Sensitivity Analysis in Stochastic Scheduling

Master Thesis

by

Carlo Attanasio

Student number	6148409
Project duration	Jan 2026 – Jul 2026

Supervisor	Prof. Dr. D. Kurowicka	Delft University of Technology
External supervisor	Dr. A. Piedrafita Postigo	TNO ESI
Committee member	Prof. Dr. J. Söhl	Delft University of Technology

TU Delft Delft University of Technology

TNO ESI Powered by industry and academia

Contents

1

Introduction

1.1. Context

Stochastic scheduling plays a key role in understanding how uncertainty propagates through complex machine logistics processes. In the domain of automated and semi-automated industrial systems, efficiency and reliability are essential.

Regular scheduling refers to the process of assigning tasks to resources over time, subject to a set of constraints. In its most basic form, it answers three questions: which tasks need to be performed, in what order, and on which resources. Constraints may arise from task dependencies, as some tasks must be completed before others can start, resource limitations, since a machine can handle only one task at a time, or timing requirements, such as deadlines or release times. The objective of a scheduling problem typically involves optimizing a performance criterion, such as minimizing total completion time, reducing delays, or maximizing resource utilization.

Deterministic and stochastic scheduling differ in how task durations are modeled and, consequently, in the nature of the resulting optimization problem. In deterministic scheduling, all task durations are assumed to be known and fixed in advance, which implies that the schedule and its performance metrics, such as total completion time, are also fixed and predictable. In contrast, stochastic scheduling models task durations as random variables, introducing uncertainty into the system. As a result, the schedule no longer yields a single deterministic outcome, but rather a distribution of possible outcomes.

To analyze these systems, individual tasks or machine operations are naturally represented as nodes in a directed acyclic graph (DAG), where directed edges denote the dependencies between tasks. Acyclicity prevents logical contradictions where task a precedes task b, while task b simultaneously precedes task a. This network is sometimes called a PERT (Project Evaluation and Review Technique) or CPM (Critical Path Method) network [24, 20, 22]. One quantity of interest for many applications is the time in which all tasks are completed, which is given by the length of the longest path in the network.

In a deterministic setting, where task durations are fixed, known quantities, the problem of determining the longest path in acyclic DAGs is solvable in linear time using topological sorting [19], although it is NP-complete for general non-acyclical directed graphs [28]. Optimization in this context has been an extensive topic of research and is well-understood.

Optimizing such systems requires more than just estimating the expected completion time; it requires a deep understanding of how local uncertainties propagate to the global system performance. From an industrial perspective, the ability to identify and mitigate these sources of delay has a direct impact on efficiency, productivity, and sustainability. By identifying which tasks most significantly influence the variability of the total production time, industries can implement targeted interventions, such as

preventive maintenance or scheduling adjustments, reducing production variability, improving energy efficiency, and allowing for less wasteful production practices.

1.2. Related work

Project Evaluation Review Technique (PERT) [24] and Critical Path Method (CPM) [20] are known standard approaches to the stochastic scheduling problem, which both aim to approximate the expected duration of the schedule. CPM accomplishes this goal by transforming the stochastic problem into a deterministic one using approximations for the length of each task, while PERT tries to directly approximate the expected longest path.

Estimating the network duration requires combining various random variables via addition and maximum operations, a fundamental and well-researched problem in operations research. Significant research focuses on symbolically computing the longest path length distribution. Ando et al. [2] and Conti [12] assume normally distributed task durations, whereas Martin [25] and Clark [10] model task random variables using polynomial density functions. Building on this, Canon et al. [7] propose correlation-aware heuristics for random-weight directed acyclic graphs (DAGs). Extensive additional literature addresses completion time bounding and conditional distributions for stochastic networks [6, 13, 21].

However, to the best of our knowledge, the current research lacks explicit treatment of exact analytical gradients for the longest path with respect to the task's parameters, which is what we aim to expand on in this thesis project.

1.3. Problem statement

The primary mathematical challenge in stochastic scheduling is that the longest path in a DAG is defined by the maximum operator over a set of path sums. When node timings are described by random variables, both the length and the identity of the longest path become random variables, since multiple distinct paths may have a non-zero probability of being the longest path.

While it is relatively straightforward to estimate the empirical distribution of the longest path via Monte Carlo simulation, this approach lacks analytical insight, as it does not clearly reveal how sensitive the global outcome is to changes in local parameters. The core problem addressed in this thesis is the derivation of an exact analytical framework to perform sensitivity analysis. Specifically, we aim to determine how the parameters of the underlying node distributions influence the distribution of the longest path duration.

Evaluating sensitivity with respect to these specific distribution parameters is crucial because, in industrial machines, these parameters represent the physical "knobs" that operators can directly control and optimize. Furthermore, in practical settings, certain parameters are often shared across multiple distinct node distributions while the underlying random variables remain independent. For example, all "move" actions of all robotic arms may be governed by independent random variables whose distributions depend on a global parameter. Determining the sensitivity to these shared parameters is necessary for assessing the system-wide impact of hardware tuning.

1.4. Research objective and structure

The primary objective of this study is to develop an analytical framework for calculating the sensitivity of the longest path distribution in stochastic directed acyclic graphs. Moving beyond traditional approximation methods, this research derives exact formulas for the partial derivatives of the longest path with respect to the network parameters. To achieve this, we first model the scheduling process as a DAG where path durations are expressed as linear combinations of node durations (Chapter 2). Subsequently, we derive closed-form analytical expressions for the gradients of the expected value of a

target function $g : \mathbb{R} \to \mathbb{R}$, which can represent different risk metrics, evaluated on the longest path duration under the assumption that the node distributions are Gaussian (Chapter 3). Then, this work extends the sensitivity analysis beyond the strict Gaussian assumption by proving a general theorem that establishes gradient formulas for a broader family of distributions (Chapter 4). Finally, we apply the developed framework to analyze the impact of wait times within the network (Chapter 5), enabling the modeling of both traditional and Just In Time (JIT) scheduling configurations [3] (Chapter 6).

The code used for the simulations shown in this paper is available as a GitHub repository [8] at:
`https://github.com/carlo-attanasio/SensitivityAnalysisStochasticScheduling`

2

Preliminaries

Before deriving the sensitivity analysis for the longest path distribution, we must establish the graph-theoretic framework and the stochastic model governing the network. This chapter introduces the notation for directed acyclic graphs (DAGs) and defines the linear algebraic relationship between node and path durations.

2.1. Graph Theoretic Framework

To model the scheduling problem, we utilize a Directed Acyclic Graph (DAG) [29], specifically following the Activity-on-Node (AoN) convention, where the set of nodes $\mathcal{V} = \{1, \ldots, k\}$ represents distinct tasks or machine operations, and the set of arcs $\mathcal{A}$ represents the precedence dependencies between them.

Definition 2.1 (Directed Acyclic Graph)**.** *A directed graph $G = (\mathcal{V}, \mathcal{A})$ is defined by a finite set of vertices $\mathcal{V} = \{1, \ldots, k\}$ (commonly referred to as nodes), and a set of ordered pairs $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$, termed arcs. Each arc $(x, y) \in \mathcal{A}$ has a weight $w_{(x,y)}$, and represents a unidirectional connection originating at node x and terminating at node y. The graph is a Directed Acyclic Graph (DAG) if it also contains no directed cycles; that is, there is no sequence of arcs $(v_1, v_2), (v_2, v_3), \ldots, (v_{m-1}, v_m)$ such that $v_1 = v_m$.*

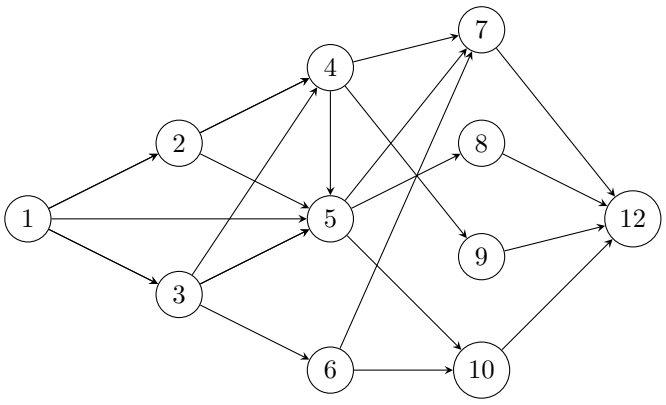


Figure 2.1: Example of a valid directed acyclic graph with 12 nodes, with a unique source (1) and sink (12) nodes.

To further characterize the structure of the graph, we introduce basic notions related to node connectivity and path construction.

The in-degree of a node $v \in \mathcal{V}$ is the number of arcs $(u, v) \in \mathcal{A}$ entering it, while the out-degree is the number of arcs $(v, w) \in \mathcal{A}$ leaving it.

Within the directed acyclic graph G, we assume the existence of a unique source, defined as a node with in-degree zero, and a unique sink, defined as a node with out-degree zero. These represent the project's formal start and completion points, respectively (Figure 2.1).

Note that any DAG can be modified to fit this assumption, and to do so, we have to find all source and sink nodes, and create two nodes in the graph: the first one will be connected to all source nodes, while all sink nodes will be connected to the second one. By doing this, we create two unique source and sink nodes that represent the start and end of the project, making the original DAG fit our assumption.

Moreover, we also assume that the DAG is connected, meaning that every node can be reached from the source node.

In the Activity-on-Node (AoN) representation adopted here, task durations are associated with nodes, meaning that all outgoing arcs from a given node have the same weight, as they represent the completion of the node. This weight corresponds to the duration of the operation performed at that node, and is independent of which subsequent task is executed.

With this information, we can consider the maximal paths of the graph, defined as a sequence of arcs forming a directed path from the source to the sink such that no further arcs can be added. For the remainder of the thesis, we will refer to them simply as paths. For instance, the graph in Figure 2.1 contains 21 distinct paths between node 1 and node 12.

Definition 2.2 (Project Network)**.** *Let $G = (\mathcal{V}, \mathcal{A})$ be a directed acyclic graph with a unique source and a unique sink. Moreover, assume that all outgoing arcs from any node in G have the same weight:*

$$\forall v \in \mathcal{V},\ \exists w_v \in \mathbb{R} \text{ such that } \forall u \in \mathcal{V} \text{ where } (v, u) \in \mathcal{A} \implies w_{(v,u)} = w_v.$$

We define the Project Network as $PN(\mathcal{V}, \mathcal{P})$, where $\mathcal{P}$ denotes the set of all source-to-sink paths in G.

As mentioned earlier, the primary focus of the analysis lies in these weights. When they are fixed and deterministic, the problem reduces to a deterministic scheduling setting. In contrast, modeling them as random variables leads to a stochastic scheduling framework.

To ensure tractability, assumptions on the admissible distributions will be introduced. In Chapter 3, all weights are assumed to be either constant or Gaussian distributed. This assumption is later relaxed in Chapter 4, where more general distributional families are considered.

We will frame the scheduling problem using the longest paths in this DAG, which will always start at the source node and end at the sink node, since by construction, those are the two farthest apart nodes. Simply, a path from the source to the sink is defined as a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that v_1 is the source, v_k is the sink, and $(v_i, v_{i+1}) \in \mathcal{A}$ for all $1 \leq i < k$.

Formally, we represent the node and path durations as vectors:

Definition 2.3 (Node Vector)**.** *Let $V \in \mathbb{R}^k$ be the vector of node durations:*

$$V = [V_1 \ \cdots \ V_k]^\top,$$

where V_j represents the duration of the j-th task.

Note that since the first node represents the "start" action, we will assume that its duration will always be zero. Its function is to signal its successors to start.

As mentioned earlier, these nodes can be random variables with Gaussian (Chapter 3) or more general distributions (Chapter 4).

We will now give a formal definition of the paths in this graph. As mentioned earlier, the paths we consider are the maximal paths, i.e. paths that are already extended to reach their maximum possible length. In the setup we have created, such paths always start at the source node and end at the sink node, therefore, this translates to enumerating all unique paths between these two nodes. In Figure 2.2 some paths are shown on the graph.

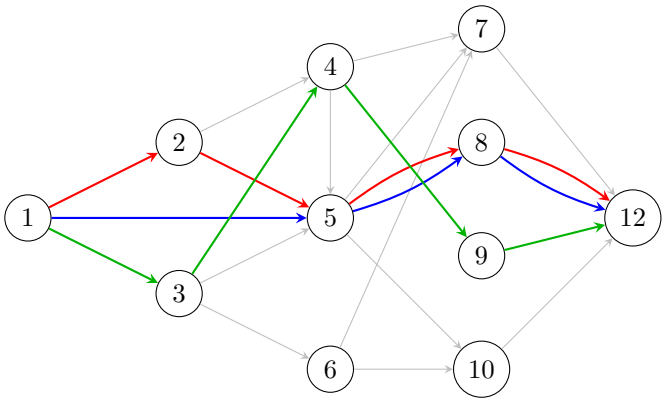


Figure 2.2: A few possible paths shown in the graph introduced in Figure 2.1

Definition 2.4 (Path Vector)**.** *Let $P \in \mathbb{R}^n$ be the vector of path durations, where n is the number of unique source to sink paths. Then:*

$$P = [P_1 \ \cdots \ P_n]^\top,$$

where P_i represents the total duration of the i-th path in the graph. Moreover, given a path P_i we will denote by $\mathcal{V}(P_i)$ the set of nodes that make up path P_i.

The relationship between the local task durations and the global path durations is linear, since the duration of any path is simply the sum of the durations of the nodes that constitute it. We formalize this using the path-node incidence matrix.

Definition 2.5 (Path-Node Incidence Matrix)**.** *Let $A \in \{0,1\}^{n\times k}$ be the matrix defined by:*

$$A_{ij} = \begin{cases} 1 & \text{if node } V_j \text{ belongs to path } X_i, \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, the vector of path durations P is expressed as a linear transformation of the vector V:

$$P = AV. \tag{2.1}$$

2.2. Graph Algorithms

Let $PN(\mathcal{V}, \mathcal{P})$ be a Project Network, and recall that $\mathcal{P}$ denotes the set of all source-to-sink paths. The analysis and manipulation of $\mathcal{P}$ rely on structural properties of the network that can be efficiently exploited through classical graph algorithms. In particular, topological sorting [19] provides an ordering of the vertices $\mathcal{V}$ such that each node appears only after all nodes corresponding to its immediate predecessors, thereby reflecting the partial order induced by the directed arcs.

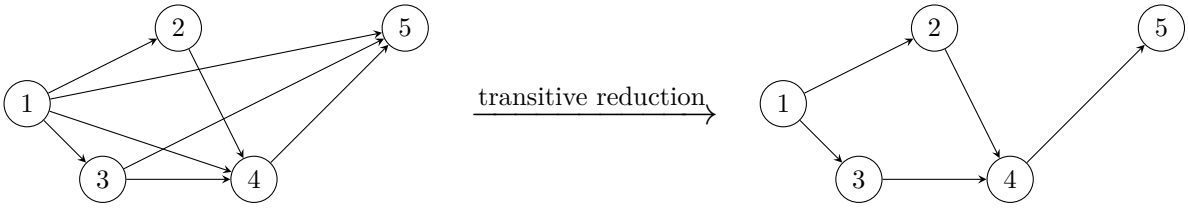
2.2.1. Transitive reduction

An examination of the paths in Figure 2.2 reveals a potential inefficiency in our current model: the presence of redundant paths. Consider, for example, the sequences $1 \to 2 \to 5 \to 8 \to 12$ and $1 \to 5 \to 8 \to 12$. The first path, which routes through node 2, is always longer than the alternative,

and since our primary objective is to identify and analyze the longest path, we can simplify the graph by removing these superfluous arcs.

This reduction also aligns with intuitive project logic: if action 5 requires the completion of both actions 1 and 2, yet action 2 strictly depends on action 1, the condition $1 \to 5$ is already satisfied by the dependency $2 \to 5$. Consequently, explicitly including the direct arc $1 \to 5$ is redundant and can be safely omitted.

The algorithm used to remove redundant arcs is called transitive reduction [1]. Formally, the transitive reduction of $G = (\mathcal{V}, \mathcal{A})$ is a graph $G^- = (\mathcal{V}, \mathcal{A}^-)$ such that for any pair of nodes $u, v \in \mathcal{V}$, there exists a directed path from u to v in G^- if and only if such a path exists in G, and $\mathcal{A}^-$ is minimal with respect to this property.



In the context of a directed acyclic graph, the transitive reduction is unique and can be obtained by removing all redundant arcs. An arc $(u, v) \in \mathcal{A}$ is said to be redundant if there exists an alternative directed path from u to v of length greater than one. Such arcs do not contribute new connectivity information and, therefore, can be eliminated without affecting the relationship of precedences between nodes described by the graph.

A constructive way to compute the transitive reduction exploits the acyclic structure of G. Given a topological ordering $(v_1, \ldots, v_k)$, one can examine each arc (u, v) and determine whether v is reachable from u via intermediate nodes that appear between them in the ordering. If such a path exists, the arc (u, v) is removed; otherwise, it is retained.

By eliminating all redundant arcs, the resulting graph G^- retains exactly the essential precedence relations of the original network. This reduction can simplify subsequent computations, including path enumeration and optimization procedures, by operating on a structurally minimal graph.

For instance, if we apply transitive reduction to the graph in Figure 2.1, the algorithm removes the following arcs:

$$2 \to 5, \qquad 4 \to 7, \qquad 3 \to 5, \qquad 1 \to 5.$$

After the reduction, the total number of paths drops from 21 to just 10.

We will therefore, without loss of generality, always assume that any project network $PN(\mathcal{V}, \mathcal{P})$ has been transitively reduced.

2.2.2. Path enumeration

The enumeration of these paths is a well-defined operation because the acyclic nature of G ensures that every directed path is finite, thereby avoiding the infinite recursion present in non-acyclic graphs [14].

Once a topological ordering of the network is found, it can be used to systematically enumerate all source-to-sink paths in $PN(\mathcal{V}, \mathcal{P})$. Let $(v_1, \ldots, v_k)$ be a topological ordering of the nodes. By construction, all arcs $(u, v) \in \mathcal{A}$ satisfy $\text{index}(u) < \text{index}(v)$, ensuring that when processing a node, all its predecessors have already been considered.

