Allocating Communication Channels to Parallel Tasks

Denis Barthou[a] Franco Gasperoni[b] and Uwe Schwiegelshohn[c]

March 14, 1997

Abstract

In this paper we study the problem of allocating communication channels to a number of processes or tasks that need to exchange data. Given a communication graph G we investigate the problem of determining the minimum channel capacity needed to execute G, the problem of finding an optimal schedule for G with minimum channel requirements and in the case where only a fixed channel capacity is available, the problem of finding a schedule which meets channel constraints and has minimum length. We provide a linear time algorithm for the first problem when G is a forest. However, both the second and third problems are shown to be NP-hard even when G is made up of chains of tasks or is a tree where every task needs the results of at most two other tasks. To somewhat compensate for such intractability we provide an efficient transformation of our communication problems to integer programming. This allows the use of off-the-shelf routines to find optimum solutions. More specifically given two integer parameters k and l and an arbitrary communication graph G we create an integer program polynomial in l and the size of G which has a feasible solution if and only if there exists a schedule s for G such that s requires a channel capacity of at most k and has a length of at most l time units.

1. Introduction

The problem of scheduling a set of tasks that need to exchange data by means of communication has been investigated by several researchers ([6, 1, 4, 5] to name a few). Most of this work has assumed that the time to transport data can be sizable and could indeed be greater than the time it took to compute the data. On the other hand the overall number of communication channels or network bandwidth was always supposed to be unbounded. Even in the absence of resource constraints (unlimited number of processors and communication channels) this problem and practically all meaningful subproblems have been shown to be NP-hard [1, 5]. Furthermore none of the attempted heuristics has been shown to yield close to optimum results in general.

In this work we look at the task communication problem from a different perspective. We assume that the time to transport data from one task to another is negligible with

respect to the time it is required to produce the data. However, we will consider the number of available communication channels or network bandwidth to be limited.

In our framework a given set T of tasks have to be executed. Processing power is always assumed to be available in sufficient quantities. Each task $\tau \in T$ requires a certain amount of processing time $d(\tau) > 0$. In addition tasks may need to exchange information. This is achieved by means of communication channels. More specifically the data produced by a task τ may be required by a task τ' before τ' can commence execution. As soon as τ starts, a communication channel C needs to be established between τ and τ'. The channel C provides a communication link so that data can be conveyed to τ'. In the case where τ' is unable to accept τ's results, because all of the data that it requires is not yet available, C provides some buffering space to hold τ's results. These will be delivered when τ' is ready, i.e. when all the data required by τ' and produced by other tasks becomes available. It is only when τ' starts that the channel C from τ to τ' can be reused by some other task. Note that the buffering space and bandwidth that need to be allocated to C while τ sends data to τ' are a function of the amount of information transferred.

Because a task models some unknown set of computational activities, its execution need not be restricted to a single processor. Therefore a communication channel is always required when forwarding data between two tasks.

The set of communication channels can be viewed as an asynchronous network providing bandwith and buffering space in which messages are temporarily saved when the recipients are not ready to accept them. If a computation requires more bandwith or buffering space that the network can provide a second network, much slower than the first one, is in charge of relaying the data between processors. Thus a computation can always be carried to completion but if it cannot be scheduled so as to meet the resource constraints of the fast network it incurs a serious time penalty. Given this framework it is interesting to investigate the following questions:

- The ACME corporation wants to buy a new parallel computer. In order to test a given platform without committing too much of their budget they ask their IS department to compute the minimum communication capacity required to execute their typical application efficiently.

- The corporation has finally selected a machine where the overall computational resources can be partitioned among a number of users. Such is for instance the case on the Connection Machine 5 [8]. Given a particular problem $\mathcal{P}$ the system administrator may now ask what is the minimum amount of communication capacity that she needs to set aside for $\mathcal{P}$ so that it can execute as quickly as possible.

- After careful elaboration the system administrator has decided to allocate the same amount of communication capacity to each user. It is now the user's turn to ask about optimality, but from a different angle, namely what is the best possible execution time that she can get for her problem with the given capacity.

Formally a communicating task system is modeled as a weighted directed acyclic graph $G = (T, E, d, w)$, called the communication graph, where the vertex set T represents the

tasks or processes that need to be executed, the edge set E represents the communication requirements among the tasks, for every $\tau \in T$ the positive integer $d(\tau)$ gives the amount of processing needed by task τ and for each edge $e = (\tau, \tau')$, the positive integer $w(e)$ gives the required buffering and bandwith capacity to forward data from τ to τ'. More specifically an edge $(\tau, \tau') \in E$ from τ to τ' implies that the data produced by τ is explicitly required by τ' and needs to be conveyed to it by the means of a communication channel with a minimum bandwith and buffering capacity of $w(e)$.

The goal is to find a valid schedule s for G, that is a mapping from T into the positive integers such that for all edge $(\tau, \tau') \in E$

$$s(\tau) + d(\tau) \leq s(\tau')$$

The length of schedule s is denoted $|s|$ and is defined to be $\max_\tau s(\tau) + d(\tau)$. The overall capacity of the communication channels required by s at time t is denoted $C(s,t)$ and is defined to be

$$C(s,t) = \sum_{e \in X(s,t)} w(e) \quad \text{where} \quad X(s,t) = \{(\tau, \tau') \in E \ : \ s(\tau) \leq t < s(\tau')\}$$

The maximum bandwith and buffering capacity needed by schedule s is denoted $C(s) = \max_t C(s,t)$. The three problems that we have formulated earlier can be formally written as follows:

Channel sufficiency: what is the minimum capacity needed to execute a communicating task system $G = (T, E, d, w)$, i.e. find a valid schedule s for G which has smallest $C(s)$.

