A Survey on Solution Methods for Task Graph Models

François Baccelli   Alain Jean-Marie   Zhen Liu
INRIA Sophia-Antipolis
2004 Route des Lucioles BP-107
06902 SOPHIA-ANTIPOLIS Cedex

Abstract

We give in this paper a survey on models developed in the literature using the concept of task graphs, focusing on solution techniques. Different types of task graphs are considered, from PERTS networks to random task graphs. Reviewed solution methods include exact computations and bounds.

1 Introduction, Concepts and Notations

The purpose of this paper is to survey models based on stochastic task graph representations and the solutions techniques that have been developed for them.

The reason for doing this in the framework of the QMIPS project is that task graphs appear to be of central importance in the modeling and analysis of parallel programs and architectures.

Yet, the solution of task graph problems is difficult in general. No really satisfactory and sufficiently general solutions have been proposed as of today, and research is still active in the area.

The term "task graphs" covers now a wide variety of models. We shall begin the survey with what appears to be the initial, and most widely used task graph model: the PERT networks (section 2). We will then give a look at three types of extensions to the PERT model: networks with "multiple logic", for which techniques similar to that of PERT can be used (section 3.1), repetitive task graphs (section 3.2) and graphs with random topology (section 3.3), for which the techniques used are rather of asymptotic nature.

Tasks graphs (or, sometimes, *activity networks* [16]) are based on the concept of a *task*, which can never be precisely defined, but which conveys the idea of an indivisible activity which can be performed sequentially in a single location using a fixed set of resources (working personnel, computer CPU/memory/communication network, manufacturing tool).

A complex activity (or *program*, in the management or computer science senses) within a project, a plant or a computer involves several such tasks, related to one another by *precedence relations* (or constraints). Such constraints specify that a certain task has to wait for the end of other ones to begin its execution. The relations between tasks are commonly represented using directed graphs, which represent efficiently the ideas of precedence and the flow of execution (progression) of the program.

Finally, an essential characteristic of the task graph models considered here is that durations associated with tasks are represented by *random variables*. Although this assumption tends to complicate the solution of models, as we shall see, it is generally believed that in contexts where task durations are not known with precision, stochastic assumptions are preferable to the simpler worst case, best case and average case assumptions.

The variety in task graph models reflects the many different situations encountered by the modeler, for instance in the area of parallel programming. The topology of the task graph can be known in advance, and sometimes have a lot of structure (the repeating patterns of numerical algorithms) or on the contrary, be unknown and random (semi-numerical algorithms). Also, the precedence constraints may be of various nature.

Let us now introduce some terminology. Throughout this paper, a task graph will be a directed, acyclic and valuated graph $\mathcal{G} = (\mathcal{E}, \mathcal{V}, \mathcal{L})$, where

- $\mathcal{E}$ is a set of *vertices* or *nodes*;
- $\mathcal{V}$ is the set of *edges*;
- $\mathcal{L}$ is a mapping from $\mathcal{V} \cup \mathcal{E}$ into $\mathbb{R}$.

Multiple edges are allowed in principle, but for the sake of keeping the notation simple, we shall identify $\mathcal{V}$ to a subset of $\mathcal{E} \times \mathcal{E}$ and note edges by the ordered pair (i, j) of their end nodes.

The function $\mathcal{L}$ indicates the duration associated with an arc or a node of the task graph.

Let $N = |\mathcal{E}|$ and $M = |\mathcal{V}|$ be the number of nodes and edges in $\mathcal{G}$.

The following definition considers that task graphs can be random, both in their topology and the durations of their activities.

Definition 1.1 *A* stochastic task graph *is a random variable on the set of all task graphs.*

The random variables used throughout this paper will be defined on some (implicit) probability space with adequate structure. We shall often omit the word "stochastic" and simply speak of "task graphs".

For a node i, let $\pi(i)$ and $\sigma(i)$ be the set of *predecessors* and *successors* of i, defined by:

$$\begin{aligned} \pi(i) &= \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\} \\ \sigma(i) &= \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\} \end{aligned}$$

Let $\mathcal{I}$ be the set of nodes with no predecessors (called *initial* nodes), and $\mathcal{T}$ the set of nodes with no successor (*terminal* nodes). A *path* of $\mathcal{G}$ is a sequence $(i_1, i_2, \ldots, i_k)$ of nodes such that $(i_l, i_{l+1}) \in \mathcal{E}$ for $l = 1, \ldots, k-1$. The number $k-1$ is often called the topological length of the path, to distinguish it from the (duration) length defined by:

$$L(p) \;=\; \sum_{l=1}^{k} \mathcal{L}(i_k) \;+\; \sum_{l=1}^{k-1} \mathcal{L}(i_l, i_{l+1}) \;.$$

Let $\mathcal{P}(i,j)$ be the set of of all paths going from node i to node j, and $\mathcal{P}$ the set of all paths of $\mathcal{G}$.

A last necessary concept needed to describe the temporal behavior of the task graph is that of *enabling function* or *synchronization constraint*. The enabling function of node i allows to say at which instant this node may begin its execution (is enabled) in function of the dates at which the arcs j, i, for $j \in \pi(i)$, have completed their own execution. Common examples of such functions are "min" and "max", but more complicated ones can be used, including order statistics.

The temporal execution of the network is then governed by the following rules. The term *activity* refers to the execution of a node or an edge of the network.

1. each activity can begin its execution when, and only when it is enabled;
2. an activity corresponding to an arc (i,j) is enabled when the activity of i completes;
3. each activity begins as soon as possible;
4. activities are not interrupted, and an activity a completes $\mathcal{L}(a)$ units of time after it has begun;
5. an initial node begins its execution at time 0.

For $k \in \mathcal{V} \cup \mathcal{E}$, let $X(k)$ denote the instant at which activity k *completes* its execution. The main quantity of interest is the *total execution* time of the graph, defined by:

$$T(\mathcal{G}) \;=\; \max_{k \in \mathcal{T}} \{X(k)\} \;. \tag{1}$$

Other measures of interest have sometimes been considered in the literature, such as: the distribution of the length of the shortest path [19], the probability that a path is critical, that is, belongs to the actual longest path [37] (sometimes called the *criticality index* of the path), or the maximum/average number of active tasks during the execution of the project [48]. Little seems to be known about these measures, which will not be discussed in this paper.

Remark 1.2 Papers differ with the localization of duration. In the literature on PERT networks, durations are set on edges which represent the tasks themselves. The beginning and end points of the task are the nodes, which represent synchronization points. On the other hand, papers using task graphs for modeling and analysis of parallel programs usually consider that the nodes represent tasks and the edges the precedence constraints between them. In that case, time is associated to nodes. The constraint being often materialized by the communication of data, some authors use models with time on both nodes and edges. In fact, all three representations are equivalent, up to a possible change in the topology of the network. Figure 1 shows how to translate a task graph model with time on both nodes and edges (in (a)) into a model with time on edges only (in (b) and (c)) or on nodes only (in (d)). The transformation of (a) into (b) does not change the topology of the network. The transformation of (a) into (c) adds N nodes and edges. However, in a stochastic setting, the first construction introduces correlations between the durations on the arcs. Therefore, it is usually better to prefer the second one, unless the duration of the node (α, in the figure) is constant. The transformation of (a) into (d) adds M nodes and edges to the graph. Note that if the original graph is series-parallel (see section 2.1.3), so is the transformed graph.