This ordering enables a dynamic programming approach to path enumeration. Define for each node $v \in \mathcal{V}$ the set $\mathcal{P}(v)$ as the collection of all directed paths from the source node s to v. Initialization is

given by $\mathcal{P}(s) = \{\,(s)\,\}$. Then, proceeding in topological order, for each node $v \neq s$, the set $\mathcal{P}(v)$ can be constructed as

$$\mathcal{P}(v) = \bigcup_{(u,v)\in\mathcal{A}} \{\, p \oplus v \mid p \in \mathcal{P}(u) \,\},$$

where $p \oplus v$ denotes the path obtained by appending node v to path p.

Because of the acyclic structure, this procedure guarantees that all paths are constructed exactly once and that no cycles are introduced. Finally, the set of all source-to-sink paths is obtained as $\mathcal{P} = \mathcal{P}(t)$, where t is the sink node.

Moreover, note that since we are using a topological ordering of the graph, we will always have that node 1 is the unique source node, while the last node, node k, is the unique sink node.

2.3. A Simple Example: Bowling Ball Manufacturing

To illustrate the problem of sensitivity analysis in stochastic scheduling, we will apply the techniques developed to a concrete running example for the rest of our work.

Figure 2.3 illustrates the sequence of actions for a machine designed to drill three holes into a bowling ball. The process begins with a robotic arm moving a ball from the input zone to a conditioner. The conditioner then heats the ball, softening the material to facilitate drilling. Next, a second robotic arm transports the ball to the drilling table, where three holes are sequentially bored. Finally, the ball is moved to the output zone.

The graph in Figure 2.3 visualizes this entire process. It contains 50 nodes, 60 arcs, and 57 distinct paths from node 1 to node 50; it is already reduced to its simplest form. It is laid out such that the horizontal axis roughly corresponds to the chronological progression of time, while the vertical axis is divided into distinct "rows." Each row contains nodes representing actions performed by a specific peripheral.

The machine consists of seven distinct peripherals, which map to the rows of the graph from top to bottom (excluding nodes 1 and 50, which represent the start and end actions of the schedule):

Row	Peripheral	Possible Actions
1	Drill Index Finger	Move, On/Off
2	Drill Middle Finger	Move, On/Off
3	Drill Thumb Finger	Move, On/Off
4	Drill Table	Rotate
5		Move
6		Clamp/Unclamp
7	Conditioner	Condition
8		Clamp/Unclamp
9	Robot 1	Move
10		Clamp/Unclamp
11	Robot 2	Move
12		Clamp/Unclamp

Table 2.1: Peripherals and their corresponding possible actions.

Additionally, each node is color-coded based on its corresponding "activity." We define an activity as

a collection of related actions that collectively achieve a specific goal. For instance, the black activity in the lower-middle section of the graph represents the sequence required to "move the ball from the conditioner to the drill table." Likewise, the magenta activity in the bottom-right corner denotes "move the ball from the drill table to the output."

While the exact definition of an activity and the rules for assigning nodes to one are not strictly rigorous, this will have no impact whatsoever on the analysis of the graph, as these groupings serve purely as a visualization aid.

Table 2.2 highlights the different activities described earlier, while a guide of all the nodes, describing which nodes represent which action, is given in Table 2.3

Color	Activity	Color	Activity	Color	Activity
	(R1) Input → Condition		(R1) Condition → Input		Condition
	(R2) Output → Condition		(R2) Condition → Drill		(R2) Drill → Output
	Drill Index Finger		Drill Middle Finger		Drill Thumb
	(DT) Thumb → Index		(DT) Index → Middle		(DT) Middle → Thumb

Table 2.2: Color legend mapping each node to the corresponding activity.

Node	Peripheral	Action	Node	Peripheral	Action
1	N/A	Start	26	Robot 2	Move
2	Robot 1	Move	27	Drill Thumb	Move
3	Drill Index	On	28	Drill Thumb	Move
4	Drill Middle	On	29	Drill Thumb	Off
5	Drill Thumb	On	30	Drill Table	Move
6	Robot 1	Clamp	31	Drill Table	Rotate
7	Robot 1	Move	32	Drill Index	Move
8	Robot 1	Move	33	Drill Index	Move
9	Robot 1	Move	34	Drill Index	Off
10	Conditioner	Clamp	35	Drill Table	Rotate
11	Robot 1	Unclamp	36	Drill Table	Move
12	Robot 1	Move	37	Drill Middle	Move
13	Conditioner	Unclamp	38	Drill Middle	Move
14	Robot 1	Move	39	Drill Middle	Off
15	Conditioner	Condition	40	Drill Table	Move
16	Robot 2	Move	41	Drill Table	Rotate
17	Conditioner	Clamp	42	Robot 2	Move
18	Robot 2	Move	43	Robot 2	Clamp
19	Robot 2	Clamp	44	Drill Table	Unclamp
20	Conditioner	Unclamp	45	Robot 2	Move
21	Robot 2	Move	46	Robot 2	Move
22	Robot 2	Move	47	Robot 2	Move
23	Robot 2	Move	48	Robot 2	Unclamp
24	Drill Table	Clamp	49	Robot 2	Move
25	Robot 2	Unclamp	50	N/A	End

Table 2.3: Description of action and peripheral for each node in Figure 2.3

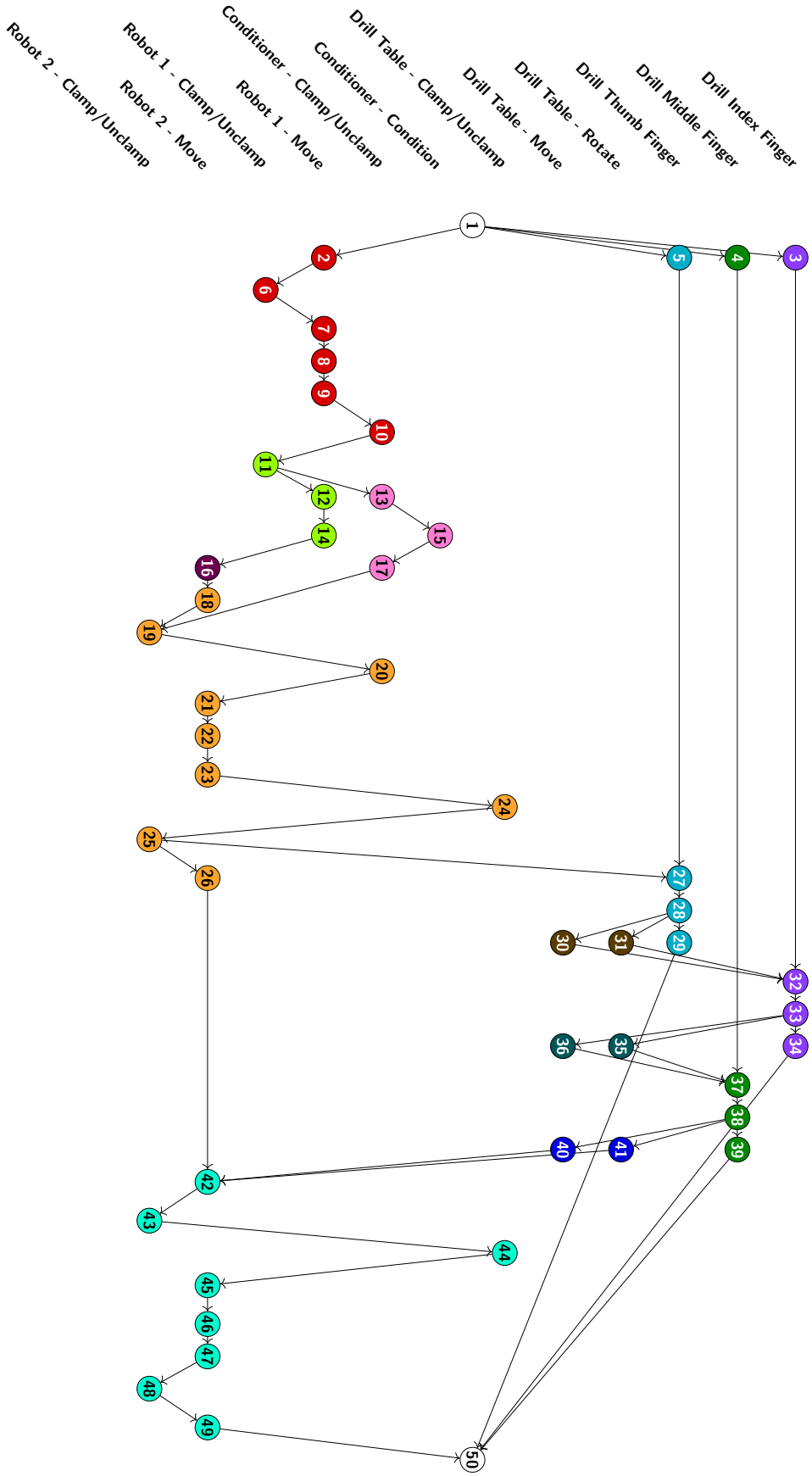


Figure 2.3: Action dependency graph for the bowling ball drilling machine. Nodes represent individual actions organized in rows by peripheral (as explained in Table 2.1), while colors group these actions into distinct, goal-oriented activities.

3

Sensitivity analysis under Gaussian assumptions

In the standard formulation of stochastic scheduling problems, task durations are traditionally modeled as purely random variables. However, to capture realistic system dynamics, we must accommodate tasks with deterministic execution times. Therefore, we assume that each task duration is either a known constant or a non-degenerate Gaussian random variable. To formalize this heterogeneous structure, we explicitly partition the network's vertices.

Definition 3.1 (Stochastic and Constant Nodes)**.** *Let $\mathcal{V}$ be the set of nodes in a project network $PN(\mathcal{V}, \mathcal{P})$. We partition $\mathcal{V}$ into two disjoint subsets: the set of stochastic nodes $\mathcal{V}_S$ and the set of constant nodes $\mathcal{V}_C$. Moreover, we denote their respective cardinalities as $k_S = |\mathcal{V}_S|$ and $k_C = |\mathcal{V}_C|$.*

Note that since Gaussian random variables could theoretically assume negative values, this formulation allows for negative task durations, which would strictly violate the causal progression of a physical schedule. However, by selecting distributions where the mean is several standard deviations above zero, the probability of realizing a negative duration becomes negligibly small in practice. Consequently, this Gaussian formulation serves as a mathematically tractable baseline to establish the exact sensitivity framework, which we will later generalize to strictly positive, bounded distributions in Chapter 4.

With this partition established, we specify the probabilistic foundation of the model:

Assumption 3.2 (Mixed Node Distributions)**.** *Given a project network $PN(\mathcal{V}, \mathcal{P})$ and a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, the node vector V is structured as follows: the constant component $\mathcal{V}_C$ is strictly deterministic, while the stochastic component $\mathcal{V}_S$ follows a non-degenerate multivariate normal distribution:*

$$V_S \sim \mathcal{N}(\mu_{V_S}, \Sigma_{V_S}).$$

where $\mu_{V_S} \in \mathbb{R}^{k_S}$ is the mean vector and $\Sigma_{V_S} \in \mathbb{R}^{k_S \times k_S}$ is the covariance matrix. Because we are considering the non-degenerate case, Σ_{V_S} is positive-definite and hence invertible.

As explained in Chapter 2, the path duration vector P is a linear combination of multivariate normal variables and deterministic constants (Equation 2.1), and thus it follows a, possibly degenerate, multivariate normal distribution. The primary quantity of interest is the total project completion time $\mathcal{L}$, defined as the maximum duration among all n possible source-to-sink paths:

$$\mathcal{L} = \max\{P_1, P_2, \ldots, P_n\}.$$

The sensitivity analysis in this thesis is performed for the random variable $\mathcal{L}$. We will denote the specific set of nodes constituting the longest path as $\mathcal{V}(\mathcal{L})$.

To perform gradient-based sensitivity analysis, $\mathcal{L}$ must be differentiable with respect to its underlying parameters. The maximum function over a finite set of smooth variables is continuously differentiable everywhere except at points where multiple arguments achieve the maximum simultaneously (i.e., when distinct paths tie for the longest duration). Let $\mathcal{T} \subset \Omega$ denote the set of these tie outcomes.

In a purely Gaussian network ($\mathcal{V}_C = \emptyset$), the absolute continuity of the joint distribution ensures that the probability of any two distinct paths sharing the exact same length is zero; thus, $\mathbb{P}(\mathcal{T}) = 0$, guaranteeing that $\mathcal{L}$ is almost surely differentiable. However, the introduction of deterministic nodes compromises this property. If two distinct paths traverse the exact same sequence of stochastic nodes but differ only in their constant nodes, the random components of their duration difference cancel out entirely. If this remaining deterministic difference evaluates to exactly zero, a tie occurs with probability one, rendering $\mathcal{L}$ locally non-differentiable. To avoid this, we must introduce a regularizing condition.

Assumption 3.3 (Partitioned Non-Degeneracy)**.** *Let $\mathcal{V}$ be a set of nodes partitioned in $\mathcal{V}_S$ and $\mathcal{V}_C$.*

1. *The joint distribution of V_S is absolutely continuous with respect to the k_S-dimensional Lebesgue measure on $\mathbb{R}^{k_S}$.*
2. *Any two distinct paths $P_i, P_j \in \mathcal{P}$ composed of the same set of stochastic nodes ($\mathcal{V}(P_i) \cap \mathcal{V}_S = \mathcal{V}(P_j) \cap \mathcal{V}_S$) must contain different subsets of deterministic durations. This requires the sum of their constant node durations to be strictly unequal:*

$$\sum_{x \in \mathcal{V}(P_i) \cap \mathcal{V}_C} V_x \neq \sum_{x \in \mathcal{V}(P_j) \cap \mathcal{V}_C} V_x.$$

In Theorem 3.4 we will show that these two conditions ensure that the set of ties has probability zero, preserving the almost-sure uniqueness of the longest path.

Our primary goal is to understand how changes in the parameters of the node distributions affect the expected project duration $\mathbb{E}[\mathcal{L}]$ and, more generally, its p-th moment $\mathbb{E}[\mathcal{L}^p]$. Specifically, we seek the gradient of $\mathbb{E}[\mathcal{L}^p]$ with respect to the mean vector of the nodes, μ_V.

Theorem 3.4 (Sensitivity of the p-th Moment of $\mathcal{L}$ with respect to node parameters)**.** *Let $\mathcal{L} = \max\{P_1, \dots, P_n\}$ be the longest path duration in a project network $PN(\mathcal{V}, \mathcal{P})$, where each path P_j is a sum of specific node durations from the vector V. Moreover, suppose that the set of nodes $\mathcal{V}$ satisfies Assumption 3.3. For any $p \geq 1$, the partial derivatives of the p-th moment of $\mathcal{L}$ with respect to the parameters of node i, μ_{V_i} and $\sigma^2_{V_i}$, are given by:*

$$\frac{\partial \mathbb{E}[\mathcal{L}^p]}{\partial \mu_{V_i}} = p\mathbb{E}\left[\mathcal{L}^{p-1} \mathbb{1}\{i \in \mathcal{V}(\mathcal{L})\}\right], \qquad \frac{\partial \mathbb{E}[\mathcal{L}^p]}{\partial \sigma^2_{V_i}} = \frac{p}{2\sigma^2_{V_i}} \mathbb{E}\left[\mathcal{L}^{p-1}(V_i - \mu_{V_i}) \mathbb{1}\{i \in \mathcal{V}(\mathcal{L})\}\right],$$

where $\{i \in \mathcal{V}(\mathcal{L})\}$ is the event that node i belongs to the longest path.

Proof. We seek the derivative of the expected value $\mathbb{E}[\mathcal{L}^p]$ with respect to the node parameters $\mu_{V_i}, \sigma^2_{V_i}$.

Let $\mathcal{T} \subset \Omega$ denote the set of outcomes where the maximum path length is not unique. Formally, $\mathcal{T}$ contains all ω such that there exist distinct paths $h_1 \neq h_2$ satisfying $P_{h_1} = P_{h_2} = \mathcal{L}$.

By Assumption 3.3, the node set $\mathcal{V}$ is partitioned into stochastic nodes $\mathcal{V}_S$ and constant nodes $\mathcal{V}_C$. To determine if two distinct paths can yield the exact same total duration, the tie condition $P_{h_1} = P_{h_2}$ is rearranged to isolate the random components on one side of the equation and the deterministic components on the other:

$$\sum_{x \in \mathcal{V}(P_{h_1}) \cap \mathcal{V}_S} V_x - \sum_{x \in \mathcal{V}(P_{h_2}) \cap \mathcal{V}_S} V_x = \sum_{x \in \mathcal{V}(P_{h_2}) \cap \mathcal{V}_C} V_x - \sum_{x \in \mathcal{V}(P_{h_1}) \cap \mathcal{V}_C} V_x.$$

The possibility of this equality holding true must be evaluated under two distinct structural scenarios.

In the first scenario, the two paths contain different sets of stochastic nodes. Because their random components do not perfectly overlap, the left-hand side of the equation forms a linear combination of continuous random variables, while the right-hand side is a strict, fixed numerical constant derived entirely from the deterministic nodes. The equation defines a $(k_S - 1)$-dimensional hyperplane in $\mathbb{R}^{k_S}$, which has k_S-dimensional Lebesgue measure of zero, since the joint distribution of the random nodes is absolutely continuous (Assumption 3.3(1)). Therefore, a tie between paths with differing random components will almost never occur.

In the second scenario, the two paths traverse the same set of stochastic nodes, thus making the left side reduce to zero. For the paths to be tied, the deterministic right-hand side must also evaluate to zero. However, Assumption 3.3(2) explicitly prohibits paths with identical stochastic footprints from sharing the same deterministic sum. Because their constant nodes must add up to different totals, the right-hand side of the equation is guaranteed to be a strictly non-zero value, making ties structurally impossible.