Channel minimality: what is the minimum capacity needed to execute a communicating task system $G = (T, E, d, w)$ in the smallest possible amount of time, i.e. find a valid schedule s for G with smallest $C(s)$ such that its length $|s|$ is equal to the longest d-weighted path in G.

Channel constrained: what is the best possible schedule for a communicating task system $G = (T, E, d, w)$ when only a capacity of k is available, i.e. find a valid schedule s with smallest length $|s|$ such that $C(s) \leq k$.

Except for the transformation to integer programming which is given in the last section, the communication graph will be restricted to a forest, that is the data produced by one task will be needed by at most one other task. We will also assume that all tasks require the same processing time which, without any loss of generality, can be assumed to be 1. Likewise will assume that communication requirements between any two tasks are identical, that is for every edge e of the communication graph $w(e) = 1$. Under these constraints a schedule s can be seen as a mapping from T into the set $\{0, 1, \ldots, |T|\}$ and for every time instant t, $C(s,t)$ gives the number of communication channels required by s at time t. In the sequel this restricted framework will be denoted $G = (T, E, 1, 1)$ and the notion of channel capacity will be identified with the notion of channel itself.

In this restricted framework communication channels can be viewed as including both processing and communication. The overall machine can be seen as a set of processors connected to a memoryless network where once a task has finished executing, the processor P involved tries to ship the results to the processor P' of the receiving task. If this is not ready then P stalls until P' is ready to accept P's messages. As such the problem can be viewed as an extension of the classical scheduling problem where a number of identical processors are provided to execute a set of precedence constrained tasks.

Even in this simple case we will show that two out of the three problems above are computationally intractable. More precisely we show that even if the communication graph is a a forest comprising only chains of tasks or a tree where each task has at most two immediate predecessors, the channel minimality and channel constrained problems are NP-hard. This is in sharp contrast with the polynomial time algorithms for the equivalent simpler classical scheduling problems where only processor constraints are considered [2].

To compensate the aforementioned intractability results we will provide in the last section of this work, an efficient polynomial time reduction of our problems to integer programming. The reduction can handle arbitrary communication graphs. Because of the amount of research that has been invested in integer programming this approach will enable us to use off-the-shelf state-of-the-art routines to solve channel minimality and channel constrained problems without having to design an exhaustive search algorithm. Of course in the worst case all integer programming routines are bound to take exponential time unless P = NP.

Note that in the restricted framework $G = (T, E, 1, 1)$ where G is a forest, the problems studied herein can be viewed as the problems of generating optimal code with no spilling for expression trees on a RISC machine comprising several functional units and sharing a common register file. Indeed by extending the result of Sethi & Ullman [7] on optimal code generation for expression trees on a one processor machine, we provide a linear time algorithm for the channel sufficiency problem.

2. The Channel Sufficiency Problem

In this section we outline a linear time algorithm to solve the channel sufficiency problem in our restricted framework, when the communication graph $(T, E, 1, 1)$ is a forest $\mathcal{F}$. The algorithm extends the ideas employed by Sethi & Ullman [7]. The approach used in this section is directly applicable to the case where task durations are different.

The minimum number of channels needed to execute $\mathcal{F}$ will be denoted $c(\mathcal{F})$. Without any loss of generality, we can assume that the communication graph is a tree rather than a forest. In fact, $c(\mathcal{F})$ is given by the maximum of the $c(\mathcal{T})$, where $\mathcal{T}$ is a tree in $\mathcal{F}$. Let $\mathcal{T} = (T, E, 1, 1)$ be a communication tree, then $c(\mathcal{T})$ is given by the following recursive formula:

- if T contains a single task: $c(\mathcal{T}) = 0$
- if not, let $\mathcal{T}_i$, for $i = 1, \ldots, a$ be the subtrees whose roots are the immediate predecessors of $\mathcal{T}$'s root. We assume that theses subtrees are indexed so that $c(\mathcal{T}_i) \geq c(\mathcal{T}_{i+1})$.

Then:

$$c(\mathcal{T}) = \max_{i=1\ldots a} \left(c(\mathcal{T}_i) + \begin{cases} i & \text{if } \mathcal{T}_i \text{ contains a single task} \\ i-1 & \text{otherwise} \end{cases} \right)$$

Note that this recursive formulation of $c(\mathcal{T})$ can be easily computed in linear time by a postorder traversal of $\mathcal{T}$.

The following lemma introduces the fundamental property on which the above formula is based.

Lemma 1 *Let $\mathcal{T}$ and $\mathcal{T}_i$, for $1 \leq i \leq a$, be communication trees as defined in the recursive formula. Let s be a valid schedule for $\mathcal{T}$. Then there exists a schedule s' such that $C(s') \leq C(s)$ and the execution of the root of one of the $\mathcal{T}_i$ is completed before any of the tasks in T_j, $j \neq i$, are initiated.*

***Proof**: For a tree $\mathcal{T}_i$, let $start(\mathcal{T}_i)$, respectively $end(\mathcal{T}_i)$, be the smallest initiation time, respectively the greatest completion time, of any of its tasks in schedule s. Let i_0 be the tree index such that*

$$start(\mathcal{T}_{i_0}) = \min_{i=1\ldots a} start(\mathcal{T}_i)$$

Then, schedule s' is defined to be:

$$s' = \begin{cases} s(\tau) & \text{if } \tau \text{ is a task in } T_{i_0} \\ s(\tau) + end(\mathcal{T}_{i_0}) - start(\mathcal{T}_{i_0}) & \text{otherwise} \end{cases}$$

Clearly s' is still a valid schedule for $\mathcal{T}$ and the execution of $\mathcal{T}_{i_0}$'s root is completed before any of the tasks in T_j, $j \neq i_0$, are initiated. It remains to show that $C(s') \leq C(s)$.

