kth bit of $D_{G_i}(v)$. Without loss of generality, we can assume $r_1 = \max_{1 \leqq i \leqq k} \{r_i = \mathrm{wd}\,(G_i)\}$. Address each node w in G with $D_G(w)$ according to the following rules:

(1) For all $w \in V_i$, encode the first m bits of $D_G(w)$ with $D_{G_S}(u_i)$.

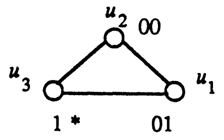
(2) For all $w \in V_i$, $m+1 \leqq k \leqq r_i + m$, **if** $D^k_{G_i}(u_i) = 1$ **then** $D^k_G(w) = \overline{D^{k-m}_{G_i}(w)}$ **else** $D^k_G(w) = D^{k-m}_{G_i}(w)$. And, let $D^k_G(w) = *$ for $r_i + m + 1 \leqq k \leqq r_1 + m$.

This theorem follows from the existence of the above addressing scheme for G, whose length is $\max_{1 \leqq i \leqq |V_S|} \{\mathrm{wd}\,(G_i)\} + m$. □

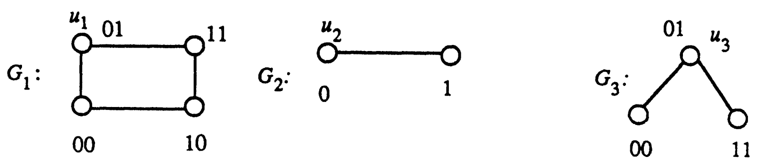
For an illustrative purpose, consider the example graph in Fig. 5(a). The induced subgraph of the node set $\{u_1, u_2, u_3\}$ is G_S in Fig. 5(b) and can be RS embedded into a Q_2. G_1, G_2, and G_3 in Fig. 5(c) are the resulting graphs after removing the edges of G_S from G. We have $\max_{1 \leqq i \leqq 3} \{\mathrm{wd}\,(G_i)\} = 2$. By encoding the last two bits of $D_G(u_i)$, $1 \leqq i \leqq 3$, with 0's and $*$'s only, inverting some corresponding bits in the address $D_{G_i}(w)$ to preserve the adjacency in G_i, and using $D_{G_s}(u_i)$ as the leading portion of the address of $w \in V_i$, we get the address of each node in G as shown in Fig. 5(d).



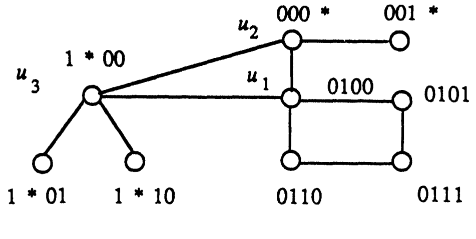
(a) An example graph G.



(b) The encoding of G_S.



(c) The encoding of G_1, G_2 and G_3 .



(d) The encoding of G.

FIG. 5. *An illustrative example for Theorem* 3.

It is important to note that there does not exist any bound tighter than the upper bound provided in Theorem 3. This can be proved by showing the existence of some graphs for which the equality relation in Theorem 3 holds. For example, if $G_i = Q_p$ for $1 \leqq i \leqq |V_S|$ and some nonnegative integer p and $G_S = Q_m$, then $\mathrm{wd}(G) = \mathrm{wd}(G_i) + m$. Furthermore, the above theorem and Proposition 1 lead to the following corollary.

COROLLARY 3.1. *Let G_i, $1 \leqq i \leqq k$, be disjoint graphs and $G = \bigcup_{i=1}^{k} G_i$. Then,* $\mathrm{wd}(G) \leqq \max_{1\leqq i\leqq k} \{\mathrm{wd}(G_i)\} + \lceil \log_2 k \rceil$.

Since every edge in a tree is a bridge, the following corollary, an immediate result of Theorem 3, can be used to determine a tighter upper bound for the weak cubical dimension of a tree.

COROLLARY 3.2. *Let $c_1(T)$ and $c_2(T)$ denote the two attached trees of the centroid edge of a tree T. Then,* $\mathrm{wd}(T) \leqq \max\{\mathrm{wd}(c_1(T)), \mathrm{wd}(c_2(T))\} + 1$.

As it will be shown in § 4, Corollary 3.2 can be applied to implement a fast algorithm for the RS embedding of a given tree. In addition, the effects of join and product operations on the weak cubical dimension of graphs can be described below by Theorems 4 and 5.

THEOREM 4. $\mathrm{wd}(G_1 + G_2) \leqq \mathrm{wd}(G_1) + \mathrm{wd}(G_2) + 1$.

Proof. Let $D_{G_j}(w)$ denote the address of $w \in G_j$, $j = 1, 2$, before a join operation. Address each node w in $G = G_1 + G_2$ according to the following rules:

(1) If $w \in V_1$, then $D_G(w) = 0* \cdots *D_{G_1}(w)$, which contains $\mathrm{wd}(G_2)$ consecutive $*$'s before $D_{G_1}(w)$.

(2) If $w \in V_2$, then $D_G(w) = 1D_{G_2}(w)* \cdots *$, which contains $\mathrm{wd}(G_1)$ consecutive $*$'s after $D_{G_2}(w)$.

Clearly, the above addressing scheme, having length $\mathrm{wd}(G_1) + \mathrm{wd}(G_2) + 1$, not only preserves the original adjacency in G_1 and G_2, but also joins every pair of nodes (u_1, u_2), $u_1 \in V_1$, $u_2 \in V_2$. □

Note that Theorem 4 provides the best upper bound, since there exist some graphs for which the equality relation in Theorem 4 holds, e.g., $\mathrm{wd}(Q_1 + Q_2) = \mathrm{wd}(Q_1) + \mathrm{wd}(Q_2) + 1$.

COROLLARY 4.1. *Let $\{V_1, V_2\}$ be a partition of the node set of a graph G_A, i.e., $V_1 \cap V_2 = \varnothing$ and $V_1 \cup V_2 = V_A$. Let the induced subgraphs of G_A with the node sets V_1 and V_2 be G_{I_1} and G_{I_2}, respectively. Then,* $\mathrm{wd}(G_A) \leqq \mathrm{wd}(G_{I_1}) + \mathrm{wd}(G_{I_2}) + 1$.

Proof. Since $G_A \subseteq G_{I_1} + G_{I_2}$, the inequality $\mathrm{wd}(G_A) \leqq \mathrm{wd}(G_{I_1} + G_{I_2}) \leqq \mathrm{wd}(G_{I_1}) + \mathrm{wd}(G_{I_2}) + 1$ follows from Proposition 1 and Theorem 4. □

Let $G_{I_{A-S}}$ denote the induced subgraph of G_A with the node set $V_A - V_S$ where $V_S \subseteq V_A$. Then, we have the following corollary.