Since ties either have zero probability, or are logically impossible across all potential network structures, the probability of the tie set occurring is zero, yielding $\mathbb{P}(\mathcal{T}) = 0$, therefore for almost all realizations $\omega \notin \mathcal{T}$, the longest path is uniquely determined, and thus there almost always exists a unique longest path $h^\star \in \mathcal{P}$.

Because the path duration functions P_h are continuous with respect to the node durations (and consequently continuous with respect to the underlying distributional parameters μ_{V_i} and $\sigma^2_{V_i}$), the strict inequality $P_{h^\star} > P_h$ for all $h \neq h^\star$ is robust to sufficiently small perturbations.

Formally, for almost all $\omega \notin \mathcal{T}$, there exists an open neighborhood around the parameters such that the maximum function is locally equal to this single, uniquely identified path $h^\star$. Within this open neighborhood, the longest path function behaves exactly as the sum of the nodes in that specific path:

$$\mathcal{L}(\omega) = P_{h^\star}(\omega) = \sum_{k \in \mathcal{V}(P_{h^\star})} V_k(\omega).$$

Since the maximum function reduces to a linear sum in this neighborhood with probability 1, and the individual node durations V_k are differentiable with respect to their parameters, it immediately follows that $\mathcal{L}$ is differentiable almost everywhere.

To compute the derivative, we first apply the chain rule to the random variable $\mathcal{L}^p$ with respect to the node duration V_i:

$$\frac{\partial \mathcal{L}^p}{\partial V_i} = \frac{\partial (\mathcal{L}^p)}{\partial \mathcal{L}} \cdot \frac{\partial \mathcal{L}}{\partial V_i}.$$

The outer derivative is simply $\frac{\partial (\mathcal{L}^p)}{\partial \mathcal{L}} = p\mathcal{L}^{p-1}$. For the inner derivative, we express $\mathcal{L}$ explicitly as $\mathcal{L} = \max_j \{\sum_{k \in \mathcal{V}(P_j)} V_k\}$. The derivative of this maximum function with respect to a specific component V_i is non-zero if and only if V_i contributes to the sum that achieves the maximum. Specifically:

$$\frac{\partial \mathcal{L}}{\partial V_i} = \sum_{j=1}^{n} \mathbb{1}\{P_j = \mathcal{L}\} \cdot \mathbb{1}\{i \in \mathcal{V}(P_j)\}.$$

The term $\mathbb{1}\{P_j = \mathcal{L}\} \cdot \mathbb{1}\{i \in \mathcal{V}(P_j)\}$ equals 1 if path j is the longest path and node i is in path j, and 0 otherwise. Summing over all paths, this is exactly the indicator that node i is on the longest path, the derivative is almost surely given by:

$$\frac{\partial \mathcal{L}}{\partial V_i} = \mathbb{1}\{i \in \mathcal{V}(\mathcal{L})\}.$$

We now reparametrize the node distribution V_i as:

$$V_i = \mu_{V_i} + \sqrt{\sigma^2_{V_i}}\xi_i, \qquad \text{where } \xi_i \sim \mathcal{N}(0,1).$$

We have immediately that $\frac{\partial V_i}{\partial \mu_{V_i}} = 1$, while the derivative with respect to $\sigma^2_{V_i}$ is:

$$\frac{\partial V_i}{\partial \sigma^2_{V_i}} = \frac{1}{2\sqrt{\sigma^2_{V_i}}}\xi_i = \frac{1}{2\sigma_{V_i}}\left(\frac{V_i - \mu_{V_i}}{\sigma_{V_i}}\right) = \frac{V_i - \mu_{V_i}}{2\sigma^2_{V_i}}.$$

We now apply the chain rule to the random variable $\mathcal{L}^p$ with respect to both parameters. Using the derivatives established earlier:

$$\frac{\partial \mathcal{L}^p}{\partial \mu_{V_i}} = \frac{\partial \mathcal{L}^p}{\partial \mathcal{L}} \cdot \frac{\partial \mathcal{L}}{\partial V_i} \cdot \frac{\partial V_i}{\partial \mu_{V_i}} = p\mathcal{L}^{p-1} \cdot \mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}, \tag{3.1}$$

$$\frac{\partial \mathcal{L}^p}{\partial \sigma^2_{V_i}} = \frac{\partial \mathcal{L}^p}{\partial \mathcal{L}} \cdot \frac{\partial \mathcal{L}}{\partial V_i} \cdot \frac{\partial V_i}{\partial \sigma^2_{V_i}} = p\mathcal{L}^{p-1} \cdot \mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}} \cdot \frac{V_i - \mu_{V_i}}{2\sigma^2_{V_i}}. \tag{3.2}$$

To complete the proof, we must justify the interchange of the derivative and the expectation operator in both cases:

$$\frac{\partial}{\partial \mu_{V_i}}\mathbb{E}\left[\mathcal{L}^p\right] = \mathbb{E}\left[\frac{\partial \mathcal{L}^p}{\partial \mu_{V_i}}\right], \qquad \frac{\partial}{\partial \sigma^2_{V_i}}\mathbb{E}\left[\mathcal{L}^p\right] = \mathbb{E}\left[\frac{\partial \mathcal{L}^p}{\partial \sigma^2_{V_i}}\right].$$

The exchange is applicable if one shows that the absolute value of both derivatives is bounded by an integrable function [17].

$$\left|\frac{\partial \mathcal{L}^p}{\partial \mu_{V_i}}\right| = \left|p\mathcal{L}^{p-1}\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right| \leq \left|p\mathcal{L}^{p-1}\right|,$$
$$\left|\frac{\partial \mathcal{L}^p}{\partial \sigma^2_{V_i}}\right| = \left|\frac{p}{2\sigma^2_{V_i}}\mathcal{L}^{p-1}(V_i - \mu_{V_i})\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right| \leq \frac{p}{2\sigma^2_{V_i}}\left|\mathcal{L}^{p-1}(V_i - \mu_{V_i})\right|.$$

Since $\mathcal{L}$ and V_i are Gaussian and maxima of Gaussians, respectively, they possess finite moments of all orders, and $\mathcal{L}^p$ is always integrable for all p, completing the justification for the interchange of the first derivative.

For the second derivative, we use Hölder's inequality to show that the expectation of the product $\left|\mathcal{L}^{p-1}(V_i - \mu_{V_i})\right|$ is finite:

$$\mathbb{E}\left[|\mathcal{L}|^{p-1}|V_i - \mu_{V_i}|\right] \leq \left(\mathbb{E}\left[|\mathcal{L}|^{2(p-1)}\right]\right)^{1/2}\left(\mathbb{E}\left[(V_i - \mu_{V_i})^2\right]\right)^{1/2} < \infty.$$

Thus we conclude the proof by using the Leibniz Integral Rule to write:

$$\frac{\partial \mathbb{E}\left[\mathcal{L}^p\right]}{\partial \mu_{V_i}} = p\mathbb{E}\left[\mathcal{L}^{p-1}\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right], \qquad \frac{\partial \mathbb{E}\left[\mathcal{L}^p\right]}{\partial \sigma^2_{V_i}} = \frac{p}{2\sigma^2_{V_i}}\mathbb{E}\left[\mathcal{L}^{p-1}(V_i - \mu_{V_i})\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right].$$

□

Theorem 3.4 allows us to derive closed-form solutions for the sensitivity of all moments of $\mathcal{L}$ with respect to both node parameters. Special cases to keep in mind are the first moment and the variance of $\mathcal{L}$:

$$\frac{\partial \mathbb{E}\left[\mathcal{L}\right]}{\partial \mu_{V_i}} = \mathbb{E}\left[\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right], \qquad \frac{\partial \mathrm{Var}\left[\mathcal{L}\right]}{\partial \mu_{V_i}} = 2\mathrm{Cov}\left(\mathcal{L}, \mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right),$$
$$\frac{\partial \mathbb{E}\left[\mathcal{L}\right]}{\partial \sigma^2_{V_i}} = \frac{1}{2\sigma^2_{V_i}}\mathbb{E}\left[\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}(V_i - \mu_{V_i})\right], \qquad \frac{\partial \mathrm{Var}\left[\mathcal{L}\right]}{\partial \sigma^2_{V_i}} = \frac{1}{\sigma^2_{V_i}}\mathrm{Cov}\left(\mathcal{L}, \mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}(V_i - \mu_{V_i})\right).$$

3.1. Concentration Bounds

The exact computation of the expectations derived in the previous Section is generally intractable for large networks, as the longest path generally does not have a closed form. Therefore, we rely on Monte Carlo simulations to estimate the expectations and their derivatives. In this section, we derive concentration bounds for these estimators, which provide insights into the convergence rate of the estimators with respect to the number of simulations N.

3.1.1. Concentration Bound for Mean Sensitivity

Let us consider the estimator for the sensitivity of the mean path duration with respect to the node mean μ_{V_i}. From Theorem 3.4, the true value is $\delta_i = \mathbb{E}\left[\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}\right]$. We define the Monte Carlo estimator based on N independent simulations as:

$$\hat{\delta}_{i,N} = \frac{1}{N}\sum_{j=1}^{N} I_i^j, , \tag{3.3}$$

where I_i^j is the indicator variable taking the value 1 if node i is on the longest path in simulation j, and 0 otherwise. Note that I_i^j are independent and identically distributed (i.i.d.), for different values of j, Bernoulli random variables, hence bounded on the interval $[0, 1]$.

Theorem 3.5 (Concentration of Mean Sensitivity Estimator)**.** *For any $\varepsilon > 0$, the probability that the estimator $\hat{\delta}_{i,N}$ defined in Equation 3.3 deviates from the true value δ_i by more than ε is bounded by:*

$$\mathbb{P}\left(\left|\hat{\delta}_{i,N} - \delta_i\right| \geq \varepsilon\right) \leq 2\exp\left\{-2N\varepsilon^2\right\}.$$

Proof. We apply Hoeffding's Inequality, which states that for independent random variables $X_1, \ldots, X_N$ such that $a_j \leq X_j \leq b_j$, the sum $S_N = \sum X_j$ satisfies:

$$\mathbb{P}\left(|S_N - \mathbb{E}\left[S_N\right]| \geq t\right) \leq 2\exp\left\{-\frac{2t^2}{\sum_{j=1}^{N}(b_j - a_j)^2}\right\}.$$

In our case the variables X_j are indicators, bounded in $[0, 1]$, so $b_j - a_j = 1$. Moreover, note that the estimator can be written as $\hat{\delta}_{i,N} = \frac{1}{N}S_N$, therefore we can rearrange the inequality inside the probability, and by setting $t = N\varepsilon$ we obtain:

$$\begin{aligned}
\mathbb{P}\left(|\hat{\delta}_{i,N} - \delta_i| \geq \varepsilon\right) &= \mathbb{P}\left(|S_N - \mathbb{E}\left[S_N\right]| \geq N\varepsilon\right) \\
&\leq 2\exp\left\{-\frac{2(N\varepsilon)^2}{\sum_{j=1}^{N}(1)^2}\right\} \\
&= 2\exp\left\{-\frac{2N^2\varepsilon^2}{N}\right\} \\
&= 2\exp\left\{-2N\varepsilon^2\right\}.
\end{aligned}$$

□

3.1.2. Concentration Bound for Variance Sensitivity

Next, we consider the estimator for the sensitivity with respect to the node variance. From Theorem 3.4 (for $p = 1$), the true gradient is $\gamma_i = \frac{1}{2\sigma_{V_i}^2}\mathbb{E}\left[\mathbb{1}_{\{i \in \mathcal{V}(\mathcal{L})\}}(V_i - \mu_{V_i})\right]$. We define the auxiliary random variable for the j-th simulation:

$$Z_i^j = I_i^j(V_i^j - \mu_{V_i}),$$

where V_i^j is the realization of the duration of node i in simulation j, and I_i^j is, as before, the indicator of path i being the longest in simulation j. The estimator is given by:

$$\hat{\gamma}_{i,N} = \frac{1}{2\sigma_{V_i}^2} \left(\frac{1}{N} \sum_{j=1}^{N} Z_i^j \right). \tag{3.4}$$

Unlike the previous case, Z_i^j is not bounded, as it depends on the Gaussian variable V_i^j. Therefore, we must rely on the properties of Sub-Gaussian random variables [31].

Definition 3.6 (Sub-Gaussian Norm)**.** *A random variable X is Sub-Gaussian if its Sub-Gaussian norm $\|X\|_{\psi_2}$ is finite, where:*

$$\|X\|_{\psi_2} = \inf \left\{ t > 0 : \mathbb{E}\left[\exp\left\{ \frac{X^2}{t^2} \right\} \right] \leq 2 \right\}.$$

A classic example of a random variable with a Sub-Gaussian distribution is a Gaussian random variable. To see this we prove the following theorem:

Theorem 3.7 (Gaussian is Sub-Gaussian)**.** *Let $X \sim \mathcal{N}(0, \sigma^2)$. Then X is a sub-Gaussian random variable, and there exists an absolute constant $c > 0$ such that*

$$\|X\|_{\psi_2} \leq c\sigma_{V_i}.$$

Proof. By definition, the sub-Gaussian norm is given by

$$\|X\|_{\psi_2} = \inf \left\{ t > 0 : \mathbb{E}\left[\exp\left(\frac{X^2}{t^2} \right) \right] \leq 2 \right\}.$$

Since $X \sim \mathcal{N}(0, \sigma^2)$, we can write $X = \sigma Z$, where $Z \sim \mathcal{N}(0, 1)$. Evaluating the expectation using the standard normal density yields

$$\begin{aligned} \mathbb{E}\left[\exp\left\{ \frac{X^2}{t^2} \right\} \right] &= \mathbb{E}\left[\exp\left\{ \frac{\sigma^2 Z^2}{t^2} \right\} \right] \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left\{ \frac{\sigma^2 x^2}{t^2} - \frac{x^2}{2} \right\} dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left\{ -x^2 \left(\frac{1}{2} - \frac{\sigma^2}{t^2} \right) \right\} dx. \end{aligned}$$

This integral is finite if and only if $\frac{1}{2} - \frac{\sigma^2}{t^2} > 0$, which requires $t > \sqrt{2}\sigma_{V_i}$. Under this condition, evaluating the Gaussian integral gives

$$\mathbb{E}\left[\exp\left\{ \frac{X^2}{t^2} \right\} \right] = \frac{1}{\sqrt{1 - \frac{2\sigma^2}{t^2}}}.$$

To satisfy the norm condition, this expectation must be bounded by 2:

$$\frac{1}{\sqrt{1 - \frac{2\sigma^2}{t^2}}} \leq 2 \implies 1 - \frac{2\sigma^2}{t^2} \geq \frac{1}{4} \implies \frac{2\sigma^2}{t^2} \leq \frac{3}{4} \implies t \geq \sqrt{\frac{8}{3}}\sigma.$$

Setting $t = \sqrt{\frac{8}{3}}\sigma$ ensures the expectation is exactly 2. Therefore,

$$\|X\|_{\psi_2} \leq \sqrt{\frac{8}{3}}\sigma.$$

□

Theorem 3.8 (Concentration of Variance Sensitivity Estimator)**.** *There exists a constant $C > 0$ such that for any $\varepsilon > 0$:*

$$\mathbb{P}\left(|\hat{\gamma}_{i,N} - \gamma_i| \geq \varepsilon\right) \leq 2\exp\left\{-CN\sigma_{V_i}^2\varepsilon^2\right\},$$

where $\hat{\gamma}_{i,N}$ is the estimator for the gradient of the longest path $\mathcal{L}$ with respect to the node variance defined in Equation 3.4.

Proof. Let $W_j = V_i^j - \mu_{V_i}$. We know that $W_j \sim \mathcal{N}(0, \sigma_{V_i}^2)$. We have proved that Gaussian variables are Sub-Gaussian with norm $\|W_j\|_{\psi_2} \leq c\sigma_{V_i}$ for some constant c. Consider the variable $Z_i^j = I_i^j W_j$; since the indicator I_i^j is bounded by 1, we have $\left|Z_i^j\right| \leq |W_j|$. Moreover, since the Sub-Gaussian norm is monotonic with respect to absolute value domination, Z_i^j is also Sub-Gaussian with $\left\|Z_i^j\right\|_{\psi_2} \leq \|W_j\|_{\psi_2}$.

We construct the centered random variable $X_j = Z_i^j - \mathbb{E}\left[Z_i^j\right]$. The sum of Sub-Gaussian variables is Sub-Gaussian. Specifically, using the triangular inequality for the ψ_2 norm:

$$\|X_j\|_{\psi_2} \leq \left\|Z_i^j\right\|_{\psi_2} + \left\|\mathbb{E}\left[Z_i^j\right]\right\|_{\psi_2} \leq 2\left\|Z_i^j\right\|_{\psi_2} \leq 2c\sigma_{V_i}.$$

Let $K = 2c\sigma_{V_i}$ be the bound on the norm. We now apply the General Hoeffding Inequality for zero-mean Sub-Gaussian variables. We seek to bound the probability:

$$\begin{aligned}\mathbb{P}\left(|\hat{\gamma}_{i,N} - \gamma_i| \geq \varepsilon\right) &= \mathbb{P}\left(\left|\frac{1}{2\sigma_{V_i}^2 N}\sum_{j=1}^{N}\left(Z_i^j - \mathbb{E}\left[Z_i^j\right]\right)\right| \geq \varepsilon\right) \\ &= \mathbb{P}\left(\left|\sum_{j=1}^{N} X_j\right| \geq 2N\sigma_{V_i}^2\varepsilon\right).\end{aligned}$$

By the general Hoeffding inequality, $\mathbb{P}\left(|\sum X_j| \geq t\right) \leq 2\exp\left\{-\frac{C't^2}{\sum\|X_j\|_{\psi_2}^2}\right\}$. Substituting $t = 2N\sigma_{V_i}^2\varepsilon$:

$$\begin{aligned}\mathbb{P}\left(|\hat{\gamma}_{i,N} - \gamma_i| \geq \varepsilon\right) &\leq 2\exp\left\{-\frac{C'(2N\sigma_{V_i}^2\varepsilon)^2}{\sum_{j=1}^{N} K^2}\right\} \\ &= 2\exp\left\{-\frac{4C'N^2\sigma_{V_i}^4\varepsilon^2}{N(2c\sigma_{V_i})^2}\right\} \\ &= 2\exp\left\{-\frac{4C'N^2\sigma_{V_i}^4\varepsilon^2}{4c^2N\sigma_{V_i}^2}\right\} \\ &= 2\exp\left\{-\left(\frac{C'}{c^2}\right)N\sigma_{V_i}^2\varepsilon^2\right\}.\end{aligned}$$

Defining the constant $C = C'/c^2$, we obtain the result. □

3.1.3. Expectation of the Approximation Error

Finally, we can use these tail bounds to derive the expected convergence rate of the estimator error. Let $X = |\hat{\delta}_{i,N} - \delta_i|$ be the error magnitude. Using the tail bound formula $\mathbb{P}\left(X \geq \varepsilon\right) \leq 2e^{-\kappa N\varepsilon^2}$ (where κ is the relevant constant from the previous proofs), we can compute the expected error. Since X is non-negative by definition, we can use the following formula to compute the expectation from the cumulative density function [5]:

$$\mathbb{E}\left[|\hat{\delta}_{i,N} - \delta_i|\right] = \int_0^\infty \mathbb{P}\left(X \geq \varepsilon\right) d\varepsilon \leq \int_0^\infty 2\exp\{-\kappa N\varepsilon^2\} d\varepsilon.$$

Solving the Gaussian integral $\int_0^\infty e^{-ax^2} dx = \frac{1}{2}\sqrt{\frac{\pi}{a}}$:

$$\mathbb{E}\left[\left|\hat{\delta}_{i,N} - \delta_i\right|\right] \leq 2 \cdot \frac{1}{2}\sqrt{\frac{\pi}{\kappa N}} = \sqrt{\frac{\pi}{\kappa}}\frac{1}{\sqrt{N}} = O\left(\frac{1}{\sqrt{N}}\right).$$

3.2. Simulation Under the Gaussian Assumption

Building upon the example introduced in Section 2.3, we assign specific probability distributions and parameters to each node according to its designated task.

