ELSEVIER

Performance Evaluation 41 (2000) 1–22

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

A hierarchical approach for bounding the completion time distribution of stochastic task graphs

Michele Colajanni [a], Francesco Lo Presti [b,1], Salvatore Tucci [b,*,2]

[a] *University of Modena, Modena 41100, Italy*

[b] *Dipartimento di Informatica, Sistemi e Produzione, Universita di Roma "Tor Vergata", Rome 00133, Italy*

Received 3 August 1999; received in revised form 2 October 1999

Abstract

The analytical evaluation of the completion time distribution of a general directed acyclic graph (DAG) is known to be an NP-complete problem. In this paper we present a new algorithm, named *Tree_Bound* , for the evaluation of bounds on the completion time of stochastic graphs assuming ideal conditions for shared resources and independent random variables as task execution times. The Tree_Bound method uses a hierarchical approach that first gives a tree-like representation of the graph, and then evaluates lower and upper bounds through a single visit of the tree. As lower bound the method takes the distribution of an embedded series–parallel graph which is evaluated by means of a simple recursion. The upper bound is based on a hierarchical application of other bounding techniques. In this paper, we use the Shogan algorithm because its determinism allows us to demonstrate some interesting properties of the Tree_Bound method.

Indeed, through stochastic ordering and stochastic comparison techniques, we demonstrate analytically that our approach provides tighter bounds than Shogan's and Yazici-Pekergin's bounds. On the other hand, we cannot compare formally the Tree_Bound accuracy to that of other important methods, such as Kleinöder and Dodin, because of their non-determinism. Various empiric comparisons show that the Tree_Bound algorithm provides analogous or superior results than heuristics derived from main non-deterministic methods. Moreover, the Tree_Bound algorithm keeps linear complexity and avoids non-determinism. Finally, it represents a useful basis for the combination of different bounding techniques which seems the only way to achieve even tighter bounds on the completion time distribution of stochastic graphs.

Keywords: Completion time distribution; Directed acyclic graphs; Performance analysis; Stochastic bounds; Stochastic ordering

* Corresponding author. Tel.: +39-06-72597385; fax: +39-06-72597460.

E-mail addresses: colajanni@unimo.it (M. Colajanni), tucci@uniroma2.it (S. Tucci)

[1] Now on leave at AT&T Labs — Research, Florham Park, NJ, USA.

[2] Now on leave as Director of the Office for Informatics, Telecommunications and Statistics of the Italian Prime Minister, Rome, Italy.

0166-5316/00/$ – see front matter
PII: S0166-5316(99)00054-1

1. Introduction

Task graphs are encountered in many models used in operations research and resource management, such as PERT diagrams, scheduling, computer and network models. Graph models are often used also to study activities that contain some concurrency. For example, a directed acyclic graph (DAG) can describe parallel computations where nodes represent tasks and directed arcs represent synchronization constraints among tasks [7,8,15,17,18]. A concurrent activity may be affected by queuing delays and synchronization delays. The former results when two or more tasks compete for the same resources, while the latter occurs when a task has to wait for the completion of other tasks. A DAG may be analyzed for mean completion time, for time of the shortest path and for various other performance measures. This paper analyzes the *completion time distribution* assuming ideal conditions for shared resources. The focus is on synchronization constraints induced by task dependencies.

If we consider tasks with constant execution times, the completion time can be easily calculated. Conversely, when we consider activities with random execution times, the analysis becomes extremely difficult [6,24] unless we restrict to graphs with a *series–parallel* (SP) structure [18]. Sahner and Trivedi demonstrate that these graphs can be solved with linear complexity through a combination of the task distributions by means of multiplication and convolution operations.

If we refer to arbitrary DAGs, the analytical evaluation of the completion time is very costly even for small graphs in which the task execution times are described by polynomial functions [16]. Therefore, many approaches evaluate performance mean values through analytical approximate methods and/or assuming exponential distributions for the task times [12,21,23]. An alternative is to obtain upper and lower bounds for the mean [5,13,15,17,22] or the distribution of the completion time. In this paper, we refer to this latter approach which is harder to investigate than the former, but even much more powerful because it allows us to compute the intervals in which any performance measures lie for any distribution of the task times.

Bounding methods for the mean and distribution of the completion time have been investigated mainly for the PERT networks [5,6,10,19]. Devroye [5] uses only the first moment and the variance of the task completion times to obtain bounds for the mean runtime of the whole graph. Shogan [19] and Kleindorfer [10] provide lower and upper bounds for the completion time distribution by means of recursive equations and exploiting properties of associated random variables. Dodin [6] obtains upper bounds for the mean completion time by transforming the original DAG into an SP graph through node duplications. Kleinöder [11] obtains SP graphs for lower and upper bounds by removing and adding arcs, respectively. Both Dodin's and Kleinöder's methods are inherently non-deterministic, and no proposed heuristics guarantees an accurate solution within a given complexity. Yazici-Pekergin and Vincent [24] propose the mean service time of the critical path as a lower bound. However, this metrics does not give good results for large task graphs because it considers only the first moment of the distributions and no parallelism. Their upper bound is based on independent path approximation in which the task time distributions are replaced by exponential distribution functions with the same first moment. The product of the distributions of all paths from a source to a sink node does not give a tight upper bound (see also some formal considerations in the last part of Section 2.2.3).

We obtain lower and upper bounds for the completion time distribution through stochastic ordering for random variables [20]. The proposed bounding procedure, named *Tree_Bound* method, consists of two steps:

1. We locate an embedded SP subgraph in the original graph and derive a tree-like representation graph. (To this purpose, we extend the notion of representation trees which are adopted for the analysis of SP

graphs [18].) The tree-like representation allows us to represent a graph as a hierarchy of subgraphs connected in series or in parallel.
2. Both lower and upper bounds are obtained through a single post-order visit of the tree-like representation graph.

The main contributions of this paper are pointed out by the two parameters that denote the quality of any bounding approach:

Algorithm complexity. We show that all main methods in the literature use a similar approach. They aim at transforming the original DAG into an SP graph by adding/removing arcs or nodes. Then, they obtain lower and upper bounds through the evaluation of the completion time distribution of the SP graph.

However, the search for the best SP graph that bounds the original DAG is an NP-complete problem. Since main bounding techniques are inherently non-deterministic, they have been implemented only as heuristics that do not achieve the approximate solution with a pre-determined complexity and accuracy. Conversely, once chosen the transformation approach (Section 3.1), the Tree_Bound method yields the tree-like representation graph in a deterministic way within linear complexity, and evaluates both lower and upper bounds in a single visit of the data structure.

Accuracy of the solution. We analytically demonstrate that the Tree_Bound method gives always tighter bounds than those obtained by Shogan's algorithm. This formal demonstration cannot be carried out for other techniques because they have been implemented as heuristics only. Therefore, their accuracy highly depends on graph topology and task distribution functions. Our experiments show that no approach is always the best. However, besides the improvements with respect to the Shogan method, Tree_Bound algorithm provides analogous or better results than Dodin and Kleinöder methods for various task graph examples. Finally, it is worth to observe that the hierarchical principle of the Tree_Bound approach could allow us to achieve even tighter bounds by combining different algorithms on different subgraphs depending on topology features. This paper focuses on Shogan's algorithm because of its determinism that allows us to demonstrate analytically the improvements achievable by Tree_Bound.

The paper is organized as follows. Section 2 describes the task graph model, and shows that the proposed bounding methods are all based on the same principle, that is, they attempt to transform the original DAG into an SP graph by adding and removing arcs or nodes. Section 3 presents the Tree_Bound method for the evaluation of lower and upper bounds on the completion time distribution. Section 4 formally demonstrates that the Tree_Bound method achieves more accurate bounds than the Shogan bounds. Section 5 compares the bounding methods applied to various task graphs. Section 6 concludes the paper with some final remarks.

2. Model for stochastic bounds

2.1. Definitions

We model concurrent activities as a directed and acyclic graph, also called *task graph*, where the nodes represent tasks and the arcs precedence constraints between tasks. We do not assume any distribution for the random variables modeling task execution times. This is an important quality of our paper because the common use of exponentially distributed execution times permits mathematical tractability but it is not representative of actual concurrent activity behavior [1]. A DAG is given by a quadruple $G = (N, A, S, T)$ where N is the set of tasks, A the set of arcs ($A \subset N \times N$), S the *source set* containing tasks that have no predecessors ($S \subseteq N$), and T is the *terminal set* containing tasks that have no successors ($T \subseteq N$).

