Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Parallel Computing 30 (2004) 103–120

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Sensitivity analysis of tree scheduling on two machines with communication delays

Frédéric Guinand [a,*], Aziz Moukrim [1], Eric Sanlaville [2]

[a] *Laboratoire d'Informatique du Havre, EA 3219, Université du Havre, 25 rue Philippe Lebon, BP 540, F-76058 Le Havre Cédex, France*

Received 15 May 2002; received in revised form 18 January 2003; accepted 19 June 2003

Abstract

This paper presents a sensitivity analysis for the problem of scheduling trees with communication delays on two identical processors, to minimize the makespan. Tasks are supposed to have unit execution time (UET), and the values associated to communication delays are supposed unknown before the execution.

The paper compares the optimal makespans with and without communication delays. The results are used to obtain sensitivity bounds for algorithms providing optimal schedules for graphs with unit execution and communication times (UECT). The notion of *processor-ordered* schedules, for two-processor systems, is introduced. It describes schedules in which all communications are oriented from one processor to the other. It is shown that these schedules are dominant for unit delays, for zero delays, but not for delays of less than or equal to one. We establish that algorithms building optimal processor-ordered schedules for UECT graphs admit an absolute sensitivity bound equal to the difference between the maximum and the minimum actual communication delays: $\omega - \omega^* \leqslant \bar{c} - \underline{c}$. This bound is tight.

Keywords: Uncertain communication delays; Trees; Unit execution and communication times; Sensitivity analysis; Two-processor scheduling

* Corresponding author.
E-mail addresses: frederic.guinand@univ-lehavre.fr (F. Guinand), aziz.moukrim@hds.utc.fr (A. Moukrim), eric.sanlaville@math.univ-bpclermont.fr (E. Sanlaville).
[1] HeuDiaSyC, UTC Compiègne, (France).
[2] LIMOS, Clermont-Ferrand, (France).

0167-8191/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0167-8191(03)00091-7

1. Introduction

Last years have seen a growing number of teams involved in the conception, the deployment and the use of infrastructures and applications distributed over a wide geographical area [26,25]. Peer-to-peer computing [24,27] as well as programs running on grids [28] are some examples of such applications. They are made of many tasks processed on computing devices located on various, possibly very distant, places. The underlying interconnection network linking together all computing elements is partially common with the internet. As a consequence, codes running on such environments may be subject to unexpected slowdowns occurring somewhere on the network. Then, it is not surprising that most peer-to-peer applications are merely based on the common master/slaves algorithmic scheme, while resources managers of grids give greater importance to high-throughput rather than to application performances. In order to avoid poor performances, the tasks of the application have to be distributed among the computing elements in order to obtain a good trade-off between load balanding and minimization of the communication overhead. Within previously described environments, both execution times of tasks and communication delays may vary. In addition, the application itself may be subject to structural variations depending on the input data, then, it is unlikely that a pure static scheduling strategy could lead to good performances. However, in our work a careful analysis of the schedule structure shows the importance of the distribution of tasks for minimizing the impact of communication variations.

In this paper, we focus our study on the degradation of schedules when communication durations estimated a priori differ from actual ones. The application model is a tree-structured task graph with unit-estimated communication delays and unit execution time (UECT task graphs). The overlapping of communications by computations is allowed and communications between tasks executed on the same processor are neglected (known as the locality assumption). This set of hypotheses corresponds to the *standard delay model* (SDM) [3,6].

Using this environment execution model, in [8], the authors proved that scheduling complete binary trees on an unlimited number of processors with arbitrary communication delays is an NP-Hard problem. The problem remains NP-Hard for binary trees when communications depend on the size of the tree. When the communication delays are equal to a constant value c, the problem can be solved polynomially for complete k-ary trees. For arbitrary trees, keeping the same assumptions, the NP-Hardness of the problem is proved in [9]. The same complexity result holds for UECT trees on an arbitrary number of processors organised as a full-connected graph [9,10,12]. When the number of processors, say m, is fixed, the problem may be solved using a dynamic programming approach (complexity $O(n^{2(m-1)})$) [20]. Finally, this problem becomes polynomial when the number of processors is 2 [9,11–13] or is not limited [4]. The main results are summed up in the following table using the notation described in [5]. A survey on scheduling problems with communication delays can be found in [14].

$\alpha \mid \beta \mid \gamma$ notation	Result	Reference
$P_\infty \mid$ complete binary tree, $p_i = 1$, $c_{ij} \mid C_{\max}$	NP-Hard	[8]
$P_\infty \mid$ tree (n vertices), $p_i = 1$, $c = f(n) \mid C_{\max}$	NP-Hard	[8]
$P_\infty \mid$ complete k-ary tree, $p_i = 1$, $c \mid C_{\max}$	Polynomial	[8]
$P_\infty \mid$ tree, $p_i = 1$, $c \mid C_{\max}$	NP-Hard	[9]
$P \mid$ tree, $p_i = 1$, $c = 1 \mid C_{\max}$	NP-Hard	[9,10,12,19]
$P_m \mid$ tree, $p_i = 1$, $c = 1 \mid C_{\max}$	$O(n^{2(m-1)})$	[20]
$P_2 \mid$ tree, $p_i = 1$, $c = 1 \mid C_{\max}$	Polynomial	[9,11,13,22]
$P_\infty \mid$ tree, $p_i = 1$, $c = 1 \mid C_{\max}$	Polynomial	[4]

While many works have been devoted to the study of scheduling precedence tasks graphs with communication delays, very few is known about the sensitivity analysis of these problems. Without communication delays, in [2] Graham studied the impact of changes for both numerical and structural parameters on the behaviour of list scheduling algorithms. When changes on processing times of the tasks are considered, Wagelmans showed that the sensitivity of a list scheduling rule increases with the quality of the schedule produced [7,15], and for the case of independent tasks Penz et al. showed that the sensitivity can be limited to the square root of the magnitude of the perturbation [16]. When communication delays are taken into account and for an unlimited number of processors, Gerasoulis and Yang [18] have proposed a relative sensitivity bound based on the notion of granularity. In [21] we proposed a stabilization scheme for m machines and arbitrary task graphs.