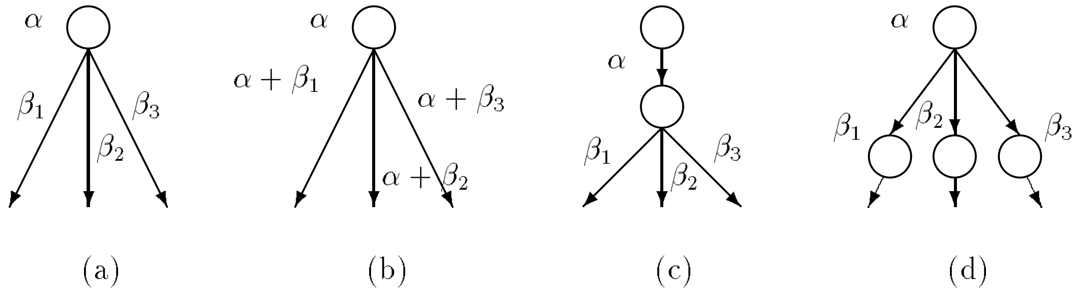


Figure 1: Equivalence of task graphs

Remark 1.3 Because of the two points of view mentioned above, the "tasks" of the project described by $\mathcal{G}$ may be assimilated either to the nodes or to the arcs. In the remainder of this paper, we shall often use the word "task" for "node", seeing the edges as communications between tasks.

Remark 1.4 It is common to consider only one initial and one terminal node. This is easily done by defining a new graph $\mathcal{G}' = (\mathcal{E}', \mathcal{V}', \mathcal{L}')$ with:

$$\mathcal{V}' = \mathcal{V} \cup \{\top, \perp\},$$

$$\mathcal{E}' = \mathcal{E} \cup \{(\top, i) \mid i \in \mathcal{I}\} \cup \{(i, \perp) \mid i \in \mathcal{T}\},$$

$$\mathcal{L}'(i) = \mathcal{L}(i),\ i \in \mathcal{E} \cup \mathcal{V};\ \mathcal{L}'(i) = 0 \text{ otherwise.}$$

The resulting network is equivalent to $\mathcal{G}$ in the sense that $\forall i \in \mathcal{E}$, $X_{\mathcal{G}}(i) = X_{\mathcal{G}'}(i)$ and in particular, $T(\mathcal{G}) = T(\mathcal{G}') = X_{\mathcal{G}'}(\perp)$ (with obvious notation) for each realization of $\mathcal{G}$.

2 PERT Networks

PERT is an acronym standing for "Performance and Evaluation Review Technique". This concept was introduced by researchers in operations research to model, evaluate and optimize the evolution of complex projects [35]. The so-called "PERT technique" is based on the operational assumption that the durations of all activities of the project have known lower and upper bounds, and a "most likely" value. This duration is then considered random with a beta distribution. An estimate (actually, a lower bound according to theorem 2.5) is obtained by replacing the random durations by their expected value (see section 2.1.1).

Although very primitive by modern standards, this technique is an improvement on methods considering only most likely durations. This concept was very successful and received much attention in the literature during the years 1960 and after. Very complete surveys, covering the research for the years 1959 to 1987 have been proposed in [10, 31, 1]. In particular, we refer to the work of Adlakha and Kulkarni [1] for its extensive discussion of approximation and simulation techniques for PERT networks. The discussion of these topics in this paper will be limited to recent ideas.

The early literature tackled the issue of the accuracy of the PERT technique from two (related) points of view [34, 33, 47]:

sensitivity of the model: how large can the error on $ET(\mathcal{G})$ be if one choses the wrong distribution;

accuracy of the solution: how large can the error be if one estimates $ET(\mathcal{G})$ by applying the "PERT technique".

The conclusion of these studies is that the error can be quite large in practice, and research has then focussed on solution techniques being able to handle general distributions with a delicate compromise between solution accuracy and computation time and precision.

In terms of the introductory discussion of section 1, PERT networks are characterized by an "and" synchronization logic meaning that a task has to wait for the completion of *all* its predecessors. Equivalently, a task is enabled when and only when all the activities of its incoming arcs are completed.

For the purposes of this paper, we adopt the following definition for PERT networks:

Definition 2.1 *A PERT network is a finite random task graph with deterministic topology, and "and" synchronization constraints.*

The execution rules above imply that the times $X(i)$ satisfy the following *evolution equations*:

$$X(i) \;=\; \max_{j\in\sigma(i)}\{X(j) \;+\; \mathcal{L}(j,i)\} \;+\; \mathcal{L}(i)\;, \tag{2}$$

with $X(i) = \mathcal{L}(i)$ if $i \in \mathcal{I}$. Another way of expressing $T(\mathcal{G})$ is using the sets $\mathcal{P}(i,j)$ and $\mathcal{P}$:

$$X(k) \;=\; \max_{p\in\mathcal{P}(i,k), i\in\mathcal{I}}\{L(p)\}\;, \tag{3}$$

and

$$T(k) \;=\; \max_{p\in\mathcal{P}}\{L(p)\}\;. \tag{4}$$

Remark 2.2 Conversely, it is clearly possible to represent any $(\max,+)$-*linear recurrence* such as (2) by a PERT network.

The approaches for computing or estimating the distribution or the statistics of $T(\mathcal{G})$ analytically fall in three categories: exact computations, bounds (sometimes called "approximation bounds") and heuristics (approximations that are not bounds). We shall concentrate on the two first approaches in the following paragraphs and refer to the bibliography (in particular, [1]) for approximations. In addition, we shall briefly discuss simulation in section 2.3.

2.1 Exact Solution

2.1.1 Deterministic PERT Networks

An important special case of PERT networks is that where all valuations are deterministic variables, with a constant and known value. In this case, the computation is reduced to that of the longest path in a valuated graph (the *critical path*). This problem can be solved in $O(M+N)$ if the network is already topologically sorted by simply using the recurrence (2). On the other hand, sorting an acyclic directed graph can be performed in $O(M)$.

2.1.2 Direct Computation

From (4), one has:

$$P(T(\mathcal{G}) \leq t) \;=\; \int_{\mathcal{D}} dF(x_1, x_2, \ldots, x_{N+M}), \tag{5}$$

where $F(x_1, x_2, \ldots, x_{N+M})$ is the joint distribution function of the timing of the nodes and edges of the graph, and $\mathcal{D}$ is the polyhedral domain:

$$\mathcal{D} \;=\; \bigcap_{p \in \mathcal{P}} \{ \sum_{i \in p} x_i \leq t \} \; .$$

If the number of variables involved in these formulas is not too large, one may integrate (5) symbolically or numerically. For instance, Hartley and Wortham [23] propose a numerical algorithm for finding the distribution of $T(\mathcal{G})$ if the graph is a "crossed network", obtained recursively from a series-parallel graph (see below) by replacing activities by Wheatstone Bridges (depicted in figure 3(a)). The same technique may apply with other small computable graphs instead of Wheatstone bridges [41].

2.1.3 Reducible Graphs

An important class of PERT networks for which the computation of the exact distribution of $T(\mathcal{G})$ is feasible is that of *series-parallel* (or *reducible*[15]) networks with independent duration distributions.

A graph is said to be series-parallel (SP) if either:

- it is reduced to a single task;
- $|\mathcal{I}| = |\mathcal{T}| = 1$ and the graph induced by $\mathcal{V} \setminus (\mathcal{I} \cup \mathcal{T})$ is composed of disconnected series-parallel graphs.

Equivalently, one can show that $\mathcal{G}$ is SP if the application of the following algorithm reduces it to a single node.

Algorithm Reduce
while $|\mathcal{G}| > 1$
do **if** $\exists i, j \in \mathcal{V}$ with multiple arcs between i and j
    then $\mathcal{G}$:= replace all arcs between i and j by a single arc
    else **if** $\exists i \in \mathcal{V}, \quad |\sigma(i)| = |\pi(i)| = 1$
        then { there are two arcs in series }
            $\mathcal{G}$:= replace node i and its two incident arcs by a single arc
        else stop
        endif
    endif
done

If the network is SP, the algorithm above is easily adapted to compute the distribution of $T(\mathcal{G})$, while performing the reduction. For this, all durations are assumed to be mutually independent.

If two arcs are in series, the resulting arc is valuated by the convolution of the distributions of the arcs and the node

If two or more arcs are in parallel, the resulting arc is valuated by the product of the distributions of the original arcs.