As previously established in Table 2.1, the manufacturing setup comprises several distinct peripherals, each capable of performing specific operations.

To accurately simulate the timing of these operations, we model the duration of each action as independent random variables, whose distribution is assigned using Table 3.1.

Action	**Distribution**
Start/End	Instantaneous
On/Off	Constant(1)
Clamp/Unclamp	$\mathcal{N}(1, 0.5)$
Move	$\mathcal{N}(2, 1)$
Condition	$\mathcal{N}(8, 2)$

Table 3.1: Probability distributions and parameters assigned to each operational action.

3.2.1. Theoretical Assumptions and Validity

In assigning these distributions, we make several assumptions: for example, we assume that all "Move" actions take the same amount of time. This can be justified by the fact that, in this example, the distances of all move actions are roughly equal, so it is reasonable to give them the same distribution. These assumptions are a simplification of the real scenario, but for this example, the goal is to highlight the mathematical meaning of the formulas, rather than to model the machine in full detail.

We can therefore apply Theorem 3.4, as the network topology (Figure 2.3) and parameterization satisfy the non-degeneracy conditions outlined in Assumption 3.3. First, the independence of the stochastic node durations ensures that their joint distribution is absolutely continuous with respect to the Lebesgue measure. Second, the structure of the graph does not allow two distinct paths traversing the exact same set of stochastic nodes to yield identical deterministic sums, eliminating the possibility of structural ties, guaranteeing that the longest path is almost surely unique.

3.2.2. Monte Carlo Simulation Setup

To evaluate the sensitivity of the system at these baseline settings, each node in the graph is initialized with the distribution parameters specified in Table 3.1. We then simulate the execution of the network over a sample size N and compute the previously discussed Monte Carlo estimators to approximate the true derivative of the expected longest path $\mathbb{E}[\mathcal{L}]$ with respect to each node's parameters.

This simulation process is highly computationally efficient, as each iteration simply requires independently sampling 50 random variables from their respective distributions, identifying the nodes comprising the longest path, and updating the estimator. To guarantee a robust approximation, we execute the simulation with $N = 10^6$ samples, a procedure that requires approximately 15 seconds of computation time. The resulting sensitivity estimates are visualized in Figure 3.1.

3.2.3. Analysis of Expectation and Critical Paths

In Figures 3.1 and 3.2, we observe the derivative of $\mathbb{E}[\mathcal{L}]$ and $\mathrm{Var}[\mathcal{L}]$ with respect to the location parameter of each node (i.e., the mean μ for Gaussian distributions, or the scalar value for deterministic nodes). For nodes 1 and 50, which represent the start and end of the project, the duration is

instantaneous and strictly parameter-less; thus, their derivative is set to zero.

In Figure 3.1, all values are strictly bounded between 0 and 1. As established previously, the derivative of the expectation with respect to a node's mean is mathematically equivalent to the probability that the node lies on the longest path. Consequently, the node values provide a direct visual heat map of path criticality. For instance, nodes depicted in red exhibit a derivative of exactly 1; across 10^6 Monte Carlo samples, they are strictly always on the longest path. Because these lower-section nodes dominate the network flow, the alternative parallel branches (such as nodes 3, 4, and 5) have a value of exactly 0.

Following node 11, a bifurcation occurs, as nodes 13, 15, and 17 have a derivative of ≈ 0.75, while nodes 12, 14, 16, 18 sit at ≈ 0.25. This indicates that in 75% of the samples the longest path routes through the upper branch of the split, while through the bottom branch the remaining 25% of the samples. A similarly balanced distribution of criticality is observed in the upper-right parallel branches, where the flow splits evenly between sub-paths (e.g., between 30 and 31 after node 28, and between 35 and 36 after node 33).

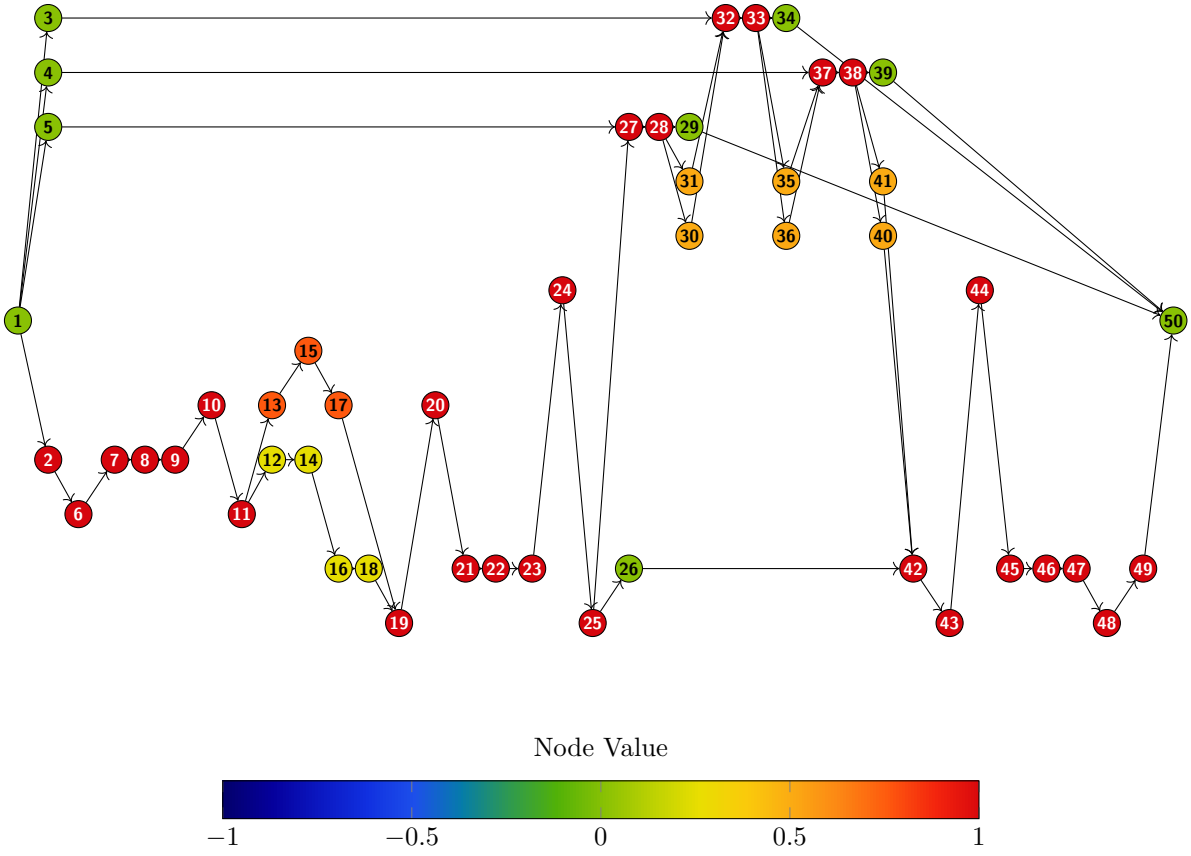


Figure 3.1: Monte Carlo simulation on the Bowling Ball example graph with 10^6 samples. The color of each node represents the value of the derivative of the expected longest path $\mathbb{E}[\mathcal{L}]$ with respect to the mean parameter of each node (either μ or the value of the constant, depending on the distribution of the node).

3.2.4. Analysis of Variance Sensitivity

Conversely, in Figure 3.2, the plotted values for the derivative of $\mathrm{Var}[\mathcal{L}]$ are no longer bounded within the unit interval and may assume negative values. A negative derivative indicates that increasing the mean duration of a specific node increases its likelihood of dictating the longest path, hence reducing the uncertainty regarding which path is the longest. This reduction in uncertainty naturally implies a decrease in the overall variance of the longest path length.

Nevertheless, the derivative for the majority of the nodes evaluates to zero. If the probability of a node

belonging to the longest path is strictly 0 or 1, an infinitesimal perturbation of its location parameter has virtually no impact on its probability of being in the longest path. This behavior diverges significantly, however, when probabilities lie strictly within the open interval $(0, 1)$. For instance, at the previously discussed split following node 11, we observe a positive derivative (≈ 0.33) for the nodes comprising the upper branch and a negative derivative (≈ -0.33) for those in the lower branch.

Given that node 15 exhibits the highest variance in this sub-network, increasing the location parameter of nodes in the upper branch of the split directly increases the variance of the total project duration. Conversely, increasing the mean of the lower-branch nodes diminishes the probability that node 15 is critical, thereby decreasing the overall variance.

Intuitively, these derivatives sum to exactly zero because increasing the mean duration of the upper branch achieves the exact same relative shift in path probabilities as decreasing the lower branch by the same amount. Therefore, an incremental change favoring the higher-variance path has the exact equal and opposite effect on the system's total variance as an equivalent change favoring the lower-variance path.

This dynamic is entirely absent in symmetric splits, such as those between nodes 30 and 31, 35 and 36, and 40 and 41. Because the parallel branches in these instances share identical distribution parameters, shifting the criticality from one sub-path to the other creates no net change in the variance of the total project length.

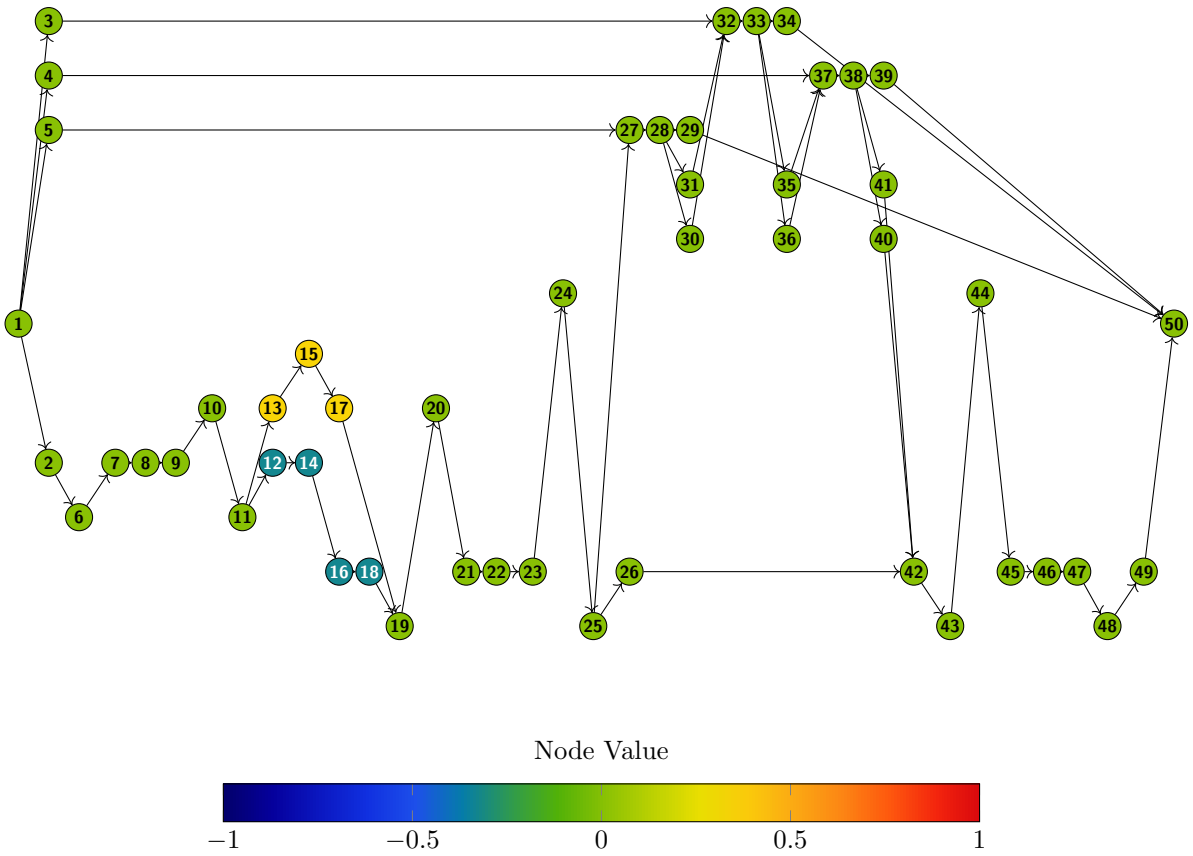


Figure 3.2: Monte Carlo simulation on the Bowling Ball example graph with 10^6 samples. The color of each node represents the value of the derivative of the variance of the longest path $\mathrm{Var}[\mathcal{L}]$ with respect to the mean parameter of each node (either μ or the value of the constant, depending on the distribution of the node).

3.2.5. Practical Implications for Manufacturing

These results can be directly used to make informed and targeted improvements to the manufacturing process. In particular, to reduce the expected project duration $\mathbb{E}[\mathcal{L}]$, the most effective parameters to act upon are those associated with operations that lie on the longest path with high probability. These actions correspond to bottleneck tasks that consistently limit the workflow. Reducing the mean duration of these actions yields the largest marginal decrease in total completion time. Conversely, modifying operations that rarely belong to the longest path has a negligible effect, as they do not significantly influence the total project length.

From a variability perspective, the sensitivity of $\mathrm{Var}[\mathcal{L}]$ highlights which parts of the system contribute most to uncertainty in completion time. Nodes associated with branches that are intermittently critical play a central role, as small parameter changes can shift the probability mass between different paths. In such cases, reducing variability in execution times can stabilize the system by enforcing a more consistent critical path.

4

Generalized Sensitivity Theory

In Chapter 3, we derived analytical expressions for the sensitivity of the moments of $\mathcal{L}$ under the assumption of Gaussian node durations. A crucial component of those proofs was the ability to decouple the distributional parameters from the source of randomness; specifically, we exploited the fact that any Gaussian random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ can be represented as an affine transformation of a standard normal variable $Z \sim \mathcal{N}(0, 1)$:

$$X = \mu + \sigma Z \quad \Rightarrow \quad X \sim \mathcal{N}(\mu, \sigma^2).$$

This representation allowed us to differentiate the random variable with respect to its parameters (μ, σ^2) while keeping the underlying probability space fixed. This property is not unique to the Gaussian distribution; later, we will see which distributions also allow some type of reparameterization.

Another important generalization is allowing different nodes to share parameters. Previously, each node V_i required its own distinct parameters μ_{V_i} and $\sigma^2_{V_i}$. However, in many real-world applications, distinct tasks are often repeated instances of the same underlying action, motivating the need for shared parameters across these equivalent nodes. For example, a robotic arm may perform the identical "move from A to B" motion multiple times across a single process. While each movement is represented by a separate node in the model, the physical dynamics remain the same. By allowing these distinct nodes to share parameters, we effectively separate the parameter space from the node space, creating a more realistic model.

In this chapter, we extend the sensitivity analysis framework beyond the Gaussian assumption to include this broader class of distributions and the use of shared parameters across multiple nodes. We establish the Generalized Sensitivity Theorem, which provides a rigorous foundation for computing the exact gradients of the expected value of an arbitrary target function of the longest path with respect to the model parameters.

4.1. Reparametrization

As we mentioned before, many distributions allow for a reparametrization that decouples the parameters from the source of randomness, and the most general transformation can be obtained by using the inverse CDF [5].

Lemma 4.1 (Inverse CDF reparameterization [5])**.** *Let F_θ be a continuous and strictly increasing CDF parameterized by some parameter(s) θ. This guarantees a well-defined inverse function $F_\theta^{-1} : (0, 1) \to supp(F_\theta)$. For a base random variable $U \sim Uniform(0, 1)$, defining $X = F_\theta^{-1}(U)$ yields a random variable X with CDF F_θ.*

Expressing X explicitly as $X(\theta) = F_\theta^{-1}(U)$ isolates the parameter(s) θ from the source of randomness

U. Assuming $F_\theta^{-1}(\cdot)$ is differentiable with respect to θ almost everywhere, the gradient is computed as:

$$\frac{\partial X(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} F_\theta^{-1}(U).$$

Although this result is quite general, it is often not very practical. Many distributions have complicated CDFs that do not have closed-form expressions, making inversion and differentiation difficult and computationally expensive. This has been widely studied, and one possible approach is to use implicit reparameterization gradients [15].