Let $C_{\mathcal{T}_i}(s,t)$ denote the number of communication channels used in schedule s at time t to execute the tasks in tree $\mathcal{T}_i$. Clearly for $start(\mathcal{T}_{i_0}) \leq t < end(\mathcal{T}_{i_0})$, we have $C(s',t) \leq C(s,t)$. Let D denote $end(\mathcal{T}_{i_0}) - start(\mathcal{T}_{i_0})$ and τ_r the root of $\mathcal{T}$. For $end(\mathcal{T}_{i_0}) \leq t < D + |s| - d(\tau_r)$, one communication channel needs to hold the results of $\mathcal{T}_{i_0}$'s root thus

$$C(s',t) = 1 + \sum_{j \neq i_0} C_{\mathcal{T}_j}(s, t-D)$$

Furthermore, at least one communication channel must be devoted in s to the tasks in T_{i_0} until $\mathcal{T}$'s root starts executing. Thus, for $start(\mathcal{T}_{i_0}) \leq t < |s| - d(\tau_r)$ we have

$$1 + \sum_{j \neq i_0} C_{\mathcal{T}_j}(s,t) \leq C(s,t)$$

and therefore, $C(s') \leq C(s)$. □

Theorem 1 *The recursive formula given at the beginning of the section gives the minimum number of channels needed to compute $\mathcal{T}$.*

***Proof**: By inductive application of the previous lemma, there exists a valid schedule s_0 for*

$\mathcal{T}$ with minimum $C(s_0)$ such that the tasks from two different subtrees $\mathcal{T}_i$ of $\mathcal{T}$ never execute simultaneously in s_0. Thus, there exists an ordering $\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, \ldots, \mathcal{T}_{i_a}$ of $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_a$ such that in s_0 we have

$$end(\mathcal{T}_{i_1}) \leq start(\mathcal{T}_{i_2}) < end(\mathcal{T}_{i_2}) \leq start(\mathcal{T}_{i_3}) \ldots$$

Let $C_{\mathcal{T}_i}(s_0)$ denote the number of channels needed in s_0 to compute $\mathcal{T}_i$. Clearly, if $c(\mathcal{T}_i) < C_{\mathcal{T}_i}(s_0)$ we can replace the piece of s_0's schedule corresponding to $\mathcal{T}_i$ with the valid schedule for $\mathcal{T}_i$ yielding $c(\mathcal{T}_i)$ without affecting $C(s_0)$ or s_0's validity for $\mathcal{T}$. Thus we can assume that $C_{\mathcal{T}_i}(s_0) = c(\mathcal{T}_i)$. Furthermore, if $C_{\mathcal{T}_i}(s_0) < C_{\mathcal{T}_j}(s_0)$ and $end(\mathcal{T}_i) \leq start(\mathcal{T}_j)$ we can swap the position of $\mathcal{T}_i$'s and $\mathcal{T}_j$'s schedules in s_0 without affecting $C(s_0)$ or s_0's validity for $\mathcal{T}$. Thus we can assume that in s_0 we have

$$c(\mathcal{T}_{i_1}) \geq c(\mathcal{T}_{i_2}) \geq \ldots \geq c(\mathcal{T}_{i_a})$$

Let us renumber the $\mathcal{T}_i$ such that $\mathcal{T}_{i_1} = \mathcal{T}_1, \mathcal{T}_{i_2} = \mathcal{T}_2$ and so on. While $\mathcal{T}_i$ is executing in s_0, $i-1$ channels are holding the results of $\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}$. Thus, the minimum number of channels needed to compute $\mathcal{T}_1$ through $\mathcal{T}_i$ is

$$\min_{j=1,\ldots,i} \left(c(\mathcal{T}_j) + \begin{cases} j & \text{if } \mathcal{T}_j \text{ contains a single task} \\ j-1 & \text{otherwise} \end{cases} \right)$$

and therefore by induction the recursive formula given at the beginning of this section is correct. □

3. Intractability Results

In this section we show that both the channel minimality and channel constrained problems are NP-hard even for very simple cases. Before getting in the heart of the matter we will outline how the channel minimality and channel constrained problems can be reduced one to the other.

Theorem 2 *If the general channel constrained problem can be solved in time $f(n,m)$, where n is the number of tasks and m the number of edges in the communication graph, then the general channel minimality problem can be solved in time $f(n,m) \cdot \log(mw_{max})$ where w_{max} is the maximum communication capacity needed to forward data between tasks. Conversely if the general channel minimality problem can be solved in time $g(n,m)$ then the channel constrained problem can be solved in time $g(n,m) \cdot \log(nd_{max})$ where d_{max} is the maximum time it takes to execute any task.*

***Proof**: Clearly if we have an overall communication capacity of mw_{max} at our disposal a trivial greedy schedule which executes everything as early as possible guarantees that any communication graph can be executed at maximum speed. Thus if we can solve the general channel constrained problem in time $f(n,m)$, then by performing a logarithmic search on the set $\{1, 2, \ldots, mw_{max}\}$ we can find the smallest number k such that the length of the*

optimum schedule when the available channel capacity is restricted to k is equal to the length of the longest weighted path in the communication graph.

Conversely suppose that l is the length of the optimum schedule for the general channel constrained problem when the overall channel capacity is k. Then if we add to the original communication graph a chain of tasks of overall length l, then the minimum channel capacity required to execute the new communication graph in the shortest possible time will be at most $k+1$. Clearly $0 < l \leq nd_{max}$. Thus we can perform a logarithmic search on the set $\{1, 2, \ldots, nd_{max}\}$ and employ the previously described transformation to compute the minimum channel capacity needed to complete the computation by time l. The smallest duration l such that the minimum number of channels needed is at most $k+1$ yields a solution to the channel constrained problem with k channels. □

Note that if the communication graph is restricted to be a tree the second transformation, from the channel constrained to the channel minimality problem, cannot be employed. However this can easily be fixed by making the root of the chain be a child of the root of the tree.