This paper is a study of the sensitivity analysis of statically computed schedules for the problem ($P_2 \mid$ tree, $p_i = 1$, $c = 1 \mid C_{\max}$) when the actual communication delays differ from the estimated ones. To this end, new results are presented for the makespan values with and without communications. We also introduce the notion of *processor-ordered* schedules, a structural property of the communications between processors, and we show their robustness when communication delays change.

The paper is organized as follows: in Section 2, the model is precisely defined. Section 3 compares the two classical problems with and without communications. Within Section 4 tight relative and absolute bounds are presented for the two-machine problem. The sensitivity analysis of several algorithms providing optimal schedules in the UECT case is presented in Section 5. We show that the cluster-based scheduling algorithm presented in [22] is by far the most robust. Some future trends for research are proposed in the conclusion.

2. Problem formulation

The model considered was first described by Rayward-Smith in [3] and extended in 1988 by Papadimitriou and Yannakakis [6]. It states that at any time, a processor can

only process one task and each task can be processed by only one processor. It considers the scheduling of a partially ordered set of n UET tasks $T_1, \ldots, T_n$ on a set of identical processors. The precedence relation is denoted by $\prec$. Given two tasks T_i and T_j such that $T_i \prec T_j$, if they are executed on different processors, a delay occurs between the end of T_i and the beginning of T_j due to the transfer of data between both processors. If they are executed on the same processor, the transfer delay is neglected. This is known as the locality assumption. The set of precedence relations forms the task graph. Throughout the paper we only consider tree-structured task graphs. This communication will be denoted $\widetilde{c_{ij}}$ for the estimated value, and c_{ij} for the actual one.

- $\bar{c}$ denotes the maximum actual communication delay,
- $\underline{c}$ denotes the minimum actual communication delay.

Another important assumption states that communications can be overlapped by computations, thus, a processor can simultaneously communicate (send and/or receive some data) and perform computations. Task preemption is not allowed. Unlike the model presented in [6], task duplication is not allowed in this work. The goal is to minimize the date of the end of the execution: the *makespan*.

On each processor a schedule is composed of three parts:

- a set of tasks corresponding to an *assignment*,
- a *sequencing* (or ordering) of these tasks on the processor,
- a set of dates (beginning time b_i for each task T_i, or b_A for a task denoted A).

When b_i is an integer for each task T_i, the schedule is called an *integer schedule*.

A scheduling algorithm provides a complete schedule from a given task graph and a given set of processors. An algorithm is static (or off-line) if the complete schedule (assignment, sequencing and dates) is determined before the execution. The sensitivity analysis described in this paper consists in studying the modification of dates when actual communication delays are taken into account.

The resulting schedule has the following properties:

- the assignment is not modified.
- the sequencing on each processor is not modified.
- the new dates are obtained by executing each task as early as possible.

For a given algorithm, let $\tilde{\omega}$ be the makespan of the schedule ($\widetilde{S}$) computed with estimated communication delays ($\tilde{c}$), and ω the makespan of the schedule (S) obtained from $\widetilde{S}$ when actual delays are reported in $\widetilde{S}$. The optimal makespan is denoted by ω^* for actual communication delays.

Definition 1. The relative sensitivity bound of an algorithm is the maximum ratio ω/ω^* over all task graphs.

Definition 2. The absolute sensitivity bound, if any, of an algorithm is the maximum difference $\omega - \omega^*$ over all task graphs.

If we denote by V the set of tasks, by E the set of communications and by P the set of processors, then, $\pi : V \rightarrow P$ denotes the function that allocates a processor to each task.

Definition 3. A communication is called effective between two tasks T_i and T_j iff $T_i \prec T_j$ or $T_j \prec T_i$ and $\pi(T_i) \neq \pi(T_j)$.

Definition 4. A processor-ordered assignment is an assignment such that all effective communications occur from one processor, denoted P_s (sender), to the second processor, denoted P_r (receiver). A processor-ordered schedule is a schedule whose assignment is processor-ordered.

As an immediate consequence, in a processor-ordered assignment any path of the task graph contains at most one effective communication. In the following PO-schedules stands for processor-ordered schedules.

3. Distance between optimal makespans with and without communication

We claim that PO-schedules are dominant for trees without communication delays.

We consider schedules built by Hu's algorithm which are optimal for UET trees. On two-processors (P_r and P_s), Hu's algorithm [1] (or Highest Level First algorithm) builds schedules such that one processor is never idle (say P_r), and the other is never idle until some time slot θ, and continuously idle thereafter. Throughout the paper such a schedule is said to be *without internal idleness*.

Theorem 1. *For any instance of* $P2 \mid tree, uet \mid C_{\max}$, *there always exists a* PO*-schedule that is optimal.*

Proof. By a sequence of exchanges, an optimal schedule without internal idleness is transformed into a PO-schedule of the same makespan. According to Fig. 1, tasks beginning after time θ form a chain and are executed on P_r. G_θ denotes the task graph obtained from tasks executed before θ. G_θ is a forest of at least two intrees. Let A be the set of tasks of the smallest intree whose root is executed at $\theta - 1$ (there are exactly two such trees). Below, the schedule is transformed so that all tasks belonging to A are executed on P_s.

A *hole slot* is a time slot containing no task in A. A *double slot* is a time slot during which two tasks from A are executed. A being the smallest intree, $|A| \leqslant \theta$ and there are at least as many holes as doubles. Within Fig. 1, capital letters identify tasks belonging to A and Greek letters are associated to tasks that do not belong to A. Suppose t is a hole, and t' is its nearest double. Assume there is no other hole between t and t', and $t' > t$ (otherwise the reasoning is completely symmetrical). Let us consider X and Y, the two tasks executed at t' (see Fig. 1a). If $t' = t + 1$ exchange Y with any of the tasks

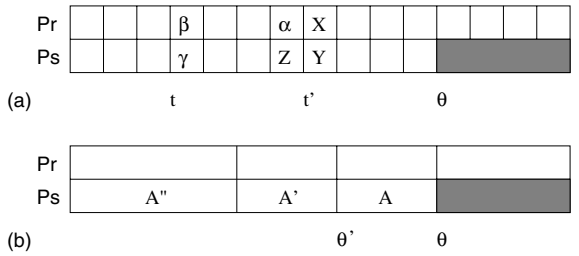

Fig. 1. Dominance of PO-schedules in the UET case.

executed at t (no precedence is violated as these tasks do not belong to A). Now, suppose $t' > t + 1$. If the task Z executed at $t' - 1$ precedes Y, then α and X are interchanged, otherwise α and Y are permuted. This entails the formation of a double at time slot $t' - 1$. Repeat the same operation until $t' = t + 1$ and remove the obtained double in the same way. The same process is repeated until no double remains.