Let $p(i)$ and $s(i)$ be the set of immediate predecessors and successors of $i \in N$, respectively. It holds, $p(i) \subset N$ and $s(i) \subset N$. A node i precedes a node j if there is a sequence $i_1, \ldots, i_n$ such that $i_1 = i$, $i_k \in s(i_{k-1})$ for $k = 2, \ldots, n$, and $i_n = j$.

The random variables T_i, $i \in N$, denoting the task execution times are independent with known distribution $C_i(t)$. Moreover, we assume ideal conditions for the system resources. This allows us to isolate and evaluate the impact of non-deterministic execution time and synchronization constraints on performance measures.

Considering the overall completion time T_G as the performance measure of interest, the problem is to find its distribution $F(t) = \Pr[T_G \leq t]$ for $t \geq 0$. If we denote with $P = \{p_1, \ldots, p_n\}$ the set of all paths in the graph, and with π_i the execution time distribution of the path $p_i \in P$, we have $\pi_j = \sum_{i \in p_j} T_i$. Since the graph completion time is given by the duration of the longest path, we have $T_G = \max_j \pi_j$, and $F(t) = \Pr[\max_j \pi_j \leq t]$.

The bound evaluation uses the *stochastic ordering* for random variables [20], and properties of *associated random variables* [2]. The *stochastic ordering* is a partial order relationship among random variables and distribution functions. Given two random variables X and Y with distribution F_X and F_Y, X is stochastically smaller than Y if for every $u \geq 0$ we have $F_Y(u) \leq F_X(u)$. In such a case, we can write $Y \geq_{st} X$ and $F_Y \geq_{st} F_X$. The basic theorem of this theory says that given two distribution families $(F_1, \ldots, F_n)$ and $(G_1, \ldots, G_n)$ such that $F_i \geq_{st} G_i$, for $i = 1, \ldots, n$, the following inequalities are true:

$$\prod_{i=1,\ldots,n} F_i \geq_{st} \prod_{i=1,\ldots,n} G_i, \tag{1}$$

$$\bigotimes_{i=1,\ldots,n} F_i \geq_{st} \bigotimes_{i=1,\ldots,n} G_i, \tag{2}$$

where $\otimes$ is the convolution operator.

The variables of a random vector $X = \{X_1, \ldots, X_n\}$ are (positively) *associated* if for all monotone non-decreasing functions $f, g : \Re^n \to \Re$ for which the expectations exist, the following inequality is true:

$$\mathrm{Cov}[f(X), g(X)] \geq 0. \tag{3}$$

In the following we will use main properties of associated random variables, that is, independent random variables are associated; the union of independent sets of associated random variables forms a set of associated random variables; any subset of a set of associated random variables forms a set of associated random variables; any monotone non-decreasing function of associated random variables generates a set of associated random variables.

Moreover, the variables of the random vector $\bar{X} = \{\overline{X_1}, \ldots, \overline{X_n}\}$ form an independent version of $X = \{X_1, \ldots, X_n\}$, if $\{\overline{X_1}, \ldots, \overline{X_n}\}$ are mutually independent, and X_i and $\overline{X_i}$ have the same probability distribution. An important result is that associated random variables can be stochastically compared with their independent versions, that is, if X is a vector of associated random variables and $\bar{X}$ is its independent version, then we have

$$\bar{X} \geq_{st} X, \tag{4}$$

$$\max_{i=1,n} \overline{X_i} \geq_{st} \max_{i=1,n} X_i, \tag{5}$$

$$\min_{i=1,n} \overline{X_i} \leq_{st} \min_{i=1,n} X_i. \tag{6}$$

2.2. Main results in literature

In this section we summarize main results for stochastic bounding methods for task graphs by giving a unifying new interpretation of the methods of Kleinöder [11], Dodin [6] and Shogan [19]. In particular, we show that all of them are based on the same principle, that is, the original DAG is transformed into an SP graph for each bound by adding/deleting arcs or nodes. Lower and upper bounds are then obtained from the evaluation of the completion time distribution of the SP graphs. Reducing a graph to an SP graph is important because its completion time distribution can be computed with linear complexity by combining the node distribution functions through convolution and multiplication operations [18]. If nodes or arcs are added, the completion time of the reduced SP graph is an upper bound of the original graph. Dually, removing nodes or arcs leads to a lower bound.

2.2.1. Kleinöder's algorithm

Kleinöder modifies the original DAG by adding and removing arcs. The goal is to derive an SP graph for which the mean completion time and, most importantly, the completion time distribution is easily computable. The insertion of arcs increases intertask synchronizations thus leading to an upper bound of the execution time, while the deletion of arcs causes less dependencies and lower execution times. The approach of adding/removing arcs is inherently non-deterministic because generally there are many possibilities to reach the SP form of a DAG. Hence, the same method can lead to more than one lower and upper bound, and the search for the tightest bounds becomes an NP-complete problem. Only heuristic approaches of the Kleinöder algorithm have been implemented. For example, the heuristics used in [4,9] consists of the following two steps, cyclically repeated:

1. Reduce the SP parts of the graph until no more reduction is possible.
2. For the upper bound, search for a pair of nodes that allows a graph reduction by adding new arcs.
   Analogously, remove arcs for the lower bound search.

The main problem is that step 2 is not deterministic. Several choices can be made, each of them leading to a more or less accurate solution. The method should aim to minimize the number of added (or removed) arcs, because this approach gives a tighter bound. However, there is no guarantee that certain choices lead to the optimum bound, and backtracking approaches are not allowed to avoid exponential complexity. Nevertheless, the proposed heuristics have a polynomial complexity which is not adequate to very high levels of parallelism, especially if compared to the linear complexity of the Dodin algorithm.

2.2.2. Dodin's algorithm

The Dodin method evaluates the upper bound for the completion time of PERT networks. It modifies the PERT network by replicating one or more arcs until the graph turns into an SP form. When applied to stochastic task graphs, this method is equivalent to a node duplication. A node i can be duplicated if it satisfies either $|p(i)| \leq 1$ and $|s(i)| \geq 2$, or $|s(i)| \leq 1$ and $|p(i)| \geq 2$.

For the *duplication* of a node i, first add a new node i', then:

1. Choose one of the successor nodes i_j of i, and add the arc from the predecessor of i to node i' and the arc (i', i_j). Then, remove the arc (i, i_j).
2. Choose one of the predecessor nodes i_j of i, and add the arc from i_j to the node i' and the arc from i_j to the successor of i. Then, remove the arc (i_j, i).

Even the original Dodin algorithm is inherently non-deterministic because various nodes may satisfy duplication conditions and there are several ways to choose the successor (case 1) or the predecessor (case

2). The heuristics proposed by Dodin [6], called *duplication process*, works as follows. First, it carries out all admissible series–parallel reductions. Then, it duplicates one source node that satisfies duplication conditions. Both steps are repeated until only one node remains.

2.2.3. Shogan's algorithm

The Shogan method obtains lower and upper bounds for the completion time distribution of PERT networks, where arcs model tasks, and nodes represent synchronization constraints [19]. In [14] we propose a version of the Shogan algorithm for stochastic task graphs. This version points out that, analogous to Kleinöder and Dodin, even the Shogan algorithm is based on a transformation of the original DAG into an SP graph. As example, let us describe in terms of our framework the upper bound estimation which is based on the following results.