The distribution of $T(\mathcal{G})$ is computed with $N-2$ convolutions and $M-N+1$ multiplications of distributions ([15], Proposition 1). This relative simplicity explains why the fact of being SP is a desirable property and that most approximation or bounding techniques use reductions to this class of graphs.

Several authors consider the exact numerical computation of the distribution of $T(\mathcal{G})$ using families of distributions that are *stable* under the "max" and "+" operations, that is the distribution functions of which are stable by multiplication and convolution. Such families include piecewise polynomial distributions, used in [36], and exponential polynomials distributions (with rational Laplace transforms) used in [44] and [32]. Another stable family is that of phase type distributions [38], but the complexity of the matrix calculus involved probably prohibits their use in practical situations.

The families quoted above are also stable under the minimization operation, which allows their use for the task graphs with "or" synchronizations of section 3.1.

Remark 2.3 It is important to note that the algorithm above guarantees that the number of distribution products and convolutions to perform on the network is linear in the size of the network. However, the number of coefficients involved in the distributions usually grows exponentially with the number of operations performed [12]. This leads to storage and accuracy problems which limit severely the method in practice. This fact is often overlooked in the literature, which insists more on the "polynomial" complexity of the series-parallel reduction.

2.1.4 Markovian Analysis

The principle of markovian analysis is to represent the evolution of the execution of the task graph by that of a Markov chain [28].

Assume that the durations in the network have all a distribution of phase type [38]. For $k \in \mathcal{V} \cup \mathcal{E}$, let ν_k be the number of phases of the distribution of k. By convention, the last stage is numbered ν_k. At an instant t, the state of activity k, $S_i(t)$ can be:

state 0: it has not started its execution;

state $i \in \{1, \ldots, \nu_{k-1}\}$: it is active, in phase i;

state ν_k: it has completed its execution.

The state of the PERT network is described by an $(N+M)$-tuple: $\mathbf{S}(t) = (S_i(t))_{i \in \mathcal{V} \cup \mathcal{E}}$. The process $\mathbf{S(t)}$ is a continuous, time Markov chain which infinitesimal generator $\mathbf{Q}$ is easily deduced from the parameters of the distributions involved and the topology of the graph. It is transient, starts from state $(0)_{i \in \mathcal{V} \cup \mathcal{E}}$ and has a single absorbing state: $(\nu_k)_{k \in \mathcal{V} \cup \mathcal{E}}$. The distribution of $T(\mathcal{G})$ is therefore also of phase type, and its statistics can be in principle computed from $\mathbf{Q}$.

The size of the state space of $\mathbf{S}$ (and number of phases of $T(\mathcal{G})$) is *a priori* $\prod_k (\nu_k + 1)$ (in the case of a parallel network). Even for exponential distributions ($\nu_k = 2$), the size of the chain grows quickly with N and M and this limits the efficiency of the technique to relatively small graphs.

However, the number of active tasks may not grow as quickly, depending on the topology of $\mathcal{G}$. This fact can be used in the solution of the Markov chain model. For instance, this number is less than $\max_k\{\nu_k + 1\}$ for a series network. If it is the case, the recursive approach proposed in [48] for exponential distributions may prove useful. Assume that only nodes have a duration and that the duration of node k is exponentially distributed with parameter μ_k. For $k \in \mathcal{V}$, let $\mathcal{G}_k$ be the graph induced from $\mathcal{G}$ by removing node k. Then

$$ET(\mathcal{G}) \; = \; \left(\sum_{k\in\mathcal{I}} \mu_k\right)^{-1} \; (1 \; + \; \sum_{k\in\mathcal{I}} \mu_k ET(\mathcal{G}_k)) \; .$$

2.2 Bounds

Monotonicity results appear in many places in the literature on PERT networks. A synthetic treatment is based on the concept of *stochastic ordering* [46], which we briefly recall here.

Let X and Y be two random variables on $\mathbb{R}^k$. One says that X *is stochastically smaller than* Y *in the strong sense* and writes:

$$X \leq_{\mathrm{st}} Y$$

if and only if:

$$E[f(X)] \; \leq \; E[f(Y)] \; ,$$

for all *increasing* function $f : \mathbb{R}^k \to \mathbb{R}$ such that both expectations exist.

One says that X *is stochastically smaller than* Y *in the increasing and convex (resp. concave) sense* and writes:

$$X \leq_{\mathrm{icx}} Y \qquad \text{(resp.)} \qquad X \leq_{\mathrm{icv}} Y$$

if and only if:

$$E[f(X)] \; \leq \; E[f(Y)] \; ,$$

for all *increasing and convex (resp. concave)* function $f : \mathbb{R}^k \to \mathbb{R}$ such that both expectations exist.

The order relations $\leq_{\mathrm{st}}$, $\leq_{\mathrm{icx}}$ and $\leq_{\mathrm{icv}}$ are called integral stochastic orders. It is plain that $X \leq_{\mathrm{st}} Y \Rightarrow X \leq_{\mathrm{icx}} Y$. By extension of the concept, we will write that two sequences of random variables $\{X(n)\}$ and $\{Y(n)\}$ compare for one of these orders if all finite subsequences do. It is possible to define other stochastic orderings, based on other classes of functions or other properties of distribution functions.

The interest for proving the existence of such orderings between two random variables is illustrated by the following implications:

$$\begin{aligned} X \leq_{\mathrm{st}} Y \; &\Leftrightarrow \; P(X > t) \leq P(Y > t) \; , t \geq 0, \qquad (6) \\ X \leq_{\mathrm{icx}} Y \text{ and } X, Y \geq 0 \; &\Rightarrow \; E[X]^p \leq E[Y]^p \; , p \geq 1 \; , \\ X \leq_{\mathrm{icv}} Y \text{ and } X, Y \geq 0 \; &\Rightarrow \; E[X]^p \leq E[Y]^p \; , p \leq 1 \; . \end{aligned}$$

Thus, the existence of stochastic orderings between random variables provides precious information on how their distribution and their moments compare. In particular, (6) shows that finding an upper strong stochastic bound for X is equivalent to "bounding its distribution from below" (see [14]).

Other bounds are obtained using the concept of *association* [9]. One says that random variables $X_1, \ldots, X_n$ are associated if for any increasing functions f and g from $\mathbb{R}^n$ in $\mathbb{R}$:

$$E[fg(X_1, \ldots, X_n)] \;\geq E[f(X_1, \ldots, X_n)]\; E[g(X_1, \ldots, X_n)].$$

In particular, independent random variables are associated, and it can be shown that increasing functions of associated random variables are associated.

In this section, we will compare stochastically performance measures of an "original" PERT network to those of another system which parameters (duration distributions, topology) are different from the original one, but can be compared stochastically with them.

The parameter and performance measures of the second system will be distinguished by a "tilde" symbol. Thus, the durations of the second network will be $\tilde{\mathcal{L}}(i), i \in \tilde{\mathcal{V}} \cup \tilde{\mathcal{E}}$. By convention, parameters of both networks will be considered as identical unless explicitly stated.