Then, all holes are gathered in the first slots. Consider a hole at t, and no hole at $t - 1$. At $t - 1$, X and α are executed and at time t, β and γ are executed. G_θ being an inforest, α has at most one successor. Assume α and β are independent. The interchange of X and β moves the hole one slot earlier. By repeating this operation, a schedule is obtained for which all tasks in A are executed on P_s between some time θ' and θ, without holes. Again, the set of tasks executed between 0 and θ' form an inforest and the same process may be applied repeatedly until the schedule has the following feature (see Fig. 1b): a set of whole intrees is executed sequentially on P_s. It follows that there is no precedence from a task executed on P_r to a task executed on P_s. □

The next theorem follows from Theorem 1.

Theorem 2. *The difference between the optimal makespans for trees with and without communication delays is bounded by* $\bar{c} : \omega^* - \omega_0^* \leqslant \bar{c}$.

Proof. Consider some PO-schedule, which is optimal when communication delays are zeroed. When communication delays are bounded by $\bar{c}$, the schedule remains feasible if all the tasks assigned to P_r are moved right by $\bar{c}$. □

Finding examples for which the bound is tight is straightforward.

4. Sensitivity bounds for algorithms which are optimal in UECT case

4.1. Bounds for small communication delays

Actual communication delays are supposed to be such that: $c_{ij} \leqslant \tilde{c}_{ij} = 1$. This case is called SCT, standing for *small communication delays*. As previously stated,

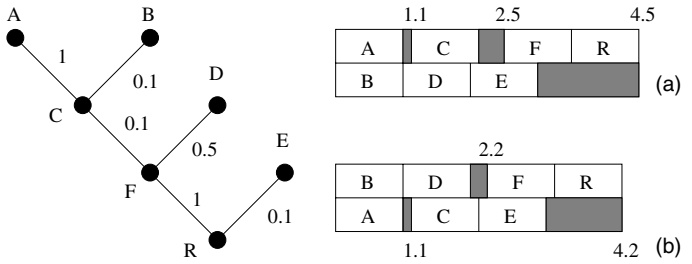

Fig. 2. No dominance of PO-schedules in the SCT case.

$\underline{c} = \min_{i,j} c_{ij}$. In this section it is proved that $1 - \underline{c}$ is an absolute bound for any optimal schedule in the SCT case. When an optimal PO-schedule exists for actual communication delays, verifying that $1 - \underline{c}$ is an absolute bound is simple. Unfortunately, the example of Fig. 2 shows that PO-schedules are not dominant for SCT problems. Indeed:

1. To finish in less than 5 time units, a schedule must execute exactly four tasks on P_r, including F and R.
2. If C, F, R are executed on P_r, the best schedule has a makespan of 4.5 (see Fig. 2a).
3. If D, F, R are executed on P_r, C or E on P_r leads to a makespan larger than 5. The only possibility is not processor-ordered and is optimal (see Fig. 2b).

It follows that the complete proof of this seemingly simple result needs a number of technical lemmas.

ω_0^* and ω_1^* denote the makespans of optimal schedules when communication delays are respectively equal to 0 and 1. Without loss of generality the root is supposed executed on P_r in all schedules.

In what follows all considered schedules are such that all tasks are executed as early as possible while respecting precedence constraints and communication delays. For a given schedule S, $t(S)$ denotes the beginning of the first idle period, if any, and $C(S)$ denotes the task executed immediately after that idle period (as shown on Fig. 3).

Consider the following property, verified by some schedule S.

Property 1. *Any task A such that $b_A < t(S)$ is a predecessor of $C(S)$.*

Lemma 1. *Consider the set of* SCT *instances of the problem for which any optimal schedule contains some idleness. Then any such schedule S^* with $t(S^*)$ maximum verifies Property* 1.

Proof. Consider an optimal schedule S^* containing some idleness and such that $t(S^*) = t$ maximum. Let us suppose, moreover, that Property 1 is not verified by S^*, then, there is at least one task A such that $b_A < t$ and A does not precede $C = C(S^*)$. Suppose X is the latest such task. Note that this implies that X is independent of any task executed between times $\tau = b_X$ and t.

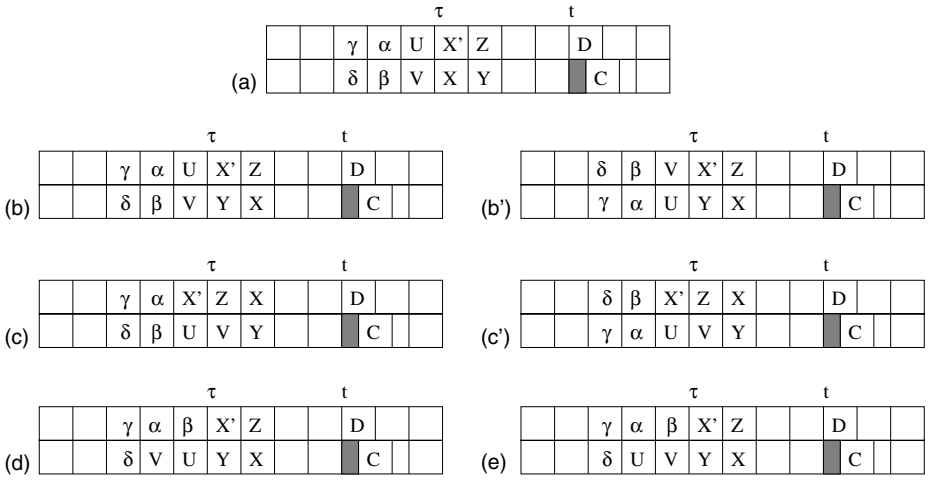

Fig. 3. Illustration of Lemma 1 proof.

If $\tau = t - 1$, then X and C are executed on the same processor, otherwise the idle period would not appear. Interchanging the tasks executed on P_r with the tasks executed on P_s, for the period $[0, t]$, results in the achievement of an optimal schedule without internal idleness, which constitutes a contradiction.