Let T_i and $F_i(t)$ denote the completion time and the distribution of the completion time of the task $i \in N$, respectively. The random variables T_i are *associated* (see definition in Section 2.1). The upper bounds for task completion times are given by the following distributions:

$$\hat{F}_i(t) = F_i(t) = C_i(t), \quad i \in S, \tag{7}$$

$$\hat{F}_i(t) = \left[\prod_{j \in p(i)} \hat{F}_j(t) \right] \otimes C_i(t), \quad i \notin S, \tag{8}$$

$$\hat{F}(t) = \prod_{j \in T} \hat{F}_j(t). \tag{9}$$

The distribution F_i, $i \in N$, and F are stochastically smaller than the distribution $\hat{F}_i$ and $\hat{F}$, respectively. Hence, we can write

$$\hat{F}_i(t) \geq_{st} F_i(t), \quad i \in N, \tag{10}$$

$$\hat{F}(t) \geq_{st} F(t). \tag{11}$$

For a given task graph G, we demonstrate in [14] that there exists always an SP graph, called the *associated graph* $\hat{G}$, the completion time distribution of which is given by the Shogan upper bound $\hat{F}$. The associated graph can be recursively obtained by considering the definition of $\hat{F}$ in (7)–(9):

- For $i \in S$, let $\hat{G}_i$ be the graph containing the single node i.
- For $i \notin S$, let $\hat{G}_{p(i)}$ be the graph obtained by connecting the graphs $\hat{G}_j$, $j \in p(i)$ in parallel. Hence, $\hat{G}_i$ is the graph obtained by connecting $\hat{G}_{p(i)}$ in series with the node i.
- Let $\hat{G}_T$ be the graph obtained by connecting in parallel the graphs $\hat{G}_j$, $j \in T$, where T is the set of terminal nodes.

If $|T| > 1$, then $\hat{G}$ is given by a series connection of $\hat{G}_T$ with a *dummy* node, else $\hat{G}$ is given by $\hat{G}_T$ itself. Let us also define, for $M \subseteq N$, the graph $\hat{G}_M$ obtained by parallel connections of the graphs $\hat{G}_j$, for $j \in M$.

Because of the construction of $\hat{G}$, we can consider independently the replicated tasks in $\hat{G}$. Hence, we have the following result.

Lemma 1. *Given a graph G, the distribution of the completion time of its associated graph* $\hat{G}$ *is* $\hat{F}$. *Moreover, the distribution of the graph* $\hat{G}_i$ *is* $\hat{F}_i$ *, for* $i \in N$.

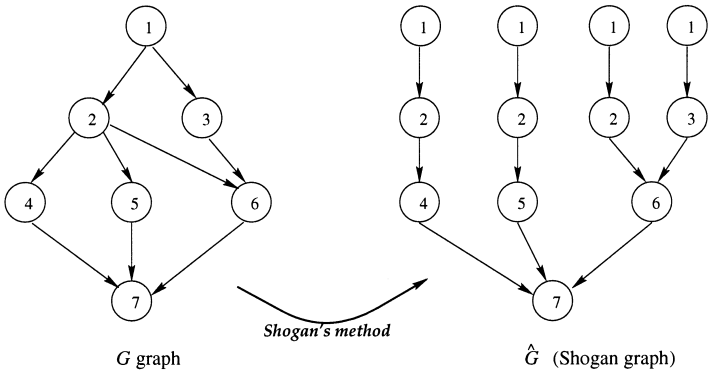

Fig. 1. The graphs G and $\hat{G}$ for the Shogan method.

By construction, the completion time of G is stochastically smaller than the completion time of the SP graph $\hat{G}$. This result shows that Shogan's method too is based on a transformation of the original DAG into an SP graph. The associated graph $\hat{G}$ of G can be viewed as a *reversed tree* (Fig. 1) with the following properties: each node $i \in N$ of G is in $\hat{G}$ as many times as the number of different paths departing from i in G; the graphs G and $\hat{G}$ share the same path, hence p_i is a path in G if and only if p_i is a path in $\hat{G}$.

In [24] it is shown that the completion time distribution F is stochastically smaller than the well-known *independent path approximation* distribution $\tilde{F}$. An immediate consequence of the latter property is the following stochastic inequality: $\tilde{F}(t) \geq_{st} \hat{F}(t) \geq_{st} F(t)$. It demonstrates the Shogan bounds to be always more accurate than the bounds given by the method of Yazici-Pekergin and Vincent, because the latter algorithm involves more independent paths than Shogan's method.

3. Tree_Bound method

In this section we propose a hierarchical method for the evaluation of lower and upper bounds on the completion time distribution of stochastic task graphs. This algorithm uses a new transformation approach of the original DAG which is based on the so-called *tree-like representation* of the graph.

3.1. Graph representation

SP graphs can be described by means of *representation trees* [18], where the leaves correspond to the nodes of the graphs, and the internal nodes are series or parallel combinations of their subtrees. This representation can be obtained in linear time with respect to the number of nodes of the representation tree.

The first step of our method aims to obtain a tree-like representation of the original DAG. Given a graph $G = (N, A, S, T)$, let us consider an SP subgraph $G^* = (N^*, A^*, S^*, T^*)$ embedded in G, such that $N^* = N$ and $A^* \subseteq A$. Let us consider a tree representation R^* for G^* where $R^* = (V^*, E^*)$. In this representation, the leaves correspond to the DAG nodes, while the internal nodes represent series and parallel combinations which are labeled by s and p, respectively. Whenever G is not SP, G^* can be viewed as a series–parallel skeleton of G, that is, A^* is strictly contained in A. By augmenting G^* with the arcs belonging to $A \backslash A^*$ we obtain again G. Therefore, we can represent G by simply augmenting R^* with the

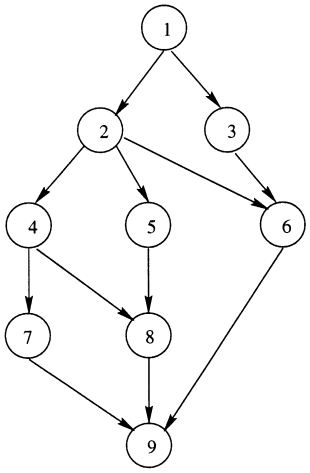

Fig. 2. An example of non-SP graph G.

arcs of $A \backslash A^*$. Let $R = (V, E)$ denote the resulting graph, where $V = V^*$ and $E = E^* \cup (A \backslash A^*)$. We consider R as the *tree-like representation* graph for G. In R we maintain the child–parent relationships that hold for R^*. In particular, $i \in V$ is a parent of $j \in V$ in R, if and only if $i \in V^*$ is a parent of $j \in V^*$ in R^*. Moreover, $j \in V$ is a child of $i \in V^*$ in R, if and only if $j \in V^*$ is a child of $i \in V^*$ in R^*.

Several possibilities exist to obtain the graph G^*. We choose two deterministic approaches named *to-series-reduction* and *to-parallel-reduction* that aim to remove from G those arcs that prevent series or parallel reductions, respectively. Let us describe how the *to-series-reduction* approach works by considering as example the graph G in Fig. 2.

1. We obtain an embedded SP graph G^* (Fig. 3) by removing from G the arcs (2,6) and (4,8) that prevent series reductions in G.
2. We give a *tree representation* of the graph G^* (Fig. 4).
3. We obtain a *tree-like representation* R of the original graph G by adding to the tree the arcs (2,6)

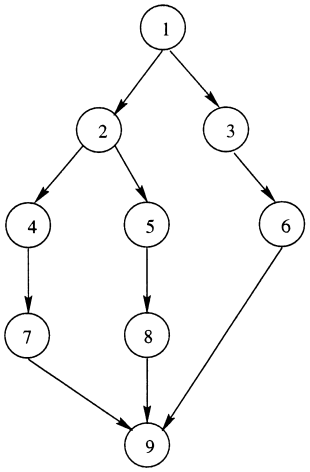

Fig. 3. The SP graph G^* embedded in G.

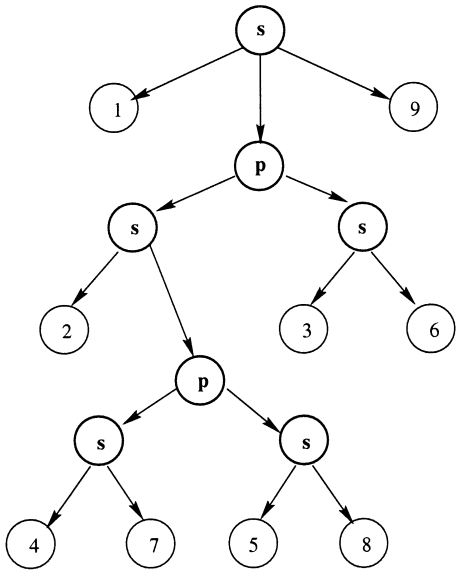

Fig. 4. The tree representation of the graph G^*.

and (4,8) which were removed at step 1 (Fig. 5). This representation describes the original DAG as a series–parallel combination of four subgraphs: three connected in series, one included in the second subgraph.