COROLLARY 4.2. $\mathrm{wd}(G_A) \leqq \mathrm{wd}(G_{I_{A-S}}) + |V_S|$.

Proof. Let G_{I_S} be the induced subgraph of G_A with V_S. From Corollary 4.1, $\mathrm{wd}(G_A) \leqq \mathrm{wd}(G_{I_S}) + \mathrm{wd}(G_{I_{A-S}}) + 1$. In addition, we get $\mathrm{wd}(G_{I_S}) \leqq |V_S| - 1$ from Corollary 2.2, and thus, this corollary follows. □

Using Corollaries 4.1 and 4.2, in § 4 we shall propose two fast algorithms for the RS embedding of a given graph. The relationship between the weak cubical dimensions of several graphs and that of their union can be described by the following corollary.

COROLLARY 4.3. *Let $G = \bigcup_{i=1}^{m} G_i$. Then,* $\mathrm{wd}(G) \leqq \sum_{i=1}^{m} \mathrm{wd}(G_i) + m - 1$.

Proof. First, prove the inequality $\mathrm{wd}(G_1 \cup G_2) \leqq \mathrm{wd}(G_1) + \mathrm{wd}(G_2) + 1$. Let G_{I^*} be the induced subgraph of G_2 with the node set $V_2 - V_1$. Clearly, $G_1 \cup G_2 \subseteq G_1 + G_{I^*}$. Then, the inequality, $\mathrm{wd}(G_1 \cup G_2) \leqq \mathrm{wd}(G_1) + \mathrm{wd}(G_{I^*}) + 1 \leqq \mathrm{wd}(G_1) + \mathrm{wd}(G_2) + 1$, follows from Theorem 4 and Proposition 1. The corollary follows by applying this inequality repeatedly. □

It is interesting to compare Corollary 4.3 with Corollary 3.1 that is applicable to disjoint graphs only. This result agrees with our intuition, since there are fewer restrictions in the RS embedding of disjoint graphs. Moreover, we have the following corollary for the complement of a graph.

COROLLARY 4.4. *Let G be a graph with n nodes. Then*, $\mathrm{wd}\,(G)+\mathrm{wd}\,(\bar{G})\geqq n-2$.

Proof. Since $K_n=G\cup\bar{G}$, we get $n-1=\mathrm{wd}\,(K_n)=\mathrm{wd}\,(G\cup\bar{G})\leqq\mathrm{wd}\,(G)+\mathrm{wd}\,(\bar{G})+1$. □

Corollary 4.4 can sometimes be used to determine tighter bounds of the weak cubical dimension of a graph. For example, Corollary 2.2 offers a loose lower bound (6) of $\mathrm{wd}\,(\overline{Q_6})$, whereas Corollary 4.4 gives a much tighter lower bound (56) of $\mathrm{wd}\,(\overline{Q_6})$.

COROLLARY 4.5. *Let T be a tree with n nodes. Then*, $\mathrm{wd}\,(T)\leqq 2\lceil\log_2 n\rceil$.

Proof. Since every tree is a bigraph, there exists an integer m such that T is a subgraph of $K_{n-m,m}$, where $K_{p,q}$ is a complete bigraph [Har69]. Without loss of generality, we can let $m\leqq n/2$. Clearly, $\lceil\log_2 m\rceil\leqq\lceil\log_2 n/2\rceil=\lceil\log_2 n\rceil-1$, and $\lceil\log_2(n-m)\rceil\leqq\lceil\log_2 n\rceil$. Then, we have $\mathrm{wd}\,(T)\leqq\mathrm{wd}\,(K_{n-m,m})\leqq\lceil\log_2(n-m)\rceil+\lceil\log_2 m\rceil+1\leqq\lceil\log_2 n\rceil+\lceil\log_2 n\rceil-1+1=2\lceil\log_2 n\rceil$. □

This corollary offers a tighter upper bound of the weak cubical dimension of a tree. Using Corollary 4.5, a fast RS embedding algorithm for a given tree will be developed in § 4.

THEOREM 5. $\mathrm{wd}\,(G_1\times G_2)\leqq\mathrm{wd}\,(G_1)+\mathrm{wd}\,(G_2)$.

Proof. For all $u_1\in V_1$, $u_2\in V_2$, let $D_{G_1}(u_1)$ and $D_{G_2}(u_2)$ be the addresses of u_1 and u_2 before the product operation. Encode the address of a node (u_1, u_2) in $G_1\times G_2$ with the concatenation of their original addresses, $D_{G_1}(u_1)D_{G_2}(u_2)$, whose length is $\mathrm{wd}\,(G_1)+\mathrm{wd}\,(G_2)$. Obviously, the adjacency requirement in $G_1\times G_2$ is preserved under the above addressing scheme, and thus the theorem follows. □

Note that Theorem 5 also provides the best upper bound. For example, $\mathrm{wd}\,(Q_r\times Q_s)=\mathrm{wd}\,(Q_r)+\mathrm{wd}\,(Q_s)$ for positive integers r and s. It can also be verified that the above addressing schemes are valid for the squashed embedding problem, i.e., $N(G_1\times G_2)\leqq N(G_1)+N(G_2)$. In addition, from the topology of a hypercube, we have the theorem below.

THEOREM 6. *Let q be an m-dimensional subcube of a Q_n, where $n\geqq m$. Then, q is adjacent to at most $(n-m)2^m$ subcubes within the Q_n.*

Proof. Without loss of generality, we can let the address of q be $00\cdots0*\cdots*$, in which there are $n-m$ consecutive 0's followed by m consecutive $*$'s. Note that the address of every Q_0 adjacent to q must have one 1 and $(n-m-1)$ 0's in its left $n-m$ bits. Among all Q_0's adjacent to q, there are 2^m different Q_0's with the kth bit equal to 1 for $m+1\leqq k\leqq n$. Thus, q is adjacent to exactly $(n-m)2^mQ_0$'s. □

In what follows, the number $(n-m)2^m$ will be referred to as the *adjacency number* of q, where q is an m-dimensional subcube of a Q_n.

COROLLARY 6.1. *Let $\{d_i\}$ be the degree sequence of a graph G_A. If $\mathrm{wd}\,(G_A)\leqq m$, then $\sum_{i=1}^{|V_A|}2^{b_i}\leqq 2^m$, where for each $1\leqq i\leqq|V_A|$, b_i is the least nonnegative integer such that the adjacency number of $Q_{b_i}\geqq d_i$.*

Proof. From Theorem 6, the dimension of the subcube assigned to a task node n_i with degree d_i cannot be less than b_i. This corollary follows from the fact that the total number of nodes in a Q_m assigned to G_A must be less than or equal to the total number of its nodes, 2^m. □

Since $b_i\geqq 0$, $1\leqq i\leqq|V_A|$, we have $\sum_{i=1}^{|V_A|}2^{b_i}\geqq|V_A|$, meaning that using the knowledge of degree sequence provides a tighter lower bound than Corollary 2.2. Moreover, the relationship between the number of edges in a graph and its weak cubical dimension can be described by the following theorem.