If $\tau < t - 1$, we will show that the partial schedule corresponding to the time interval $[0, \tau + 1]$ can be transformed into another schedule of the same length where X is moved strictly forward.

There are three different ways to move X forward. Fig. 3a. illustrates the situation before the moves.

1. $\tau = 0$, or *U and V are not both predecessors of Y*. In that case X and Y can be permuted.
   (a) If V precedes Y, nothing more is needed (see Fig. 3b).
   (b) If U precedes Y, tasks executed on P_r are interchanged with the tasks executed on P_s for the time period $[0,\tau]$ (see Fig. 3b′).
2. $\tau \geqslant 1$ *and*, *U and V both precede Y* or *α and β are not both predecessors of X'* (see Fig. 3a). In that case X' and Z are moved backward so that X is moved forward by 1 time unit $(\tau + 1)$.
   (a) If $\tau = 1$, or if β does not precede X', the changes are obvious (see Fig. 3c).
   (b) If β precedes X', an interchange of tasks between both processors is necessary (see Fig. 3c′).
3. $\tau \geqslant 2$ *and U and V precede Y*, *and α and β both precede X'*. The solution for moving X forward consists in gathering tasks U, V and Y on one processor, and α, β and X' on the other one. According to the precedence relation between γ and V, $b_U < b_V$ if $\gamma \prec V$ (see Fig. 3d), or $b_V < b_U$ otherwise (Fig. 3e).

In all cases X is moved forward by one time unit. Note that the new schedule always respects the precedence relations. The process is iterated until X is moved forward to time slot $[t-1,t]$. At the end of the process, if $X \prec D$, and if both X and C are allocated to the same processor, the set of tasks belonging to the time interval $[0,t]$ are permuted on both processors, otherwise no extra move is needed. In both cases, the idleness before C is removed, which leads to a contradiction, so, X also precedes C. □

Lemma 2. *Consider the set of* SCT *instances of the problem, with* $\underline{c} > 0$, *such that* $\omega_0^* = \omega^*$. *Then any optimal schedule is without idleness on* P_r.

Proof. Consider some optimal schedule S^*, with idleness on P_r. Let Y be the task executed immediately after the last idle period on P_r. As $\underline{c} > 0$, Y waits for a non-zero communication from P_s. If this communication delay is zeroed, Y is moved backward, creating a new idleness. Let Y' follow Y. Y' is free to move backward or to wait for a non-zero communication. In the second case this delay is again zeroed, so that Y' is moved backward. The process is iterated until R, and a schedule with a makespan strictly smaller than ω_0^* is obtained. □

Remark on the proof. If we consider any feasible schedule for the SCT problem, zeroing all the communications does not necessarily lead to a schedule without internal idleness (see Fig. 4).

Lemma 3. *Consider the set of* SCT *instances of the problem such that* $\omega_0^* < \omega^*$. *If there exists some optimal schedule for which the amount of idleness on* P_r *is at least* $\underline{c}$, *then*

$$\omega^* \geqslant \omega_0^* + \underline{c}$$

Proof. Consider S^* an optimal schedule such that $\omega^* = \omega_0^* + q$ with $q < \bar{c}$. The presence of idleness on P_r entails that the number of tasks executed on it is less than or equal to $\omega_0^* + q - 1$ and is at most equal to $\omega_0^* - 1$. So, the amount of idleness on P_r is at least equal to $1 + q$.

If all the communications are removed from S^*, the makespan of the resulting schedule is equal to ω_0^*. As P_r executes at most $\omega_0^* - 1$ tasks, at least one idle slot remains on it. Let $[u-1,u]$ be that slot, and Y the task executed on P_r such that $b_Y = u$.

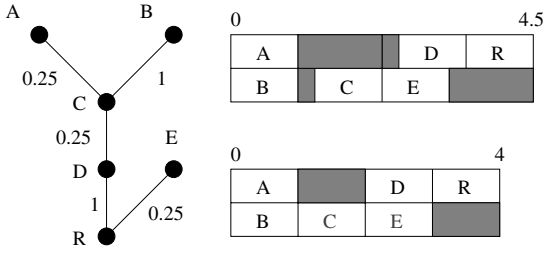

Fig. 4. Zeroing communications does not eliminate idleness.

Y is not executed earlier because it has one predecessor, say X, executed on P_s during $[u-1,u]$. Thus, when adding communication delays, the makespan increases up to $\omega_0^* + \underline{c}$ which is strictly greater than $\omega_0^* + q$ and leads to a contradiction. □

Lemma 4. *Consider the set of* SCT *instances of the problem such that* $\underline{c} > 0$. *If* $\omega^* < \omega_0^* + \underline{c}$, *then*

$$\omega_0^* = \omega^* = \omega_1^*$$

Proof. Consider an optimal schedule S^* such that $\omega^* < \omega_0^* + \underline{c}$.

If S^* is without internal idleness, then all tasks are executed at integer times and two communicating tasks executed on different processors are separated by at least one time unit. If all communication delays are set to 1, the schedule remains unchanged: $\omega^* = \omega_1^*$. As no internal idleness appears within S^*, $\omega^* > \omega_0^*$ implies $\omega^* \geqslant \omega_0^* + 1$ which contradicts the hypothesis. Then, $\omega_0^* = \omega^* = \omega_1^*$.

We want to prove that S^* cannot admit any internal idleness. For that purpose, we introduce $t = t(S^*)$ and $C = C(S^*)$ as defined previously for Lemma 1.

Suppose t is the maximum for S^* among all optimal schedules, then any task executed before t is a predecessor of C (Lemma 1). If C is executed on P_r, an idleness of at least $\underline{c}$ appears on P_r. According to Lemma 2, $\omega^* > \omega_0^*$. Moreover, from Lemma 3, $\omega^* \geqslant \omega_0^* + \underline{c}$, which leads to a contradiction with the hypothesis.