2.2.1 Monotonicity with Respect to Durations

Theorem 2.4 *If*

$$\{\tilde{\mathcal{L}}(i)\}_{\mathcal{V}\cup\mathcal{E}} \leq_{\mathrm{st}} \{\mathcal{L}(i)\}_{\mathcal{V}\cup\mathcal{E}} \;,$$

then for all $k \in \mathcal{V}$*,*

$$\{\tilde{X}(k)\}_{k\in\mathcal{V}} \leq_{\mathrm{st}} \{X(k)\}_{k\in\mathcal{V}} \;.$$

In particular,

$$T(\tilde{\mathcal{G}}) \leq_{\mathrm{st}} T(\mathcal{G}) \;.$$

A similar result holds for the increasing and convex ordering, namely:

Theorem 2.5 *If*

$$\{\tilde{\mathcal{L}}(i)\}_{\mathcal{V}\cup\mathcal{E}} \leq_{\mathrm{icx}} \{\mathcal{L}(i)\}_{\mathcal{V}\cup\mathcal{E}} \;,$$

then for all $k \in \mathcal{V}$*,*

$$\{\tilde{X}(k)\}_{k\in\mathcal{V}} \leq_{\mathrm{icx}} \{X(k)\}_{k\in\mathcal{V}} \;.$$

In particular,

$$T(\tilde{\mathcal{G}}) \leq_{\mathrm{icx}} T(\mathcal{G}) \;.$$

These theorems appear for instance in [46]. Their proof is done by induction based on (2) using the increasingness and convexity of the (max, +) operators involved.

Kleindorfer [25] proposes an upper bound for $T(\mathcal{G})$ in the $\leq_{\mathrm{st}}$ sense, by computing successively the distributions:

$$Y(i) \stackrel{d}{=} \operatorname*{maxst}_{(j,i)\in\mathcal{E}}\{Y(j) + \mathcal{L}(j,i)\} \;+\; \mathcal{L}(i) \;, \tag{7}$$

with $Y(i) = 0$ if $i \in \mathcal{I}$. In this formula, "maxst" denotes the smallest distribution larger in the $\leq_{\mathrm{st}}$ sense than all the other. It is obtained by taking the common lower envelope of the distribution functions. This technique appears to be difficult to implement outside the case of discrete and bounded distributions. For continuous distributions, a preliminary bounding by a discrete distribution my be necessary.

Robillard and Trahan [42] propose a family of lower bounds for $ET(\mathcal{G})$ based on Jensen's inequality: for any $D \subset \{\mathcal{L}(i) | i \in \mathcal{V} \cup \mathcal{E}\}$,

$$E[X(i)] \;\geq\; E\big(\max_{(j,i) \in \mathcal{E}} \{E[X(j) + \mathcal{L}(j,i) + \mathcal{L}(i) | D]\}\big) \; .$$

Various bounds are derived by chosing proper subsets D, but a tradeoff has to be found between the accuracy of the bound (perfect if D contains all relevant random variables, weak if $D = \emptyset$) and the complexity of the computation.

In [43], the same authors prove an upper bound in the $\leq_{\mathrm{st}}$ sense for $T(\mathcal{G})$, based on Bonferonni's inequality (see also [29]):

$$P(T(\mathcal{G}) \;>\; x) \;\leq\; \max\{0, \sum_{p \in \mathcal{P}} P(L_p > x) \;\} \; .$$

The distributions of the lengths L_p are then computed numerically by inverting the product of the characteristic functions (or Laplace transforms) of the distributions of durations on the path. It is unlikely that this method performs well for large graphs, because of the number of paths involved, and the well known instability of the inversion of characteristic functions.

Using the same formula, Nádas [37] derives an upper bound for $T(\mathcal{G})$ in the $\leq_{\mathrm{icx}}$ sense, which holds even if the durations of nodes and edges in the graph are correlated. The expectation of the upper bound is computed as the solution of a constrained minimization problem. The number of variables in this problem is, here also, the number of paths in the network.

Remark 2.6 According to Theorem 2.4, replacing all random variables by their expected values yields a lower bound for the distribution of $T(\mathcal{G})$ because for any random variable X: $EX \leq_{\mathrm{icx}} X$. In particular, the PERT technique gives a lower bound on the expected completion time $ET(\mathcal{G})$. This result stems also from Robillard and Trahan's bounds, with $D = \emptyset$.

Remark 2.7 Theorems 2.4 and 2.5 do not use any independence assumptions. They hold *a fortiori* when all the durations are independent and for all $i \in \mathcal{V} \cup \mathcal{E}$, $\tilde{\mathcal{L}}(i) \leq_{\mathrm{st}} \mathcal{L}(i)$ or $\tilde{\mathcal{L}}(i) \leq_{\mathrm{icx}} \mathcal{L}(i)$, respectively.

2.2.2 Monotonicity with Respect to the Topology

Another intuitive result is that if one adds edges and nodes to the graph of a PERT network, it will be more constrained and have a longer execution time.

Theorem 2.8 *Assume that:*

$$\tilde{\mathcal{V}} \subset \mathcal{V}, \quad \textit{and} \quad \tilde{\mathcal{E}} \subset \mathcal{E}.$$

Assume further that for all $i \in \tilde{\mathcal{V}} \cup \tilde{\mathcal{E}}$, $\tilde{\mathcal{L}}(i) = \mathcal{L}(i)$. Then,

$$\forall k \in \tilde{\mathcal{V}}, \qquad \tilde{X}(k) \;\leq\; X(k) \; .$$

Moreover,

$$T(\tilde{\mathcal{G}}) \;\leq T(\mathcal{G}) \; .$$

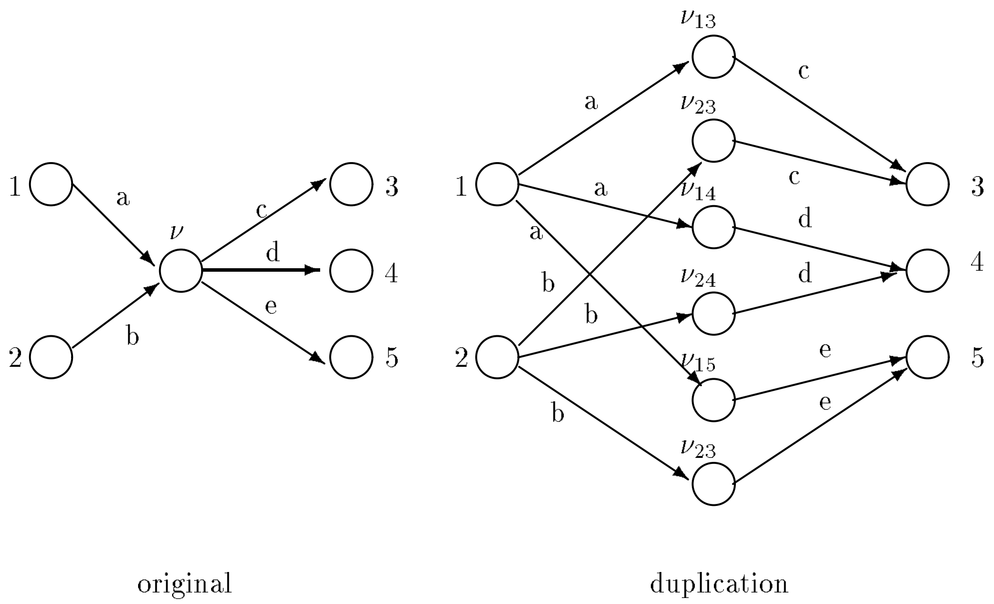


Figure 2: Duplication of a node in a task graph

2.2.3 Bounds Based on Duplications

In [14] and [15], Dodin proposes a bound on the execution time based on task duplications. Task duplications have the property to increase the distribution of $T(\mathcal{G})$ in the strong stochastic sense.

A graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{L}')$ is obtained by node duplication from the graph $\mathcal{G}$ if there exists a node $\nu \in \mathcal{V}$ such that:

- $\mathcal{V}'=\mathcal{V} \setminus \{\nu\} \cup \{\nu_{j,k} | 1 \leq j \leq |\pi(\nu)|, 1 \leq k \leq |\sigma(\nu)|\}$,
- $\mathcal{E}'=\mathcal{E}\setminus(\{(i,j)|j \in \sigma(\nu)\}\cup\{(j,i)|j \in \pi(\nu)\})\cup(\{(j,\nu_{j,k})|1 \leq k \leq |\sigma(\nu)|\}\cup\{(\nu_{j,k},k)|1 \leq j \leq |\pi(\nu)|\}$,

and the durations are such that, in distribution:

- $\mathcal{L}'(i) = \mathcal{L}(i)$, $i \in (\mathcal{V} \cup \mathcal{E}) \cap (\mathcal{V}' \cup \mathcal{E}')$,
- $\mathcal{L}'(\nu_{j,k}) = \mathcal{L}(\nu)$, $1 \leq j \leq |\pi(\nu)|, 1 \leq k \leq |\sigma(\nu)|$,
- $\mathcal{L}'(j, \nu_{j,k})$=$\mathcal{L}(j, \nu)$ and $\mathcal{L}'(\nu_{j,k}, k) = \mathcal{L}(\nu, k)$ for all $1 \leq j \leq |\pi(\nu)|$, $1 \leq k \leq |\sigma(\nu)|$.

