計算機アーキテクチャ 81－1
（1990．3．23）

有界グラフに関する最適マッピング問題の一般的な解法

アィサム.A.ハミド.　白鳥 則郎　　野口 正一

東北大学電気通信研究所,

〒980 仙台市片平2-1-1

並列アルゴリズムの性能は、対象とする並列システムの相互接続網のトポロジに依存する。並列システムの要素間のコミュニケーションは並列アルゴリズムに応じて変化するので、再配置可能な接続網が、そのような要求を満足するために効果的となる。本稿では、与えられた問題に対して最適となる相互接続網をいかに生成するかについて述べる。具体的には、入力としてグラフ表現された問題が与えられた時、適合する相互接続網のトポロジを出力するアルゴリズムを構成する。生成されたトポロジは最適化のために分析し、修正が試みられる。

A genetral Solution for the Optimal Mapping Problem on Bounded Degree Graph

Issam A. Hamid　Norio Shiratori　Shoichi Noguchi

Research Institute of Electrical Communication, Tohoku University

2-1-1, Katahira, Aoba-ku, Sendai shi, 980, Japan.

The performance of a parallel algorithms depends highly on the interconnection topology of the target parallel architecture. The reconfigurable interconnection network can efficiently support application algorithms of different communication requirements whose the communication pattern change from one algorithm to another. In this research, we have presented an algorithm which can generate a network topology which is optimal due to the communication pattern of a given task. This algorithm generate a task graph with optimal topology that closely matches the optimal (*i.e.*, high cardinality) input task graph. The performance of these algorithms have been analyzed for proof of optimality.

1-Introduction

Parallel computers have a big role in executing complex real time tasks in efficient execution time. But an extensive need is necessary to map the parallel tasks on the parallel machine topology efficiently without increasing the system communication overhead for message handling between these parallel tasks.

Suppose we have at hand a computational task and to perform it we are also given a suitable algorithm to be used. In addition assuming this algorithm has been divided into some number of subtasks in away that some number of subtasks may be run concurrently. As these subtasks run (on the different processors) they may need to exchange data amongst themselves, and it may not be possible to start the execution of one subtask before the completion of some other subtasks. If we can estimate the extent of this data exchange, *i.e.*, communication between subtasks, and their data dependencies, then we can estimate the general behavior of the algorithm on the system.

The scheduling problem is how does one find an optimum allotment of these subtasks among the processors so that the maximum possible speedup (or equivalently some other performance parameters) may be achieved.

Numerous authors have considered the problem of allocating non interacting tasks in a distributed environment. They have not considered the effects of data dependency or communication amongst these tasks.

Several approaches to task assignment in distributed computing system have been suggested. They can roughly classified into *three* categories, namely graph theoretic, mathematical programming and heuristic methods.

The graph theoretic method uses a graph to represent a task and applies the minimal cut algorithm to the graph to get the task assignment with interprocess communication.

Bokhari [Bo80] showed that when assigning tasks to the processors, pairs of tasks or module that need to communicate with each other should be placed, if possible on processors that are directly connected. The property that characterize such an assignment is known in graph theory as the *cardinality* of the mapping, which is the number of edges in the problem graph that fall on the links in the system graph. In this case, all the problem edges are considered identical.

2- The mapping problem in graph theory

G_P, represents the undirected graph related to tasks problem such that: $G_P=\langle V_P, E_P\rangle$, where V_P, represents the set of modules (vertices) representing the subtasks of task graph. E_P denotes the communication requirements between these subtasks.

$G_S=\langle V_S, E_S\rangle$, represents the system configuration of the parallel model of machine architecture, where V_S, denotes the set of processors, and E_S, is the interconnection pattern between these processors. E_S, in our model is not statical because it is assumed to be changeable with G_P' configuration such that to make the requirements of tasks graph comply with the new processors topology represented in our model by Benes interconnection network which is capable to support $N!$ bijection mapping, where N is the number of input terminals t the network[Ha88]. If we have M_P which denotes one-to-one mapping between V_P onto V_S, then the quality of this mapping is determined by the number of edges, E_P that fall on processor edges, E_S. This number is called *Cardinality*[Bo80] such that;
$|M_P| = 1/2 \Sigma G_P(x,y) * G_S(M_P(x),M_P(y))$.

Cardinality refers to the number of data transfers of the communicating subtasks falling upon physically or directly connected processors. In other words it gives a number indicating how well the algorithm has mapped on the architecture in a physical sense, It has been shown that the mapping problem falls into a class of intractable problems called NP-complete[Ah74].