4.2. Generalized Sensitivity Theorem

We will now lay the groundwork for the statement and proof of the theorem.

Definition 4.2 (Parameter Space and Parameter Vector)**.** *Define the parameter space $\Theta \subset \mathbb{R}^d$ to be an open subset of $\mathbb{R}^d$. Any $\theta \in \Theta$ is a parameter vector.*

We require Θ to be open to allow differentiation with respect to any $\theta \in \Theta$. Note that the dimension d represents the total number of independent parameters required to characterize the distributions of all node durations in the network.

Next, consider a project network $PN(\mathcal{V}, \mathcal{P})$ with k nodes. The node durations are defined as a vector $V(\theta, \omega) = [V_1(\theta, \omega), \ldots, V_k(\theta, \omega)]$. Our goal is to decouple the parameter vector $\theta \in \Theta$, and the source of randomness by defining a family of transformation functions $\phi_i(\theta, Z_i)$.

Assumption 4.3 (Reparametrization of node distributions)**.** *Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, assume that there exists a family of transformation functions ϕ_i such that all node durations V_i can be mapped from a base random vector Z:*

$$V_i(\theta, \omega) = \phi_i(\theta, Z(\omega)),$$

where $\omega \in \Omega$, and $Z = [Z_1, \ldots, Z_s]^\top$ is a random vector whose joint distribution is independent of the parameter space Θ. Moreover, we assume that the mapping $\theta \mapsto \phi_i(\theta, \cdot)$ is continuously differentiable almost everywhere for all $i = 1, \ldots, k$.

Here s represents the dimension of the base noise vector. If we want a non-degenerate node distribution, we assume that $s \geq k$, since if we had $s < k$, the resulting mapped vector $V = \Phi(\theta, Z)$ would be restricted to a s-dimensional manifold within $\mathbb{R}^k$, with k-dimensional Lebesgue measure of zero. However, s could also be bigger than k, which represents the case where there are multiple sources of randomness that affect node distributions.

For each variable V_i, this function could be given by the inverse CDF of the distribution of V_i, if it satisfies the correct assumptions of differentiability, or by any other function ϕ_i that satisfies Assumption 4.3. For instance, in Chapter 3, especially in the proof of Theorem 3.4, the affine transformation $X = \phi((\mu, \sigma), Z) = \mu + \sigma Z$, where $Z \sim \mathcal{N}(0, 1)$, is used to decouple the parameters (μ, σ^2) from the source of randomness Z.

A simple class of distributions for which Assumption 4.3 fails is given by discrete distributions with parameter-dependent probabilities, such as the Bernoulli distribution. Let

$$V_\theta \sim \text{Bernoulli}(\theta), \qquad \theta \in (0, 1).$$

Although this random variable can be generated by inverse transform sampling as

$$V_\theta = \mathbf{1}_{\{U \leq \theta\}}, \qquad U \sim \text{Uniform}(0, 1),$$

the map $\theta \mapsto \mathbf{1}_{\{U \leq \theta\}}$ is a step function for each fixed realization of U. Hence it is not continuously differentiable almost everywhere in the pathwise sense required by Assumption 4.3. Its derivative is

zero except at the discontinuity, so the resulting pathwise derivative cannot capture the sensitivity of the Bernoulli distribution with respect to θ.

Our goal is to make a general theorem, and because of this, we will introduce the target function $g : \mathbb{R} \to \mathbb{R}$, allowing for the calculation of sensitivities for moments (e.g., $g(x) = x^p$) or other risk metrics.

In the proof of the theorem, we will be exchanging the derivative and expectation operators. To do so, we require the following conditions on g:

Assumption 4.4 (Growth condition on g)**.** *The target function g must be continuously differentiable, and its derivative g' must satisfy a polynomial growth bound $|g'(x)| \leq C(1 + |x|^m)$ for some $m \in \mathbb{R}$.*

Examples of possible risk metrics functions for g which satisfy Assumption 4.4 include:

- **Soft-Hinge:** widely recognized in optimization literature as a smooth, infinitely differentiable approximation for the standard non-smooth hinge loss. It is used to penalize schedules that exceed a critical project deadline T, a smoothed hinge loss can be utilized:

  $$g(x) = \ln(1 + \exp(x - T))$$

  This metric evaluates the expected lateness of the schedule. Its derivative is given by

  $$g'(x) = \frac{\exp(x - T)}{1 + \exp(x - T)}.$$

  Since this derivative is strictly bounded between 0 and 1 for all $x \in \mathbb{R}$, it satisfies the polynomial growth condition with $m = 0$.

- **Quadratic Risk Measure:** To simultaneously optimize for a short expected completion time and low variability, a quadratic risk function can be applied [26]:

  $$g(x) = x + \frac{\gamma}{2}(x - \bar{x})^2$$

  where $\gamma > 0$ is a risk-aversion parameter and $\bar{x}$ is the expected value of the project length. The derivative is

  $$g'(x) = 1 + \gamma(x - \bar{x}).$$

  Because the derivative is a linear function of x, it satisfies the polynomial growth bound with $m = 1$.

- **Smoothed Probability of Exceedance:** If the objective is to evaluate the sensitivity of the probability that the completion time exceeds a critical threshold T, a sigmoid barrier function can approximate the step-like behavior [30]:

  $$g(x) = \frac{1}{1 + \exp(-k(x - T))},$$

  where $k > 0$ controls the steepness of the approximation. The derivative is

  $$g'(x) = k \cdot g(x)(1 - g(x)).$$

  Since the sigmoid function $g(x)$ is strictly bounded in the interval $(0, 1)$, its derivative is bounded by $k/4$, thereby satisfying the growth condition with $m = 0$.

The following last assumption is needed to exchange the derivative and expectation operators.

Assumption 4.5 (Finite moments)**.** *The node durations and their sensitivities must have finite moments of order $m+1$ locally, where $g(x)$ is a target function satisfying Assumption 4.4, and m is the degree of the polynomial function that bounds the derivative $|g'(x)|$. Formally, there exists an open neighborhood $O \subset \Theta$ such that:*

$$\mathbb{E}\left[\sup_{\theta' \in O} |V_i(\theta')|^{m+1}\right] < \infty \quad \textit{and} \quad \mathbb{E}\left[\sup_{\theta' \in O} \left|\frac{\partial V_i}{\partial \theta_j}(\theta')\right|^{m+1}\right] < \infty,$$

for all $i = 1, \ldots, k$ and all $j = 1, \ldots, d$.

Finally, we can state Theorem 4.6.

Theorem 4.6 (Generalized Sensitivity)**.** *Consider a project network $PN(\mathcal{V}, \mathcal{P})$, where $\mathcal{V}$ satisfies Assumption 3.3. Let $\Theta \subset \mathbb{R}^d$ be a parameter space and suppose that there exists a family of transformation functions ϕ_i that satisfy Assumption 4.3. Let $g : \mathbb{R} \to \mathbb{R}$ be a target function that satisfies Assumption 4.4. Finally, suppose that Assumption 4.5 is met. Then the derivative of the expected target function with respect to parameter θ_j is given by:*

$$\frac{\partial}{\partial \theta_j}\mathbb{E}\left[g(\mathcal{L})\right] = \mathbb{E}\left[g'(\mathcal{L}) \sum_{i \in \mathcal{V}(\mathcal{L})} \frac{\partial V_i}{\partial \theta_j}\right]. \tag{4.1}$$

Proof. We start by assuming that $\mathcal{L}$ is almost-surely differentiable, since this is already discussed in the proof of Theorem 3.4.

The longest path length is given by the sum of all nodes that belong to the longest path:

$$\mathcal{L}(\theta, \omega) = \sum_{i \in \mathcal{V}(\mathcal{L})} V_i(\theta, \omega) = \sum_{i=1}^{k} \mathbb{1}\{i \in \mathcal{V}(\mathcal{L})\} V_i(\theta, \omega)$$

Thus, the partial derivative with respect to a parameter θ_j is given by:

$$\frac{\partial \mathcal{L}}{\partial \theta_j}(\theta, \omega) = \sum_{i=1}^{k} \mathbb{1}\{i \in \mathcal{V}(\mathcal{L})\} \frac{\partial V_i}{\partial \theta_j}(\theta, \omega).$$

We aim to compute the partial derivative $\frac{\partial}{\partial \theta_j} g(\mathcal{L})$. Since g is continuously differentiable (Assumption 4.4) and $\mathcal{L}$ is almost surely differentiable, the Chain Rule applies for almost all realizations $\omega \in \Omega$:

$$\frac{\partial}{\partial \theta_j} g(\mathcal{L}(\theta, \omega)) = g'(\mathcal{L}(\theta, \omega)) \sum_{i=1}^{k} \mathbb{1}\{i \in \mathcal{V}(\mathcal{L})\} \frac{\partial V_i}{\partial \theta_j}(\theta, \omega). \tag{4.2}$$

We differentiate under the integral sign using the Leibniz Integral Rule. This requires showing that the sample path derivative is locally dominated by an integrable random variable $K(\omega)$ (i.e., $\mathbb{E}[K] < \infty$).

We define the following bounding variables on the neighborhood U (existence guaranteed by Assumption 4.5):

$$M_i(\omega) = \sup_{\theta' \in O} |V_i(\theta', \omega)|, \quad D_{ij}(\omega) = \sup_{\theta' \in O} \left|\frac{\partial V_i}{\partial \theta_j}(\theta', \omega)\right|.$$

Using Equation (4.2), we bound the magnitude uniformly for all $\theta' \in O$:

$$\sup_{\theta' \in O} \left|\frac{\partial}{\partial \theta_j} g(\mathcal{L}(\theta', \omega))\right| \leq C \left(1 + \left(\sum_{i=1}^{k} M_i(\omega)\right)^m\right) \cdot \sum_{i=1}^{k} D_{ij}(\omega) =: K(\omega).$$

To prove $K(\omega)$ is integrable, we apply Hölder's Inequality with conjugate exponents $r = m + 1$ and $q = (m + 1)/m$.

Crucially, Hölder's Inequality applies to the expectation of products of random variables defined on the same probability space, regardless of their underlying dependence structure. By Assumption 4.5, the suprema M_i and D_ij have finite moments of order $m + 1$. Therefore, the sum $\sum_i D_{ij} \in L^{m+1}(\Omega)$. Similarly, since $M_i \in L^{m+1}(\Omega)$, the algebraic expansion of $(\sum_i M_i)^m$ belongs to $L^{(m+1)/m}(\Omega) = L^q(\Omega)$.

As $K(\omega)$ is a linear combination of products of L^q and L^r functions, $K \in L^1(\Omega)$ by Hölder's Inequality. Since the domination condition is met, the Leibniz Integral Rule justifies the interchange:

$$\frac{\partial}{\partial \theta_j}\mathbb{E}\left[g(\mathcal{L})\right] = \mathbb{E}\left[\frac{\partial}{\partial \theta_j} g(\mathcal{L})\right] = \mathbb{E}\left[g'(\mathcal{L}) \sum_{i \in \mathcal{V}(\mathcal{L})} \frac{\partial V_i}{\partial \theta_j}\right].$$

□

4.3. Cascading Dependencies

The formulation of Theorem 4.6 is structurally robust to autoregressive or cascading dependencies within the network. Because Assumption 4.3 defines the transformation ϕ_i over a shared multivariate noise vector Z, it allows individual node durations to depend explicitly on the realizations of preceding nodes.

Consider a scenario where a sequence of tasks physically influences one another, such as a machining process V_1 that alters the required duration of a subsequent recalibration task V_2.

Let Z_1 and Z_2 be independent components of the base random vector Z. The duration of the first node depends strictly on its own parameter θ_1:

$$V_1(\theta_1, \omega) = \phi_1(\theta_1, Z_1(\omega)).$$

However, the second node is topologically dependent on the realization of the first. Its duration is governed by its own parameter θ_2, its noise Z_2, and the realized duration of V_1:

$$V_2(\theta_1, \theta_2, \omega) = \phi_2(\theta_2, V_1(\theta_1, \omega), Z_2(\omega)).$$

Because Theorem 4.6 relies on the total sample path derivative, we evaluate the sensitivity of V_2 with respect to the preceding parameter θ_1 by applying the chain rule:

$$\frac{\partial V_2}{\partial \theta_1} = \frac{\partial \phi_2}{\partial V_1}\frac{\partial V_1}{\partial \theta_1}.$$

If both nodes V_1 and V_2 lie on the critical path, the summation inside the expectation of Theorem 4.6 can be rewritten as:

$$\sum_{i \in \mathcal{V}(\mathcal{L})} \frac{\partial V_i}{\partial \theta_1} = \frac{\partial V_1}{\partial \theta_1} + \frac{\partial V_2}{\partial \theta_1} + \cdots = \frac{\partial V_1}{\partial \theta_1}\left(1 + \frac{\partial \phi_2}{\partial V_1}\right) + \dots$$

To make this abstract formulation concrete, consider a nonlinear coupling model. Let V_1 represent a machining task, modeled as a random variable with a mean duration parameter θ_1, driven by standard environmental noise $Z_1 \sim \mathcal{N}(0, 1)$:

$$V_1(\theta_1, \omega) = \theta_1 + Z_1(\omega).$$

Let V_2 represent a subsequent cooling task. The cooling time requires a base expected duration θ_2, subject to its own independent ambient noise $Z_2 \sim \mathcal{N}(0, 1)$. Crucially, the total cooling time also increases based on the heat generated by the machining task V_1, following a smooth and differentiable function $f(x)$:

$$V_2(\theta_1, \theta_2, \omega) = \underbrace{\theta_2 + Z_2(\omega)}_{\text{base}} + \underbrace{f(V_1(\theta_1, \omega))}_{\text{coupled delay}}.$$

In this framework, the transformation function for the second node is defined explicitly as $\phi_2(V_1, \theta_2, Z_2) = \theta_2 + Z_2 + f(V_1)$. We can easily compute the necessary partial derivatives to evaluate the sensitivity of V_2 with respect to the initial mean duration parameter θ_1:

$$\frac{\partial \phi_2}{\partial V_1} = f'(V_1(\theta_1, \omega)) \quad \text{and} \quad \frac{\partial V_1}{\partial \theta_1} = 1.$$

Applying the chain rule, the sample path derivative of the cooling task with respect to the expected machining time is:

$$\frac{\partial V_2}{\partial \theta_1} = \frac{\partial \phi_2}{\partial V_1} \frac{\partial V_1}{\partial \theta_1} = f'(V_1(\theta_1, \omega)) \cdot 1 = f'(V_1(\theta_1, \omega)).$$

If we substitute this into the summation inside the Generalized Sensitivity Theorem, assuming both nodes lie on the critical path, we obtain:

$$\sum_{i \in \mathcal{V}(\mathcal{L})} \frac{\partial V_i}{\partial \theta_1} = 1 + f'(V_1(\theta_1, \omega)).$$

4.4. Numerical estimation and concentration bounds

Theorem 4.6 is a powerful result that generalizes Theorem 3.4 under many aspects. However, by the theorem, we are left with computing the expected value of a very complicated random variable whose distribution is difficult to express. As a result, we employ a standard Monte Carlo approach to approximate this expectation and obtain a numerical result that can be used in practice.

In Section 3.1 we established bounds on the tail probability $\mathbb{P}\left(|\hat{\gamma}_{i,N} - \gamma_i| \geq \varepsilon\right)$ for the Gaussian case. In this section, we formalize the numerical estimators for these derivatives and analyze their convergence properties as the sample size N increases, demonstrating that the estimators developed previously extend naturally to this general case.

Lemma 4.7 (Expected absolute error for sample mean estimator)**.** *Consider a random variable $X \in L^2$ with mean $\mu = \mathbb{E}[X]$ and variance $\sigma^2 = Var(X)$. Given N independent samples $X_1, \ldots, X_N$ of X, define the sample mean estimator $\hat{\mu}_N$:*

$$\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^{N} X_i,$$

then, we have:

$$\mathbb{E}\left[|\hat{\mu}_N - \mu|\right] \leq \frac{\sigma}{\sqrt{N}}.$$

Proof. By Jensen's inequality applied to the concave square root function, for any random variable Y, $\mathbb{E}[|Y|] \leq \sqrt{\mathbb{E}[Y^2]}$.

Let $Y = \hat{\mu}_N - \mu$. Then:

$$\mathbb{E}\left[|\hat{\mu}_N - \mu|\right] \leq \sqrt{\mathbb{E}\left[\left(\hat{\mu}_N - \mu\right)^2\right]}.$$

Since the estimator $\hat{\mu}_N$ is unbiased ($\mathbb{E}\left[\hat{\mu}_N\right] = \mathbb{E}\left[\mu\right]$), we have that $\mathbb{E}\left[Y\right] = 0$, thus the term $\mathbb{E}\left[\left(\hat{\mu}_N - \mu\right)^2\right]$ coincides with the variance of the sample mean $\hat{\mu}_N$. Since $X_1, \ldots, X_N$ are independent and identically distributed with variance σ^2:

$$\mathrm{Var}(\hat{\mu}_N) = \mathrm{Var}\left(\frac{1}{N} \sum_{i=1}^{N} X_i\right) = \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{Var}(X_i) = \frac{N\sigma^2}{N^2} = \frac{\sigma^2}{N}.$$

Substituting the variance into the inequality yields:

$$\mathbb{E}\left[|\hat{\mu}_N - \mu|\right] \leq \sqrt{\frac{\sigma^2}{N}} = \frac{\sigma}{\sqrt{N}}.$$

□

Lemma 4.7 establishes that the sensitivity measure derived in Theorem 4.6 can be evaluated using a standard Monte Carlo estimator. Define the sample path derivative:

$$X = g'(\mathcal{L}) \sum_{i \in \mathcal{V}(\mathcal{L})} \frac{\partial V_i}{\partial \theta_j}.$$

Under Theorem 4.6, Assumptions 4.4 and 4.5 guarantee that $X \in L^2$. Because X has finite variance ($\sigma_X^2 < \infty$), Lemma 4.7 applies. For N independent realizations $X^{(1)}, \ldots, X^{(N)}$, the expected absolute error of the estimator $\hat{\mu}_N = \frac{1}{N} \sum_{n=1}^{N} X^{(n)}$ is strictly bounded:

$$\mathbb{E}\left[\left|\hat{\mu}_N - \frac{\partial}{\partial \theta_j} \mathbb{E}\left[g(\mathcal{L})\right]\right|\right] \leq \frac{\sigma_X}{\sqrt{N}}.$$

While Lemma 4.7 guarantees the convergence of the Monte Carlo estimator, a strictly stronger optimality condition holds. As a matter of fact, it can be shown that the sample mean estimator is a consistent estimator, hence increasing the sample size increases the probability of the estimator being close to the true value of the expectation.