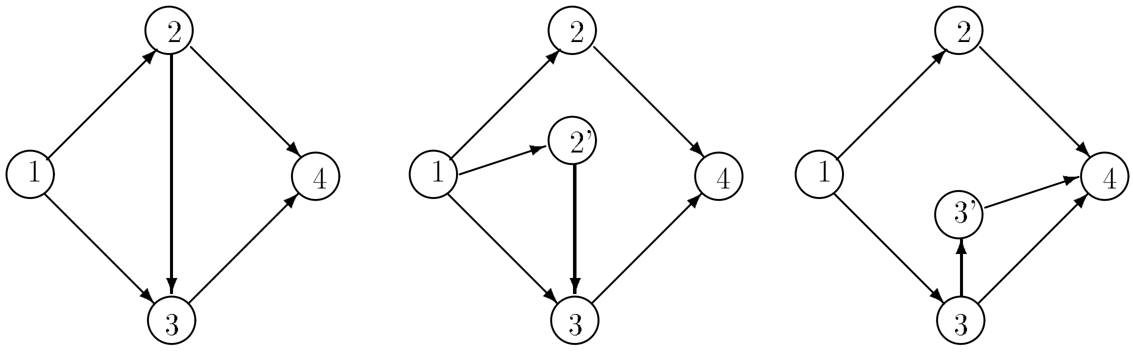
All resulting random variables are independent.

This duplication principle is illustrated in figure 2. The following theorem holds due to properties of associated random variables.

Theorem 2.9 *Assume all the durations $\{\mathcal{L}(i)\}_{i \in \mathcal{V} \cup \mathcal{E}}$ are independent random variables. If $\tilde{\mathcal{G}}$ is constructed by duplication from $\mathcal{G}$, then:*

$$T(\tilde{\mathcal{G}}) \leq_{\mathrm{st}} T(\mathcal{G}).$$

Figure 3 shows this principle on one of the simplest "irreducible graph", the Wheatstone Bridge. This graph is very popular in the literature, as the smallest non-SP task



(a) (b) (c)

Figure 3: Dodin's possible duplications for the Wheatstone bridge

graph. Autors test bounds and approximations are tested on it, often overlooking the fact that the exact distribution can be quite easily computed using (5). One can see that two duplication moves are possible to transform the original graph (a) into a SP graph (b)-(c). The resulting SP graph may have different completion time distribution, depending on the distribution of the durations of duplicated activities.

As described above, the duplication process seems to multiply nodes and edges and lead to large SP graphs. Dodin proposes an algorithm combining duplication and reduction which allows to reduce the network to a single task in at most $N^2/2$ duplications. It consists in duplicating only arcs with one neighbor (it is then useful to consider that $|\mathcal{I}| = 1$). Conceptually, however, the algorithm computes the execution time distribution of a SP task graph obtained by duplicating nodes as in figure 2.

2.2.4 Other Bounds

An immediate application of Theorem 2.4 is when the distributions of all durations in the graph $\mathcal{G}$ have a finite support. Accordingly, assume $\mathcal{L}(i) \in [a_i, b_i]$ for any $i \in \mathcal{V} \cup \mathcal{E}$. Let $\underline{\mathcal{G}}$ and $\overline{\mathcal{G}}$ be task graphs with nodes $\mathcal{V}$, edges $\mathcal{E}$ and valuations $\underline{\mathcal{L}}(i) = a_i$ and $\overline{\mathcal{L}}(i) = b_i$,$i \in \mathcal{V} \cup \mathcal{E}$. Then, obviously:

$$T(\underline{\mathcal{G}}) \leq_{\mathrm{st}} T(\mathcal{G}) \leq_{\mathrm{st}} T(\overline{\mathcal{G}}) \ .$$

An application of Theorem 2.5 allows to compare a PERT network with NBUE distributions with another one with exponentially distributed durations [48]. Recall that a distribution is NBUE (New Better then Used in Expectation) if $\int_0^\infty P(X > s|X > t)ds \leq \int_0^\infty P(X > s)ds$ for any t. If X is NBUE and Y is exponentially distributed with mean EX, then $X \leq_{\mathrm{icx}} Y$.

Using bounds on order statistics, and starting from the representation (4), Lefèvre and Laloyaux [30] propose the bound

$$ET(\mathcal{G}) \leq \mu \ + \ \sqrt{\frac{\mu - 1}{\mu} \sum_{p \in \mathcal{P}} \{\mathrm{V}ar(L(p)) + [E(L(p)) - \mu]^2\}} \ ,$$

where $\mu = 1/(|\mathcal{P}|) \sum_{p \in \mathcal{P}} EL(p)$ is the average path length. This bound depends only on the first two moments of the distributions of $\mathcal{L}(i)$. Likewise, Devroye [13] proposes bounds on $ET(\mathcal{G})$ and Var $T(\mathcal{G})$, by computing recursively the numbers $\{m_i, v_i\}_{i\in\mathcal{V}}$ as follows. Let $m_i = v_i = 0$, i$\in \mathcal{I}$, and (assuming durations associated with nodes are equal to 0):

$$\begin{aligned} v_i &= \sum_{j\in\pi(i)} (v_j + \mathrm{Var}\mathcal{L}(j,i)), \\ m_i &= \max_{j\in\pi(i)}\{m_j + E\mathcal{L}(j,i)\} \; + \; \sqrt{|\pi(i)| \max_{j\in\pi(i)}\{v_i + \mathrm{Var}\mathcal{L}(j,i)\}} \; . \end{aligned}$$

Then, if $\mathcal{T} = \{N\}$, one has in particular:

$$ET(\mathcal{G}) \leq m_N, \qquad \mathrm{Var}\ T(\mathcal{G}) \leq v_N \; .$$

2.2.5 Bounding Strategies

As mentioned above, most bounding (or approximation) strategies aim at transforming the original PERT network in as SP network. The resulting graph is considered as easily solvable, despite the potential problems pointed out in Remark 2.3.

Careful Duplications When reducing the network according to Dodin's algorithm, many choices can by possible for nodes to be duplicated. The problem of finding the "best" bound is estimated to be $\mathcal{NP}$-hard in [14] (although the quality criterion for the bounds is not precised).

Less Careful Duplications When ultimately repeated starting from nodes in $\mathcal{T}$, the duplication procedure leads to an in-forest with roots corresponding to the nodes of $\mathcal{T}$ in $\mathcal{G}$ (the "in-forest graph"). Further duplication separates all paths of the initial graph and produces a "path graph". This gives the two following theorems. The bounds obtained with these graphs are of course less accurate than ones obtained by the procedure above, but they have the advantages being more easily computable recursively (and possibly in parallel), and to lend themselves to an asymptotic analysis if the graph $\mathcal{G}$ has a regular structure (see 3.2 and [5]). A form of Theorem 2.11 appears in [48]. Theorem 2.10 is proved in [25] for discrete distributions, and is also implicit in [5].