The intractability proofs that follow are based on a reduction from 3-partition. An instance $I3P = (X, m, B, size)$ of the 3-partion problem is a set X of $3m$ elements, a positive integer bound B and a positive integer size $size(x)$ for each $x \in X$, such that $B/4 < size(x) < B/2$ and such that $\sum_{x \in X} size(x) = mB$. The goal is to partition X into m disjoint sets, X_1, ..., X_m such that for $1 \leq i \leq m$, $\sum_{x \in X_i} size(x) = B$. Because of the constraint imposed on $size(x)$ each X_i must contain exactly 3 elements. The 3-partition problem is strongly NP-hard [3]. In this particular case it means that even if we restrict B and the weights $size(x)$ to be polynomial in m, the 3-partition problem remains NP-hard. Without any loss of generality one can assume that B is a multiple of 4 so that $B/4$ is an integer, for otherwise it suffices to multiply B and all $size(x)$ by 4 which does not change the substance of the 3-partition problem. This will be convenient in the proof of theorem 4.

Given the reduction given in theorem 2 the NP-hardness of any particular subproblem of channel minimality immediately implies the NP-hardness of the corresponding channel constrained subproblem.

Theorem 3 *The channel minimality problem is NP-hard even when task durations and communication capacities are equal to 1 and the communication graph is a set of chains.*

***Proof**: Let $I3P = (X, m, B, size)$ be a given instance of the 3-partition problem. We create a communication graph G where for all $x \in X$ G contains a chain of $size(x)$ unit time tasks. Finally G contains a chain of B unit time tasks. We will show that $I3P$ has a solution if and only if the minimum number of channels needed to execute the communication graph in B time steps is exactly $m+1$.*

Clearly any schedule which employs less than $m+1$ channels at every time step will be unable to compute the communication graph in time B. If there exists a schedule s such that $|s| = B$ and $C(s) = m+1$ then $I3P$ has a solution. In fact at every time step there must be exactly $m+1$ tasks executing. Clearly if $C(s) = m+1$ at most

$m+1$ tasks can be executing every time step. If in a given time step there are less than $m+1$ tasks executing then because $\sum_{x\in X} size(x) = mB$ we must have $|s| > B$. Thus the tasks belonging to a same chain must execute in contiguous time steps. For $x \in X$ let $chain(x)$ be the corresponding chain in the communication graph. Then by the previous remark there must be exactly m chains, apart for the chain of length B, $chain(x_1)$, ..., $chain(x_m)$ whose first task starts executing at time 0. Each of the x_i will be placed in a set X_i, $1 \leq i \leq m$. Then for each set X_i we adjoin the element x'_i such that $chain(x'_i)$ starts executing just after the last task in $chain(x_i)$ has completed. Again by the previous remark such chain must exist. We break ties arbitrarily. We repeat this step one more time using x'_i instead of x_i. Clearly for all $1 \leq i \leq m$, $\sum_{x\in X_i} = B$.

Conversely if 3-partition has a solution we can very easily construct a schedule s whose length $|s| = B$ and such that $C(s) = m+1$. □

Theorem 4 *The channel minimality problem is NP-hard even when task durations and communication capacities are equal to 1 and the communication graph is a tree where every task has at most two immediate predecessors.*

***Proof**: In the proof we will interchangeably use the words task and node. Given an instance $I3P = (X, m, B, size)$ of the 3-partition problem we will construct a communication tree $\mathcal{T}$ which can be executed at maximum speed with exactly $6m + 2B$ channels if and only if $I3P$ has a solution. Every task in $\mathcal{T}$ will require 1 unit of time to execute and will depend on at most two other tasks.*

The idea is to construct a critical subtree which creates a rigid pattern in which the remaining subtrees, one for each element $x \in X$, will be able to fit the overall use of $6m + 2B$ channels if and only if $I3P$ has a solution. The critical subtree comprises $3m$ subtrees called type-1 subtrees, B identical subtrees called type-2 subtrees and a glue subtree to attach together the type-1 and type-2 subtrees. As we will see later on, the glue subtree will also provide attachments for the trees which are associated to each $x \in X$. This last kind of subtrees is called a gadget subtree. There are $3m$ of these.

Let $S = \max_{x\in X} size(x)$. There are $3m$ type-1 subtrees. A type-1 subtree has $m(2S+1)$ levels. For $1 \leq i \leq 3m$, the i-th subtree has 2 nodes in the first $\lceil i/3 \rceil (2S+1) - 1$ levels and one node in the remaining levels. See figure 1(a). When we put all the type-1 subtree next to each other we obtain the pattern indicated in figure 1(b).

Type-2 subtrees are all identical. There are B of those. Each has $m(2S+1)+1$ levels. In the first $2S$ levels there is one node per level. Level $2S+1$ has two nodes. The next $2S$ levels have again one node per level and the the level after that, level $4S+2$ has again two nodes. This alternation of $2S$ one node levels followed by one two nodes levels is repeated. Figure 2 gives a type-2 tree. To its right we give the pattern obtained when we put all the type-2 trees next to each other.

For each element $x \in X$ we create the gadget tree shown in figure 3(a). All the gadget trees have the same number of levels, namely $2S$. Finally the glue tree pastes together all type-1, type-2 and gadget subtrees as shown in figure 3(b). The final communication tree will be denoted $\mathcal{T}$.