This representation allows us to obtain both lower and upper bounds on the completion time distribution. In particular, the distribution of G^* is a lower bound, while the upper bound is obtained through two steps.

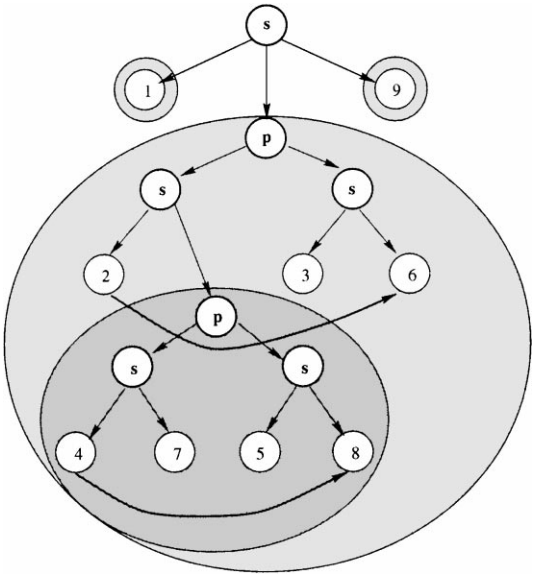

Fig. 5. The tree-like representation of the graph G.

We compute the upper bound of each subgraph by means of another bounding technique (here, we use Shogan's algorithm because of its determinism) and hierarchically combine these bounds in accordance with the tree structure.

3.2. Tree_Lower bound

We take the completion time distribution of the graph G^* as a lower bound for the completion time distribution. We recall that G^* is obtained from G by removing arcs until the graph becomes series–parallel reducible, hence its completion time is stochastically smaller than the overall completion time of G [8]. Unlike other heuristics based on the Kleinöder algorithm, we make deterministic graph modifications that lead to series or parallel reductions. However, this is not a constraint of our method that could be implemented even through a combination of the *to-series-reduction* and *to-parallel-reduction* approaches. This would introduce some non-determinism that could be solved by taking into account the graph topology. However, this issue is out of the scope of this paper.

The completion time distribution of G^* is evaluated through the tree representation method [18]. The algorithm requires a post-order visit of the tree during which we assign a distribution to each node. In particular, we assign to each leaf the distribution of the corresponding graph node. Internal nodes represent the series or parallel combinations of subgraphs. Accordingly, we convolute or multiply the distributions of the child nodes and assign the result as the distribution of the node itself. The distribution of the root node is the distribution of G^*, that we retain as the lower bound. Both visit and assignment are described by the following procedure, hereafter called Tree_LB:

```
procedure Tree_LB (k:node)
begin
   for j ∈ children(k) Tree_LB(j)
   if k has label s, then assign to k the distribution L_k = ⊗_{j∈children(k)} L_j
   if k has label p, then assign to k the distribution L_k = ∏_{j∈children(k)} L_j
end
```

3.3. Tree_Upper bound

The main idea behind the evaluation of the Tree_Upper bound is the following:

1. The tree-like representation isolates the non-SP parts of the graph. We replace each non-SP part with a node that has the distribution given by some bounding technique. In this paper, we use the Shogan algorithm because this allows us to give a formal demonstration of the improvements achievable by the Tree_Bound method. However, there is no theoretic limit to the use of other bounding techniques.
2. Through this approach, we obtain an SP graph for which we know the exact distribution for some nodes and upper bounds on the distributions for other nodes.
3. The distribution of the resulting SP graph can be easily computed using the method of Sahner and Trivedi [18].

We formalize the upper bounding approach by introducing the concept of *close* subgraphs for R and G. Let R_j^* ($j \in V^*$ defined in Section 3.1) denote the subtree of R with root j, and $(A \backslash A^*)_j$ the set of the arcs connecting nodes of R_j^* in G, that is, $(A \backslash A^*)_j = \{(l, m) \in A \backslash A^* | l, m \in V_j^*\}$. We define the graph $R_j = (V_j, E_j)$, with $V_j = V_j^*$ and $E_j = E_j^* \cup (A \backslash A^*)_j$. Among the nodes of R_j we keep same child–parent relationships that hold for R_j^*.

A graph $R_j = (V_j, E_j)$, $j \in V$, is *close* if for each arc $(l, m) \in (A \backslash A^*)$, $l \in V_j$ if and only if $m \in V_j$. R_j is the tree-like representation graph for the *close subgraph* of G denoted by $G_j = (N_j, A_j, S_j, T_j)$. In G_j we denote the predecessor $p(G_j)$ and successor set $s(G_j)$ with $s(l)$ and $p(m)$, $l \in T_j$, $m \in S_j$, respectively. A close graph G_j has the following properties: the source nodes of G_j have the same set of predecessors in G (that is, for each $l, m \in S_j$, $p(l) = p(m)$); the terminal nodes of G_j have the same set of successors in G (that is, for each $l, m \in T_j$, $s(l) = s(m)$); there are no arcs connecting nodes of G_j with nodes which are not in G_j except for those provided above.

Let us now define the *close subgraph reduction operation*. We can reduce a graph G, given one of its *close subgraphs* G_j, through the application of the close subgraph reduction operation as follows: *substitute G_j with a single node g_j such that $s(g_j) = s(G_j)$, $p(g_j) = p(G_j)$ and using the Shogan upper bound of G_j as distribution*.

In addition to the well-known series and parallel reduction operations, the upper bounding procedure uses the reduction operation for close subgraphs. Given the graph G, let R be its tree-representation graph. To calculate the upper bound, we visit the nodes of the graph R in post-order and assign distribution to nodes $j \in V$ if and only if R_j is a *close* graph. The upper bound is the distribution of the root node. Both visit and assignment are described by the following procedure, hereafter called Tree_UB:

```
procedure Tree_UB (k :node)
begin
  for j ∈ children(k) Tree_UB(j)
  if R_k is close then
    begin
      Denote with G_k the graph which has R_k as representation graph
      Carry out all series–parallel reductions in G_k, and let G'_k be the resulting graph
      Assign to node k the Shogan upper bound U_k of G'_k
      Remove the subgraph R_k except the node k
    end
end
```

The procedure works as follows. Given a graph G, the tree-like representation graph R is visited in post-order, in accordance with the parent–child relationship inherited from R^*. Whenever a visited node k is such that R_k is *close*, let G_k denote the subgraph of G having R_k as tree-like representation graph and proceed as follows. First, reduce all series–parallel subgraphs in G_k and let G'_k denote the reduced subgraph. Then, reduce G'_k in accordance with the close subgraph reduction operation. The procedure ends when we visit the root node of R and the graph is reduced to a single node. One property of our method is that the lower and upper bound evaluation can be jointly performed. We need to visit only once the tree-like representation and assign lower and upper distributions to each node. To this purpose, we use the following Tree_Bound procedure:

```
procedure Tree_Bound (k:node)
begin
  for j ∈ children(k)
    Tree_Bound(j )
  if k has label s, then assign to k the distribution L_k = ⊗_{j∈children(k)} L_j
  if k has label p, then assign to k the distribution L_k = ∏_{j∈children(k)} L_j
```

```
if R_k is close then
  begin
    Denote with G_k the graph which has R_k as representation graph
    Carry out all series–parallel reductions in G_k, and let G'_k be the resulting graph
    Assign to node k the Shogan upper bound U_k of G'_k
    Remove the subgraph R_k except the node k
  end
end
```

4. Accuracy of the Tree_Bound algorithm

In this section we formally demonstrate that Tree_Upper bound is tighter than the upper bound obtained by the Shogan method. Let F^{T} be the upper bound distribution assigned to the root node of R, and $\hat{F}$ the Shogan upper bound. The demonstration is based on the following theorems (to better follow the arguments, the reader may initially skip the proofs of Theorems 2 and 3).

Theorem 2. *Let $G_j = (N_j, A_j, S_j, T_j)$ be a close subgraph of the graph G. Let us denote with G' the graph resulting by substituting G_j with a single node g_j such that $s(g_j) = s(G_j)$ and $p(g_j) = p(G_j)$. Its distribution is given by the Shogan upper bound of G_j denoted with $Sh(G_j)$.*

We have $\hat{F}' \leq_{st} \hat{F}$, where $\hat{F}'$ and $\hat{F}$ are the Shogan upper bounds for G' and G, respectively.