Under the conditions of Lemma 4.7, the sample mean estimator $\hat{\mu}_N$ is strongly consistent, converging almost surely to the true expectation $\mu = \frac{\partial}{\partial \theta_j} \mathbb{E}\left[g(\mathcal{L})\right]$ as $N \to \infty$. This is because, by definition, strong consistency demands convergence with probability one:

$$\mathbb{P}\left(\lim_{N \to \infty} \hat{\mu}_N = \mu\right) = 1.$$

Under Lemma 4.7, the sample path derivative X is in L^2, which then guarantees $X \in L^1$ ($\mathbb{E}\left[|X|\right] < \infty$). Because the sequence $X^{(1)}, \ldots, X^{(N)}$ is independent and identically distributed, Kolmogorov's Strong Law of Large Numbers applies directly. Therefore, the sample mean $\hat{\mu}_N$ converges almost surely to μ:

$$\hat{\mu}_N \xrightarrow{a.s.} \mu.$$

Moreover, we can get a naive bound on the tail probability, as we did in Section 3.1, by applying Markov's inequality:

$$\mathbb{P}\left(|\hat{\mu}_N - \mu| \geq \varepsilon\right) \leq \frac{\mathbb{E}\left[|\hat{\mu}_N - \mu|\right]}{\varepsilon} = \frac{\sigma_X}{\varepsilon \sqrt{N}}.$$

4.5. Simulation in the Generalized Setting

As we did in Section 3.2, we use the bowling ball example introduced in Section 2.3 to run simulations with the new Generalized Sensitivity theory. We will assign distributions to each action, keeping independence across each node, according to the following table.

Action	**Distribution**
Start/End	Instantaneous
On/Off	PERT$(0.5, 1, 1.5)$
Clamp/Unclamp	Beta$(3, 2)$
Move	$\mathcal{N}(2, 1)$
Condition	Gamma$(2, 4)$

Table 4.1: Probability distributions and parameters assigned to each operational action.

The PERT distribution [9] is a modified version of the four-parameter beta distribution. It includes the additional assumption that the expected value is given by $\mu = \frac{a+4b+c}{6}$. While the standard beta

distribution, defined by the shape parameters α and β, is supported on the interval $[0, 1]$, it can be extended to have support on a general interval $[a, b]$ with two extra parameters to control its location and scale [18]. Because of this additional assumption, the PERT distribution has only three free parameters: the minimum value, the maximum value, and the mode. It is widely used in risk analysis to model uncertainty when values are based on subjective estimates, since these three parameters are easy to understand and specify.

In contrast to the node-specific parameterization employed in Section 3.2, the Generalized Sensitivity Theorem enables the incorporation of shared parameters across multiple nodes, allowing for a more realistic simulation. Recall that each node within the project network represents a specific action executed by a specific peripheral. To accurately model this architecture, the family of the probability distribution is determined by the action type, whereas the specific distributional parameters are bound directly to the peripheral performing it.

This structural choice ensures that all identical actions executed by a given peripheral are governed by a shared parameter space rather than isolated, node-specific values. Consequently, distinct peripherals performing the same class of action maintain independent parameterizations. For instance, a "Clamp" action performed by Robot 1 and an identical "Clamp" action performed by Robot 2 may both follow a Beta distribution, yet they are governed by independent parameter sets. From an industrial perspective, this accurately reflects hardware tuning: adjusting the mechanical settings of Robot 1 alters the execution times of all the operations performed by Robot 1, without influencing the performance of Robot 2.

Because these parameters are now distributed globally across subsets of nodes rather than localized to individual tasks, the node-level visualization previously utilized in Figure 3.1 is no longer structurally applicable, and instead the sensitivity must be evaluated directly with respect to the shared parameters.

Following the execution of the Monte Carlo simulation with $N = 10^6$ samples, the gradients of the expected project duration $\mathbb{E}[\mathcal{L}]$ with respect to the shared peripheral parameters were computed. The results are summarized in Table 4.2.

In Table 4.2, the gradients of the expected value with respect to the mean movement time (μ) for Robot 2 and Robot 1 are large (8.985 and 4.985). Since the derivative of a Normal variable with respect to its mean is exactly 1, these values can be interpreted as the expected number of times these movements appear on the longest path. Even more clearly, the gradients for the three Drill peripherals are exactly 2.000. This means that, for each drill finger, the two movement steps are always on the longest path, with no slack, in every realization.

For all "Clamp/Unclamp" operations modeled by the Beta distribution, the sensitivity with respect to the second shape parameter (β) is strictly negative. This is a direct consequence of the Beta probability density function: increasing β shifts the probability mass toward the lower bound of the support interval, hence decreasing the node duration. Conversely, the "On/Off" operations for the drills, modeled by the PERT distribution, exhibit a sensitivity of exactly zero across all three parameters, as these specific nodes never belong in the longest path.

Peripheral	**Action**	**Distribution**	**Sensitivity Gradients**		
			Param 1	**Param 2**	**Param 3**
Robot 1	Move	$\mathcal{N}(\mu, \sigma)$	4.985	0.155	–
	Clamp/Unclamp	Beta(α, β)	0.160	-0.241	–
Robot 2	Move	$\mathcal{N}(\mu, \sigma)$	8.985	0.151	–
	Clamp/Unclamp	Beta(α, β)	0.321	-0.482	–
Conditioner	Condition	Gamma(k, θ)	2.580	1.502	–
	Clamp/Unclamp	Beta(α, β)	0.241	-0.363	–
Drill Table	Move	$\mathcal{N}(\mu, \sigma)$	1.501	0.848	–
	Rotate	$\mathcal{N}(\mu, \sigma)$	1.499	0.845	–
	Clamp/Unclamp	Beta(α, β)	0.160	-0.241	–
Drill Thumb	Move	$\mathcal{N}(\mu, \sigma)$	2.000	0	–
	On/Off	PERT(a, b, c)	0	0	0
Drill Index	Move	$\mathcal{N}(\mu, \sigma)$	2.000	0	–
	On/Off	PERT(a, b, c)	0	0	0
Drill Middle	Move	$\mathcal{N}(\mu, \sigma)$	2.000	0	–
	On/Off	PERT(a, b, c)	0	0	0

Table 4.2: Gradient of expected project length with respect to shared peripheral distribution parameters. Parameters 1, 2, and 3 correspond to the standard arguments of each respective distribution.

5

Sensitivity Analysis of Component Waiting Times

After introducing the Generalized Sensitivity Theorem in Chapter 4, we now apply it to the internal logistics of the machines. While total project completion time remains an important performance measure, it does not capture the full dynamics of product-specific behavior during production.

In practice, each product follows a fixed and known sequence of processing steps which corresponds to a predetermined specific path within the scheduling network, as shown in Figure 5.1. However, the timing of these steps is governed by a broader dependency structure that includes numerous auxiliary operations. As a result, even though the processing order is fixed, the product may experience idle periods between consecutive operations due to unresolved dependencies.

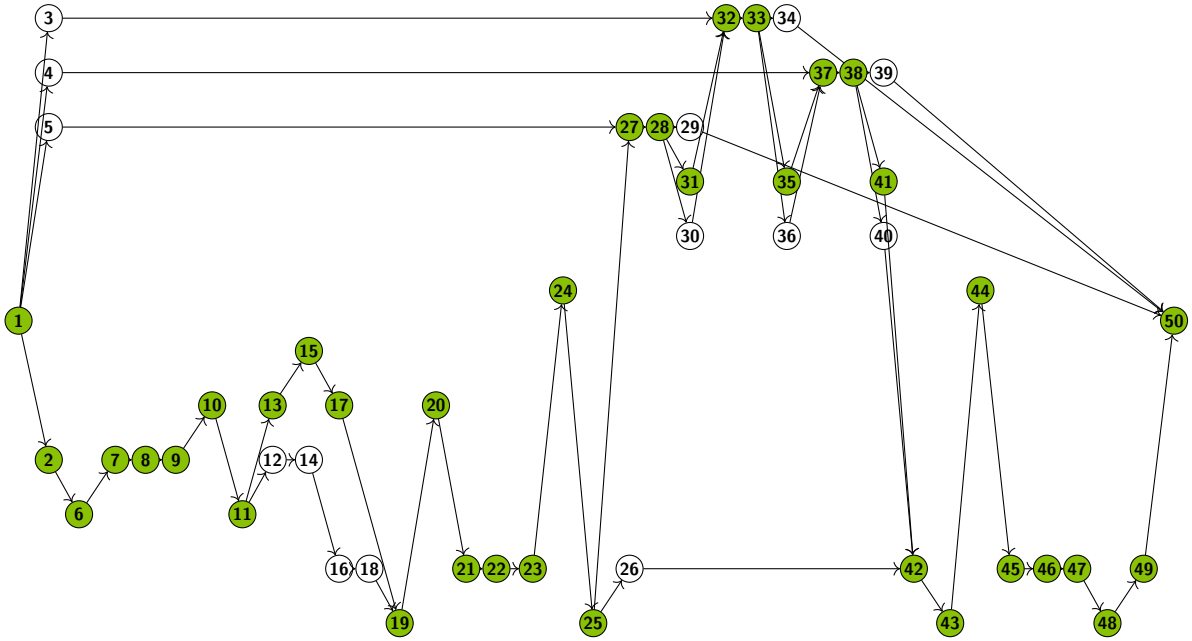


Figure 5.1: Bowling Ball example graph, the internal trajectory of the ball through the machine is represented by highlighted green nodes.

These waiting times can directly influence the physical state of the component: for instance, if removal from an oven is delayed because a required machine task has not yet been completed, the component

remains exposed to heat beyond its planned duration, potentially altering its temperature profile. Such effects cannot be inferred from task durations alone and require a lifecycle-based perspective.

The central objective of this chapter is to quantify how uncertainty in task durations propagates to the waiting times along the component path. Specifically, we derive analytical expressions for the sensitivity of the p-th moment of component waiting times with respect to the distributional parameters of the underlying tasks, enabling targeted improvements in system performance.

5.1. Stochastic Formulation of Waiting Times

Before giving the definition of the waiting time between two actions, we first want to generalize the notion of the longest path in the graph. Given any node i in the graph, we will denote the length of the longest path from the source node to the node i with $\mathcal{L}_i$, while the path itself, containing all the nodes that form the longest path, will be indicated with $\mathcal{V}(\mathcal{L}_i)$ as before. Recall that the definition of $\mathcal{V}(\mathcal{L}_i)$ excludes the node i in the set of nodes. Note that if the first action of the graph starts at time $t = 0$, $\mathcal{L}_i$ indicates the starting time of action i, and the value $\mathcal{L}_i + V_i$ represents the ending time of action i.

For completeness, the overall longest path previously denoted by $\mathcal{L}$ is formally equivalent to $\mathcal{L}_k$. Because the nodes follow a topological ordering, k is always the terminal node, meaning $\mathcal{L} = \mathcal{L}_k$; however, we will continue using $\mathcal{L}$ as a shorthand to refer to the longest path through the entire graph.

Definition 5.1 (Waiting Time)**.** *Given $PN(\mathcal{V}, \mathcal{P})$, and two nodes i and j such that there exists a path $i \to j$, we define the waiting time $W_{i\to j}$ as:*

$$W_{i\to j} = \text{start}(j) - \text{end}(i) = \mathcal{L}_j - \mathcal{L}_i - V_i. \tag{5.1}$$

Because task i is a topological ancestor to j in the component's path, the longest path to j must be at least as long as the path traversing through i. Thus, the waiting time is strictly non-negative ($W_{i\to j} \geq 0$).

Note that i and j do not need to be directly connected by an arc, as it suffices that there exists a path from i to j for the result to hold. In general, when no direct connection is present, we have $W_{i\to j} > 0$, as intermediate nodes that lie between the completion of action i and the start of node j cannot be bypassed.

By the linearity of expectation, the mean waiting time can be trivially decomposed as

$$\mathbb{E}\left[W_{i\to j}\right] = \mathbb{E}\left[\mathcal{L}_j\right] - \mathbb{E}\left[\mathcal{L}_i\right] - \mathbb{E}\left[V_i\right].$$

However, analyzing higher-order moments, such as $\mathbb{E}\left[W_{i\to j}^p\right]$, requires navigating the complex correlation between $\mathcal{L}_j$ and $\mathcal{L}_i$. We leverage the Generalized Sensitivity Theorem to derive exact analytical gradients for these moments without relying on Gaussian assumptions.

5.2. Gradient of the p-th Moment of Waiting Time

Theorem 5.2 (Sensitivity of Waiting Time Moments)**.** *Let $W_{i\to j} = \mathcal{L}_j - \mathcal{L}_i - V_i$ be the waiting time between two component tasks i and j such that there exists a path $i \to j$. Assume that all the node durations $V_k(\boldsymbol{\theta}, \omega)$ satisfy the assumptions outlined in Theorem 4.6. Furthermore, assume all node durations possess finite moments of orders up to $p \geq 1$.*

The partial derivative of the p-th moment of the waiting time with respect to an arbitrary parameter θ_k is given by:

$$\frac{\partial}{\partial \theta_k}\mathbb{E}\left[W_{i\to j}^p\right] = p \cdot \mathbb{E}\left[W_{i\to j}^{p-1} \sum_{v\in\mathcal{V}} \left(\mathbb{1}_{\{v \in \mathcal{V}(\mathcal{L}_j)\}} - \mathbb{1}_{\{v \in \mathcal{V}(\mathcal{L}_i)\}} - \mathbb{1}_{\{v = i\}}\right) \frac{\partial V_v}{\partial \theta_k}\right], \tag{5.2}$$

where $\mathcal{V}(\mathcal{L}_j)$ and $\mathcal{V}(\mathcal{L}_i)$ denote the almost surely unique sets of nodes belonging to the longest paths leading to j and i, respectively, and $\mathbb{1}_{\{\cdot\}}$ is the indicator function.

Proof. The proof relies on defining an appropriate target composition and applying Theorem 4.6.

Step 1: Verification of Target Function Conditions
Let the target function be $g(W_{i\to j}) = W_{i\to j}^p$. This function is continuously differentiable on $\mathbb{R}$. Its first derivative is:

$$g'(W_{i\to j}) = pW_{i\to j}^{p-1}.$$

Since $g'(W_{i\to j})$ is a polynomial of degree $p-1$, it trivially satisfies the polynomial growth bound condition $|g'(x)| \leq C(1+|x|^{p-1})$. Furthermore, $W_{i\to j}$ is a linear combination of maxima of node durations which have all p-moments by assumption, and therefore $W_{i\to j}$ has a finite p-moment, guaranteeing the integrability of the bounding function required by the Leibniz Integral Rule.

Step 2: Sample Path Derivative of the Waiting Time
We must determine the derivative of the waiting time $W_{i\to j}$ with respect to the parameter θ_k. By the linearity of differentiation, we write:

$$\frac{\partial W_{i\to j}}{\partial \theta_k}(\boldsymbol{\theta}, \omega) = \frac{\partial \mathcal{L}_j}{\partial \theta_k}(\boldsymbol{\theta}, \omega) - \frac{\partial \mathcal{L}_i}{\partial \theta_k}(\boldsymbol{\theta}, \omega) - \frac{\partial V_i}{\partial \theta_k}(\boldsymbol{\theta}, \omega). \tag{5.3}$$

From Step 1 of the Generalized Sensitivity Theorem's proof, the absolute continuity of the joint distribution ensures that the longest paths leading to V_j and V_i are unique almost everywhere. Thus, for any node $m \in \mathcal{V}$, the sample path derivative of the start time $\mathcal{L}_m$ with respect to θ_k is the sum of the partial derivatives of the nodes precisely on that path:

$$\frac{\partial \mathcal{L}_m}{\partial \theta_k} = \sum_{v \in \mathcal{V}(\mathcal{L}_m)} \frac{\partial V_v}{\partial \theta_k} = \sum_{v \in \mathcal{V}} \mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_m)\} \frac{\partial V_v}{\partial \theta_k}.$$

Substituting this structural identity into Equation (5.3) for both $\mathcal{L}_j$ and $\mathcal{L}_i$, we obtain the sample path derivative of the waiting time:

$$\frac{\partial W_{i\to j}}{\partial \theta_k}(\boldsymbol{\theta}, \omega) = \sum_{v \in \mathcal{V}} \left(\mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_j)\} - \mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_i)\} - \mathbb{1}\{v = i\} \right) \frac{\partial V_v}{\partial \theta_k}(\boldsymbol{\theta}, \omega). \tag{5.4}$$

Step 3: Expectation Interchange
Applying the Chain Rule, the sample path derivative of our target function $g(W_{i\to j}) = W_{i\to j}^p$ is:

$$\frac{\partial}{\partial \theta_k}(W_{i\to j}^p) = pW_{i\to j}^{p-1} \frac{\partial W_{i\to j}}{\partial \theta_k}.$$

Because the conditions for almost sure differentiability and integrable local domination are satisfied, the Leibniz Integral Rule justifies the interchange of the expectation and derivative operators:

$$\frac{\partial}{\partial \theta_k} \mathbb{E}\left[W_{i\to j}^p\right] = \mathbb{E}\left[\frac{\partial}{\partial \theta_k}(W_{i\to j}^p)\right].$$

Substituting Equation (5.4) into the expectation yields the initial gradient expression:

$$\frac{\partial}{\partial \theta_k} \mathbb{E}\left[W_{i\to j}^p\right] = \mathbb{E}\left[pW_{i\to j}^{p-1} \sum_{v \in \mathcal{V}} \left(\mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_j)\} - \mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_i)\} - \mathbb{1}\{v = i\} \right) \frac{\partial V_v}{\partial \theta_k}\right].$$

□

To simplify this expression for computational applications, let us define $\mathcal{V}(\mathcal{L}_i)^+ = \mathcal{V}(\mathcal{L}_i) \cup \{i\}$ as the complete set of nodes on the component's critical path up to the conclusion of task V_i. We can consolidate the indicators:

$$\mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_i)\} + \mathbb{1}\{v = i\} = \mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_i)^+\}.$$

Because of this, we introduce an auxiliary variable that describes the influence that a certain node has on a given waiting time.