THEOREM 7. *Let* $m = \mathrm{wd}\,(G_A)$ *and* $k = \lfloor \log_2 (2m/|V_A|)\rfloor$. *Then,*

$$|E_A| \leqq \tfrac{1}{2}[(2^{m-k} - |V_A|)\min\{(m-k-1)2^{k+1}, |V_A|-1\} + (2|V_A| - 2^{m-k})\min\{(m-k)2^k, |V_A|-1\}].$$

Note that Theorem 7 provides a necessary condition for a task graph with a given number of edges to be RS embedded into a cube. This condition provides information useful for the decomposition of a task into (interacting) modules in such a way that the resulting task graph can be embedded into a cube of a given dimension. In order to simplify the proof of Theorem 7, first it is necessary to introduce the following two lemmas.

LEMMA 1. *The maximal adjacency number of a subcube in a* Q_n *is* 2^{n-1}, *which is attained by a subcube of dimension* $n-1$ *or* $n-2$.

Proof. Let $F(k) = (n-k)2^k$. Since $dF(k)/dk = 2^k[(n-k)\log_e 2 - 1] > 0$ for $0 \leqq k \leqq n-2$, and $F(n-1) = F(n-2) = 2^{n-1} > F(n) = 0$, this lemma follows. □

LEMMA 2. *Let* a_i, $1 \leqq i \leqq r$, *be nonnegative integers and* $\sum_{i=1}^{r} 2^{a_i} \leqq 2^m$. *Then,* $f(a_1, a_2, \cdots, a_r) = \sum_{i=1}^{r}(m-a_i)2^{a_i} \leqq 2^{m-k} - r)(m-k-1)2^{k+1} + (2r - 2^{m-k})(m-k)2^k$, *where* $k = \lfloor \log_2 (2^m/r)\rfloor$.

Proof. Let $(a_1^*, a_2^*, \cdots, a_r^*)$ be the vector that maximizes $f(a_1, a_2, \cdots, a_r)$, i.e., $f^* = f(a_1^*, \cdots, a_r^*)$. From the proof of Lemma 1, we know that $g(a_i) = (m-a_i)2^{a_i}$ is a monotonically increasing function in the integer variables a_i, where $1 \leqq a_i \leqq m-1$. Let $a_p^* = \min_{1\leqq i \leqq r}\{a_i^*\}$ and suppose $2^m - \sum_{i=1}^{r} 2^{a_i^*} > 0$. Then, $2^m - \sum_{i=1}^{r} 2^{a_i^*}$ must be an integral multiple of $2^{a_p^*}$. This is impossible, since the a_p^* in the function f can be replaced with $a_p^* + 1$, resulting in a larger f-value than f^*. Thus, $\sum_{i=1}^{r} 2^{a_i^*} = 2^m$, implying that there exist a_p^* and a_x^*, $p \neq x$, such that $a_p^* = a_x^* = \min_{1\leqq i\leqq r}\{a_i^*\}$. We claim that $\max_{1\leqq i\leqq r}\{a_i^*\} - \min_{1\leqq i\leqq r}\{a_i^*\} \leqq 1$, and then this lemma follows from the fact that $2^{m-k} - r$ variables among a_i^*'s are $k+1$ and $2r - 2^{m-k}$ variables among a_i^*'s are k.

Let $a_y^* = \max_{1\leqq i\leqq r}\{a_i^*\}$, $a_p^* = a_x^* = \min_{1\leqq i\leqq r}\{a_i^*\}$ and suppose $a_y^* - a_p^* \geqq 2$. Then, we have $2^{a_y^*} + 2^{a_p^*} + 2^{a_x^*} = 2^{a_y^*-1} + 2^{a_y^*-1} + 2^{a_p^*+1}$ and $(m-a_y^*)2^{a_y^*} + (m-a_p^*)2^{a_p^*} + (m-a_x^*)2^{a_x^*} \leqq (m-a_y^*+1)2^{a_y^*-1} + (m-a_y^*+1)2^{a_y^*-1} + (m-a_p^*-1)2^{a_p^*+1}$. This leads to a contradiction, because a_y^*, a_p^*, and a_x^* in the function f can be replaced by a_y^*-1, a_y^*-1, and a_p^*+1, respectively, yielding a larger f-value than f^*. Therefore, the claim $\max_{1\leqq i\leqq r}\{a_i^*\} - \min_{1\leqq i\leqq r}\{a_i^*\} \leqq 1$ is proved and, thus, this lemma follows. □

Proof of Theorem 7. Let a_i be the dimension of the subcube assigned to a task node n_i in G_A, $1 \leqq i \leqq |V_A| = r$. Then, $\sum_{i=1}^{r} 2^{a_i} \leqq 2^m$ follows from the capacity constraint of a Q_m. Note that the adjacency number of the subcube assigned to n_i is $(m-a_i)2^{a_i}$ and the degree of any node in $G_A \leqq |V_A| - 1$. This theorem follows from Lemma 2 and the fact that $\sum_{i=1}^{r} d_i = 2|E_A|$. □

When a graph belongs to some regular families, its weak cubical dimension can be determined by the theorem below.

THEOREM 8. *The weak cubical dimensions of a cycle* C_m, *a path* P_m, *and a star* S_m *can be determined by the following formulas*:

(i) $\mathrm{wd}\,(C_m) = \lceil \log_2 m \rceil$;

(ii) $\mathrm{wd}\,(P_m) = \lceil \log_2 m \rceil$;

(iii) $\mathrm{wd}\,(S_m) = \lceil \log_2 (m-1) \rceil + 1$.

Proof. Consider (i) first. Clearly, $\mathrm{wd}\,(C_m) \geqq \lceil \log_2 m \rceil \equiv k$. From the existence of Hamiltonian cycles in a Q_k, we know that a C_m can be RS embedded into a Q_k by embedding $2^k - m$ nodes of the C_m into Q_1's and $2m - 2^k$ nodes of the C_m into Q_0's, and thus (i) is proved. Part (ii) follows from (i) immediately.

Consider (iii). Let δ_k be a trivial graph with k nodes and no edges. Note that $S_m = \delta_{m-1} + \delta_1$ and $\mathrm{wd}\,(\delta_k) = \lceil \log_2 k \rceil$. Then, we have $\mathrm{wd}\,(S_m) \leqq \lceil \log_2 (m-1) \rceil + 1$ by

Theorem 4. From Lemma 1, we know that the maximal adjacency number of a subcube in a $Q_{\lceil \log_2 (m-1) \rceil}$ is $2^{\lceil \log_2 (m-1) \rceil - 1} < m-1$. Thus, $\mathrm{wd}\,(S_m) > \lceil \log_2 (m-1) \rceil$ and (iii) follows. □

COROLLARY 8.1. *Let $m_{c\times d}$ denote a $(c \times d)$-dimensional mesh. Then,* $\mathrm{wd}\,(m_{c\times d}) \leqq \lceil \log_2 c \rceil + \lceil \log_2 d \rceil$.