Bokhari[Bo80] has assumed a fix topology, therefore his problem is to map G_P onto G_S such that to have a task graph be mapped on directly connected processors, *i.e.*, of maximum cardinality. In our model we assume that; G_S is changeable according to the type of the permutations representing the communication demands between processors. Such type of mapping is supported by Benes IN whose control algorithm has complexity of $O(k \log_2 N)$ where $k \leqq \log_2 N$, $N=2^n$ [Ha89]. Hence, the degree of every vertex in G_S is bounded due to constant fanout of each processor connected to Benes IN. Therefore the problem is to map G_S such that the resulted mapping has a bounded degree per vertex.

Spanning subgraph is a subgraph containing all of the nodes of a graph G, which is connected if every pair of nodes are joined by a path. A maximal connected subgraph of G is called a connected components of G. A *factor* of a graph is a subgraph whose vertex set is that of the whole graph. If every vertex of a *factor* has degree γ then we call it an γ-factor. $d_G(x)$ is the *degree* of node x in G and represents the number of edges in $E(G)$ that are incident with x.

A graph is regular of degree γ if all vertices of G is γ. If a,b, are integers such that $0 \leqq a \leqq b$, then a G graph is called an $[a,b]$-graph if $a \leqq d_G(x)$ for all $x \in V(G)$. A factor ϕ of G is called an $[a,b]$-factor if $a \leqq d_\phi(x) \leqq b$, for all $x \in V(\phi)$. If $a=0$, and $b=\gamma$, we call such a factor a deficient γ-factor. The deficiency δ, of a deficient γ-factor ϕ is; $\delta = \Sigma \gamma - d_\phi(x)$. A maximal deficient γ-factor is an γ-factor with the maximum number of edges.

Spanning subgraph ϕ of a graph is called $[a,b]$-*factor* of G if $a \leqq d_{\phi}(x) \leqq b$ for all $x \in V(\phi)$. A *bridge* of a graph G is an edge whose removal increase the number of components.

We call the assignment of task modules to processors as; a *mapping*.

The problem of *maximizing* the number of pairs of communicating modules that fall on pairs of directly connected processors is called the *mapping problem*[Bo80].

3- The algorithm problem representation

Hence the problem can be stated as follows:

A connected and undirected graph $G_S = \{V(G_S), E(G_S)\}$ with N nodes. A set of N nodes $V(P)$ and a bijective function g: $V(G) \rightarrow V(P)$; and an integer $\gamma \geqq 2$.

Find a function c: $V(P) \times V(P) \rightarrow \{0,1\}$, such that $P = \{V(P), E(P)\}$, $E(P) = \{(u,v);\ c(u,v) = 1\}$ is a connected graph with $d_P(v) \leqq \gamma$ for all $v \in V(P)$ and cardinality of $= |\{(u,v), (u,v) \in E(G)$ and $(g(u), g(v)) \in E(p)\}|$ is maximized.

G is a task graph that represents the algorithm to be executed on the system. G is a graph$= \{V(G), E(G)\}$, $V(G)$ denotes the graph nodes, and $E(G)$ denotes the graph edges. Nodes in the graph correspond to individual tasks and an edge between two nodes signifies that communication occurs between these two tasks. The processor system is also represented as a graph with nodes corresponding to processors and edges corresponding to communication links. The problem is to find the set of edges that maximizes the number of pairs of intercommuicating tasks that fall on pairs of directly connected processors. At the same time because of the limited number of links each processor can access the degree of each node must be bound. The resultant graph must also be connected since a message sent to processor that is not directly connected to a source processor, must be forwarded through intermediate processors. graphs meeting these conditions are called γ-degree constrained connected graph. The problem is related to such graphs.

Most of the above methods adopt some types of cost function to evaluate the effectiveness of task assignment algorithm. The most commonly used cost function is defined as the sum of the interprocess communication cost and the processing cost. But these two types of cost are measured in different units and it is difficult to give a reasonable meaning to the resulting cost summation.

Then the problem can be represented as to maximize the cardinality such that if the edge $(i,j) \in G_P$, and $(M_P(i), M_P(j)) \in G_S$, then G_P should be maximized.

Spanning subgraph is a subgraph containing all of the nodes of G.

A graph is connected if every pair of nodes are joined by a path. A maximal connected subgraph of G is called a connected component of G. A factor of G is a spanning subgraph of G that is not totally disconnected.