Proof. The proof is based on three steps.

As first step, we give the equation for $\hat{F}'_{g_j}$ recalling that $p(g_j) = p(i)$ and $s(g_j) = s(i)$, for $i \in S_j$ and $j \in T_j$. It holds that

$$\hat{F}'_{g_j} = \left[\prod_{l \in p(g_j)} \hat{F}'_l \right] \otimes Sh(G_j). \tag{12}$$

From (1) and (2), for $k \in N$ only nodes preceding k are involved in the evaluation of $\hat{F}_k$. If g_j does not precede l in G', then $\hat{F}'_l(t) =_{st} \hat{F}_l(t)$ for $t \geq 0$, that is, the upper bound for the completion time distribution of this node is given by the same distribution in G and G'. Therefore, we can write $\hat{F}'_l =_{st} \hat{F}_l$, which implies $\hat{F}'_l \leq_{st} \hat{F}_l$. In particular, for $l \in p(g_j)$ we have $\hat{F}'_l =_{st} \hat{F}_l$, therefore

$$\hat{F}'_{g_j} = \left[\prod_{l \in p(g_j)} \hat{F}'_l \right] \otimes Sh(G_j) = \left[\prod_{l \in p(g_j)} \hat{F}_l \right] \otimes Sh(G_j). \tag{13}$$

As second step, we demonstrate that

$$\hat{F}'_{g_j} \leq_{st} \prod_{k \in T_j} \hat{F}_k. \tag{14}$$

We simplify the proof by giving a graph interpretation for this equation. In particular, $\hat{F}'_{g_j}$ can be considered the distribution of the graph in Fig. 6, that is, the series composition of the graph $\hat{G}_j$ with the graph $\hat{G}_{p(g_j)}$.

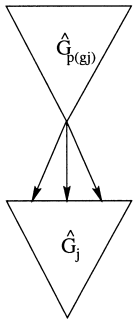

Fig. 6. Graph interpretation for $\hat{F}'_{g_j}$.

$\hat{G}_j$ is the associated graph for G_j having distribution $Sh(G_j)$ in accordance with Lemma 1 in Section 2.2.3. From the definition of *associated* graphs, we have that $\hat{G}_{p(g_j)}$ is obtained as parallel compositions of $\hat{G}_l$ for $l \in p(g_j)$, and has distribution $\prod_{l\in p(g_j)} \hat{F}_l$. Analogously, $\prod_{k\in T_j} \hat{F}_k$ is the distribution of $\hat{G}_{T_j}$.

We observe that for $l, m \in S_j$ we have $p(l) = p(m) = p(G_j) = p(g_j)$, hence $\hat{G}_{p(l)} = \hat{G}_{p(m)} = \hat{G}_{p(g_j)}$. By using the recursive procedure for the construction of the associated graph, it is easy to verify that $\hat{G}_{T_j}$ is given by the graph in Fig. 7. Clearly, the latter graph can be viewed as the associated graph of the former. Therefore, we can conclude that its distribution $\prod_{l\in T_j} \hat{F}_l$ is stochastically greater than $\hat{F}'_{g_j}$.

As third and last step, we demonstrate that $\hat{F}' \leq_{st} \hat{F}$. The proof based on induction aims at demonstrating that $\hat{F}'_m \leq_{st} \hat{F}_m$, for all $m \in N' = [N - N_j] \cup \{g_j\}$.

- For $m \in S'$, we have

$$\hat{F}'_m = C_m = \hat{F}_m \quad \Rightarrow \quad \hat{F}'_m \leq_{st} \hat{F}_m. \tag{15}$$

- For $m \notin S'$, we distinguish two instances after the demonstration that $\hat{F}'_k \leq_{st} \hat{F}_k$ for $k \in p(m)$ and $k \neq g_j$:
  1. If $g_j \notin p(m)$ in G', then for Lemma 1,

$$\hat{F}'_m = \prod_{k\in p(m)} \hat{F}'_k \otimes C_m \leq_{st} \prod_{k\in p(m)} \hat{F}_k \otimes C_m = \hat{F}_m. \tag{16}$$

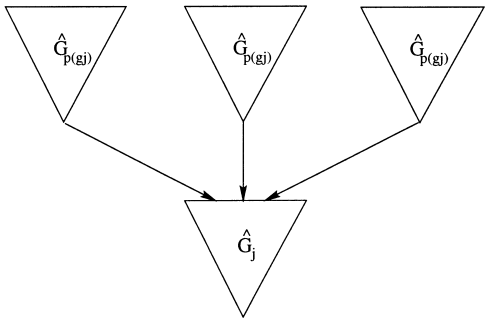

Fig. 7. Associated graph for $\hat{G}_{T_j}$.

2. If $g_j \in p(m)$ in G', then

$$\hat{F}_m = \left[\hat{F}'_{g_j} \prod_{k \in p(m) \setminus \{g_j\}} \hat{F}'_k \right] \otimes C_m =_{st} \left[\left(\prod_{l \in p(g_j)} \hat{F}'_l \otimes Sh(G_j) \right) \prod_{k \in p(m) \setminus \{g_j\}} \hat{F}'_k \right] \otimes C_m, \tag{17}$$

recalling that if $g_j \in p(m)$ in G', then $T_j \subseteq p(m)$ in G.

From Lemma 1 and inequality (14), we can write

$$\hat{F}_m = \prod_{k \in p(m)} \hat{F}_k \otimes C_m =_{st} \prod_{k \in p(m) \setminus T_j} \hat{F}_k \prod_{k \in T_j} \hat{F}_k \otimes C_m \geq_{st} \hat{F}'_m. \qquad \square \tag{18}$$

Theorem 3. *Let G_R denote the graph obtained by reducing series–parallel subgraphs into a given graph G. If $\hat{F}_R$ and $\hat{F}$ are the Shogan upper bounds for the completion time distribution of G_R and G, then $\hat{F}_R \leq_{st} \hat{F}$.*

Proof. Reducing series–parallel subgraphs consists in carrying out all feasible *graph reduction* operations. Hence, we have only to show that for the graph $G' = (N', A', S', T')$, which is obtained after a single graph reduction operation from G, we have $\hat{F}' \leq_{st} \hat{F}$, where $\hat{F}'$ is the Shogan upper bound for G'. We demonstrate this inequality by distinguishing the *series* from the *parallel* reductions.

Let us first consider that the graph operation is a *series reduction*. Let i and j be two nodes in series in G, and denote with $\widetilde{ij}$ the reduced node in G'. For $l \in p(\widetilde{ij})$, we have that $\hat{F}'_l =_{st} \hat{F}_l$, $p(\widetilde{ij}) = p(i)$ and $s(\widetilde{ij}) = s(j)$. Therefore, it follows that

$$\hat{F}'_{\widetilde{ij}} = \prod_{k \in p(\widetilde{ij})} \hat{F}'_k \otimes C_{\widetilde{ij}} = \prod_{k \in p(i)} \hat{F}_k \otimes C_{\widetilde{ij}} = \prod_{k \in p(i)} \hat{F}_k \otimes C_i \otimes C_j = \hat{F}_j. \tag{19}$$

This equation implies $\hat{F}' =_{st} \hat{F} \Rightarrow \hat{F}' \leq_{st} \hat{F}$.

Let us now consider that the graph operation is a *parallel reduction*. Let i and j be the two nodes in parallel in G, and denote with $\overline{\overline{ij}}$ the reduced node in G'. Recalling that $p(i) = p(j) = p(\overline{\overline{ij}})$ and $s(i) = s(j) = s(\overline{\overline{ij}})$, from (2) we have

$$\hat{F}_i = \prod_{l \in p(i)} \hat{F}_l \otimes C_i, \qquad \hat{F}_j = \prod_{l \in p(j)} \hat{F}_l \otimes C_j. \tag{20}$$

Since $\hat{F}'_l =_{st} \hat{F}_l$ for $l \in N$ (l preceding $\overline{\overline{ij}}$), the equation for $\hat{F}'_{\overline{\overline{ij}}}$ in G' is

$$\hat{F}'_{\overline{\overline{ij}}} = \prod_{l \in p(\overline{\overline{ij}})} \hat{F}'_l \otimes (C_i C_j) = \prod_{l \in p(i)} \hat{F}_l \otimes (C_i C_j). \tag{21}$$

Moreover, the following result is known (for example, a proof is in [6]):

$$\hat{F}_i \hat{F}_j \geq_{st} \hat{F}_{\overline{\overline{ij}}}. \tag{22}$$

We can now use induction to demonstrate that $\hat{F}' \leq_{st} \hat{F}$. The proof aims at showing that $\hat{F}'_m \leq_{st} \hat{F}_m$, for $m \in N' = [N - \{i, j\}] \cup \{\overline{\overline{ij}}\}$.