Proof. Since a $(c\times d)$-dimensional mesh is $P_c \times P_d$, this corollary follows from Theorem 5 and (ii) of Theorem 8. □

Due to its nature of NP-completeness, the weak cubical dimension of a graph is in general very difficult to characterize. However, as we shall show in the following section, the mathematical properties derived in this section can play a significant role in designing efficient algorithms for the RS embedding of a given graph.

4. Algorithms for relaxed squashed embedding.

The mathematical properties derived in § 3 are applied to the design of algorithms for the RS embedding. Fast algorithms of polynomial time complexity that are efficient but may not provide the minimal cube required for a given task graph are presented first. Then, a heuristic search algorithm is developed to determine the weak cubical dimension of a graph.

4.1. Fast algorithms for RS-embedding.

Since every tree is a bigraph, we have an efficient addressing scheme for a tree with n nodes as described below.

ALGORITHM $A_1(T)$/*. This algorithm uses the property that every tree T is a bigraph and determines an efficient addressing scheme for T.*/

```
Step (1). Choose an arbitrary node in $T$. Label it with a symbol $+$.
Step (2). Label with $-$'s all the nodes adjacent to each node labeled with $+$. **If**
          every node in $T$ has been labeled with $+$ or $-$ **then** goto Step (4).
Step (3). Label with $+$'s all the nodes adjacent to each node labeled with $-$. **If**
          every node in $T$ has been labeled with $+$ or $-$ **then** goto Step (4) **else**
          goto Step (2).
Step (4). Suppose there are $j$ nodes with $+$ and $k$ nodes labeled with $-$. Then,
          encode all the nodes labeled with $+$ with $0*\cdots*B^{(+)}(i)$, $0 \leqq i \leqq j-1$,
          where $B^{(+)}(i)$ is a binary representation of the number $i$ with $\lceil \log_2 j \rceil$
          bits, which follows $\lceil \log_2 k \rceil$ $*$'s. Also, encode all the nodes labeled with
          $-$ with $1B^{(-)}(i)*\cdots*$, $0 \leqq i \leqq k-1$, where $B^{(-)}(i)$ is a binary representa-
          tion of the number $i$ with $\lceil \log_2 k \rceil$ bits, followed by $\lceil \log_2 j \rceil$ $*$'s.
```

By Corollary 4.5, the length of the above addressing scheme must be less than or equal to $2\lceil \log_2 n \rceil$. (This, in general, is significantly less than $n-1$ for a large n.) Although the required length of the addressing scheme used in A_1 may be larger than the weak cubical dimension of the tree, A_1 is favorable in some cases due to its *linear* complexity.

Corollary 3.2 suggests the following algorithm that also determines a cube required to accommodate a tree.

ALGORITHM $A_2(T)$/*. This algorithm determines the dimension of a cube to accommodate a task tree T.*/

```
Step (1). **If** $T$ is a star or a path **then** determine wd $(T)$ by Theorem 8 and return
          wd$(T)$ **else** compute the weight of each edge and determine the centroid
          edge of the tree.
Step (2). Let $T_1$ and $T_2$ be the two attached trees of the centroid edge of $T$. Return
          $\max\{A_2(T_1), A_2(T_2)\}+1$.
```

A_2 is recursive and uses the divide-and-conquer technique. We decompose a tree by removing its centroid edge first, and then continue to decompose the remaining trees in the same way until only stars or paths are left, whose weak cubical dimension can be determined by Theorem 8. An illustrative example for this algorithm can be found in § 5. Note that the complexity of A_2 depends on the degree of sophistication in the way of determining the centroid edge. Nevertheless, it is easy to verify that A_2 requires only polynomial time.

Consider the case when the task graph is an *arbitrary* graph. Clearly, using Corollary 2.1, we can derive a straightforward algorithm for the RS embedding of each graph: address any first two nodes with $0\cdots 0$ and $0\cdots 01$, respectively, each of which consists of $n-1$ bits, and then the kth node, $3 \leqq k \leqq n$, with $0\cdots 01{*}\cdots{*}$ that consists of $n-k$ consecutive 0's and $k-2$ consecutive $*$'s. However, despite its linear complexity, this naive algorithm is not used for a better system utilization. Instead, we present Algorithm A_3 below that is derived from Corollary 4.2.

ALGORITHM $A_3(G)$/*. Using the technique of node-removing, this is a fast algorithm to determine the size of the cube required for a given task graph.*/

Step (1). Let n^* be the node with the largest degree among all the nodes in G. **If** $d(n^*) \leqq 2$ **then** goto Step (2) **else** goto Step (3).

Step (2). Determine all the cycles in G, denoted by $C^1, C^2, \cdots, C^m$, and the least integer p such that $2^p \geqq \sum_{i=1}^{m} 2^{\lceil \log_2 |C^i| \rceil} + 2^{\lceil \log_2 (|V| - \sum_{i=1}^{m} |C^i|) \rceil}$, where $|C^i|$ is the number of nodes in the cycle C^i. Return p.

Step (3). Let $G_1 := G - n^*$ and return $A_3(G_1)+1$.

Using A_3, a graph is reduced by removing the node with the largest degree from the graph. The reduction steps are performed repeatedly until the graph is reduced to the extent that it contains only disjoint cycles and paths. By Corollary 4.2, the size determined in Step (2) plus the total number of nodes removed will be the dimension of a cube required to accommodate the original task graph. Note that in A_2 and A_3, it is required to determine if a graph belongs to some families of graph such as paths, stars, and cycles. For this purpose, an adjacency matrix [Har69] can be used to represent each task graph, since these families of graph can be easily identified if they are represented with adjacency matrices. Moreover, by Corollary 4.1 we can modify Step (3) of A_3 as follows and get a generalized version of A_3, called Algorithm A_4.

Step (3′). Partition V into V_1 and V_2. Let G_{I_1} and G_{I_2} be, respectively, the induced subgraphs of G with the node sets V_1 and V_2. Return $A_4(G_{I_1}) + A_4(G_{I_2}) + 1$.

Several heuristic approaches can be employed in determining how to partition the node set V into V_1 and V_2 in Step (3′) of A_4. Clearly, a more sophisticated method will lead to an addressing scheme with a shorter length at the cost of higher computational costs of A_4.

Although the above proposed algorithms are efficient in determining the required cube for a given graph, the resulting cube may not be minimal. As far as the system utilization is concerned, we want to find the minimal subcube required for a given task graph. This is explored in § 4.2.