$d_G(x)$ degree of node x in G represents the number of edges in $E(G)$ that are incident with x. A graph G is regular of degree γ or simply γ-regular if $d_G(x) = \gamma$ for all $x \in V(G)$.

The graph follows the above conditions may named as γ-degree constrained connected graph, which is NP-complete. But there is a polynomial time solution for such graphs if the restriction that the resultant graph be connected is removed[Ah74].

This is accomplished by observing that the number of edges in a maximal deficient γ-factor of (G) gives an upper bound on cardinality.

Such a maximal deficient γ-factor can be found using a polynomial time graph matching algorithm[Ga83].

Using this algorithm we develop an algorithm for finding suboptimal γ-degree constrained connected graph. After finding a maximal deficient γ-factor the components of the factor are connected using heuristics.

These heuristics which are developed and analyzed in the following, must guarantee that discontinuties are not introduced when deleting edges between nodes within a component.

A *bridge* of a graph is an edge whose removal; increases the number of components.

(1) We note that if a component consists only of bridge edges it is a tree and thus has at least two nodes with degree one.

Hence (2) every regular graph with degree greater than 1 has at least on non bridge edge.

These two properties are used in the coming Th-1 to show that the algorithm is correct.

The idea of algorithm methodology :

(1) Find a maximal γ-factor of G using a polynomial time graph matching algorithm of [Ga83]

(2) Use heuristics such that; the connected components of the factor are then clustered into larger components by adding edges between nodes with degree less than r until no more components can be combined in this manner. The resulting components includes at most one nonregular component and all other components are regular.

(3) The algorithm next selects the non regular component if it exists or an arbitrary component otherwise. Then it selects another component such that there is an edge in the original graph G between the two selected components.

Accordingly the function **Join1** which is shown in Fig.1, tries to merge these two components using the edge in G .

If it is not possible to use that edge function **Join1** merges the two components by adding an edge between the different nodes.

After the first two components are merged into a component named as ω the remaining components

are merged into ω one at a time, using the function **Join2**, shown in Fig. 2.

Note that after the execution of the synthesis algorithm there may still exist nodes with degree less than γ.

Edges can be added to these nodes however they do not increase the cardinality.

In the following we have proved the synthesized algorithm should be correct.

(1) The fact that every node in the resultant graph has degree less than or equal to γ, is true, is trivial since the algorithm adds edges between nodes with degree less than γ by removing edges first if necessary.

(2) To show that the algorithm terminates is true, note that the clustering step shown in Fig. 3, terminates since the number of joined components in a maximal deficient γ-factor ϕ is finite and each iteration reduces the number of nonregular components in ϕ by one.

Similarly, the connection step terminates since there is only a finite number of components in ϕ, after gathering and one component is removed after each call to **Join2**.

To show the resultant graph is connected is true, note that functions **Join1** and **Join2** remove only nonbridge edges and thus they never partition connected components.

Function **Join1** connects ω and ω' since ω' is a regular component and thus has a nonbridge edge and ω has a node with degree less than γ or has a nonbridge edge. Thus an edge can always be added to connect ω and ω'.

Each time a regular component is connected to ω the resulting component is non regular.

Thus function **Join2** also always successfully connects ω and a regular component ω'. Therefore the resulted graph is connected.

The next step is to show that the algorithm has a time complexity of $O(N\ |E(G)|)$, N is the number of nodes.

By using the algorithm of Gabow[Ga83] a maximal deficient γ-factor can be found in $O(\sqrt{(N\gamma)}\ |E(G)|)$ time. Locating the connected components can be done in $O(|E(G)|)$ time [Ah74].

Since there are at most N nonregular components and since each iteration in the clustering step can be performed in $O(N^2)$ time.

Determing whether all edges incident with a node are bridges, can be done in $O(\gamma)$ time and locating a nonbridge edge can be done in $O(|E(G)|)$ time.

Thus both **Join1** and **Join2** have a maximum running time of $O(|E(G)|)$.

Since there are at most N components in ϕ the connection step can be performed in $O(N\ |E(G)|)$ time. Therefore the total running time of the algorithm is;

$O(\sqrt{(N\gamma)}\ |E(G)|+N^2+N|E(G)| = O(N|E(G)|)$.

4- Analysis of the synthesized algorithm

From now on we will discuss the analysis of the synthesized algorithm from the worst case analysis point view.

How close the topology comes to the optimum topology. Since the optimum cardinality is not known, then an exact answer is impossible.