Theorem 2.10 *Let $\tilde{\mathcal{G}}$ be the "in-forest" generated from $\mathcal{G}$:*

$$T(\mathcal{G}) \leq_{\mathrm{st}} T(\tilde{\mathcal{G}}).$$

Theorem 2.11 *Let $\tilde{\mathcal{G}}$ be the "path graph" generated from $\mathcal{G}$:*

$$T(\mathcal{G}) \leq_{\mathrm{st}} T(\tilde{\mathcal{G}}).$$

Examples of the tree graph and the path graphs for the Wheatstone bridge of figure 3 are shown in figure 4.

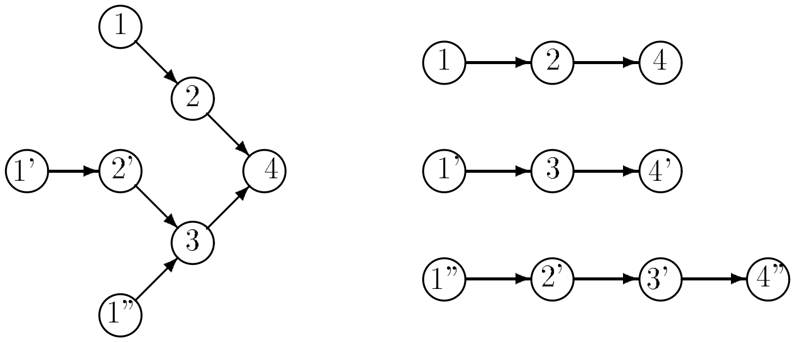


Figure 4: In-Forest Graph and Path Graph of the Wheatstone bridge

Distribution Simplification An idea for circumventing the problem of the (numerical) complexity of the SP reduction method by using a family of distributions characterized by a small and fixed number of parameters.

Stoyan [46] proposes an upper bound for $T(\mathcal{G})$ in the $\leq_{\mathrm{icx}}$ sense using uniform random variables. This family is not stable under "+" or "max" operations, and the method consists in systematically replacing the result of convolutions and products by stochastically larger uniform distributions.

Kamburowski [24] uses the class of "power distributions" with two parameters a, b:

$$P(X \leq t) = (t/b)^a , \qquad 0 \leq t \leq b .$$

This class is not stable by "+" and "max" operations either, but the parameters of power distributions bounding (in the $\leq_{\mathrm{icx}}$ order) the sum and the maximum of variables with a power distribution can be easily computed. This, combined with Theorem 2.9, allows to derive a bound for $T(\mathcal{G})$ in the $\leq_{\mathrm{icx}}$ sense which has the advantage of being quickly computable. Cubaud [12] uses in addition the family of Pareto distributions to obtain a lower bound for $ET(\mathcal{G})$ in the case of "or" networks. A combination of both gives a heuristic for mixed networks (section 3.1).

Topological Simplifications Based on Theorem 2.8, it is possible to perform structural changes to a PERT network while guaranteeing that the execution of the new network we bound above or below the original one. Removing nodes and edges ("undersynchronizing") will provide a lower bound, and adding nodes and edges ("oversynchronizing") will give an upper bound in the $\leq_{\mathrm{st}}$ sense [27, 32].

The aim is to reduce the network to one more easily solvable, such as SP networks (down to networks with disjoint paths [45]), or networks with small irreducible components. Figure 5 illustrates this principle.

However, as for the duplication technique above, no efficient, general and fast procedure is known, and the simplifications have to be performed on a heuristic basis.

2.3 Simulation

Several simulation techniques can be envisioned to estimate the statistics of $T(\mathcal{G})$.

The most commonly used technique for simulating random dynamical systems is the "Monte Carlo" (discrete event) simulation of the evolution of activities using an event

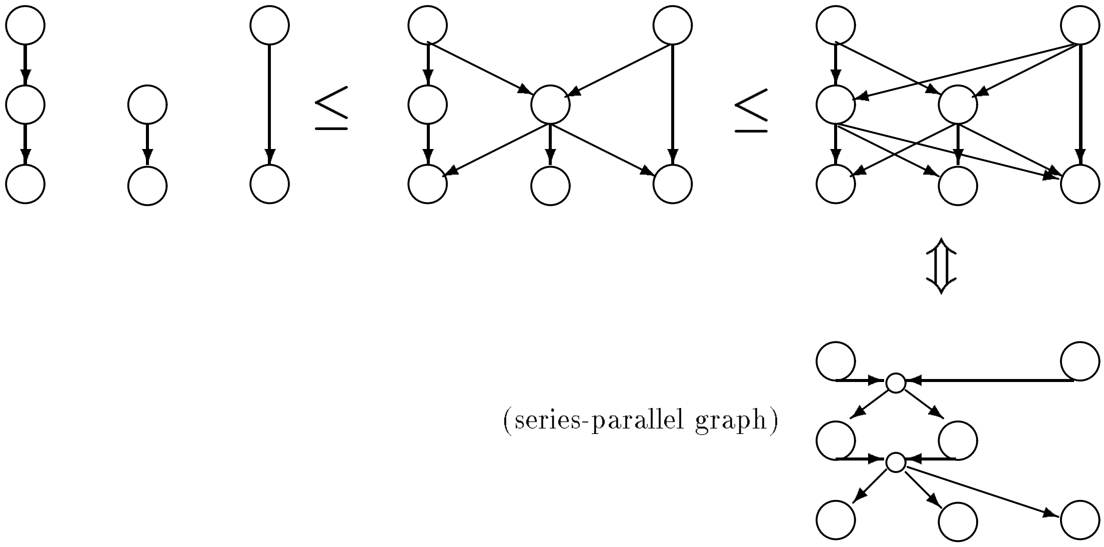


Figure 5: Bounds based on structural simplifications

list and computing the successive dates of beginning and end of activities. The statistics are performed by computing the execution time of a large number of graphs (replication method). Variance reduction techniques have been investigated by several authors, and we refer to [1] again for more bibliography and details.

In the case of PERT networks, however, the existence of mathematically known relations between the instants $X(k), k \in \mathcal{V}$ allow an "equational" simulation of the system. Accordingly, one may successively compute values of $X(k)$ using (3). This property was used in 2.1.1 and allows to compute the execution time of a graph very quickly. Note that the sequence of computed instants is not necessarily increasing, which makes this technique different from the above one. In particular, computing quantities such as the number of active task at a particular instant requires additional work with equational simulation whereas it is readily available with discrete event simulation. This approach applies also to some extensions of the PERT model in section 3.

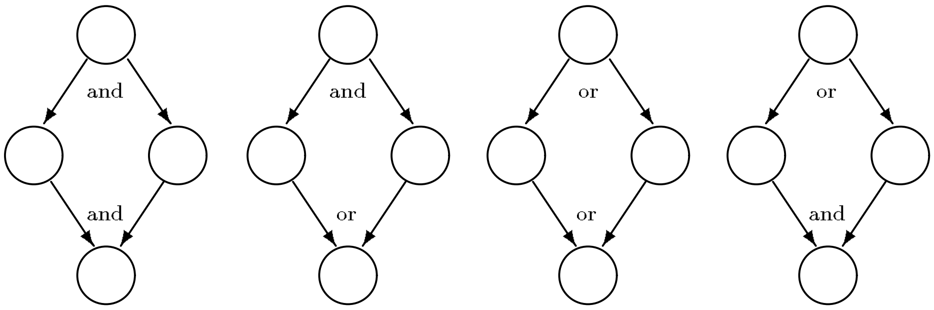
Note finally that the particular form of (3) makes it suitable for parallel simulation, as discussed in [22, 3].

3 Extension to the PERT model

3.1 Logic Networks

Other synchronization structures are necessary in order to represent the execution of complex projects, or parallel programs on parallel architectures. Several authors have proposed generalizations of PERT networks (which use the "and" synchronization primitive) and their solution techniques to networks using other logic operations on events [40, 8, 26]. For instance, Kleindorfer and Kleindorfer generalize in [26] the bounds of [25] (see (7) and theorem 2.10) to a class of networks using logic operations and (integer and bounded) random delays.