4.2. An algorithm for determining the weak cubical dimension. To determine the weak cubical dimension of a task graph, first we present an algorithm that determines whether or not there is an RS embedding from a given task graph into a cube. Then, the algorithm is applied to determine the weak cubical dimension of the graph. To facilitate our discussion, we label the task graph as follows. Label the node with the largest degree with n_1 and let $X := \{n_1\}$ and $i := 2$. Then, among all the nodes that are adjacent to any node in X and are not in X, choose a node with the largest degree and label this node with n_i. Then, let $X := X \cup \{n_i\}$ and $i := i+1$. Repeat the same procedure until all nodes are labeled.

Now, we want to assign a subcube within the n-cube to each node in a task graph, node by node, subject to the adjacency requirement in the RS embedding. Clearly, this problem is a graph matching problem and can be solved by a state-space search similar to the one in [ShT85]. In what follows, we shall formulate a heuristic function, and the A^* search algorithm [Nil80] will then be used to determine the existence of an RS embedding from a given graph into a cube. The following definitions are necessary to facilitate our presentation.

DEFINITION 2. The *merge* operation, denoted by $\odot$, of two sets of subcubes, U_1 and U_2, is defined as

$$U_1 \odot U_2 = \{\tau \mid \tau = \operatorname{lcm}(\alpha, \beta) \text{ for } \alpha \in U_1 \text{ and } \beta \in U_2\}.$$

The merge operation among $k \geqq 2$ sets of subcubes is written as $\odot_{1 \leqq i \leqq k} U_i$.

DEFINITION 3. The *exclusion* operation of two sets of subcubes, U_1 and U_2, is defined as

$$U_1 - U_2 = \{r \mid r \in U_1 \text{ and } \gcd(t, r) = \emptyset, \forall t \in U_2\}.$$

DEFINITION 4. The *reduced set* of a set of subcubes U is defined as

$$\operatorname{Rd}(U) = U - \{r \mid r \in U \text{ and } t \subset r \text{ for some } t \in U\}.$$

For example, let $U_1 = \{0{*}{*}, 0{*}0, 01{*}, 001\}$, $U_2 = \{00{*}, 10{*}\}$, and $U_3 = \{001\}$. Then, $\operatorname{Rd}(U_1) = \{0{*}0, 01{*}, 001\}$, $U_1 - U_2 = \{01{*}\}$, and $U_2 \odot U_3 = \{00{*}, {*}0{*}\}$. Recall that $B(n_i)$ is the set of all nodes adjacent to n_i in the task graph G_T. Let M^i denote the partial mapping for the task node n_j, $1 \leqq j \leqq i$. Let $A_{n_j}^{(i)}$ be the set of unoccupied Q_0's that are adjacent to $D(n_j)$ under the partial mapping M^i. Also, define the set of *essential subcubes* of n_j under the partial mapping M^i, denoted by $E_{n_j}^{(i)}$, as the reduced set of unoccupied subcubes that are adjacent to the subcubes assigned to all $n_k \in B(n_j)$, $1 \leqq k \leqq i$. For example, suppose that in the graph of Fig. 3, we have $D(n_1) = 00{*}$, $D(n_2) = 010$. Then, $A_{n_1}^{(2)} = \{011, 100, 101\}$, $A_{n_2}^{(2)} = \{110, 011\}$, and $E_{n_3}^{(2)} = \{011, 1{*}0\}$. That is, the subcube to be assigned to n_3 should contain either 011 or $1{*}0$ to satisfy the adjacency requirement. Then, the $E_{n_k}^{(i)}$ generated under M^i can be expressed as follows:

$$E_{n_k}^{(i)} = Rd\left(\bigodot_{\substack{n_j \in B(n_k) \\ 1 \leqq j \leqq i}} A_{n_j}^{(i)} \right) - \bigcup_{j=1}^{i} D(n_j) \quad \forall k > i. \tag{1}$$

From this formula, we can determine the sets of all essential subcubes of unassigned task nodes. Note that $E_{n_k}^{(i)}$ is determined by $A_{n_j}^{(i)}$, for all $n_j \in B(n_k)$, and the adjacency requirement, and the term $\bigcup_{j=1}^{i} D(n_j)$ in equation (1) is necessary to exclude the possibility of allocating the already occupied subcubes. Given a partial mapping M^i, the set of all possible subcubes that can be assigned to the task node n_{i+1} is represented by

$$\operatorname{Sp}(E_{n_{i+1}}^{(i)}) = \left\{ q \,\middle|\, \begin{array}{l} \text{there exists } t \in E_{n_{i+1}}^{(i)},\ t \subseteq q \text{ and} \\ \sum_{j=1}^{i} |D(n_j)| + |q| < 2^n - (|V_T| - i - 1) \end{array} \right\} - \bigcup_{j=1}^{i} D(n_j). \tag{2}$$

The inequality in equation (2) is to ensure that after the allocation of q to n_{i+1}, there is a sufficient number of nodes in the Q_n to be assigned to the remaining task nodes. From equation (2), it is easy to see that *both* (i) more subcubes in $E_{n_{i+1}}^{(i)}$ *and* (ii) subcubes of smaller dimensions in $E_{n_{i+1}}^{(i)}$ will allow for more freedom in allocating a required subcube to n_{i+1}. This in turn implies that the sets of essential subcubes of unassigned nodes can be used in determining the heuristic value of the node in the search tree associated with the partial mapping made thus far.

Suppose that a node p in the search tree corresponds to the allocation of a subcube q to the task node n_i. Then, $A_{n_i}^{(i)}$ can be determined by the method introduced in the proof of Theorem 6, and the sets $A_{n_k}^{(i)}$, $1 \leqq k < i$, can be updated from their predecessors by equation (3) below. They are in turn used to determine $E_{n_k}^{(i)}$, $k > i$, by using equation (1).

(3) $$A_{n_k}^{(i)} = A_{n_k}^{(i-1)} - \{q\}, \qquad 1 \leqq k < i.$$

Combining all the results and findings discussed thus far, a heuristic function for each node p in the search tree can be contructed as follows:

(4) $$f(p) = g(p) + h(p) \text{ where } g(p) = i2^n,$$
$$h(p) = \sum_{i+1}^{|V_T|} V(E_{n_k}^{(i)}), \quad \text{and} \quad V(E_{n_k}^{(i)}) = \sum_{t \in E_{n_k}^{(i)}} \frac{1}{2^{|t|}}.$$

Note that the heuristic value (or h-value) of a node p is defined so that *both* more subcubes in $E_{n_k}^{(i)}$ *and* subcubes of smaller dimensions in $E_{n_k}^{(i)}$, $k > i$, will result in a larger h-value of p. Applying the above heuristic function to the A^* search algorithm, we propose the following RS-embedding algorithm.