Therefore, the first internal idleness period should be on P_s. We denote as A and B the tasks beginning at $t-1$ respectively on P_s and P_r. D corresponds to the task executed at t on P_r. By Lemma 1, A and B are predecessors of C and D has no predecessor. Consider the first effective communication from P_s (task X) to P_r (task Y) occuring after t. Last, define E to be the first task executed on P_s independent of C, if any. E has no predecessor on P_s. □

1. X is executed after E, or $E = X$. In such a case, B is moved on P_s ($b_B = t$), and E can be moved on P_r. If E has no predecessor it is executed after D. Otherwise it is moved just after its predecessor. On P_s, tasks that were before E are moved forward by $1 - c_{BC}$ time units. Tasks that were after E can be left unchanged. Fig. 5a′ illustrates this set of moves. If $X = E$ the precedence constraint between

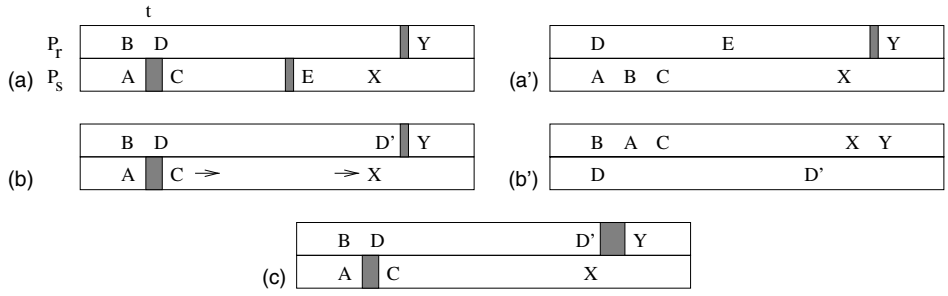

Fig. 5. Illustration of Lemma 4 proof.

X and Y is respected. So, the first idleness period on P_s is removed without increasing the makespan which is in contradiction with the hypothesis on S^*.

2. If X is executed just before E (or if E does not exist), then, $C,\ldots,X,Y$ form a chain (see Fig. 5b). Consider D' the task executed on P_r such that $b_{D'} > b_X$ and $b_{D'} < b_Y$. Such a task always exists since the internal idleness on P_r is strictly lower than $\bar{c}$. Between D and D', as no communication occurs from P_s to P_r, no idleness appears on P_r, so, the number of tasks $A, C, \ldots, X$ is less than or equal to the number of tasks $D,\ldots,D'$. Consider the new schedule obtained by permuting $D,\ldots,D'$ with $A, C, \ldots, X$ (see Fig. 5b′). Each task of $D,\ldots,D'$ can be scheduled on P_s one time unit earlier than it was scheduled on P_r while each task of $A, C, \ldots, X$ is scheduled on P_r later than it was scheduled on P_s without violating any precedence constraint. The other tasks on both processors can be left unchanged. As a consequence, the idle period on P_s beginning at t is removed, which contradicts the hypothesis.

Finally, we have proved that, within the SCT context, if $\omega^* < \omega_0^* + \bar{c}$, then no internal idleness could appear on either processor leading to the result that $\omega_0^* = \omega^* = \omega_1^*$.

Theorem 3. *Optimal schedules for* UECT *intrees have an absolute sensitivity bound in the* SCT *case of at most* $1 - \underline{c}$:

$$\omega - \omega^* \leqslant 1 - \underline{c}$$

Proof. Let $\widetilde{S}$ be such a schedule, and ω the makespan of the schedule obtained from $\widetilde{S}$ with the real SCT communication delays. According to the SCT characteristic of the communications, $\omega \leqslant \omega_1^*$ and from Theorem 2: $\omega \leqslant \omega_1^* \leqslant \omega_0^* + 1 \leqslant \omega^* + 1$. If $\omega^* \geqslant \omega_0^* + \underline{c}$, then, $\omega \leqslant \omega_0^* + 1 \leqslant \omega^* + 1 - \bar{c}$, otherwise ($\omega^* < \omega_0^* + \underline{c}$) from Lemma 4, $\omega_0^* = \omega^* = \omega_1^*$. Finally, in both cases, the theorem holds. □

4.2. Bounds for large communication delays

In this section, we suppose (the LCT case) that actual communication delays are such that: $c_{ij} \geqslant \tilde{c}_{ij} = 1$. $\bar{c}$ still denotes the maximum value of actual communication delays: $\bar{c} = \max_{i,j} c_{ij}$.

Theorem 4. *Optimal schedules for* UECT *intrees have a relative sensitivity bound in the* LCT *case of at most* $\frac{\bar{c}+1}{2}$.

Proof. We consider $\widetilde{S}$ an optimal schedule considering unit communication delays: $\widetilde{\omega} = \omega_1^*$. We divide this schedule into k slices of width 2 (depending on the parity of the makespan, the last one can be of width 1 or 2), and, between any two consecutive slices, we insert an idle period of width $\bar{c} - 1$. The obtained schedule is feasible for the set of actual communication delays. Indeed, consider X and Y such that (X,Y) is an effective communication, which implies that X and Y do not belong to the same

slice. Whatever the situation, before the insertion of idleness $b_Y \geqslant b_X + 1$, and after the insertion $b_Y \geqslant b_X + 1 + \bar{c} - 1 = b_X + \bar{c}$. Then, if we denote ω the makespan of the resulting schedule,

- $\tilde{\omega} = 2k$: then $\omega = 2k + (k-1)(\bar{c}-1) = (\bar{c}+1)k - \bar{c} + 1$.
  Furthermore $\omega^* \geqslant \tilde{\omega}$, so

$$\frac{\omega}{\omega^*} \leqslant \frac{\omega}{\tilde{\omega}} = \frac{(\bar{c}+1)k - \bar{c} + 1}{2k} \leqslant \frac{\bar{c}+1}{2} (\text{as } \bar{c} \geqslant 1)$$

- $\tilde{\omega} = 2k - 1$: then $\omega = 2k - 1 + (k-1)(\bar{c}-1) = (\bar{c}+1)k - \bar{c}$. So,

$$\frac{\omega}{\omega^*} \leqslant \frac{\omega}{\tilde{\omega}} = \frac{(\bar{c}+1)k - \bar{c}}{2k-1} \leqslant \frac{\bar{c}+1}{2} (\text{as } \bar{c} \geqslant 1) \qquad \square$$

Remark. The theorem is valid for arbitrary task graphs.

As soon as two processors are involved in the computation of an UECT intree which is not a chain and contains more than four tasks, at least one communication has to be effective. The maximum value of the difference between the estimated communication cost and the actual one may concern this communication. In such a case, the difference between ω^* and ω is at least $c - 1$ (as illustrated by Fig. 6).