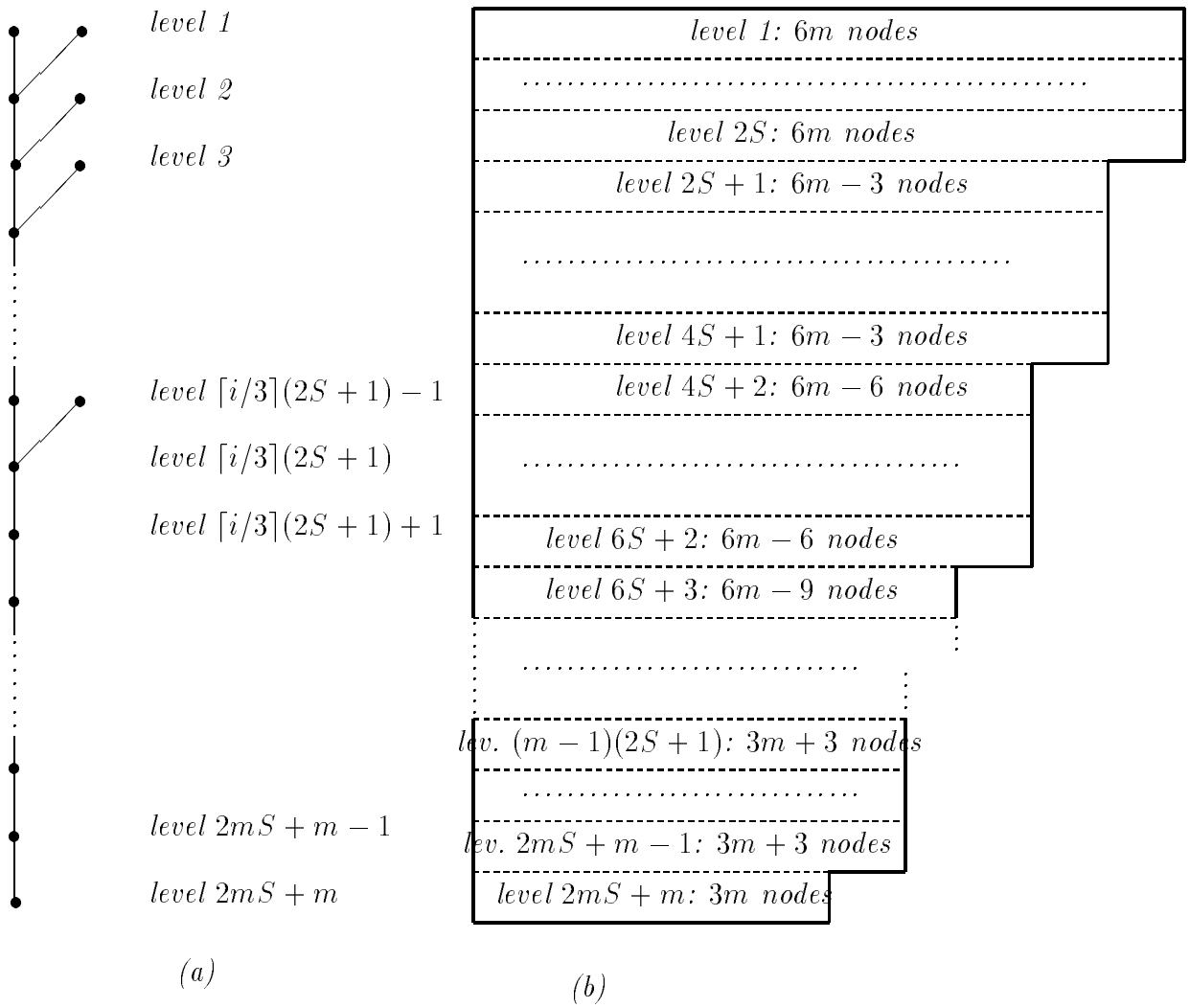

Figure 1. (a) A type-1 subtree. There are $3m$ of these. In the figure we have portrayed the i-th, for $1 \leq i \leq 3m$. (b) The pattern obtained by putting the $3m$ type-1 subtrees next to each other.

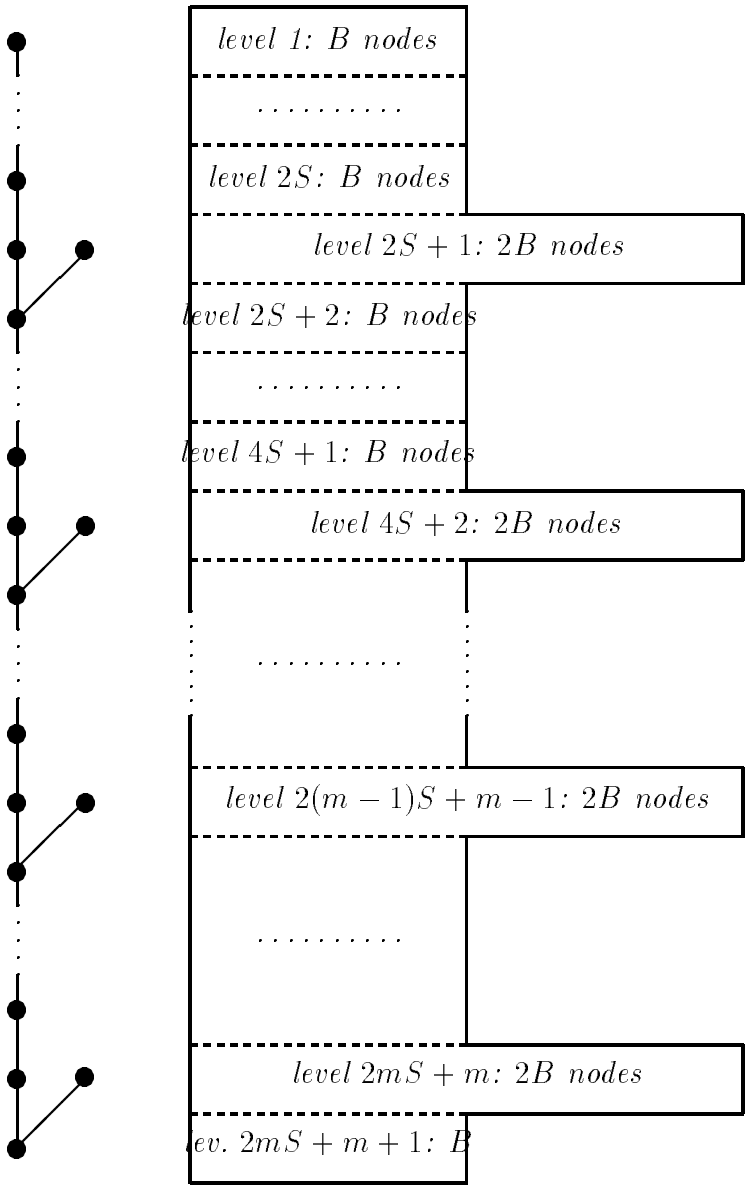

Figure 2. A type-2 subtree. There are B of these. To its right the pattern obtained by putting the B type-2 subtrees next to each other. The levels in the pattern indicate the corresponding level in the type-2 tree to its left.