ALGORITHM RS-embedding $(G_T, k)/*$. This algorithm determines the existence of an RS embedding from a task graph G_T into a Q_k.*/

Step (1). Without loss of generality, let the list OPEN be $\{00\cdots0, 0\cdots0*, \cdots, 0*\cdots*\}$ consisting of k strings of length $k-1$ each. Check the validity of these nodes by using Theorem 6. Compute equations (1), (3), and (4) for nodes in the list OPEN.

Step (2). If OPEN $= \varnothing$, report *false* and exit. Determine the node p with the maximal f-value from the list OPEN. Remove it from OPEN and put it into the list CLOSE. If node p is associated with the allocation of the last node, report *true* and exit.

Step (3). Determine the successors of p by equation (2). Check the validity of successors by using Theorem 6, evaluate equations (1), (3), and (4) for valid nodes, and put these nodes in the list OPEN.

Step (4). Go to Step 2.

According to the formulation of the heuristic function, the h-value of any node in the search tree must be less than 2^n. This means that our heuristic function satisfies the monotone restriction [Nil80]. In other words, the goal nodes whose distances from the root node are $|V_T|$ have the maximal f-value. Thus, if there exist goal nodes in the search tree, then one of them should be reached in a finite number of steps. Note that there may be more than one goal node in the search tree. However, we are concerned only with the existence of such nodes, rather than the number of such nodes in the search tree.

Using the RS embedding algorithm, we can determine the existence of an RS embedding of a given graph into a cube. For some graphs whose weak cubical dimensions are in a narrow range, a linear search algorithm is suggested as follows. Since an unsuccessful search usually involves more computational costs than a successful one, the linear search algorithm is designed to perform a top-down search for the weak cubical dimension of a graph. Let *ub* and *lb* be, respectively, the upper and lower bounds of the weak cubical dimension of the graph determined by the mathematical properties in § 3. Using a linear search, the expected number of times to execute the RS embedding algorithm is $(ub-lb+1)/2$, containing $(ub-lb-1)/2$ successful searches and one unsuccessful search.

On the other hand, the bounds for the weak cubical dimension of some other graphs may be quite loose, making any linear search algorithm inefficient. For those graphs, a binary search algorithm is suggested. Then, the expected number of times used to execute the RS embedding algorithm becomes $\lceil \log_2 (ub-lb+1)\rceil$, in which successful and unsuccessful searches have the same likelihood of occurrence.

5. Examples. In this section, examples are presented to illustrate the application of the results developed in § 3 and the execution of the algorithms proposed in § 4.

Example 1. Consider the example graph G shown in Fig. 6. From Corollary 2.2, we have $\lceil \log_2 6\rceil = 3 \leqq \text{wd}(G) \leqq 6-1=5$. Moreover, from Corollary 6.1 we get $\text{wd}(G) \neq 3$. Let $V_1=\{n_1, n_2, n_5, n_6\}$ and $V_2=\{n_3, n_4\}$. Denote the induced subgraph with the node in V_j by $G_{I_j}, j=1,2$. Clearly, $G_{I_1}=C_4$ and $G_{I_2}=P_2$. Thus, from Corollary 4.1 we obtain $\text{wd}(G) \leqq \text{wd}(C_4)+\text{wd}(P_2)+1=2+1+1$. From the above results, we get $\text{wd}(G)=4$.

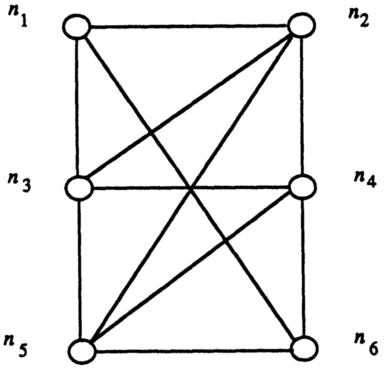
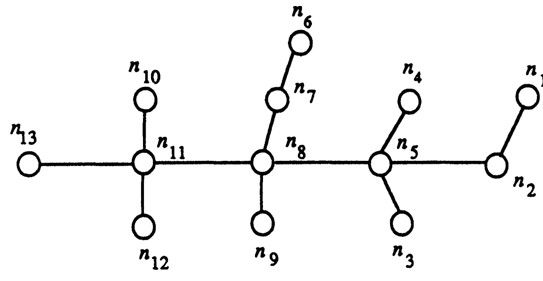


FIG. 6. *An example graph G.*

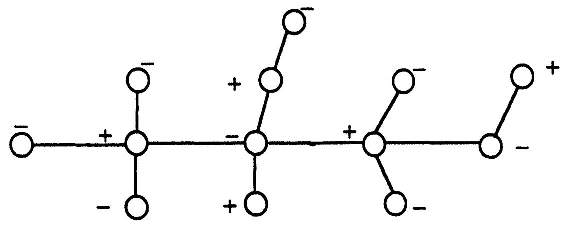
Example 2. Consider the task tree shown in Fig. 7(a). Using A_1, we obtained a labeled tree as shown in Fig. 7(b), and then derived an addressing scheme with the length $\lceil \log_2 5\rceil + \lceil \log_2 8\rceil + 1 = 7$. For example, under this addressing scheme the assigned address of n_5, the second node labeled with +, is 0010*** and that of n_8, the fourth node labeled with −, is 1***100.

The application of A_2 to the tree in Fig. 7(a) can be described by Fig. 8. The tree with the weight of each edge specified is given in Fig. 8(a), and the operations of A_2 are illustrated by the binary tree in Fig. 8(b). Each internal node in Fig. 8(b) has two children that are the disjoint trees resulting from the removal of its centroid edge. For example, T_1 and T_2 are the two attached trees of the edge (n_5, n_8), while n_8 is in T_1 and n_5 in T_2. Using A_2, we get $A_2(T_3)=3, A_2(T_4)=2, A_2(T_5)=2, A_2(T_6)=1, A_2(T_1)=4, A_2(T_2)=3$, and $A_2(T)=5$.

Example 3. Consider the example graph $G=(V, E)$ of Fig. 3. Again, we have $\text{wd}(G) \geqq \lceil \log_2 |V|\rceil = 3$ from Corollary 2.2. In addition, the induced subgraph of G with the node set $\{n_2, n_3, n_4, n_5, n_6\}$ is P_5 whose weak cubical dimension is 3. From Corollary 4.2, we get $3 \leqq \text{wd}(G) \leqq 4$. To determine $\text{wd}(G)$, we must apply the RS embedding algorithm.



(a) An example tree T.



(b) The labeling of the tree T.