However the cardinality of the maximal deficient γ-factor is an upper bound on the optimum cardinality. In addition, the loss in cardinality due to connecting the connected components of the factor can be bounded.

Hence, we can determine how close the topologies generated by our algorithm come to realizing the cardinality of an optimum topology in the worst case.

We first bound the maximum reduction in cardinality due to the connection algorithm and then show that the bound is tight.

Lemma 1

Let G be a non empty γ-regular graph when $\gamma \geq 2$, suppose G has a node u such that all edges in G incident with u are bridges then γ must be greater than 2 and G has more than $3(\gamma+1)$ nodes.

proof:

If $\gamma=2$ then G is a cycle and thus it cannot have a bridge. Therefore $\gamma>2$. Since u is incident with only bridges and G is regular u must be adjacent to γ distinct γ-regular component. Since the minimal size of any γ-regular component is $(\gamma+1)$ G has at least $\gamma(\gamma+1)+1$ nodes which is greater than $3(\gamma+1)$. ■

Theorem-1

Given a deficient γ-factor ϕ the loss in cardinality due to the connection step of the algorithm is less than $\lfloor N/(\gamma+1) \rfloor$

proof:

Since the minimum size of a regular component is $(\gamma+1)$ and since there is at most one nonregular component after the clustering step there are at most $\lfloor N/(\gamma+1)+1 \rfloor$ components in ϕ immediately after the clustering step.

The component ω after **Join1** is nonregular throughout each iteration of the second while loop. Since ω is nonregular each remaining regular component can be connected to ω using the function **Join2** with cardinality loss at most one.

Thus it suffices to show that if the cardinality loss due to **Join1** is k where $k=0,1$ or 2, then there are less than $\lfloor N/(\gamma+1) \rfloor - k$ regular components left in ϕ after **Join1**. Let (u_0,v_0) where $u_0 \in \omega$ and $v_0 \in \omega'$ be the edge in G that is passed as an argument to the function **Join1**. Let (u,v) where $u \in \omega$ and $v \in \omega'$ be the edge added to connect ω and ω'.

When $k=0$: Since the regular component ω' connected to ω using **Join1** has at least $(\gamma+1)$ nodes at most $\lfloor (N-(\gamma+1))/(\gamma+1) \rfloor = \lfloor N/(\gamma+1) \rfloor - 1$ regular components are in ϕ after **Join1**.

When $k=1$:

Since ω' is regular an edge must be removed from ω'. Thus $k=1$ implies that either ① $(u,v) \in E(G)$ and an edge incident with u is removed from ω or ② $(u,v) \notin E(G)$ and the degree of u in ω before **Join1** is less than γ.

In ① the fact that an edge incident with u needs to be deleted implies that $d_u(u)=\gamma$ and thus ω has at least $(\gamma+1)$ nodes. Since ω' also has at least $(\gamma+1)$ nodes the resultant component must have at least $2(\gamma+1)$ nodes. Thus at most $\lfloor (N-2(\gamma+1)) / (\gamma+1) \rfloor = \lfloor N/(\gamma+1) \rfloor - 2$, regular component are left in ϕ after **Join1**.

In ② since **Join1** tries to use the edge $(u_0,v_0) \in E(G)$ to connect ω and ω' $(u,v) \notin E(G)$ means that all of the edges incident with either $u_0 \in \omega$ or $v_0 \in \omega'$ are bridges.

The former implies that ω' has at least $3(\gamma+1)$ nodes from lemma-1.
In either case the resultant component has at least $2(\gamma+1)$ nodes.

The case for $k=2$. Cardinality loss equals *two* only if $(u,v) \notin E(G)$ and edges are removed from both ω and ω'. Edge $(u,v) \notin E(G)$ means either $u_0 \neq u$ or $v_0 \neq v$ or both.

If $u_0 \neq u$ then an edge is removed from ω implies that ω is regular and all edges incident with u are bridges. Thus ω has at least $3(\gamma+1)$ nodes (lemma-1) similarly $v_0 \neq v$ implies that ω' has at least $3(\gamma+1)$ nodes . Thus at most $\lfloor N-3(\gamma+1) / (\gamma+1) \rfloor = \lfloor N/(\gamma+1) \rfloor - 3$, regular component are left in ϕ after **Join1**. ■

We should note that the loss in cardinality due to the connection step of the algorithm is less than or equal to the number of components in ϕ after clustering. This is because the function **Join1** connects two components and reduces the cardinality by at most two. Also, each call to function **Join2** reduces their cardinality by at most one for each component.