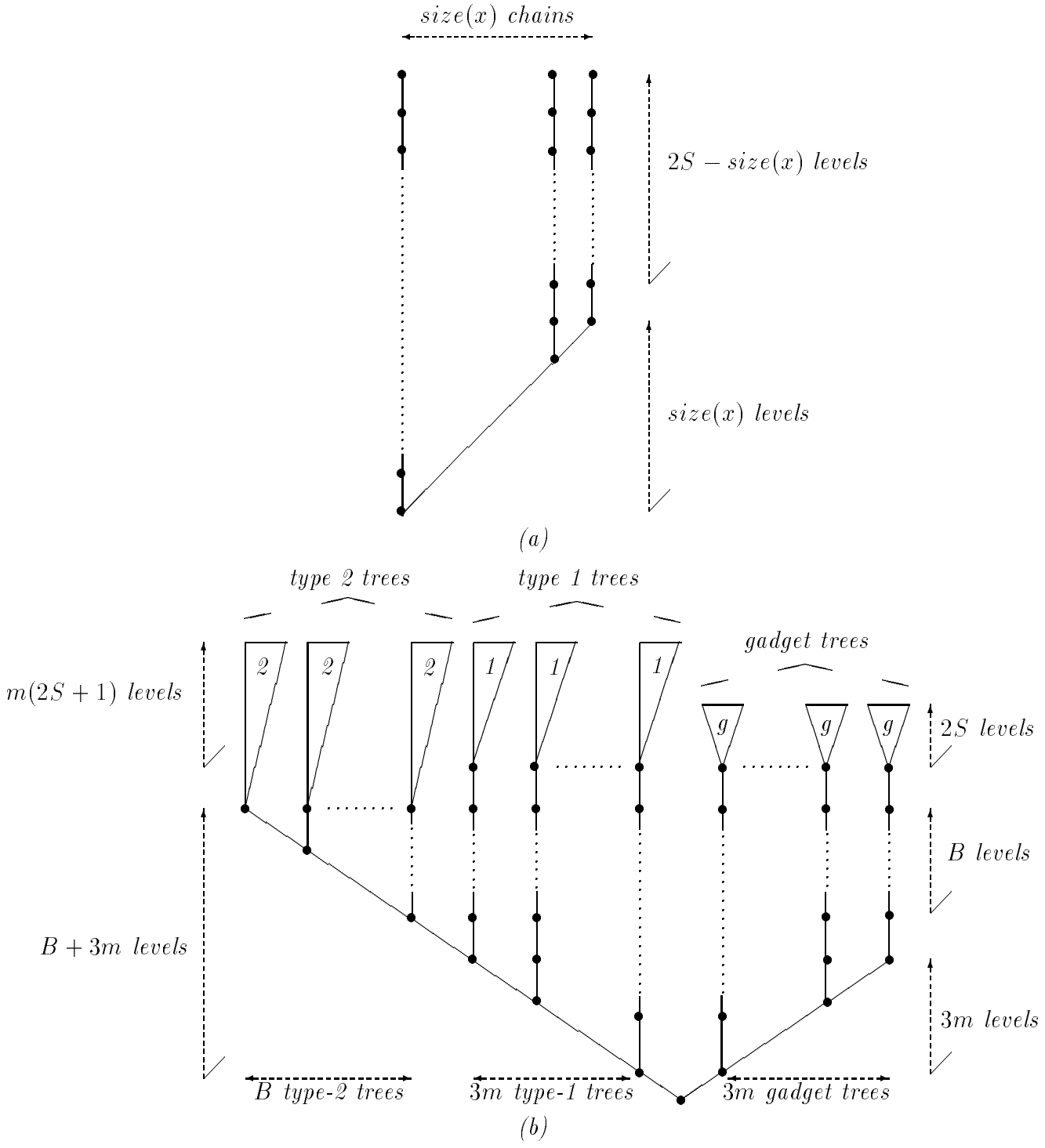

Figure 3. (a) The gadget subtree for $x \in X$. There are $3m$ of these. (b) The final tree.

Let us now show that the minimum number of channels needed to execute $\mathcal{T}$ as fast as possible is at least $2B + 6m$. First of all it is clear that to execute $\mathcal{T}$ at maximum speed all tasks in type-1 and type-2 trees must be executed as soon as possible, basically tasks at level l have to start executing at time $l-1$ and complete at time l. To be able to accomplish this we must have at least $2B + 3m$ channels at our disposal since there are that many tasks in level $2mS + m$ of our type-1 and type-2 trees. Furthermore because the $3m$ roots of the gadget trees cannot be scheduled after the tasks which belong to level $2mS + m$ in the type-1/type-2 trees we overall need at least $2B + 6m$ channels.