FIG. 7. *An example of labeling a tree.*

Figure 9 shows the state-space search tree for a Q_3 to accommodate the task graph. Let L denote the list of occupied subcubes. By using the heuristic search algorithm in § 4.2, initially we get OPEN $=\{000, 00*, 0**\}$. Note that the allocation $(n_1 \leftarrow 000)$ and $(n_1 \leftarrow 0**)$ will be eliminated by Theorem 6 and equation (2), respectively. Thus, node A is the only node to be expanded. According to equations (1), (3), and (4), we have the following:

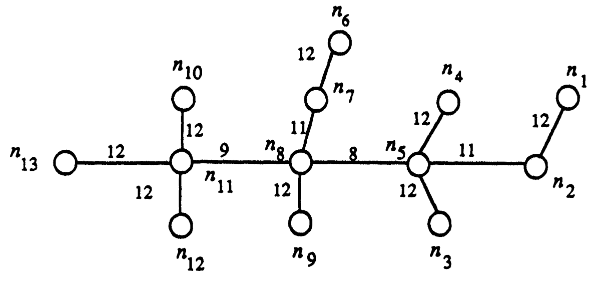
(1) Node $A(n_1 \leftarrow 00*)$: $A^{(1)}_{n_1}=E^{(1)}_{n_2}=E^{(1)}_{n_3}=E^{(1)}_{n_4}=E^{(1)}_{n_6}=\{010, 011, 100, 101\}$, $E^{(1)}_{n_5}=\{010, 011, 100, 101, 110, 111\}$, and $L=\{00*\}$. $g(A)=2^3$, $h(A)=4+4+4+4+6$, and $f(A)=30$.

Under the allocation $M^1=(n_1 \leftarrow 00*)$, we get Sp $(E^{(1)}_{n_2})=\{010, 011, 101, 100, 01*, 10*, *10, *11, 1*0, 1*1\}$ from equation (2). Due to the symmetry, only the computation for the nodes B, C, and D is shown below.

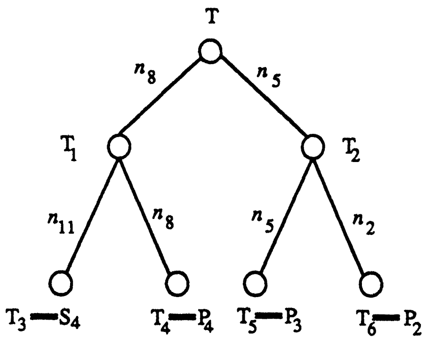
(2) Node B $(n_2 \leftarrow 010)$: $A^{(2)}_{n_1}=\{011, 100, 101\}$, $A^{(2)}_{n_2}=\{011, 110\}$, $E^{(2)}_{n_3}=\text{Rd}\,(A^{(2)}_{n_1} \odot A^{(2)}_{n_2})=\{011, 1*0\}$, $E^{(2)}_{n_4}=E^{(2)}_{n_6}=\{011, 100, 101\}$, $E^{(2)}_{n_5}=\{011, 110\}$, and $L=\{00*, 010\}$. $g(B)=2^3 2=16$, $h(B)=1\frac{1}{2}+3+2+3$, and $f(B)=25\frac{1}{2}$.

(3) Node C $(n_2 \leftarrow 01*)$: $A^{(2)}_{n_1}=\{100, 101\}$, $A^{(2)}_{n_2}=\{110, 111\}$, $E^{(2)}_{n_3}=\text{Rd}\,(A^{(2)}_{n_1} \odot A^{(2)}_{n_2})=\{1*0, 1*1\}$, $E^{(2)}_{n_4}=E^{(2)}_{n_6}=\{100, 101\}$, $E^{(2)}_{n_5}=\{110, 111\}$, and $L=\{00*, 01*\}$. $g(C)=2^3 2=16$, $h(C)=\frac{1}{2}+\frac{1}{2}+2+2+2$, and $f(C)=23$.

(4) Node D $(n_2 \leftarrow *10)$: $A^{(2)}_{n_1}=\{011, 100, 101\}$, $A^{(2)}_{n_2}=\{100, 011, 111\}$, $E^{(2)}_{n_3}=\{011, 100, 1*1\}$, $E^{(2)}_{n_4}=E^{(2)}_{n_6}=\{101, 100, 011\}$, $E^{(2)}_{n_5}=\{100, 011, 111\}$, and $L=\{00*, *10\}$. $g(D)=2^3 2=16$, $h(D)=2\frac{1}{2}+3+3+3$, and $f(D)=27\frac{1}{2}$.



(a) A tree with the weight of each edge specified.



(b) The determination of the subcube required for a tree using divide and conquer.

FIG. 8. *An application example for Corollary* 3.2.

Since node D has the maximal f-value among the three nodes, D is now the next node to be expanded. Note that under the partial mapping $M^2=(n_1\leftarrow 00*,\ n_2\leftarrow *10)$, Sp $(E^{(2)}_{n_3})=\{011, 100\}$. Then, using the same procedure, the remaining computation for the heuristic search algorithm is given below.

(5) Node E $(n_3\leftarrow 011)$: $g(E)=2^3 3=24$, $h(E)=\frac{1}{2}+2+2$ and $f(E)=28\frac{1}{2}$.

(6) Node F $(n_3\leftarrow 100)$: $g(F)=2^3 3=24$, $h(F)=1+2+2$ and $f(F)=29$.

Node F is now the next node to be expanded. Using the same procedure, it is easy to verify that all the children of nodes F and E can be pruned, and node B becomes the next node to be expanded, since $f(B)>f(C)$. Thus, $M^2=(n_1\leftarrow 00*,\ n_2\leftarrow 010)$, leading to the following results: $f(G)=27\frac{1}{2}$, $f(H)=28\frac{1}{2}$, and $f(I)=28\frac{1}{2}$.

Now, node H is to be expanded. Continuing the same procedure, we obtain the following results: $f(J)=34$, $f(K)=41$, and $f(L)=48$.

Since the node L is associated with the allocation of the last node, *true* will be reported, meaning that an RS embedding of G into Q_3 has been found and wd $(G)=3$. It is easy to see that the proposed heuristic function plays an important role in guiding and, thus, speeding up the state-space search. Use of the f-value of a node as an indication of the likelihood for the node to lead to a successful mapping results in a significant improvement over a blind search. However, as the size of the task graph increases, large amounts of computation will be required for the node expansion of the heuristic search, and the necessity of applying this state search algorithm to every graph calls for an optimization in some sense. For example, depending on the system's objective function, one can strike a compromise between the system utilization and the computational cost.