- For $m \in S'$, we have

$$\hat{F}'_m = C_m = \hat{F}_m \Rightarrow \hat{F}'_m \leq_{st} \hat{F}_m. \tag{23}$$

- For $m \notin S'$, we have to consider two instances, after the demonstration that $\hat{F}'_k \leq_{st} \hat{F}_k$, for $k \in p(m)$ and $k \neq \overline{\overline{ij}}$:
  1. If $\overline{\overline{ij}} \notin p(m)$, then

$$\hat{F}'_m = \prod_{k \in p(m)} \hat{F}'_k \otimes C_m \leq_{st} \prod_{k \in p(m)} \hat{F}_k \otimes C_m = \hat{F}_m. \tag{24}$$

  2. If $\overline{\overline{ij}} \in p(m)$, then for (1), (2) and (22), we have

$$\hat{F}'_m = \left(\prod_{k \in p(m) \setminus \{\overline{\overline{ij}}\}} \hat{F}'_k \right) \hat{F}'_{\overline{\overline{ij}}} \otimes C_m \leq_{st} \left(\prod_{k \in p(m) \setminus \{i,j\}} \hat{F}_k \right) \hat{F}_i \hat{F}_j \otimes C_m = \hat{F}_m, \tag{25}$$

  recalling that if $\overline{\overline{ij}} \in p(m)$ in G', then $i, j \in p(m)$ in G. □

Combining the results of Theorems 2 and 3, we can now demonstrate the main result concerning the Tree_Bound algorithm.

Theorem 4. *Given a graph G with completion time distribution F, let R be its tree-like representation graph. Moreover, let F^{T} and $\hat{F}$ denote the distribution of the root node R obtained through the Tree_Upper bound algorithm and the Shogan method, respectively. The following stochastic inequalities are true:*

$$F \leq_{st} F^{\mathrm{T}} \leq_{st} \hat{F}. \tag{26}$$

Proof. The Tree_UB procedure evaluates an upper bound through series–parallel and close subgraph reductions. Let us denote with $G^{(i)}$ the graph obtained at step i, where $G^{(0)} = G$. Moreover, let us denote with $F_{G(i)}$ and $\hat{F}_{G(i)}$ the distribution and the Shogan upper bound of the graph $G^{(i)}$, respectively. We have two instances.

If step i is a *series–parallel* reduction, then $F_{G(i)} =_{st} F_{G(i-1)}$ that implies $F_{G(i)} \geq_{st} F_{G(i-1)}$. Moreover, from Theorem 3 we have $\hat{F}_{G(i)} \leq_{st} \hat{F}_{G(i-1)}$.

If step i is a *close subgraph* reduction, then $F_{G(i)} \geq_{st} F_{G(i-1)}$. Moreover, from Theorem 2 we have $\hat{F}_{G(i)} \leq_{st} \hat{F}_{G(i-1)}$.

If we assume that the graph G is reduced to a single node in k steps, then we have the following chain of stochastic inequalities that demonstrates the theorem:

$$\begin{aligned} F &=_{st} F_{G(0)} \leq_{st} F_{G(1)} \leq_{st} \ldots \leq_{st} F_{G(k)} =_{st} F^{\mathrm{T}} \\ &=_{st} \hat{F}_{G(k)} \leq_{st} \hat{F}_{G(k-1)} \leq_{st} \ldots \leq_{st} \hat{F}_{G(0)} =_{st} \hat{F}. \end{aligned} \tag{27}$$

□

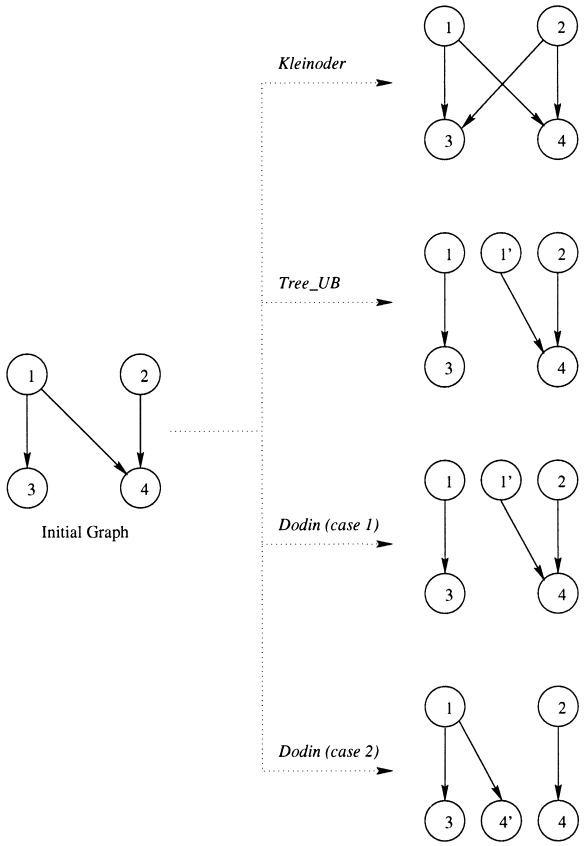

Fig. 8. How the bounding techniques work on the simplest form of non-SP graph?

5. Performance analysis and comparison

In this section we discuss the quality of the Tree_Bound approach by comparing its results with the heuristics derived by main bounding methods such as Dodin and Kleinöder.

For the sake of a fair comparison, we want to point out that no method performs always as the best. The accuracy of each approach highly depends on graph topology and distribution of task times. We have already demonstrated that all techniques try to transform the original DAG into an SP graph. Hence, let us take the simplest form of a non-SP graph (initial graph in Fig. 8) and show how each method obtains the corresponding SP graph for the upper bound evaluation. This simple example gives also an insight on the working method of the considered bounding algorithms.

Graph	Tree_Upper bound	Shogan	Dodin		Kleinöder
			Case 1	Case 2	
All nodes, mean 1	3.05	3.05	3.05	3.05	3.00
Node 3, mean 2	3.75	3.75	3.75	3.74	3.83

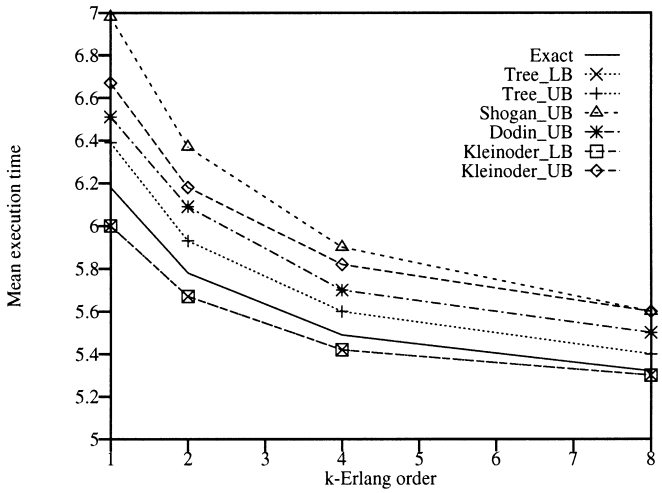

Fig. 9. Bounds on mean execution time for the graph in Fig. 2.

Kleinöder's method inserts the arc (2,3), while Shogan and Tree_UB duplicate node 1 in accordance with the graph interpretation. The Dodin approach has a non-deterministic choice: it can duplicate node 1 (case 1) or node 4 (case 2). The table shown above shows the upper bounds for the mean execution time obtained by these methods. In the first row we assume that the task execution times are independent and identically distributed random variables having exponential distribution with mean 1. In this instance, Kleinöder provides the best bound, while both Dodin duplications lead to the same results. If we slightly change the characteristics of the graph, by assuming the node 3 with mean 2, we obtain the results shown in the second row of the table. Now, Kleinöder provides the worst result, while Dodin (case 2) provides the best. However, we have to carry out both feasible duplications to single out which is the best Dodin's upper bound.