Proposition 1. *The absolute sensitivity bound of any scheduling algorithm providing optimal schedules for* UECT*-intrees on two identical processors is at least* $\bar{c} - 1$.

4.3. PO-algorithms

The previous proposition states that when actual communication delays are larger or equal to 1, (the LCT hypothesis), any scheduling algorithm has an absolute sensitivity bound equal to $\bar{c} - 1$. In the sequel, we show that for PO-schedules this bound is tight.

In the schedule S, with a makespan of ω, that considers actual communication delays, the tasks are executed as early as possible. This leads to the property below.

Property 2. *Let* $\widetilde{S}$ *be some* PO*-schedule in the* UECT *case. Let* S' (*whose makespan is* ω') *be the schedule built from* $\widetilde{S}$ *by moving forward any task of* P_r *by exactly* $\bar{c} - 1$ *time units. Then* S' *is feasible for the* LCT *case, and* $\omega \leqslant \omega' = \tilde{\omega} + \bar{c} - 1$.

Lemma 5. *Optimal* PO*-schedules for* UECT *intrees have an absolute sensitivity bound in the* LCT *case of at most* $\bar{c} - 1$: $\omega - \omega^* \leqslant \bar{c} - 1$.

Proof. Let $\widetilde{S}$ be such a schedule. According to Property 2, $\omega \leqslant \tilde{\omega} + \bar{c} - 1$. Moreover, $\tilde{\omega} = \omega_1^*$ and $\omega_1^* \leqslant \omega^*$, hence, the lemma holds. $\square$

Now we can state a general result on PO-schedules.

Theorem 5. *Consider some problem with* $\bar{c} \geqslant 1 \geqslant \underline{c}$. *Optimal PO-schedules for* UECT *intrees have an absolute sensitivity bound of at most* $\bar{c} - \underline{c}$.

Proof. Let $\widetilde{S}$ be some PO-schedule which is optimal in the UECT case, and S (makespan ω) the resulting schedule when actual delays are reported in $\widetilde{S}$. Let S_{SCT} (makespan ω_{SCT}) and S_{LCT} (makespan ω_{LCT}) be the schedules obtained from S when respectively, the communication delays larger than one are set to one and the communication delays smaller than one are set to one.

As $\widetilde{S}$ is processor-ordered Fig. 6:

- $\omega_{\mathrm{LCT}} \leqslant \omega_1^* + \bar{c} - 1$ (from Lemma 5), and,
- $\omega_{\mathrm{SCT}} \leqslant \omega_{\mathrm{SCT}}^* + 1 - \underline{c}$ (from Lemma 3).

Two cases may occur.

If $\omega_{\mathrm{SCT}}^* \geqslant \omega_0^* + \underline{c}$, then $\omega_0^* = \omega_1^* - 1$, and we obtain:

$$\omega \leqslant \omega_{\mathrm{LCT}} \leqslant \omega_1^* + \bar{c} - 1 = \omega_0^* + \bar{c} \leqslant \omega_{\mathrm{SCT}}^* - \underline{c} + \bar{c} \leqslant \omega^* + \bar{c} - \underline{c}$$

Otherwise, according to Lemma 4, $\omega_{\mathrm{SCT}}^* = \omega_0^* = \omega_1^*$, and we obtain:

$$\omega \leqslant \omega_{\mathrm{LCT}} \leqslant \omega_1^* + \bar{c} - 1 = \omega_{\mathrm{SCT}}^* + \bar{c} - 1 \leqslant \omega^* + \bar{c} - 1 \leqslant \omega^* + \bar{c} - \underline{c} \qquad \square$$

Remark. The bound $\bar{c} - \underline{c}$ is tight. Indeed, $\forall \bar{c}, \underline{c}$ with $\bar{c} \geqslant 1 \geqslant \underline{c}$, a task graph (tree) exists such that $\omega = \omega^* + \bar{c} - \underline{c}$ (see Fig. 7).

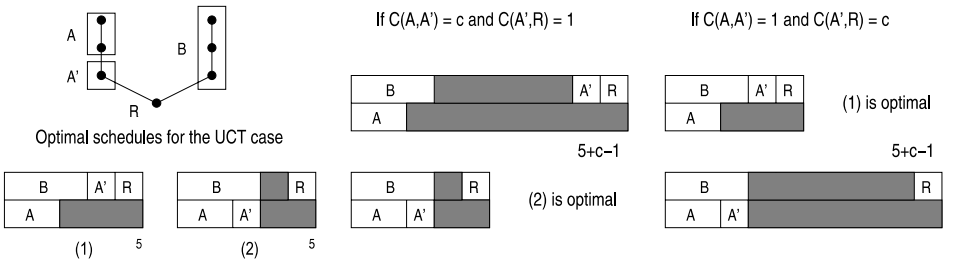

Fig. 6. The absolute sensitivity bound is $c - 1$. A special case could be $c = \bar{c}$.

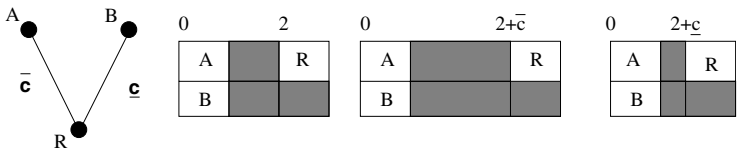

Fig. 7. Tightness of the absolute bound.

5. Sensitivity analysis of known algorithms

There are four optimal algorithms presented in the literature. We will first briefly recall the first three that are not processor-ordered.

Lawler's algorithm [11] builds a delay-free tree and then applies a level-by-level strategy. Veldhorst [13] builds identical schedules with a different algorithm. Picouleau [9] presents an algorithm in two phases. It first builds clusters of tasks, then in the second phase (sequencing) these clusters can be broken.

None of these algorithms build PO-schedules. Furthermore, examples can be provided (see research report [23]) showing that their relative sensitivity bound for arbitrary communication delays reaches the value of $\frac{\bar{c}+1}{2}$, which is the worst possible (see Theorem 4).

In the following pages we study the sensitivity of one particular scheduling algorithm, building optimal schedules for UECT intrees: CBoS (standing for *clustering based on subtrees*) [22]. This algorithm builds clusters of tasks with respect to a load balancing indication. All the clusters are assigned to one processor. Hence, the schedules built by CBoS are PO-schedules. Moreover, CBoS produces optimal schedules so, Theorem 1 can be extended to the case of UECT intrees.