Therefore the algorithm will performs better on the average if the clustering minimize the number of components.

One way to achieve this is to sort the components of ϕ in decreasing order of deficiency and then cluster components with the largest deficiencies first.

There is another consequence result comes from Th-1. Such that for a given task graph G, if $\omega_{Opt.}(G,\gamma)$ is the cardinality of an optimum γ-bounded connected graph, and $\omega_{Ach.}(G,\gamma)$ is the achieved cardinality of the given G and γ, then $\omega_{Ach.}(G,\gamma) > \omega_{Opt.}(G,\gamma) - \lfloor N/(\gamma+1) \rfloor$. This follows from Th-1 because $\omega_{Opt.}(G,\gamma)$ is less than or equal to the number of edges in a maximal deficient γ-factor.

Although our connection algorithm is not optimal we can show that the worst case performance of our algorithm equals that of any algorithm. That is for every N and γ there exists a graph with a maximum deficient γ-factor such that the connection of this factor using any algorithm reduces the cardinality by $\lfloor N/(\gamma+1) \rfloor - 1$. We now explain how to construct such a graph given N and γ. Letting $m=\lfloor N/(\gamma+1) \rfloor$ we first construct m γ-regular graphs each with $(\gamma+1)$ nodes.

Then let G' be the union of these regular graphs and a connected graph called ω with $N-m(\gamma+1)$ nodes. We note that ω may be empty however if ω exists it contains at least one node v such that $d_{G'}(v)<\gamma$. Choose one node from each of the m regular components and label them as uu_i, $0 \leqq i < m-1$. Let $E_u=\{(u_0,v_i)|\ 1 \leqq i \leqq m-1\}$. $G=\{v(G'), E(G') \cup E_u \cup (u_0,v)\}$ where (u_0,v) is omitted if ω is empty.

One maximal deficient γ-factor of G consists of the edges in the regular graphs and a maximal deficient γ-factor of ω. This factor has $\lfloor N/(\gamma+1) \rfloor + 1$ (or $\lfloor N/(\gamma+1) \rfloor$ if ω is empty) connected components.

The component with u_0 can be connected to ω without any loss in cardinality.

To connect the remaining $\lfloor N/((\gamma+1) \rfloor - 1$ regular components one edge has to be removed from each of them. Hence the total reduction in cardinality is $\lfloor N/(\gamma+1) \rfloor - 1$.

For graphs constructed as above we now show that any connection algorithm will reduce the cardinality by the bound given in Th-1.

First at least one edge to be removed from each regular component.

Second the only edges that increase the cardinality during the connection step are those incident with u_0.

Third an edge incident with u_0 cannot be used without first removing another edge incident with u_0. Hence the connection of each of the γ-regular components except the one with u_0 to another component will decrease the cardinality by at least one. We next bound the worst case cardinality of a maximal deficient γ-factor of [a,b]-graph.

There are *three* cases to consider depending on the values of a and b.

The *first* case is for $a \leqq \gamma$ and $b \leqq \gamma$. Here the cardinality obviously equals $|E(G)|$.

The *second* case is for $a>\gamma$, and $b>\gamma$.

The *third* case is for $a<\gamma$ and $b \geqq \gamma$.

The following four theorems analyze those such cases.

For these theorems a certain result has been used from [Bo74] as follows. In the case where G is an $(\gamma+1)$-regular graph a maximal deficient γ-factor can be generated by first finding a maximal 1-factor ϕ of G and then deleting the corresponding edges of ϕ from G. After deleting these edges G has $2|E(\phi)|$ nodes with degree γ and $N-2|E(\phi)|$ nodes with degree $\gamma+1$. From each of the nodes with degree $(\gamma+1)$ we delete an edge. The deletion of an edge from each node with degree $(\gamma+1)$ results in a deficient γ-factor. The deficiency of the resulting factor is less than $N-2\,|E(\phi)|$. Thus to bound the deficiency we need to know the smallest value that

$|E(\phi)|$ can be taken. This value has been studied thoroughly by [Bo74] due to the following lemma.

Lemma 2 [Bo74]

Define a function $m_e(N,\gamma,\lambda)$ on the set of all N-node γ-regular graphs with λ edge connectivity to be; $m_e(n,\gamma,\lambda) = min\ \{\ \beta_1(G)\colon |V(G)|=N,\ G$ is γ-regular and λ edge connected $\}$, where $\beta_1(G)$ is the number of edges in a maximal 1-factor of G.