Definition 5.3 (Path-Difference Variable)**.** *Given the waiting time $W_{i\to j}$ between two component tasks i and j such that there exists a path $i \to j$, the discrete path-difference variable $\Delta_{i\to j}(v) \in \{-1, 0, 1\}$ for any node $v \in \mathcal{V}$ is defined as:*

$$\Delta_{i\to j}(v) = \mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_j)\} - \mathbb{1}\{v \in \mathcal{V}(\mathcal{L}_i)^+\}.$$

Substituting this variable into the expectation yields the simplified, closed-form gradient:

$$\frac{\partial}{\partial \theta_k}\mathbb{E}\left[W_{i\to j}^p\right] = \mathbb{E}\left[pW_{i\to j}^{p-1}\sum_{v\in\mathcal{V}}\Delta_{i\to j}(v)\frac{\partial V_v}{\partial \theta_k}\right]. \tag{5.5}$$

The variable $\Delta_{i\to j}(v) \in \{-1, 0, 1\}$ acts as a discrete state indicator isolating the marginal impact of any network node v on the waiting time $W_{i\to j}$. Its three potential values map to distinct physical bottleneck conditions:

- $\Delta_{i\to j}(v) = 1$: Node v lies strictly on the machine's longest path for task j ($\mathcal{V}(\mathcal{L}_j)$), but not on the component's longest path concluding task i ($\mathcal{V}(\mathcal{L}_i)^+$). Physically, node v delays the machine's readiness to begin task j without delaying the component's arrival. Increasing the duration of V_v strictly increases the waiting time.
- $\Delta_{i\to j}(v) = -1$: Node v lies strictly on the component's longest path concluding task i ($\mathcal{V}(\mathcal{L}_i)^+$), but not on the machine's longest path for task j ($\mathcal{V}(\mathcal{L}_j)$). Physically, node v delays the component's arrival without delaying machine readiness. Increasing the duration of V_v holds the component back, causing it to arrive closer to the machine's ready state, thereby decreasing the waiting time gap.
- $\Delta_{i\to j}(v) = 0$: Node v exists on both longest paths, on neither, or there is no waiting time ($W_{i\to j} = 0$). Marginal changes to V_v either shift the component's arrival and the machine's readiness by identical amounts or shift neither. The net impact on the waiting time is zero.

5.3. Simulation and Sensitivity Analysis of Waiting Times

As previously established, analyzing waiting times provides useful insight into how variability in the starting times of each task physically impacts the component. To contextualize this theoretical framework, we refer to the bowling ball manufacturing process introduced in Section 2.3. In this network, node 15 represents the conditioning of the ball: a thermal process designed to soften the material. The duration the ball remains in the conditioner strictly determines its internal temperature and, consequently, its final structural integrity. Therefore, precisely quantifying the time spent in the conditioner is critical for quality control.

Based on the network topology, the operation corresponding to the physical removal of the ball from the conditioner is represented by node 21. It is important to note that between the completion of the programmed conditioning cycle (node 15) and the physical extraction of the ball (node 21), the machine must execute several preparatory actions, such as maneuvering Robot 2 and engaging the

clamps. However, from the perspective of the component, it remains physically housed within the active conditioner during these intermediate operations.

By applying the distributional assumptions established in Chapter 4, we investigate the expected waiting time $\mathbb{E}[W_{15\to21}]$. This analysis allows us to quantify the severity of the delay and identify optimal mitigation strategies.

A Monte Carlo simulation utilizing $N = 10^6$ samples yields an expected waiting time of:

$$\mathbb{E}[W_{15\to21}] \approx 1.658.$$

Given that the expected duration of the conditioning task is $\mathbb{E}[V_{15}] = 8$, this expected delay induces an approximate 20.7% increase in the component's exposure to the conditioning environment. While a naive scheduling approach might attempt to mitigate this by simply subtracting the expected delay from the programmed conditioning time, our analytical sensitivity framework enables a more rigorous identification of the underlying dynamics.

Evaluating the gradient of the expected waiting time with respect to the network's shared distributional parameters yields the sensitivities presented in Table 5.1.

Peripheral	**Action**	**Distribution**	**Sensitivity Gradients**	
			Param 1	**Param 2**
Robot 1	Move	$\mathcal{N}(\mu, \sigma)$	0.975	0.148
	Clamp/Unclamp	Beta(α, β)	0	0
Robot 2	Move	$\mathcal{N}(\mu, \sigma)$	0.982	0.154
	Clamp/Unclamp	Beta(α, β)	0	0
Conditioner	Condition	Gamma(k, θ)	-1.421	-0.499
	Clamp/Unclamp	Beta(α, β)	-0.080	0.119

Table 5.1: Gradient of $\mathbb{E}[W_{15\to21}]$ with respect to shared peripheral distribution parameters. Parameters not included in the table have a sensitivity of exactly zero, as they do not lie on the relevant critical paths.

The computed gradients align rigorously with the topological structure of the network and the behavior of the discrete path-difference variable $\Delta_{15\to21}(v)$. Operations topologically succeeding nodes 15 and 21, such as downstream drilling operations, exhibit a sensitivity of strictly zero.

Furthermore, the positive gradients associated with the movement parameters of Robot 1 and Robot 2 indicate that they frequently act as constraints on the machine's longest path without delaying the component's path ($\Delta_{15\to21}(v) = 1$). Conversely, the negative gradients associated with the Conditioner dictate that artificially extending the conditioning task delays the component's arrival without affecting the robots' readiness ($\Delta_{15\to21}(v) = -1$), shrinking the time gap. Consequently, lowering this waiting time structurally requires either reducing the expected operational duration of the robotic arms or extending the conditioning duration to absorb the delay.

6

Just In Time Scheduling

The analytical framework established in the preceding chapters models traditional scheduling, where in each operation initiates immediately upon the resolution of its precedence constraints. As defined in Chapter 5, assuming the initial network task starts at time $t = 0$, the start time of task i is exactly $\mathcal{L}_i$.

Conversely, Just In Time (JIT) scheduling aims to delay the start of operations to minimize the duration components spend in idle states. This delay directly minimizes the component waiting times defined in Chapter 5.

An example of this is shown in Figure 6.1, which includes a Gantt chart, a helpful and commonly used representation in project scheduling settings; it lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis.

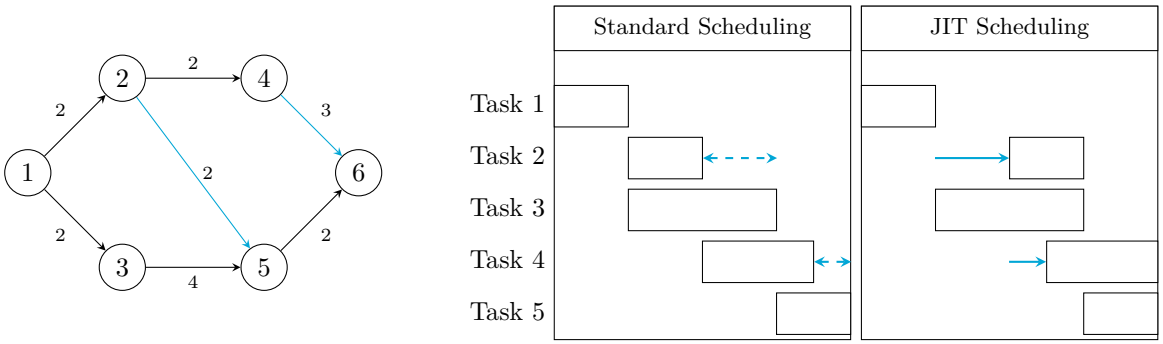


Figure 6.1: On the left, we have a project network with 6 tasks, each task duration is constant and is indicated by the number on the outgoing edge. On the right are two Gantt charts corresponding to standard and JIT scheduling, respectively. Note that task 6 is not reported in the charts since it represents the "end" task, and thus it has no duration. In both cases, the duration of the project is the same (reported on the x axis), but in the JIT scheduling, the starting times of tasks 2 and 4 are postponed to minimize the waiting time between task 2 and 5, and between task 4 and 6, respectively.

Under deterministic conditions, executing an JIT schedule is a well-defined problem [3], since all task durations are known constants, start times can be precisely shifted forward in time without risk of overlap. However, in a stochastic environment, task durations and critical paths are random variables, rendering true JIT scheduling strictly impossible; perfectly aligning the completion of a random path with the start of another necessitates a priori knowledge of their exact realizations.

To approximate JIT behavior within a stochastic network, we present a structural tradeoff between the overall expected project duration and the localized waiting time of specific components. This approximation is achieved by inserting parameterized artificial dependencies into the project network.

These auxiliary arcs artificially delay the start of specific tasks, forcing JIT execution, while allowing the framework to analytically track the resulting probabilistic impact on the global longest path.

6.1. Artificial Delay Optimization

To postpone an operation's start time, we introduce a fixed-value auxiliary dependency into the network.

Definition 6.1 (Artificial Delay)**.** *Consider a project network $PN(\mathcal{V}, \mathcal{P})$ with nodes $\mathcal{V} = \{1, \dots, k\}$. We can set a lower bound on the start time of any node i by inserting a new delay node $v_i^{(d)}$ into the network, as a predecessor to task i, and as a successor to the project source (node 1), hence the two arcs are created $\left(1, v_i^{(d)}\right)$, $\left(v_i^{(d)}, i\right)$. We set $V_{v_i^{(d)}} = d_i$, and define d_i to be the artificial delay.*

For clarity, even after adding nodes to the network, we always keep node 1 and node k to be the source and sink nodes of the graph.

For any two nodes i and j such that there exists a path $i \to j$, the objective is to determine an optimal d_i that minimizes the local waiting time $W_{i \to j}$. While this waiting time decreases as d_i increases, this reduction is bounded by its impact on the total project duration $\mathcal{L}$.

In a stochastic context, assigning the right duration for the delays becomes a guessing game. On the one hand, you don't want to guess too short and leave unnecessary waiting time, on the other hand, you don't want to guess too long and delay the whole schedule. As a consequence, JIT scheduling becomes an optimization problem defined by the tradeoff between minimizing the local waiting time and maintaining the expected global completion time.

Fortunately, this tradeoff can be formalized by a loss function with a parameter, which acts as a preference for minimizing local waiting time $W_{i \to j}$ as opposed to prioritizing shorter project lenghts.

Definition 6.2 (Loss function for tradeoff)**.** *Given a parameter $\alpha \in (0, 1)$ and two nodes i and j such that there exists a path $i \to j$, we define the following loss function for the optimization problem:*

$$L_{i \to j}^{\alpha}(d_i) = \alpha \mathbb{E}\left[\mathcal{L}\right] + (1 - \alpha)\mathbb{E}\left[W_{i \to j}\right].$$

The two opposing terms in the loss function and their respective trends as d_i varies are illustrated in Figure 6.2.

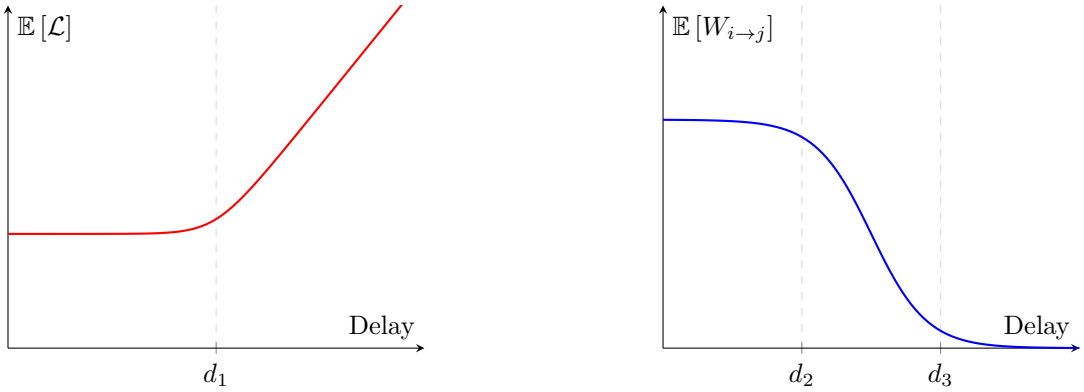


Figure 6.2: The two opposing terms in the loss function and their trend as d_i changes.

The first term, $\mathbb{E}[\mathcal{L}]$, remains invariant for sufficiently small artificial delays, since in this regime the delay d_i does not affect the global completion time. However, as d_i is incrementally increased, we eventually reach a critical threshold d_1 where the accumulated delay consumes the available slack, forcing node i to become part of the longest path. Beyond this point, $\mathbb{E}[\mathcal{L}]$ exhibits a linear increase,

as governed by the following relationship:

$$\frac{\partial \mathbb{E}\left[\mathcal{L}\right]}{\partial d_i} = \mathbb{E}\left[\sum_{v \in \mathcal{V}(\mathcal{L})} \frac{\partial V_v}{\partial d_i}\right] = \mathbb{E}\left[\underbrace{\frac{\partial V_{v_i^{(d)}}}{\partial d_i}}_{=1} \cdot \mathbb{1}\left\{v_i^{(d)} \in \mathcal{V}(\mathcal{L})\right\}\right] = \mathbb{P}\left(v_i^{(d)} \in \mathcal{V}(\mathcal{L})\right).$$

Analytically, this formula reveals that the marginal impact of the artificial delay on the expected project duration is exactly the probability of the delay node $v_i^{(d)}$ belonging to the longest path. When d_i is small, this probability is negligible, resulting in a near-zero slope. As d_i surpasses the threshold d_1, the probability $\mathbb{P}\left(v_i^{(d)} \in \mathcal{V}(\mathcal{L})\right)$ converges toward 1, causing the expected project length to increase proportionally with the delay. Note that in a general scenario, d_1 might be zero, indicating that the node already is in the longest path, and any perturbation in its starting time will impact the expected total project length $\mathbb{E}\left[\mathcal{L}\right]$.

The second plot in Figure 6.2 illustrates the sensitivity of the expected waiting time $\mathbb{E}\left[W_{i\to j}\right]$ with respect to the value of the artificial delay d_i. For values below the threshold d_2, the waiting time remains invariant, as the constraints of the network already dictate a start time for node i that exceeds the artificial delay. Once d_i surpasses d_2, the artificial delay becomes the dominant constraint on the starting time of node i, shifting its execution interval further into the future.

$$\frac{\partial \mathbb{E}\left[W_{i\to j}\right]}{\partial d_i} = \mathbb{E}\left[\sum_{i \in \mathcal{V}} \Delta_{i\to j}(v) \frac{\partial V_i}{\partial d_i}\right] = \mathbb{E}\left[\Delta_{i\to j}\left(v_i^{(d)}\right)\right].$$

This delay reduces the overlap between the completion of node i and the active durations of other parallel predecessors of node j, therefore diminishing the expected waiting time. This reduction continues until the delay reaches a second critical point d_3, where node i is virtually guaranteed to finish after all other merging dependencies. In this regime, the waiting time $\mathbb{E}\left[W_{i\to j}\right]$ converges to zero and the function plateaus, signifying that the local component waiting time has been effectively eliminated.

Given that both terms of the loss function exhibit flat regions, the resulting surface $L^{\alpha}_{i\to j}(d_i)$ naturally inherits plateaus where the gradient is close to zero. In these areas, a standard gradient-based search would fail to move because the derivative provides no direction.

To resolve this initialization problem, the gradient descent algorithm must begin in the active, sloping region of the loss function. Ideally, this starting value would exceed both thresholds d_1 and d_2 to ensure a non-zero gradient for the expected waiting time. However, because these thresholds are an emergent property of the network's joint probability distribution, their exact value is analytically intractable to compute a priori.

To avoid the initial plateau without needing exact threshold calculations, we use expected path metrics from a simpler baseline evaluation to set a target synchronization value.

$$d_i^{(0)} = \mathbb{E}\left[\mathcal{L}_j\right] - \mathbb{E}\left[V_i\right].$$

This heuristic sets the artificial delay so that task i is expected to finish exactly when task j is ready on average. By choosing the delay to match the average gap between the two tasks, $d_i^{(0)}$ becomes the main factor controlling the start time of task i in many simulated scenarios. At this point, the delay is large enough to avoid the initial plateau, since any small increase in d_i will push task i later, which reduces the expected waiting time $\mathbb{E}\left[W_{i\to j}\right]$.

In the next section, we will apply this theory to approximate JIT scheduling for a handful of nodes in the Bowling Ball Drilling simulation, as it was outlined in 4.5.

6.2. JIT Scheduling Simulation

To illustrate the practical application of this optimization tradeoff, we return to the bowling ball manufacturing process introduced in Section 2.3. Specifically, we focus on delaying the activation sequence of the three drills, represented by nodes 3, 4, and 5. Because their sole topological predecessor is the project source (node 1), an ASAP scheduling policy forces these machines to turn on immediately at the start of the production cycle. However, the drills are not actively used until the component physically arrives at the drilling table later in the process (nodes 32, 37, and 27 respectively). This premature activation results in substantial idle time and unnecessary energy consumption, presenting a clear target for JIT optimization.