Theorem 6. *For any instance of* $P2 \mid tree, \text{UECT} \mid C_{\max}$, *there always exists a* PO-*schedule that is optimal.*

Theorems on PO-schedules imply that CBoS has a tight absolute sensitivity bound of $\bar{c} - \underline{c}$ for arbitrary communication delays.

6. Experiments

We have previously shown that CBoS is the most robust of the currently available algorithms for scheduling trees with communication delays, in the sense that it achieves the tightest bounds and is not dominated by any other algorithm. Numerical tests were conducted to confirm these theoretical results. The first experiments presented compare CBoS and Lawler's algorithm. As will be shown, when actual communication delays are unknown before the execution, CBoS achieves better results than Lawler's algorithm.

6.1. CBoS vs. Lawler's algorithm

The experiment was conducted assuming a fixed number of tasks ($n = 20, 50, 100$) and a fixed maximum breadth for the tree (maximum number of tasks at the same level, respectively 5, 8, 12). One thousand trees are randomly generated for each case. Communication times are uniformly taken in four different intervals : $[1,2]$, $[1,5]$, $[1,10]$, $[0,2]$. For each instance the ratio $R = \frac{\omega_{\text{CBoS}}}{\omega_{\text{Lawler}}}$ is computed. Table 1 provides five measures: the maximum, the minimum and the mean values of R, and the number of

Table 1
Comparison of Lawler's algorithm and CBoS

Interval of actual delays	Number of tasks	Maximum value of R	Minimum value of R	Mean value	Percentage of time $R > 1$	Percentage of time $R < 1$
$[1, 2]$	20	1.07	0.84	0.98	16.1	61.6
	50	1.03	0.87	0.97	4.3	86.6
	100	1.02	0.91	0.98	1.7	92.4
$[1, 5]$	20	1.14	0.59	0.93	17.7	70.4
	50	1.04	0.59	0.82	1.1	98.8
	100	1.00	0.64	0.85	0.0	100.0
$[1, 10]$	20	1.19	0.39	0.89	17.0	73.7
	50	1.06	0.41	0.68	1.2	98.3
	100	0.95	0.46	0.64	0.0	100.0
$[0, 2]$	20	1.14	0.83	0.99	14.3	46.5
	50	1.07	0.88	0.98	6.8	72.3
	100	1.02	0.93	0.99	11.1	63.6

times (in percentage) Lawler's algorithm was strictly better ($R > 1$) and the number of times CBoS was strictly better ($R < 1$).

As expected, the performance ratio can be very poor for Lawler, with a minimum value of 0.39. Its mean value also is significant, especially when $\bar{c} \geqslant 5$. When $\bar{c} = 5$ or 10 and $m = 100$, CBoS is always strictly better. Note also that admitting small communication times ($c \in [0, 2]$) does not change the results (compare with $c \in [1, 2]$) although in that case Lawler's algorithm should theoretically be better. The strict SCT case is not presented: indeed, both algorithms result in the same makespan in more than 99.8% of the instances whatever the number of tasks!

6.2. Behaviour of CBoS sequencing

The theoretic results of the previous sections and the numerical tests above justify the use of CBoS at least to compute the task assignment off-line. It was mentionned earlier that in most cases it is unadvisable (or impossible) to change the assignment at execution time. Now at execution time we can choose to keep CBoS sequencing, or try to adjust the sequencing to the actual communication times. We present here results on the average and maximum losses of performance induced by CBoS sequencing, when compared with the optimal one.

A basic Branch and Bound algorithm was used to compute the optimal sequencing. The problem is NP-Hard for PO assignments. Indeed, it simply reduces to the two machines flow shop with transportation costs, which is strongly NP-Hard [17]. Branching is done by ordering the roots of the subtrees assigned to P_s. The computation of the lower bound is then equivalent to solve an easy one machine problem.

Table 2
Experimental behaviour of CBoS with respect to optimality

Interval of actual delays	Mean loss of performances	Maximum loss (measured)	Maximum loss (theory)	Number of optima
[1, 2]	0.59	6.02	6.25	58
[1, 5]	4.82	19.28	25	22
[1, 10]	12.90	43.12	56.25	7

All measures are in percentage.

The experiment was conducted assuming a number of tasks fixed at 30 and a maximum breadth of 6. One hundred trees were randomly generated. Large actual communication times were assumed. These times were uniformly taken in three different intervals : [1, 2], [1, 5], [1, 10]. The average and largest losses of performance are given in Table 2 (in percentage), with the number of times CBoS found the optimal makespan. The B&B algorithm occasionally failed to exhibit the optimal sequencing (respectively 5, 3 and 0 times) because the number of nodes of the search tree became too large when the number of effective communications was close to 10. This sometimes happenned with CBoS sequencing even with 30 tasks, and became more frequent when n increases. These cases were removed from the table.

Let us first consider the case of very large actual communication times (up to 5 or 10 times the estimation). It is no surprise that CBoS can behave badly, as it forces P_r to wait for some communications. However, these results must be considered together with the absolute bound presented in previous section: remember that the absolute makespan difference will never exceed $\bar{c} - 1$, between the CBoS schedule and an optimum schedule. However, *this bound is reached also when the CBoS assignment is fixed*. Indeed, consider the case of two identical subtrees A and B of size $\bar{c} - 1$, immediately preceding the root, and both executed on P_s. CBoS sequences them indifferently. But actual delays might be of 1 and $\bar{c}$ for the two effective communications, and the makespan difference may then reach the above bound.

From this bound we derive a maximum loss of performance (in percentage) of $\frac{c-1}{\omega^*}$. For trees with 30 tasks, the optimal makespan is at least 16, so this maximum percentage is equal to 56.25% for $\bar{c} = 10$, and to 25% for $\bar{c} = 5$. These upper values of percentages were not reached for the set of 100 generated trees, and the mean behaviour is far from these values.

When the largest actual communication delay do not exceed 2, the mean loss of performance is not significant, even if the maximum value of the loss can sometimes be close to the theoretical bound (6.25% in that case). Very often CBoS remains optimal.

It is clear that if large deviations in the communication delays are assumed, a fully on-line sequencing will be preferred. But if it is not possible in practice, or if the communication delays are expected to remain reasonably small, or if the number of tasks is very large in comparison with the maximum actual value of the communication delay, then CBoS sequencing provides good average performances.