Define, $m_c(\gamma)=\gamma'(\gamma^2-2\gamma)+2(\gamma-1)/\ (2(\gamma'(\gamma^2-2\gamma)+\gamma(\gamma-1)))$, where γ' is the least even integer not less than γ. If γ is odd then $m_e(N,\gamma,1)\geqq \lfloor m_c(\gamma)N-1/2\rfloor$ if γ is even of $\lambda>1$ when γ is odd then, $m_e(N,\gamma,\lambda)\geqq min\{\lfloor N/2\rfloor,\ N(\gamma\gamma'+2\lambda')\ /\ 2(\gamma(\gamma'+1)+\lambda'))\}$ where λ' is the least even (or odd) integer not less than λ if γ is even (or odd). ■

Th-2 is a consequence from the above lemma[Bo74] and given without prove.

Theorem-2

If G is an $(\gamma+1)$-regular graph then a maximal deficient γ-factor has deficiency

$\delta \leqq N-2\ m_e(N,\gamma+1,\lambda)$ ■

Several observation make it easier to predict the behavior of the equations in the preceding theorem.

The *first* observation is that the maximum possible number of edges in a 1-factor is $\lfloor N/2\rfloor$.

The *second* observation, that the number of edges in a 1-factor increases with the edge connectivity of the graph until it reaches the maximum value $\lfloor N/2\rfloor$. The bound on the number of edges also increase as γ increases.

As an example of the range of values that the equations in the above lemma, takes on, let us consider graphs with $\gamma=4$ and edge connectivity of one. Here, $2m_e(N,4,1)\ \geqq\ 5N/11$ and hence, the efficiency of an γ-factor of an $\gamma+1$-regular graph varies between 0 and approximately $N/11$.

The next lemma taken from [Ka83] and our Th-3 establish bounds for $[a,b]$-graphs where both a and b are greater than γ. In any case Th-3, holds only when b is close to a.

Lemma 3 [Ka83]

Let $0\leqq k\leqq a$, $0\leqq s$, and $1\leqq t$. If $(ks)\leqq(at)$, then an $[a,a+s]$-graph has a $[k,k+t]$-factor. ■

Theorem-3

Let G be an $[a,a+s]$-graph with N nodes if $s+0$ and $\gamma-1\leqq a/s$, then there exists a deficient γ-factor with deficiency $\delta\leqq N$.

proof:

Let G be a graph meeting the conditions of the theorem and let $t=1$ in Lemma-3, from which there exists an $[\gamma-1,\gamma]$-factor ϕ of G.

Let $R_1=\{x\colon\ x\in V(G)$ and $d_F(x)=\gamma-1\}$, and $R_2=\{x\colon x\in V(G)$ and $d_F(x)=\gamma\}$, δ is maximum when $|R_1|=N$, and $|R_2|=0$, which implies $\delta\leqq N$. ■

The next theorem bounds the least number of edges a maximal deficient γ-factor can have for an $[a,b]$-graph G where $a<\gamma$ and $b\geqq\gamma$.

Note that, since some nodes of G have degree less than γ, the most number of edges any maximal deficient γ-factor can have, is

$\Sigma\ min(d_G(x),\gamma)$.

Theorem-4

Suppose G is an $[a,a+s]$-graph where $a<\gamma$ and $a+s>\gamma$.

(1) If $\gamma-1\leqq\gamma/(a+s-\gamma)$, then there exits a deficient γ-factor ϕ such that

$\Sigma\ min(d_G(x),\gamma)-\Sigma\ d_\phi(x)\leqq N.$ (1)

(2) If $\gamma-1>\ \gamma/(a+s-\gamma)$, then there exists a deficient γ-factor ϕ such that;

$\Sigma d_\phi(x)\ \geqq\ \Sigma\ (\gamma-1-(d-d_G(u)))+(\gamma-1)\ |\ \{x\in V(G)\colon d_G(x)\geqq d\}|$ (2)

Where d is the least integer such that $(\gamma-1)\leqq d(/a+s-d)$ and $S=\{u\in V(G)\colon d_G(u)<d$ and $d-d_G(u)<\gamma\}$.

proof:

To show these bounds we embed G into an $[z,a+s]$-graph G' such that every node $x\in V(G)$ with $d_G(x)<z$ has degree z in G'.

The embedding of G into G' is accomplished by adding extra nodes to G, and then adding edges between these extra nodes and the nodes in G with degree less than z. We let z be γ for (1) and d for (2).