This naive example shows that no bounding method can always guarantee the best result. This also implies that no stochastic ordering relation among the three methods can be demonstrated in a formal way. Typically, we have to apply all the methods to know which approach gives the tightest bounds.

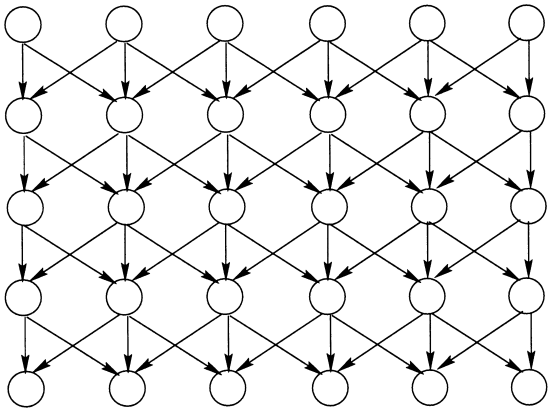

Fig. 10. Task graph of the (5×6) neighbor synchronization algorithm.

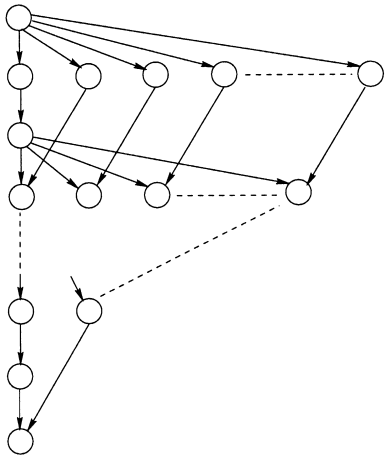

Fig. 11. Task graph of the LU factorization algorithm.

Moreover, the most accurate result is often obtained through a combination of different methods applied to subset of the graph.

We consider now the graph G in Fig. 2, assuming each task with unitary mean and independent k-Erlang distribution, we obtain the results shown in Fig. 9 with the k-Erlang order in abscissa. In this instance,

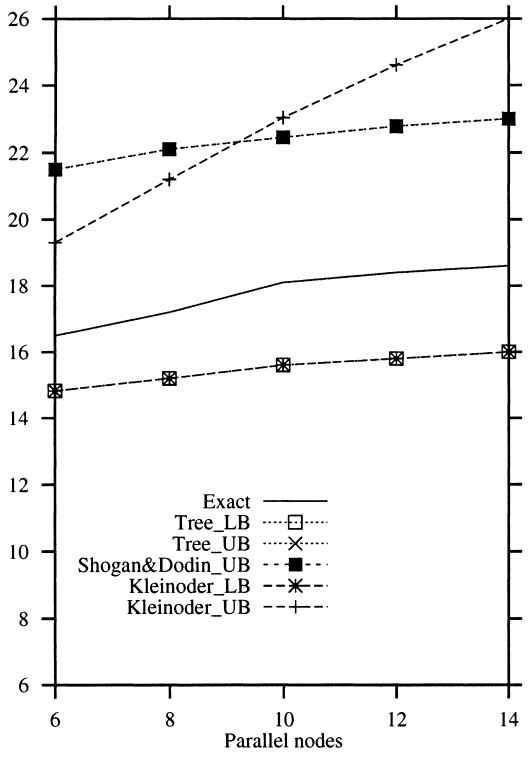

Fig. 12. Bounds on mean execution time for the near neighbor synchronization graph.

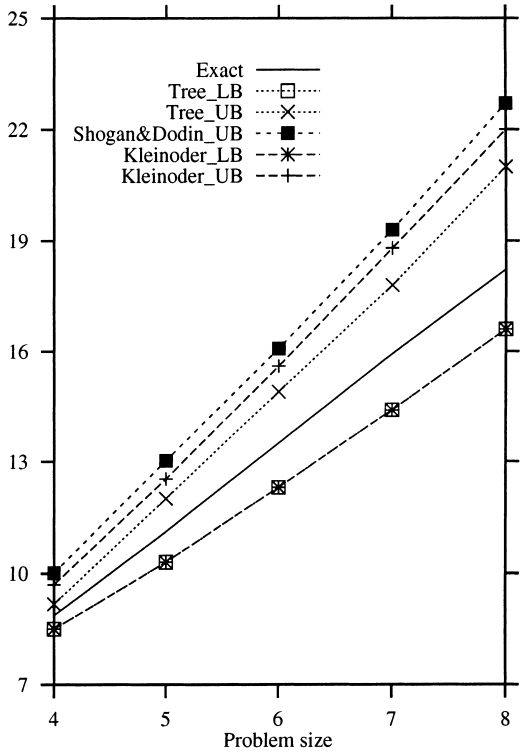

Fig. 13. Bounds on the mean execution time for the LU factorization graph.

Kleinöder's method provides same tight lower bounds as Tree_LB, while Tree_UB obtains the best upper bounds. It is close to Dodin's result and much better than Kleinöder's and Shogan's bounds.

Besides artificial graphs, we explore the differences among the bounding techniques for some real graphs. We first focus on the neighbor synchronization of an $(n \times m)$ structure (Fig. 10) which is a popular task graph in scientific applications. The graph has height n (iteration number) and width m (number of parallel nodes). Fig. 11 shows the bounds for the mean execution time of an $(8 \times m)$ graph for m ranging from 4 to 12, in which we assume that the task execution times are independent random variables exponentially distributed with mean 1. Kleinöder's upper bound performs best for very small m, while it does not provide accurate bounds for large m. The other methods, including Tree_UB, are preferable because they obtain tightest bounds for large and more realistic numbers of parallel nodes. Tree_LB gives same accurate lower bounds as Kleinöder's algorithm.

In Fig. 12, we describe the task graph for the parallel LU factorization algorithm, while Fig. 13 shows the bounds for the mean execution time as a function of the problem size. We assume that the task execution times are independent random variables exponentially distributed with mean 1. For this example, Tree_UB obtains tightest upper bounds for all problem sizes, while Tree_LB and Kleinöder methods give same lower bounds.

In Fig. 14, we consider the task graph of the parallel multiplication algorithm for block decomposed matrices [3]. We assume that the task execution times are independent random variables having k-Erlang distribution with mean 1. Fig. 15 shows the bounds for the mean execution time as a function of the k-Erlang order of the task execution time. Tree_UB provides always the tightest

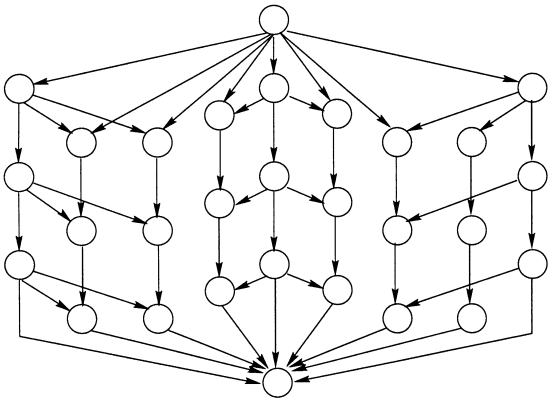

Fig. 14. Task graph of the block matrix multiplication algorithm.

bound except for the exponential instance ($k = 1$) in which Kleinöder's upper bound is slightly more accurate. Tree_LB is always tighter than the lower bound obtained by the Kleinöder algorithm.

Although we cannot demonstrate analytically that one method is always the best, from the experiments we observe the following results:

- In all instances the Dodin algorithm, in the form of the duplication process heuristics [6], provides same upper bounds as the Shogan method which we demonstrate to be less tight than the Tree_Bound technique.
- In some experiments Kleinöder method obtains better upper bounds than Tree_UB. However, the latter is often the best and, most importantly, it does not show very poor worst cases such as the Kleinöder algorithm (for example, in Fig. 12 for high numbers of parallel nodes).
- The Shogan method does not give good lower bounds because it is always less tight than the lower bound obtained by removing arcs [8], while the Dodin algorithm does not provide a lower bound.
- Although through quite different approaches, Kleinöder and Tree_Bound methods get same lower bounds in all test cases but for matrix multiplication in which Tree_LB is more accurate than Kleinöder.