7. Conclusions

In this paper, we have characterized the sensitivity of algorithms for scheduling trees with communication delays on two processors. The study focuses on changes concerning communication delays. We have shown that the best absolute sensitivity bound is $\bar{c} - \underline{c}$ and that the worst relative sensitivity bound is equal to $\frac{\bar{c}+1}{2}$. We have also proved that no algorithm can present better robustness than CBoS since it verifies the absolute bound. This is due to the fact that CBoS is processor-ordered. By contrast the three other known algorithms reach the worst relative bound. It is shown in the paper that algorithms producing processor-ordered schedules are dominant when all communication delays are equal to 0, or equal to 1. Unfortunately, they are not dominant in the SCT case.

In [7] a sensitivity analysis of list-scheduling algorithms was presented. When changes in processing times are considered, the author shows that the sensitivity of a schedule increases with its quality. Fortunately, our work shows that when changes in communication delays are considered, some algorithms can provide optimal schedules for estimated values while being the most robust.

Further directions of research will focus on the sensitivity analysis where communication delays and/or execution times may vary, and when the number of available processors is equal to m. The goal is to identify cases for which optimality and robustness are compatible.

References

[1] T.C. Hu, Parallel sequencing and assembly line problems, Operations Research 6 (9) (1961) 841–848.
[2] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM Journal on Applied Mathematics 2 (17) (1969) 416–429.
[3] V.J. Rayward-Smith, UET scheduling with interprocessor communication delays, Discrete Applied Mathematics 18 (1987) 55–71.
[4] P. Chrétienne, A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints, European Journal of Operational Research 43 (1989) 225–230.
[5] E. Lawler, J.K. Lenstra, A.H.G, Rinnooy Kan, D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, Technical Report BS-R8909, CWI Amsterdam, 1989.
[6] C.H. Papadimitriou, M. Yannakakis, Towards an architecture-independent analysis of parallel algorithms, SIAM Journal of Computing 19 (2) (1990) 322–328.
[7] A. Wagelmans, Sensitivity Analysis in Combinatorial Optimization, Ph.D. Thesis, Erasmus Universiteit Rotterdam, 1990.
[8] A. Jakoby, R. Reischuk, The complexity of scheduling problems with communication delays for trees, in Lecture Notes in Computer Sciences 621 (1992) 165–177.
[9] C. Picouleau, Etude des problèmes d'optimisation dans les systèmes distribués, Ph.D. Thesis, Paris VI, 1992.
[10] R. Saad, Scheduling with communication delays, Technical Report 754, LRI, University of Paris-Sud, Paris, France, 1992.
[11] E.L. Lawler, Scheduling trees on multiprocessors with unit communication delays, Workshop on Models and Algorithms for Planning and Scheduling Problems, Villa Vigoni, Lake Como, Italy, June 14–18, 1993.
[12] B. Veltman, Multiprocessor scheduling with communication delays, Ph.D. Thesis, University of Technology of Eindhoven, Department of Operations Research, Amsterdam, The Netherland, 1993.

[13] M. Veldhorst, A Linear Time Algorithm for Scheduling Trees with Communication Delays Optimally on two Machines, Technical Report RUU-CS-93-04, Department of Computer Sciences, Utrecht, 1993.
[14] Ph. Chrétienne, C. Picouleau, Scheduling with communication delays: a survey, in: P. Chrétienne, E.G. Coffman, J.K. Lenstra, Z. Liu (Eds.), Scheduling Theory and its Applications, John Wiley Ltd, 1994.
[15] A.W.J. Kolen, A.H.G. Rinnooy Kan, C.P.M. van Hoesel, A.P.M. Wagelmans, Sensitivity analysis of list scheduling heuristics, Discrete Applied Mathematics 2 (55) (1994).
[16] B. Penz, C. Rapine, D. Trystram, Sensitivity analysis of scheduling algorithms, European Journal of Operational Research 134 (2001) 606–615.
[17] P. Brucker, T.C.E. Cheng, S. Knust, N.V. Shakhlevich, Complexity Results for Shop Problems with Transportation Delays, Osnabrücker Schriften zur Mathematik, Reihe P, No. 228, 2001. Available from: <http://www.mathematik.uni-osnabruck.de/staff/phpages/bruckerp.rdf.html>.
[18] A. Gerasoulis, T. Yang, Applications of graph scheduling techniques in parallelizing irregular scientific computations, in: A. Ferreira, J. Rolim (Eds.), Parallel Algorithms for Irregular Problems: State-of-the-Art, Kluwer Academic Publishers, 1995, pp. 245–267, Chapter 13.
[19] J.K. Lenstra, M. Veldhorst, B. Veltman, The complexity of scheduling trees with communication delays, Journal of Algorithms 20 (1996) 157–173.
[20] T.A. Varvarigou, V.P. Roychowdhury, P. Kailath, E.L. Lawler, Scheduling in and out forests in the presence of communication delays, IEEE Transaction on Parallel and Distributed Systems 7 (10) (1996).
[21] A. Moukrim, E. Sanlaville, F. Guinand, Scheduling with communication delays and on-line disturbances, Springer Lecture Notes in Computer Science 1685 (1999) 350–357.
[22] F. Guinand, D. Trystram, Optimal scheduling of UECT trees on two processors, RAIRO Operation Research 34 (2000) 131–144.
[23] F. Guinand, A. Moukrim, E. Sanlaville, Sensitivity Analysis of Tree-Scheduling Algorithms, Technical Report 2002, Université de Technologie de Compiègne, 2002.
[24] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer, Seti@Home: an experiment in public resource computing, Communication of the ACM 45(11) (2002) 56–61.
[25] The European DataGrid: Project Overview and Testbed Experience, 2nd ApGrid Workshop, Taipei, Taiwan, May 2002.
[26] I. Foster, C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, 11(2), 115–128, 1997. Web: http://www.globus.org.
[27] G. Fedak, C. Germain, V. Néri, F. Cappello, XtremWeb: a generic global computing system, Proceedings of CCGrid 2001, may 2001.
[28] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Seymour, K. Sagi, Z. Shi, S. Vadhiyar, User's guide to Netsolve v 1.4.1, Technical Report No ICL-UT-02-05. University of Tennessee, Knoxville, 2002.