(1) $\gamma-1\leqq\gamma/(a+s-\gamma)$ embed G into an $[\gamma,\ a+s]$-graph G'. Let U be the set of edges in $E(G')$ that are not in $E(G)$. Since $\gamma-1\leqq\gamma/(a+s-\gamma)$, lemma-3 states that theγe exists an $[\gamma-1,\gamma]$-factor ϕ' of G'. Define $F=\{V(G),E(G)\cap E(\phi')\}$. For each node $x\in G$ with $d_G(x)<\gamma$ there are $\gamma-d_G(x)$ edges incident with x in U. Since $d_{\phi'}(x)\geqq\gamma-1$, we have $d_\phi(x)\geqq d_{G(x)}-1$. Furthermore for all nodes $x\in G$ with $d_G(x)\geqq\gamma$, we have $d_{F(x)}\geqq\gamma-1$. Therefore (1) follows.

(2) $\gamma-1>\gamma/(a+s-\gamma)$, embed G into an $[d,a+s]$-graph G' and define the set U as in part (1). Since d was chosen to satisfy $\gamma-1\leqq d/(a+s-d)$, lemma-3 states that there exists an $[\gamma-1,\gamma]$-factor ϕ' of G'.

Define $\phi=\{V(G),E(G)\cap E(\phi')\}$

Let x be a node in G with $d_G(x)<d$, if $d-d_G(x)\geqq\gamma$, then all of the edges of ϕ' incident with x can be in U.

Here, $d_F(x)\ \geqq\ \gamma-1-(d-d_G(u)))$, where $S=\{u\in V(G)\colon d_G(u)<d$ and $d-d_G(u)<\gamma\}$.

Furthermore for every node y such that $d_G(y)\geqq d$, we have $d_F(y)\geqq\gamma-1$. Therefore, (2) follows. ■

Theorem-5

If $\gamma-1\leqq\gamma/(a+s-\gamma)$, then the difference between the maximum number of edges a deficient γ-factor could have, and the number of edges it might actually have, is less than or equal to N, the same results as obtained in Th-5.

This is surprising since in Th-6 some nodes have degree $<\gamma$ whereas in Th-5 all nodes have degree greater than $\gamma-1$.

If both a and b are greater than γ but b is such that Th-3 does not apply we can bound the least number of edges in a maximal deficient γ-factor by using the embedding technique of Th-4. This bound is shown in the next theorem.

Theorem-6

Let G is an $[a,a+s]$-graph where $a\geqq\gamma$, $\gamma-1>a/s$, and if d is the least integer such that $\gamma-1\ \leqq\ d$

$/(a+s-d)$, then there exists a deficient γ-factor ϕ such that, $\Sigma d_\phi(G) \geqq \Sigma(\gamma-1-(d-d_G(u)))+(\gamma-1)|\{x \in V(G): d_G(x) \geqq d\}|$, where $s=\{u \in V(G): d_G(u)<d$ and $d-d_G(u)<\gamma\}$.

Proof:

Embed G into a $[d,a+s]$-graph G'. Since d was chosen to satisfy $\gamma-1 \leqq d/(a+s-d)$, lemma 3 states that G' has an $[\gamma-1,\gamma]$-factor. The rest of the proof is the same to the case that of $\gamma-1>\gamma/(a+s-\gamma)$, case of the previous theorem. ■

Eq-2 in theorem-5 and eq-3 in Th-6 show that if the difference between the minimal degree and maximal degree of node in a graph is large then the least number of edges a maximal deficient γ-factor can have is small.

Consider for example the star graph G on N nodes, It has a minimal degree of 1 and a maximal degree of $(N-1)$; the number of edges in a maximal deficient γ-factor of G is only γ.

The lower bound also depends on the relative number of nodes with minimal degree that is the more nodes with degree less than γ or d the smaller the bound becomes.

Conclusion

Algorithm given here generates a network topology for a given task graph. Its analysis has given the best, worst and average cases performance behavior of it.

On average case it was shown that the algorithm produces almost optimum topologies with respect to cardinality. The algorithm uses an arbitrary mapping between tasks and processors. One alternative mapping would entail sorting the tasks and processors according to its degree. A task would then be mapped to the processor with the same relative degree.

The use of reconfigurable networks allows one to select the network topology that closely matches the communication requirements of the algorithm to be executed.

References

[Ah74] Aho, A..V., et.al., The design and analysis of computer algorithms, Reading, MA:Addison-wesley, 1974.