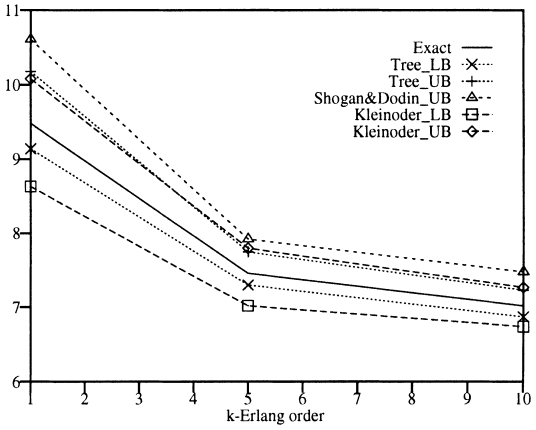

Fig. 15. Bounds on mean execution time for the block matrix multiplication graph.

6. Conclusions

In this paper, we propose a new hierarchical approach to obtain lower and upper bounds for the completion time distribution of stochastic DAGs. The method is based on a tree-like representation of the graph, which is derived from the classical tree representation of SP graphs. This decomposition allows us to represent the original non-SP graph as a hierarchy of subgraphs which are connected in series or in parallel. The lower bound obtained as the distribution of the embedded SP graph is similar to other bounding methods, while we introduce the concept of close subgraph reduction operation to derive a new algorithm for the upper bound. The lower and upper bounds can be calculated jointly through a single post-order visit of the tree-like representation graph.

Our approach avoids the non-determinism of other proposed heuristics while keeping linear complexity. For what concerns the accuracy of the bounds, we analytically demonstrate that the Tree_Bound method gives always equal or tighter bounds than the Shogan algorithm [19] which, in its turn, is always better than the Yazici-Pekergin and Vincent method [24]. Moreover, we show that none of the other heuristics can guarantee the best result because the accuracy is deeply related to the graph topology and task time distribution. Nevertheless, we demonstrate that the Tree_Bound approach provides analogous or better results than other well-known heuristics for various DAGs modeling artificial and real applications.

The *tree-like representation* which decomposes the original DAG into subgraphs is a further contribution of this paper because it intrinsically permits to combine different bounding approaches. Indeed, we have investigated several approaches for improving the accuracy of bounding techniques. Our conclusion is that further improvements can be achieved only through a meta-algorithm that combines different approaches and applies the most appropriate to each part of a DAG. To this purpose, we are studying for which characteristics of the graph topologies one method works better than others.

References

[1] V.S. Adve, M.K. Vernon, The influence of random delays on parallel execution times, Performance Evaluation 21 (1) (1993) 61–73.
[2] R.E. Barlow, F. Proshan, Statistical Theory of Reliability and Life Testing, Hold, New York, 1975.
[3] M. Calzarossa, G. Serazzi, Workload characterizations: a survey, Proc. IEEE 81 (8) (1993) 1136–1150.
[4] P. Dauphin, F. Hartleb, M. Kienow, V. Mertsiotakis, A. Quick, PEPP: performance evaluation of parallel programs — User's guide — Version 3.1, Technical Report 5/92, University of Erlangen, Nürnberg, IMMD VII, April 1992.
[5] L.P. Devroye, Inequalities for the completion time of stochastic PERT networks, Math. Oper. Res. 4 (1980) 441–447.
[6] B. Dodin, Bounding the project completion time distribution in PERT networks, Oper. Res. 33 (4) (1985) 862–881.
[7] E. Gelenbe, Multiprocessor Performance, Wiley, London, 1989.
[8] F. Hartleb, V. Mertsiotakis, Bounds for the mean runtime of parallel programs, Proceedings of the Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Edinburgh, 1992.
[9] R. Hofman, R. Klar, B. Mohr, A. Quick, M. Siegle, Distributed performance monitoring: methods, tools, and applications, IEEE Trans. Parallel Distributed Systems 5 (6) (1994) 585–598.
[10] G.B. Kleindorfer, Bounding distributions for a stochastic acyclic network, Oper. Res. 19 (1971) 1586–1601.
[11] W. Kleinöder, Stochastic analysis of parallel programs for hierarchical multiprocessor systems, Ph.D. Thesis, University of Erlangen, Nürnberg, 1982.
[12] K.N. Lalgudi, D. Bhattacharya, P. Agrawal, On the performance prediction of parallel algorithms, Proceedings of Parallel and Distributed Computing Systems, Las Vegas, October 1994, pp. 330–335.
[13] K. Li, Stochastic bounds for parallel program execution times with processor constraints, IEEE Trans. Computers 46 (5) (1997) 630–636.
[14] F. Lo Presti, M. Colajanni, S. Tucci, Stochastic bounds on execution times of parallel computations, Proceedings of the IEEE Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Durham, NC, February 1994.

[15] S. Madala, J. Sinclair, Performance of synchronous parallel algorithms with regular structures, IEEE Trans. Parallel Distributed Systems 2 (1) (1991) 105–115.
[16] J.J. Martin, Distribution of the time through a directed acyclic network, Oper. Res. 13 (1) (1965) 44–66.
[17] J.T. Robinson, Some analysis technique for asynchronous multiprocessor algorithms, IEEE Trans. Software Engrg. SE-5 (1) (1979) 24–31.
[18] R. Sahner, K. Trivedi, Performance and reliability analysis using directed acyclic graph, IEEE Trans. Software Engrg. SE-13 (10) (1987) 1105–1114.
[19] A.W. Shogan, Bounding distributions for a stochastic PERT network, Networks 7 (1977) 359–381.
[20] D. Stoyan, Comparison Methods for Queues and Other Stochastic Models, Wiley, New York, 1983.
[21] J. Sztrik, D. Kouvatsos, Asymptotic analysis of a heterogeneous multiprocessor system in a randomly changing environment, IEEE Trans. Software Engrg. 17 (10) (1991) 1069–1075.
[22] A.B. Tayyab, J.G. Kuhl, Stochastic performance models of parallel task systems, Proceedings of Sigmetrics'94, Santa Clara, CA, May 1994.
[23] A. Thomasian, P.F. Bay, Analytic queueing network models for parallel processing of task systems, IEEE Trans. Parallel Distributed Systems C-35 (12) (1986) 1045–1054.
[24] N. Yazici-Pekergin, J. Vincent, Stochastic bounds on execution times of parallel programs, IEEE Trans. Software Engrg. 17 (10) (1991) 1005–1012.

Michele Colajanni is an Associate Professor in the Department of Information Engineering at the University of Modena and Reggio Emilia, Italy. His research interests include parallel and distributed systems, high performance Web services, parallel computing, load balancing, and performance analysis. Dr. Colajanni received the Laurea degree in computer science from the University of Pisa and the Doctorate degree in computer engineering from the University of Roma "Tor Vergata". From 1992 to 1998, he was with the University of Roma "Tor Vergata" as a researcher in computer engineering. In 1996, he was a visiting researcher at the IBM T.J. Watson Research Center in Yorktown Heights, NY. Dr. Colajanni is a member of the IEEE Computer Society and the ACM.

Francesco Lo Presti received the Laurea degree in electrical engineering and the Doctorate degree in computer science from the University of Rome "Tor Vergata" Rome, Italy, in 1993 and 1997, respectively. He is currently with the Computer Science Department, University of Massachussets at Amherst. His research interests include high-speed and wireless networks, measurements, modeling and performance evaluation of computer and communications networks.

Salvatore Tucci received the Doctorage degree in physics from the Universtiy of Pisa, Italy in 1972. From 1973 to 1975 he held a post-graduate fellowship in computer science at IEI-CNR, Pisa. In 1975, he joined the Department of Computer Science of the Universtiy of Pisa as an Assistant Professor. In 1987, he joined the Department of Electronics Engineering of the University of Roma "Tor Vergata" as a Full Professor. He held visiting positions in 1987 at INRIA, Paris, and in 1981/1982 at the IBM T.J. Watson Research Center, Yorktown Heights, NY. From 1990 to 1999 he was the Dean of the computer science curricula at the Faculty of Engineering. He is currently on leave as the Director of the Office for Informatics, Telecommunications and Statistics of the Italian Prime Minister. His research interests include performance evaluation, parallel and distributed systems, multimedia applications, conception and design of large information systems for decision-making and government.