Now assume that I3P has a solution, then we will show that $\mathcal{T}$ can be executed at maximum speed with exactly $2B + 6m$ channels. If I3P has a solution then X can be partitioned in m disjoint sets $X_1, \ldots, X_m$, such that for $1 \leq i \leq m$, $\sum_{x \in X_i} size(x) = B$. Let us take the 3 gadget trees corresponding to the elements in X_1 and schedule the trees concurrently with the first $2S$ level tasks in the type-1/type-2 trees. These 3 gadget trees fit perfectly but 3 channels need to be withdrawn from the pool of available channels until the tasks in level $2mS + m$ of type-1/type-2 trees complete. In general the gadget trees corresponding to the elements in X_i can be scheduled concurrently with the type-1/type-2 tasks whose levels range between $2(i-1)S + i$ and $2iS + i - 1$. Again 3 channels need to be withdrawn from the pool of available channels from the time the type-1/type-2 tasks in level $2iS + i$ start executing until the type-1/type-2 tasks in level $2mS + m$ complete. The shape of the type-1 subtrees is exactly designed to counterbalance this need for withdrawing more and more channels from the overall channel pool. More precisely type-1 trees $3i-2$, $3i-1$ and $3i$ all need one channel less to execute from the moment the task in level $2iS+i$ start. Thus if I3P has a solution the communication tree $\mathcal{T}$ can be executed at maximum speed with exactly $2B + 6m$ channels.

Conversely suppose that the communication tree $\mathcal{T}$ can be executed at maximum speed with $2B + 6m$ channels. Because at level $2iB + i$, for $1 \leq i \leq m$, type-1 and type-2 trees require $2B + 6m - 3i$ channels, at most $3i$ gadget trees (or pieces thereof) can be scheduled above that level. Thus at least 3 gadget trees must be scheduled between levels $2(m-1)S + m - 1$ and $2mS + m$. We claim that there must be exactly 3. For if there are 4 then we need $3m + 3 + B$ channels for the type-1/type-2 tasks, $3(m-1) - 1$ channels for the roots of the gadget trees scheduled above level $2(m-1)S + m - 1$ and at least $4(B/4+1) = B+4$ channels for the 4 gadget trees scheduled between levels $2(m-1)S+m-1$ and $2mS + m$ (remember B was assumed to be a multiple of 4). Thus the overall number of channels needed would be at least $2B + 6m + 3$ which is prohibited. Henceforth there are exactly 3 gadget trees executing between levels $2(m-1)S + m - 1$ and $2mS + m$. The above reasonment can be repeated for the levels $2(m-2)S + m - 2$ and $2(m-1)S + m - 1$ and so on. Thus for each $1 \leq i \leq m$, there are exactly 3 gadget trees scheduled between type-1/type-2 levels $2(i-1)S + i$ and $2iS + i$. As there are only B channels left to carry out the execution of these 3 gadget trees and since $\sum_{x \in X} size(x) = mB$ it must be that the overall size of the elements associated with each of the 3 sets of gadget trees is exactly B. Therefore if the communication tree $\mathcal{T}$ can be executed at maximum speed with $2B + 6m$ channels, I3P has a solution: for $1 \leq i \leq m$, X_i contains the elements whose gadget trees are scheduled between levels $2(i-1)S + i - 1$ and $2iS + i$ type-1/type-2 trees. □

4. Reduction to Integer Programming

To compensate for the intractability of the previous section we provide an efficient transformation of our communication problems to integer programming. This allows the use of off-the-shelf routines to find optimum solutions. More specifically given two integer parameters $1 \leq k$ and $1 \leq l$ and an arbitrary communication graph $G = (T, E, d, w)$ we create an integer program polynomial in l and the size of G which has a feasible solution if and only if there exists a schedule s for G such that $C(s) \leq k$ and $|s| \leq l$. This transformation can be readily used to solve the channel minimality or channel constrained problems by emploing a logarithmic search to find the smallest value of k or l for which a valid schedule exists. For the channel minimality problem the value of l is equal to the longest path in the communication graph. For the channel constrained problem the value of k is given by the channel capacity at our disposal.

Let $|T| = n$ and $|E| = m$ then our reduction employs $O(nl + ml^2)$ variables whose values are restricted to be 0 or 1, $O(n + ml^2)$ equations and a total of $O(nl + ml^3)$ non null equation coefficients.

Let us denote $L_u(\tau)$ the maximum length, with respect to the delay function d, of a path from τ to any sink task of G and $L_l(\tau)$ the maximum length, w.r.t. d, of a path from any source task in G to τ. In any valid schedule s for G we have $L_l(\tau) \leq s(\tau) \leq |s| - L_u(\tau)$. The reduction to integer programming proceeds as follows:

Variables: There are two sets of variables. In the first set we associate a variable $x_{\tau,t} \in \{0, 1\}$ to each task $\tau \in T$ and each integral time instant $t \in [L_l(\tau), l - L_u(\tau)]$. The value of $x_{\tau,t}$ will be 1 if and only if task τ is scheduled at time t, otherwise $x_{\tau,t} = 0$. In the second set we associate a variable $y_{e,t,t'} \in \{0, 1\}$ to each edge e in the communication graph G and all time instants $1 \leq t < t' \leq l$. Let $e = (\tau, \tau')$. The value of $y_{e,t,t'}$ will be 1 if and only if τ is scheduled at time t, i.e. $x_{\tau,t} = 1$ and τ' is scheduled at time t', i.e. $x_{\tau',t'} = 1$, otherwise $y_{e,t,t'} = 0$. The variable $y_{e,t,t'}$ pinpoints the starting and ending times in which a communication channel needs to be allocated from task τ to τ'. Let n be the number of tasks and m the number of edges in G. Then there are $O(nl)$ x variables and $O(ml^2)$ y variables.

Constraints: Our goal in the integer program will be to find values for the x and y variables that satisfies the following constraints:

1. For every task $\tau \in T$ exactly one element $x_{\tau,t}$ must be set to one, all the other have to be null, that is $x_{\tau,t} \geq 0$ and

$$\sum_{t=L_l(\tau)}^{l-L_u(\tau)} x_{\tau,t} = 1$$

There are n of these equations, for an overall total of $O(nl)$ non null coefficients.