[Bo81] Bokhari, S., "On the mapping problem," IEEE, Trans., Comput., vol. C-30., pp.207-214, March, 1981.

[Bo74] Bollobas, B., et.al., "Maximal matchings in graph with given minimal and maximal degree," in Proc. Cambridge Phil, Soc., vol., 79., 1974, pp.221-234.

[Ga83] Gabow, H.N., "An efficient reduction technique for a degree-constrained subgraph and bidirected networks flow problems," in Proc., 1983, *ACM*, Symp., theory Comput., 1983, pp.448-456.

[Ka83] Kano, M., et.al., "$[a,b]$-factors of graphs," Discrete Math., vol. 47. pp.113-116, 1983.

[Le88] Lee, I., et.al., "A synthesis algorithm for reconfigurable interconnection networks," *IEEE, Trans. Comput.*, vol.37, No.6, June, 1988, pp.691-699.

[Ha87] Hamid, I.A., et.al.,"A new fast control mechanism for rearrangeable interconnection network useful for supersystems," *Tran. IEICE*, vol. E70 No. 10, Oct. 1987, pp. 997-1008.

[Ha88] Hamid, I.A., et.al.,"A new controlling algorithm for Benes interconnection network without symmetry," *Trans., IEICE*, vol.E71, No.9, E71, Sep., 1988, pp.895-904.

[Ha89] Hamid, I.A., et.al.,"A new fastparallel computation model for setting Benes rearrangeable interconnection network," *Trans., IEICE*, vol.E72, No.4, April, 1989, pp.393-405.

[Wi81] Wittie, L., "Communication structures for large networks of mircocomputers," *IEEE, Trans., Comput.*, vol.C-30, pp.264-273, April, 1981.

Acknowledgment

This research has been done when the first author was with NEC, Tohoku software group under a visiting fellowship program sponsored by NEC. Much thanks go to NEC Tohoku software group, which has giving me such research environment. Also, much thanks go to the research group of the large computer center of Tohoku University, for their various assistance.

Function **Join1**(ω,ω': Component, *e*:edge):
/* ω may not be regular */
/* ω' is always regular */
/ * *e* is an edge between ω and ω' */
begin
let (i,j) be edge e, where $i \in V(\omega)$ and $j \in V(\omega')$
if $d_\omega(i)=\gamma$, and all (i,i') in ω are bridges then
if there is a node *ii* in ω such that $d_\omega(ii)<\gamma$ then
Let *i* be *ii* **else** choose a non-bridge edge (x,x') in ω
/* ω is regular */ Remove (x,x')
let *i* be *x*
end if
else
if $d_J(i)=\gamma$ **then** remove a non bridge edge (i,i') from ω
endif
ifall (j,j') in ω' are bridges **then** choose a non-bridge edge (y,y') in ω'
remove (y,y')
let *j* be *y*
else remove a non-bridge edge (j,j') from ω'
endif
connect ω and ω' by adding edge (i,j)
return the resulting component
end Join1

Function **Join2**(ω,ω':component): component
/* ω is always non regular */
/* ω' is always regular */
begin
find *i* in ω such that $d_\omega(i)<\gamma$
find a non-bridge edge(j,j') in ω'
remove (j,j') from ω'
connect ω and ω' by adding edge (i,j)
return the resulting component
end **Join2**

/* **Joining** Algorithm; connect components by adding and deleting edges */
if there is a non-regular component in *F then*
let ω be it *else*
Let ω be any component in *F*
endif
Remove J from *F*
Find ω' in *F* such that there is an edge *e* in *G* between *G* and *G'*.
Remove *J'* from *F*, *J'*:=**Join1**(ω , ω',*e*)
while there is a component in *F do*
choose any *J'* from *F*, *remove* ω' from *F*,
ω:=**Join2**(ω,ω')
endwhile

Input= a connected graph and an integer $\gamma \geqq 2$
Output= an γ-degree constrained connected graph.
Use a matching algorithm to find a maximal deficient γ-factor of *G*
Let *F* be a set of connected components in the factor.
/***Gathering** Algorithm--Gather components by adding Edges only */
While there is more than one-regular component in *F do*
Choose any two non-regular components ω_1 and ω_2 in *F*
Remove ω_1 and ω_2 from *F*
Find a node *x* in ω_1 such that $d_{\omega_1}(x)<\gamma$
Find a node *y* in ω_2 such that $d_{\omega_2}(x)<\gamma$
Connect ω_1 and ω_2 by adding edge(x,y)
Add the new component to *F*.
End While