In [12], two synchronizations primitives are used: "and" and "or". Moreover, the



(a) (b) (c) (d)

Figure 6: Different precedence structures

set of edges of the graph is not assumed to be deterministic anymore. One may see this as the possibility for tasks not to induce dependencies (or not to send messages) to other tasks in a probabilistic fashion. The graph must combine synchronizations and dependencies carefully so as to avoid deadlocks. For instance, only the three first of the four combinations of figure 6 give valid task graphs.

Other synchronization primitives of the "k out of n" family, familiar in reliability theory, can be used. The corresponding enabling function of a nodes uses then order statistics of the execution dates of its input arcs.

If the graph contains nodes with synchronization constraints of one of these three valid types above, theorem 2.4 still holds. If the graph contains only "or" synchronization constraints, one can state a theorem similar to theorem 2.5 [12]:

Theorem 3.1 *If $\mathcal{G}$ is a task graph with "or" synchronization constraints, then:*

$$\{\tilde{\mathcal{L}}(i)\}_{\mathcal{V}\cup\mathcal{E}} \leq_{\text{icv}} \{\mathcal{L}_i(n)\}_{\mathcal{V}\cup\mathcal{E}} ,$$

then for all $k \in \mathcal{V}$,

$$\{\tilde{X}(k)\}_{k\in\mathcal{V}} \leq_{\text{icv}} \{X(k)\}_{k\in\mathcal{V}} .$$

In particular,

$$T(\tilde{\mathcal{G}}) \leq_{\text{icv}} T(\mathcal{G}) .$$

However, if the graph contains both "or" and "and" synchronizations, no result is known for stochastic orderings weaker than $\leq_{\text{st}}$.

Theorem 3.2 *If $\mathcal{G}$ is a task graph with "and" or "or" synchronization constraints, then:*

$$\{\tilde{\mathcal{L}}(i)\}_{\mathcal{V}\cup\mathcal{E}} \leq_{\text{st}} \{\mathcal{L}_i(n)\}_{\mathcal{V}\cup\mathcal{E}} ,$$

then for all $k \in \mathcal{V}$,

$$\{\tilde{X}(k)\}_{k\in\mathcal{V}} \leq_{\text{st}} \{X(k)\}_{k\in\mathcal{V}} .$$

In particular,

$$T(\tilde{\mathcal{G}}) \leq_{\text{st}} T(\mathcal{G}) .$$

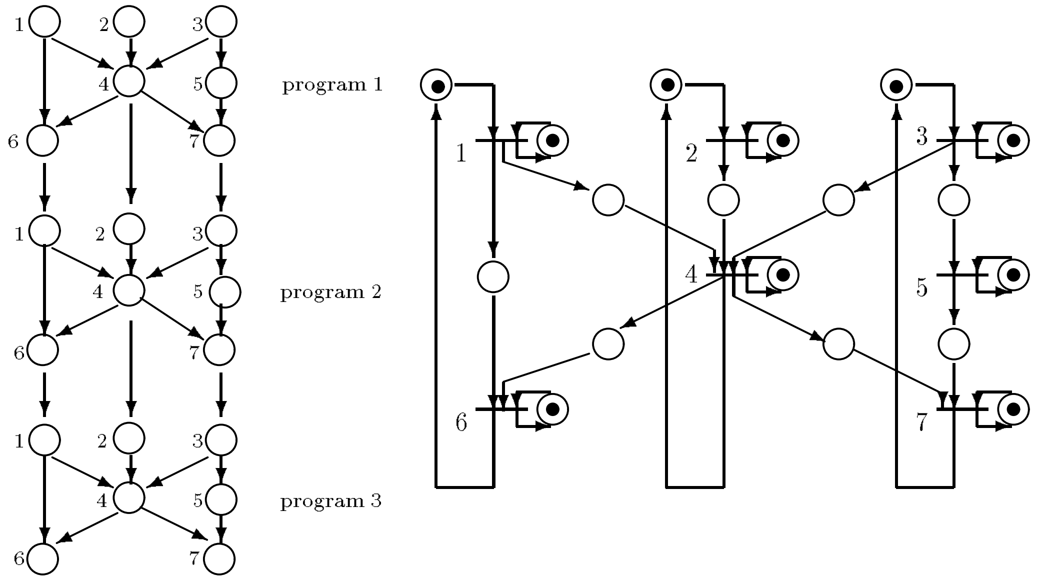


Figure 7: A repeating task graph and its Petri Net equivalent

3.2 Repeating Task Graphs

A repeating task graph $\mathcal{G}$ is loosely defined as a task graph (sometimes considered as infinite) in which one can identify a sequence of subgraph $\mathcal{G}_n$ such that $\mathcal{G}$ is the "concatenation" of the $\mathcal{G}_n$, and all these subgraph have the same characteristics. For instance, all graphs of the sequence $\{\mathcal{G}_n\}_n$ may be PERT networks with the same topology and identical duration distributions (periodic task graphs) or may have a random topology, with a certain distribution. An example of repeating PERT network appears in figure 7.

For such graphs, the problem of finding the distribution of $T(\mathcal{G})$ is reformulated in asymptotic terms. A quantity of interest is then the limit (in distribution, when it is defined):

$$T \ = \ \lim_{N\to\infty} \frac{1}{N} \, T(\mathcal{G}(N)) \ ,$$

where $\mathcal{G}(N)$ denotes the restriction of $\mathcal{G}$ to the subgraphs $\{\mathcal{G}_n, n \leq N\}$. This quantity represents the asymptotic delay between two successive completion of subgraphs, and it is related to the *throughput* θ of the system modeled (measured in subgraphs executed by unit time) by:

$$\theta \ = \ \frac{1}{E[T]} \ .$$

In this case, one tries to use the existence of the pattern to assert that this limit indeed exists, and to compute or estimate it based on the structure of the smaller graphs $\mathcal{G}_n$.

Repeating PERT networks are equivalent to a class of Timed Petri Nets called Marked graphs. This property has been used [11] for solving periodic scheduling problems in the case of deterministic execution times.

An illustration of the transformation is given in figure 7.

This equivalence allows to use theoretic results [4] and techniques that are known for computing, bounding or approximating the performance of Stochastic Petri Nets [6]. In particular, the usual Markovian technique [2] (which is similar to that of section 2.1.4

applies when task durations have exponential (or, in extensions, Cox) distributions or zero duration. The evolution of the execution of the task graph is then described by a Markov Chain. If a certain *dependency graph* (derived from the underlying graph of the $\mathcal{G}_n$'s) is strongly connected, the marked graph is "autonomous" [4] and the chain is finite and can be solved in principle. If it has two strongly connected components, matrix geometric techniques [38] may be applied [18]. The throughput of the task graph is then computed from the stationary distribution of the chain.

3.3 Graphs with Random Topology

Task graphs with random topology occur in models representing the execution of parallel programs on multi-purpose multiprocessor machines. The randomness of the structure tries to take into account the facts that the exact structure of the task graph of a particular instance of a program may not be known beforehand, and also that on a multiprogrammed system, programs of different nature may follow one another, making the successive task graphs appear to be "random". Another occurrence of random task graphs appears when a (determined) parallel algorithm is applied to a random data.

Recursively defined graphs. In [17], Fayolle *et al.* consider task graphs with an out-tree topology. The tree is randomly generated, each task of level i being able to generate a random number of parallel tasks at level $i+1$. In some cases, the average execution time of the tree can be computed explicitly or numerically.

In [20], Gelenbe *et al.* study a family of series-parallel task graphs with random topology. The graphs are generated as above according to a sort of branching process, where each node of some graph generation is now able to give birth to (transform into) a pure series or pure parallel subnetwork. An approach using generating functions allows to compute statistics of the total execution time.