2. For every edge e in G there is exactly one $y_{e,t,t'}$ which is one, all the other have to be null, that is $y_{e,t,t'} \geq 0$ and

$$\sum_{t=1}^{l-1} \sum_{t'=t+1}^{l} y_{e,t,t'} = 1$$

There are m of these equations for an overall total of $O(ml^2)$ non null coefficients.

3. The communication constraints enforced by the graph G have to be preserved, that is for every edge $e = (\tau, \tau')$ in G if τ is scheduled at time t, i.e. $x_{\tau,t} = 1$, then τ' has to be scheduled on or after time $t + d(\tau)$, i.e. for each $t' < t + d(\tau)$ we must have $x_{\tau',t'} = 0$. Formally this can be written as:

$$x_{\tau,t} + \sum_{t'=L_l(\tau')}^{t+d(\tau)-1} x_{\tau',t'} \leq 1$$

There are ml of these equations for an overall total of $O(ml^2)$ non null coefficients.

4. The allocation of communication channels needs to be taken into account, that is for each edge $e = (\tau, \tau')$ of G and time instants $1 \leq t < t' \leq l$ we have:

$$x_{\tau,t} + x_{\tau',t'} - y_{e,t,t'} \leq 1$$

There are at most ml^2 of these equations for an overall total of $O(ml^2)$ non null coefficients.

5. The final constraint is that at every time instant q a channel capacity of at most k is employed, that is

$$\sum_{t=1}^{q-1} \sum_{t'=q}^{l} \sum_{e \in E} w(e) y_{e,t,t'} \leq k$$

There are l of these equations for an overall total of $O(ml^3)$ non null coefficients.

The overall number of number of non-null coefficients is $O(ml^3)$. It is fairly straightforward to see that the above integer program has a feasible solution if and only if there exists a valid schedule s for the communication graph G such that $|s| = l$ and $C(s) \leq k$. More specifically $s(\tau) = t$ if and only if $x_{\tau,t} = 1$.

Note that in the case where all communication capacities are equal The above transformation yields a 0-1 integer program. Alternatively we can provide a better transformation in terms of the overall number of non null coefficients needed in the constraint equations. For each task $\tau \in T$ we respectively define $pn(\tau)$ and $sn(T)$ to be the number of predecessors/successors of τ in the communication graph G. In this new transformation we only need the x variables and therefore constraint 2 above disappears. Constraints 4 and 5 get merged into a sigle new constraint:

Let k_t denote the number of tasks τ such that $l - L_u(\tau) = t$, that is the number of channels that would be required at time t by the schedule s_0 where $s_0(\tau) = l - L_u(\tau)$. Then for all $1 \leq t \leq l$ we have

$$k_t + \sum_{\substack{\tau \in T \text{ and} \\ l - L_u(\tau) > t}} \sum_{t' = L_l(\tau)}^{t} (sn(\tau) - pn(\tau)) \cdot x_{\tau,t'} \leq k$$

There are l of these equations for an overall total of $O(nl^2)$ non null coefficients.

Thus this second transformation only needs $O(ml^2)$ non null coefficients.

Theorem 5 *The modified integer program in the case of identical edge capacities has a feasible solution if and only if there exists a valid schedule s for the communication graph G such that $|s| \leq l$ and $C(s) \leq k$.*

***Proof**: We only need to show that the new constraint accurately accounts for the fact that at every time step no more than k communication channels are needed.*

Let s_0 denote the schedule such that $s_0(\tau) = l - L_u(\tau)$ for every task $\tau \in T$. We have $C(s_0, t) = k_t$. Clearly the left hand side of the t-th new constraint equation correctly reports the number of channels needed at time t in s_0. Each schedule can be reached from this starting schedule s_0 by simply scheduling tasks at earlier time instants.

Assume that for a given schedule s the channel usage is correctly described by the left hand side of the new constraints. Consider now a second schedule s' which is almost identical to s except for the fact that a task τ is scheduled at time $t-1$ in s' rather than at time t in s. The number of channels needed at time t in s' remains unchanged whereas the number of channels needed at time $t-1$ in s' increases by $sn(\tau) - pn(\tau)$ since the $pn(\tau)$ channels that where needed to convey data to τ in s are not necessary in s' but $sn(\tau)$ new channels need to be allocated at time $t-1$ in s' to carry data from τ to its immediate successors in the communication graph G. It is easy to see that the new constraint equations correctly account for such changes in communication channel needs. Inductive application of this procedure shows the correct description of the channel usage by the left hand side of the new constraint equations. □

1 P. Chrétienne, *Task scheduling over distributed memory machines*, in Parallel and Distributed Algorithms, M. Cosnard, ed., Elsevier Science Publishers (North Holland), 1989, pp. 165–176.

2 E. G. Coffman, *Computer and Job-shop Scheduling Theory*, John Wiley and Sons, New York, New York, 1976.

3 M. R. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, Freeman, New York, New York, 1979.

4 A. Gerasoulis, S. Venugopal, and T. Yang, *Clustering task graphs for message passing architectures*, in International Conference on Supercomputing, ACM, June 1990, pp. 447–456.

5 C. H. Papadimitriou and M. Yannakakis, *Towards an architecture-independent analysis of parallel algorithms*, SIAM Journal of Computing, 19 (1990), pp. 322–328.

6 V. Sarkar, *Partitioning and Scheduling Parallel Programs fro Execution on Multiprocessors*, MIT Press, 1989.

7 R. Sethi and J. D. Ullman, *The generation of optimal code for arithmetic expressions*, Journal of the ACM, 17 (1970), pp. 715–728.

8 Thinking Machine Corporation, *CM5–Technical Summary*, Thinking Machine Corporation, Oct. 1991.