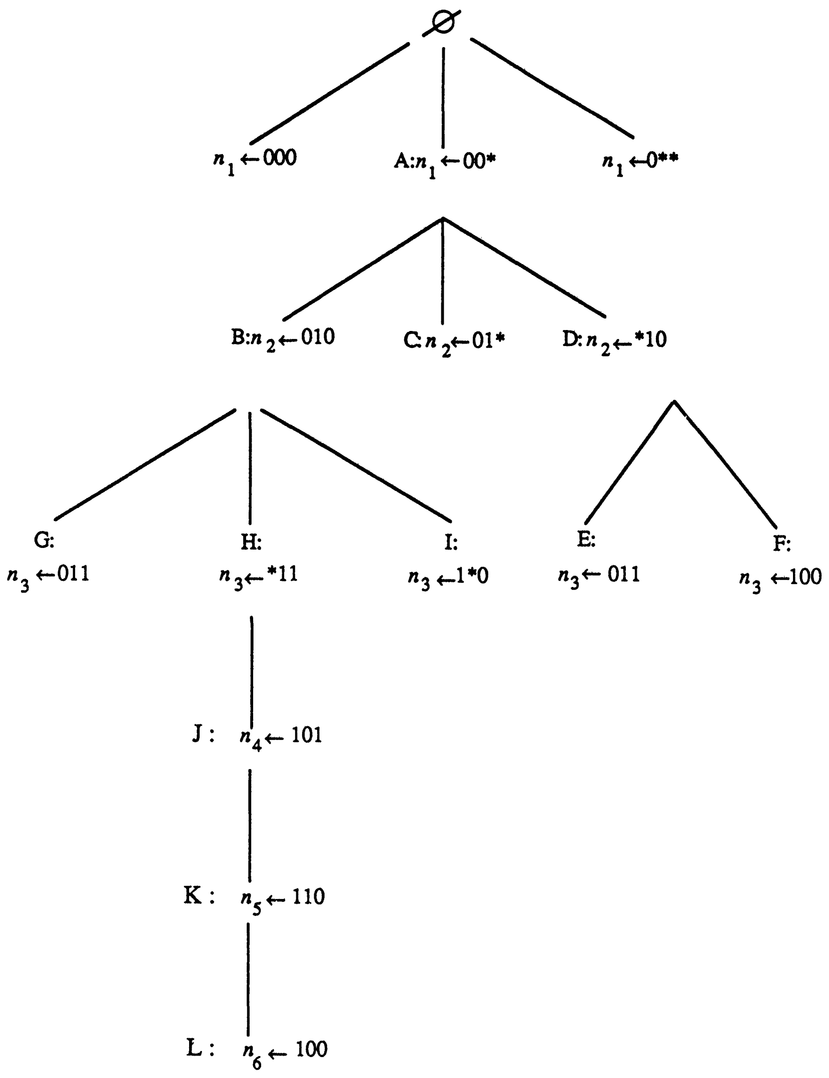


FIG. 9. *Part of the search tree.*

6. Discussion and conclusion. We have proposed and investigated a new type of embedding, called the RS embedding, that was motivated by the problem of allocating tasks in a hypercube multicomputer. Several mathematical properties for the weak cubical dimension have been derived that are not only applied to develop fast algorithms for the RS embedding, but also used to guide the heuristic search for an RS embedding.

The problem studied in this paper can be generalized by considering both the computation load of each module and the communication load between modules in a task graph. The task graph can then be represented by a labeled graph. The number assigned to a node of the graph denotes the dimension of a subcube required for the corresponding module to perform the computation load of the module. The number assigned to an edge of the task graph represents the required number of communication links between the two subcubes assigned to the two task nodes incident to this edge to provide enough communication capacity between them. Note that two adjacent subcubes could have different numbers of connecting links. For example, $*10*$ and $00*0$ are connected by a link (0100, 0000), and $010*$ and $00**$ are connected by two links, (0100, 0000) and (0101, 0001). Thus, the constraint treated in this paper is a special case of the generalized version, since one is assigned to every node and every edge of the task graph.

Clearly, the inclusion of computation and communication loads of modules increases the number of constraints to meet, and thus, makes the RS embedding more realistic but complicated.

REFERENCES

[BGK72] L. H. Brandenburg, B. Gopinath, and R. P. Kurshan, *On the addressing problem of loop switching*, Bell System Tech. J., 51 (1972), pp. 1445-1469.

[ChS87] M.-S. Chen and K. G. Shin, *Processor allocation in an N-cube multiprocessor using Gray codes*, IEEE Trans. Comput., 36 (1987), pp. 1396-1407.

[Cor85] N. Corp., NCUBE/ten: an overview, November 1985.

[CKV87] G. Cybenko, D. W. Krumme, and K. N. Venkataraman, *Fixed hypercube embedding*, Inform. Process. Lett., 25 (1987), pp. 35-39.

[Fir65] V. V. Firsov, *On isometric embedding of a graph into a Boolean cube*, Cybernetics, 1 (1965), pp. 112-113.

[GaG75] M. R. Garey and R. L. Graham, *On cubical graphs*, J. Combin. Theory Ser. B, 18 (1975), pp. 84-95.

[GaJ79] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, San Francisco, CA, 1979.

[GrP71] R. L. Graham and H. O. Pollak, *On the addressing problem for loop switching*, Bell System Tech. J., 50 (1971), pp. 2495-2519.

[GrP72] ———, *On embedding graph in squashed cubes*, Springer Lecture Notes Math., 303 (1972), pp. 99-110.

[Har69] F. Harary, *Graph Theory*, Addison-Wesley, MA, 1969.

[Har86] ———, *The Topological Cubical Dimension of a Graph*, Lecture Notes on the first Japan Conference on Graph Theory and Applications, Hakone, Japan, June 3, 1986.

[Har80] J. Hartman, *On homeomorphic embeddings of $K_{m,n}$ in the cube*, Canad. J. Math., 32 (1980), pp. 644-652.

[HaM72] I. Havel and J. Moravek, *B-valuations of graphs*, Czech. Math. J., 22 (1972), pp. 338-351.

[KVC85] D. W. Krumme, K. N. Venkataraman, and G. Cybenko, *Hypercube embedding is* NP-*complete*, Proc. First Hypercube Conference, Knoxville, TN, August 1985, pp. 148-157.

[Nil80] N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.

[Pie72] J. R. Pierce, *Network for block switching of data*, Bell System Tech. J., 51 (1972), pp. 1133-1145.

[Sei85] C. L. Seitz, *The cosmic cube*, Commun. Assoc. Comput. Mach., 28 (1985), pp. 22-33.

[ShT85] C. C. Shen and W. H. Tsai, *A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion*, IEEE Trans. Comput., 34 (1985), pp. 197-203.

[Val82] L. G. Valiant, *A scheme for fast parallel communication*, SIAM J. Comput., 11 (1982), pp. 350-361.

[Wil87] P. Wiley, *A parallel architecture comes of age at last*, IEEE Spectrum, 24 (1987), pp. 46-50.

[Win83] P. M. Winkler, *Proof of the squashed cube conjecture*, Combinatorica, 3 (1983), pp. 135-139.

[Yao78] A. C. Yao, *On the loop switching addressing problem*, SIAM J. Comput., 7 (1978), pp. 515-523.