Graphs with random precedence. Another class of random task graphs is generated using probabilistic precedence constraints: a node i a node $j > i$ with a certain fixed probability. For this class of graphs, closed form formulas for $T(\mathcal{G})$ seem unlikely to exist, and, as in the case of the repetitive graphs of section 3.2, one looks for the asymptotics of $T(\mathcal{G})$ when N grows. Under suitable statistical assumptions, it is possible to prove the existence of a limit to the ratio $T(\mathcal{G}_N)/N$ as $N \to \infty$ [21, 7]. The computation of this limit is however still an open problem. Under slightly different assumptions, Newman [39] obtains precise asymptotics for $T(\mathcal{G}_N)$.

References

[1] V. Adlakha and V. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966-1987. Technical report, Dept. of Op. Res., University of North Carolina, Chapel Hill, 1987.

[2] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* MIT Press, Cambridge, Mass., 1986.

[3] F. Baccelli and M. Canales. Parallel simulation of stochastic petri nets using recursive equations. *ACM Tomacs*, 1993.

[4] F. Baccelli, G. Cohen, G.-J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. Wiley, 1992.

[5] F. Baccelli and T. Konstantopoulos. Large deviation estimates of cycle times in stochastic petri nets. In *Lecture Notes in Computer Science*, volume 177. Springer Verlag, 1991.

[6] F. Baccelli and Z. Liu. Comparison properties of stochastic decision free petri nets. In *First QMIPS Workshop, Sophia-Antipolis*, Nov. 1992.

[7] F. Baccelli and Z. Liu. On a class of stochastic recursive sequences arising in queueing theory. *The Annals of Probability*, 20(1):350–374, 1992.

[8] M. Bakes, M. Bramson, S. Freckelton, P. Roberts, and D. Ryan. Stochastic-network reduction and sensitivity techniques in a cost effectiveness study of a military communications system. *Operational Research Quartely*, 21(conference issue):45–67, 1970.

[9] R. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing*. Holt, Rinehart and Wintson, New York, 1975.

[10] C. Bigelow. Bibliography on project planning and control by network analysis: 1959-1961. *Operations research*, 10:728–731, 1962.

[11] J. Carlier and P. Chrétienne. *Problèmes d'ordonnancement*. Masson, 1988.

[12] P. Cubaud. *Modèles et outils pour la prédiction de performances des systèmes informatiques parallèles*. PhD thesis, Université René Descartes, Paris V, Mar. 1992.

[13] L. Devroye. Inequalities for the completion times of stochastic PERT networks. *Mathematics of Operations Research*, 4(4):441–447, Nov. 1979.

[14] B. Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33(4):862–881, 1985.

[15] B. Dodin. Reducibility of stochastic networks. *OMEGA Intl. J. of Mgmt. Sci.*, 13(3):223–232, 1985.

[16] S. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. Wiley, New York, 1977.

[17] G. Fayolle, P. King, and I. Mitrani. On the execution of programs by many processors. In A. Agrawala and S. Tripathi, editors, *Performance'83*. North Holland, 1983.

[18] G. Florin and S. Natkin. Les réseaux de Petri stochastiques. *TSI*, 4(1):143–160, 1985.

[19] H. Frank. Shortest paths in probabilistic graphs. *Operations Research*, 17:583–599, 1969.

[20] E. Gelenbe, E. Montagne, R. Suros, and C. Woodside. A performance model of block structured parallel programs. In Cosnard et al., editor, *Parallel Algorithms & Architectures*, pages 127–138. North-Holland, 1986.

[21] E. Gelenbe, R. Nelson, T. Philips, and A. Tantawi. Asymptotic processing time of a model of parallel computation. In *Proc. National Computer Conference, Las Vegas*, Nov. 1986.

[22] G. Greenberg, I. Mitrani, and B. Lubachevsky. Algorithms for unboundedly parallel simulations. *ACM Transactions on Computers Systems*, 9(3), Nov. 1991.

[23] H. Hartley and A. Wortham. A statistical theory for PERT critical path analysis. *Management Science*, 12(10):B–469–B–481, June 1966.

[24] J. Kamburowski. Two-point approximations for activity times in PERT networks. *RAIRO Recherche Opérationelle*, 19(3):301–313, Aug. 1985.

[25] G. Kleindorfer. Bounding distributions for a stochastic acyclic network. *Operations Research*, 19(7):1586–1601, 1971.

[26] G. Kleindorfer and P. Kleindorfer. Bounding distributions for stochastic logic networks. *Operational Research Quarterly*, 25(3):465–479, 1974.

[27] W. Kleinöder. Evaluation of task structures for a hierarchical multiprocessor system. In *Intl. Conf. on Modelling Techniques and Tools for Performance Analysis*, Paris, May 1984.

[28] V. Kulkarni and V. Adlakha. Markov and Markov-regenerative PERT networks. *Operations Research*, 34(5):769–781, Sept. 1986.

[29] T. Lai and H. Robbins. A class of dependent random variables and their maxima. *Zeit. für Wahrsc. und Verwandte Geb.*, 42(2):89–111, 1977.

[30] C. Lefèvre and P. Laloyaux. Note sur les problèmes d'ordonnancement PERT aléatoires. *R.A.I.R.O. Recherche Opérationnelle/Operations Research*, 19(1):27–33, Feb. 1985.

[31] S. Lerda-Olberg. Bibliography on network-based project planning and control techniques: 1962-1965. *Operations Research*, 14(5):925–931, 1966.

[32] Z. Liu. *Modèles d'exécution parallèles sur des systèmes multiprocesseurs: analyse et optimisation.* PhD thesis, Université Paris Sud, Orsay, Jan. 1989.

[33] J. Łukaszevicz. On the estimation of errors introduced by standard assumptions concerning the distribution of activity duration in PERT calculations. *Operations Research*, 13:326–327, 1965.

[34] K. MacCrimmon and C. Ryavec. An analytical study of the PERT assumptions. *Operations Research*, 12:16–37, 1964.

[35] P. Malcolm, J. Roseboom, C. Clark, and W. Fazar. Application of a technique for research and development. *Opns. Res.*, 7:646–669, 1959.

[36] J. Martin. Distribution of the time through a directed, acyclic network. *Operations Research*, 13(1):46–66, 1965.

[37] A. Nádas. Probabilistic PERT. *IBM Journal of Research and Development*, 23(3):339–347, May 1979.

[38] M. Neuts. *Matrix Geometric Solution of Stochastic Models.* John Hopins University Press, Baltimore, 1981.

[39] C. Newman. Chain lengths in certain random directed graphs. *Random Structures and Algorithms*, 3(3):243–253, 1992.

[40] A. Pritsker and W. Happ. GERT: graphical avaluation and review technique – I. fundamentals. *J. Industr. Eng.*, XVII(5), 1966.

[41] L. Ringer. Numerical operators for statistical PERT critical path analysis. *Management Science*, 16(2):B–136–B–143, Oct. 1969.

[42] P. Robillard and M. Trahan. Expected completion time in PERT networks. *Operations Research*, 24(1):177–182, 1976.

[43] P. Robillard and M. Trahan. The completion time of PERT networks. *Operations Research*, 25(1):15–29, Jan. 1977.

[44] R. Sahner and K. Trivedi. Performance and reliability analysis using directed acyclic graphs. *IEEE Transactions on Software Engineering*, SE-13(10):1105–1114, Oct. 1987.

[45] H. G. Spelde. Bounds for the distribution function of network variables. In *First Symposium on Operations Research*, 3, pages 113–123. University Heidelberg, 1977.

[46] D. Stoyan. *Comparison Methods for Queues and other Stochastic Models.* Wiley, New York, 1984.

[47] D. Welsh. Errors introduced by a PERT assumption. *Operations Research*, 13:141–143, 1965.

[48] N. Yazici-Pekergin and J.-M. Vincent. Stochastic bounds on execution times of parallel programs. *IEEE Transactions on Software Engineering*, 17(10):1005–1012, Oct. 1991.