We establish a baseline by first evaluating the waiting times of these components under the standard ASAP scheduling policy. The expected idle periods for the three drills are significant:

$$\mathbb{E}\left[W_{5\to 27}\right] \approx 28.069, \qquad \mathbb{E}\left[W_{3\to 32}\right] \approx 34.632, \qquad \mathbb{E}\left[W_{4\to 37}\right] \approx 41.195.$$

To mitigate this inefficiency, we optimize the artificial delays $\mathbf{d} = (d_5, d_3, d_4)$ jointly. A simultaneous optimization is required because adjusting the start time of one component can shift the global longest path, inadvertently affecting the waiting times of the other components. We define the set of targeted dependencies as $\mathcal{A} = \{(3, 32), (4, 37), (5, 27)\}$, and formulate the objective using the previously established tradeoff loss function:

$$L^{\alpha}(\mathbf{d}) = \alpha \mathbb{E}\left[\mathcal{L}\right] + (1-\alpha) \sum_{(u,v)\in\mathcal{A}} \mathbb{E}\left[W_{u\to v}\right].$$

Similarly, the derivative of the loss function with respect to the l-th component of the delay vector $\mathbf{d}$ is given by:

$$\frac{\partial}{\partial \mathbf{d}_l} L^{\alpha}(\mathbf{d}) = \alpha \mathbb{P}\left(v_l^{(d)} \in \mathcal{V}(\mathcal{L})\right) + (1-\alpha) \sum_{(u,v)\in\mathcal{A}} \mathbb{E}\left[\Delta_{u\to v}\left(v_u^{(d)}\right)\right].$$

To find the optimal delay values, we use RMSProp [23], an adaptive optimization method that adjusts the learning rate based on past gradients. It is well-suited for problems where the loss is estimated through sampling and can be noisy or vary between evaluations.

The algorithm demonstrates rapid convergence, reaching the optimal delay configuration in just 11 iterations:

$$d_5 = 26.556, \qquad d_3 = 31.708, \qquad d_4 = 40.643.$$

By introducing these artificial delays into the network, the expected waiting times are drastically reduced:

$$\mathbb{E}\left[W_{5\to 27}\right] \approx 2.726, \qquad \mathbb{E}\left[W_{3\to 32}\right] \approx 4.226, \qquad \mathbb{E}\left[W_{4\to 37}\right] \approx 2.625.$$

To quantify this improvement, the introduction of these artificial delays eliminates over 90% of the initial idle time for each drill machine. Specifically, the waiting time for the drill represented by node 4 drops from roughly 41.2 down to just 2.6, translating into a massive reduction in wasted energy.

While minimizing local idle time is the primary objective, we must also assess the impact on the global project duration. The optimization incurs a minor penalty on the expected project length, increasing the mean duration from 60.540 to 62.629. However, this introduces a highly beneficial secondary effect: the variance of the total project duration is significantly reduced, falling from 41.623 to 24.758. The histograms of the distribution of $\mathcal{L}$ both with and without the artificial delay nodes are shown in Figure 6.3.

This steep drop in variance occurs because the artificial delays act as deterministic constants within the network. By inserting large delays, we increase the probability that these constant values dominate the longest path, thereby masking the natural stochastic variance of the other tasks.

Mechanically, this variance reduction occurs because the artificial delay acts as a hard lower bound on the start time of the tasks. In a purely ASAP schedule, early finishes of preceding stochastic tasks allow subsequent tasks to start early, which contributes to a wider lower tail in the overall duration distribution. An analysis of the criticality indices confirms this behavior, showing the probability of each delay node residing on the critical path:

$$\mathbb{P}\left(v_5^{(d)} \in \mathcal{V}(\mathcal{L})\right) \approx 0.145, \qquad \mathbb{P}\left(v_3^{(d)} \in \mathcal{V}(\mathcal{L})\right) \approx 0.008, \qquad \mathbb{P}\left(v_4^{(d)} \in \mathcal{V}(\mathcal{L})\right) \approx 0.404.$$

Cumulatively, there is a 0.557 probability that the longest path incorporates at least one of these long deterministic durations, which heavily stabilizes the overall project completion time.

If the minor increase in the expected project duration is deemed unacceptable, the tradeoff can be adjusted by increasing the parameter α. Rerunning the simulation with $\alpha = 0.9$ prioritizes the minimization of the global project length. Under these new conditions, the gradient descent converges in 12 steps, yielding a more conservative set of delays:

$$d_5 = 22.509, \qquad d_3 = 30.390, \qquad d_4 = 35.342.$$

Implementing this revised delay vector results in expected waiting times that are slightly higher than the previous optimization, but still vastly superior to the baseline ASAP configuration:

$$\mathbb{E}\left[W_{5\to 27}\right] \approx 5.749, \qquad \mathbb{E}\left[W_{3\to 32}\right] \approx 4.723, \qquad \mathbb{E}\left[W_{4\to 37}\right] \approx 6.378.$$

Critically, this conservative approach successfully restricts the growth of the expected total project time, resulting in a mean duration of 61.078 and a variance of 35.006. This demonstrates the flexibility of the parameterized loss function in achieving a precise, controllable balance between local operation efficiency and global project performance.

Conversely, setting $\alpha = 0.25$ shifts the optimization toward prioritizing waiting-time reduction over minimizing the overall project duration. This produces substantially larger artificial delays:

$$d_5 = 32.657, \qquad d_3, = 41.283 \qquad d_4 = 49.229.$$

Under this configuration, the targeted waiting times are driven much closer to zero:

$$\mathbb{E}\left[W_{5\to 27}\right] \approx 0.743, \qquad \mathbb{E}\left[W_{3\to 32}\right] \approx 0.579, \qquad \mathbb{E}\left[W_{4\to 37}\right] \approx 0.594.$$

This represents a substantial reduction relative to previous regimes. The improvement in local efficiency, however, comes at the cost of a pronounced increase in the expected total project duration, which rises to 69.186, as shown in Figure 6.4.

A summary of all the results shown in this section is shown in Table 6.1.

Configuration (α)	**Optimal Delays d**			**Expected Waiting Times**			**Global Project**	
	d_5	d_3	d_4	$\mathbb{E}[W_{5\to 27}]$	$\mathbb{E}[W_{3\to 32}]$	$\mathbb{E}[W_{4\to 37}]$	$\mathbb{E}[\mathcal{L}]$	$\mathrm{Var}\,[\mathcal{L}]$
No Delays	0	0	0	28.069	34.632	41.195	60.540	41.623
$\alpha = 0.90$	22.509	30.390	35.342	5.749	4.723	6.378	61.078	35.006
$\alpha = 0.75$	26.556	31.708	40.643	2.726	4.226	2.625	62.629	24.758
$\alpha = 0.25$	32.657	41.283	49.229	0.743	0.579	0.594	69.186	17.043

Table 6.1: Summary of artificial delay optimization results and the resulting trade-offs between local waiting times and global project duration across different loss function weights (α).

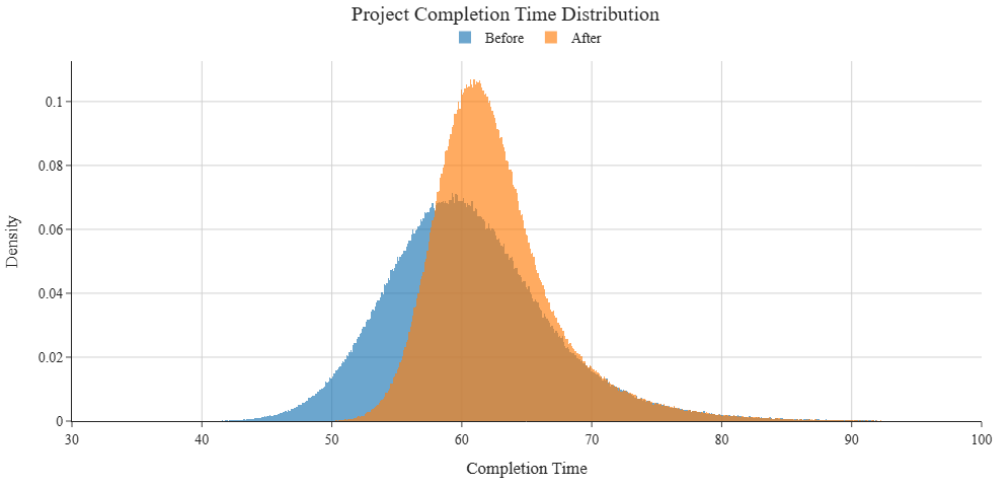


Figure 6.3: Distribution of project completion times before and after delay optimization with $\alpha = 0.75$. The higher weight on the expected project duration produces moderate artificial delays that balance reductions in local waiting times with a limited increase in the mean completion time.

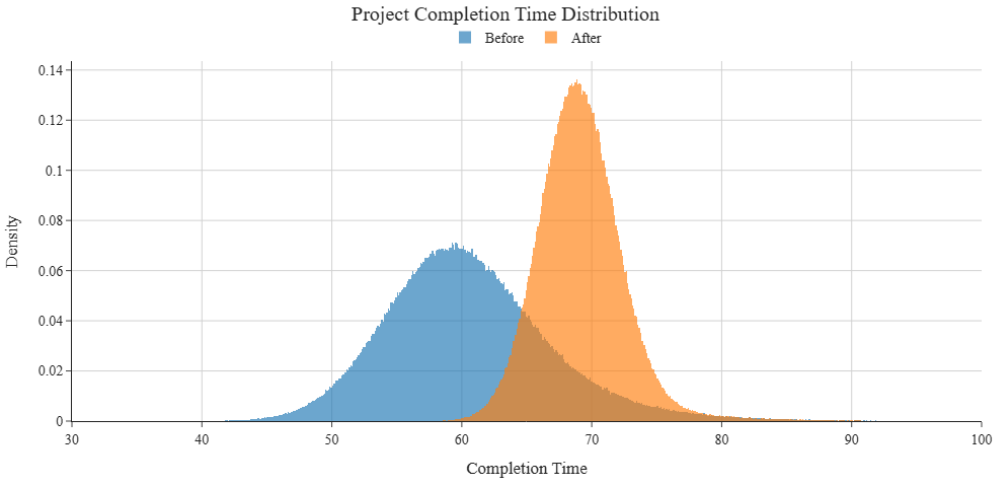


Figure 6.4: Distribution of project completion times before and after delay optimization with $\alpha = 0.25$. The stronger emphasis on reducing local waiting times induces larger artificial delays, significantly shifting probability mass toward longer project durations.

7

Conclusion

Stochastic scheduling is an essential tool for understanding how uncertainty propagates through complex manufacturing processes, as introduced in Chapter 1. While traditional methods provide ways to estimate expected completion times, they often lack the analytical depth required to see exactly how local variations affect global system performance. The primary objective of this thesis, outlined in Section 1.3, was to bridge this gap by developing an exact analytical framework to calculate the sensitivity of the longest path distribution in directed acyclic graphs. By identifying which task parameters most significantly influence production time, industries can implement targeted interventions.

To achieve this, we first established a foundation using the graph-theoretic framework defined in Chapter 2 alongside Gaussian node distributions. This allowed us to derive exact, closed-form formulas for the gradients of the expected project duration, as proven in Theorem 3.4 in Chapter 3. Recognizing that real-world applications require more flexibility, we expanded this foundation into the Generalized Sensitivity Theorem in Chapter 4. This significant generalization, detailed in Theorem 4.6, accommodates a much broader class of probability distributions and allows for shared parameters across multiple nodes. This ensures we can accurately model realistic industrial scenarios where multiple distinct actions are governed by the same physical peripheral.

Beyond looking only at the total completion time, this framework proved highly effective for analyzing internal machine logistics, specifically component waiting times explored in Chapter 5. In practical manufacturing, components often experience idle periods waiting for unresolved dependencies. By quantifying how task uncertainty affects these local delays, we were able to introduce a mathematical approach to approximate Just In Time (JIT) scheduling within stochastic environments in Chapter 6.

This JIT approximation was achieved by introducing parameterized artificial delays into the network. We formulated an optimization problem using the loss function defined in Definition 6.2, which explicitly balances the tradeoff between minimizing local component waiting times and maintaining a short overall expected project length. The simulations, visualized in Figures 6.3, 6.4, demonstrated that optimizing these delays not only drastically reduces unnecessary idle time but also significantly reduces the variance of the total project duration, leading to more predictable manufacturing cycles.

This thesis develops a direct analytical way to evaluate system performance, rather than relying only on empirical approximations. The results expose where delays and inefficiencies actually originate in the underlying structure of the process, providing a clearer mathematical basis for identifying bottlenecks and understanding how they affect production.

7.1. Future Research

While the analytical framework developed in this thesis provides a solid basis for stochastic scheduling, several important directions remain open to address its computational and methodological limitations.

7.1.1. Non-Smooth Extensions of Sensitivity Analysis for Longest Path

A fundamental assumption of the current model is that the joint distribution of stochastic nodes must be absolutely continuous. This guarantees that the probability of multiple paths resulting in the exact same duration is zero, ensuring the longest path is almost surely differentiable. Future work could attempt to adapt the generalized gradient calculations to handle discrete task durations or systems where structural path ties occur with a non-zero probability. When task durations include discrete components or identical deterministic branches, the longest-path operator introduces points where the objective is no longer differentiable.

Addressing this breaks the smoothness assumptions of the current framework and naturally leads to non-smooth analysis. Future work would therefore need to use tools such as subgradients [4] or Clarke generalized derivatives [11] to properly characterize sensitivity in these cases. A key extension would be the development of methods to compute expected subgradients when multiple critical paths coexist, broadening the applicability of the approach to networks with more rigid timing structures.

7.1.2. Dynamic JIT Scheduling and Stochastic Control

The optimization of artificial delays d_i presented in Chapter 6 is formulated as a static strategy where the delay vector is determined before the start of the project. However, in a real-world manufacturing environment, the realization of early node durations provides critical information that alters the conditional distributions of the remaining paths. A natural extension of this work is the transition from a static framework to a dynamic, online scheduling policy.

Future research could investigate the formulation of the JIT objective within the context of stochastic optimal control or Markov Decision Processes (MDPs) [27]. Specifically, one could aim to dynamically update the optimal delay d_i conditioned on the filtration $\mathcal{F}_t$, representing the exact realizations of all tasks completed up to time t. This would involve calculating the sensitivity of a time-dependent value function with respect to the real-time state of the directed acyclic graph, allowing the system to absorb stochastic shocks as they occur and maintain the project deadline more robustly.

7.1.3. Advanced Variance Reduction in High-Dimensional Estimators

The numerical validation of Theorem 4.6 relied on standard Monte Carlo simulation, which exhibits a convergence rate of $\mathcal{O}(1/\sqrt{N})$. While effective for the examples provided, the computational cost of this approach scales poorly as the network dimension k and the parameter space Θ increase. This is particularly relevant when sensitivity analysis is embedded within iterative optimization loops such as RMSProp.

A possible extension is the use of Multilevel Monte Carlo (MLMC) [16] for the longest path operator. The main idea is to combine estimators of different accuracy, ranging from coarse approximations based on expected path lengths to full stochastic simulations. This hierarchy can reduce variance and improve efficiency compared to standard Monte Carlo. In addition, control variates can be introduced by exploiting simpler analytical approximations as baselines, for instance, those derived under Gaussian assumptions. These techniques can reduce variance further and accelerate gradient estimation in the general setting. Together, they would make the approach more scalable to large networks where direct simulation becomes computationally expensive.

Bibliography

[1] A. V. Aho, M. R. Garey, and J. D. Ullman. "The transitive reduction of a directed graph". In: *SIAM J. Comput.* 1.2 (1972), pp. 131–137. ISSN: 0097-5397. DOI: 10.1137/0201008. URL: https://doi.org/10.1137/0201008.

[2] Ei Ando, Toshio Nakata, and Masafumi Yamashita. "Approximating the longest path length of a stochastic DAG by a normal distribution in linear time". In: *Journal of Discrete Algorithms* 7.4 (2009), pp. 420–438. ISSN: 1570-8667. DOI: https://doi.org/10.1016/j.jda.2009.01.001. URL: https://www.sciencedirect.com/science/article/pii/S1570866709000033.

[3] Zoltan Baruch. "Scheduling Algorithms for High-Level Synthesis". In: *ACAM Scientific Journal* 5.1-2 (1996), pp. 48–57.

[4] Dimitri P. Bertsekas. *Convex Optimization Algorithms*. MIT Lecture Notes / Athena Scientific. 2015. URL: https://web.mit.edu/dimitrib/www/CONVEX_ALGORITHMS_Short_View.pdf.

[5] Joseph K Blitzstein and Jessica Hwang. *Introduction to probability, second edition*. en. 2nd ed. Chapman & Hall/CRC Texts in Statistical Science. London, England: CRC Press, Feb. 2019.

[6] John M. Burt and Mark B. Garman. "Conditional Monte Carlo: A Simulation Technique for Stochastic Network Analysis". In: *Management Science* 18.3 (1971), pp. 207–217. DOI: 10.1287/mnsc.18.3.207. URL: https://doi.org/10.1287/mnsc.18.3.207.

[7] Louis-Claude Canon and Emmanuel Jeannot. "Precise Evaluation of the Efficiency and the Robustness of Stochastic DAG Schedules". In: *10ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision - ROADEF 2009*. Nancy, France, Feb. 2009, pp. 13–24. URL: https://hal.science/hal-00431193.

[8] Attanasio Carlo. *Sensitivity Analysis in Stochastic Scheduling*. https://github.com/carlo-attanasio/SensitivityAnalysisStochasticScheduling. 2026.

[9] C. E. Clark. "The PERT Model for the Distribution of an Activity Time". In: *Operations Research* 10.3 (1962), pp. 405–406. DOI: 10.1287/opre.10.3.405. URL: https://www.scirp.org/reference/referencespapers?referenceid=2157989.

[10] Charles E. Clark. "The Greatest of a Finite Set of Random Variables". In: *Operations Research* 9 (1961), pp. 145–162. DOI: 10.1287/OPRE.9.2.145. URL: https://pubsonline.informs.org/doi/abs/10.1287/opre.9.2.145.

[11] Frank H. Clarke. "Generalized gradients and applications". In: *Transactions of the American Mathematical Society* 205 (1975), pp. 247–262. DOI: 10.1090/S0002-9947-1975-0367131-6.

[12] Duccio Conti. *Stochastic scheduling for machine logistics*. Tech. rep. Delft University of Technology, 2025.

[13] Bajis Dodin. "Bounding the Project Completion Time Distribution in PERT Networks". In: *Operations Research* 33.4 (1985), pp. 862–881. DOI: 10.1287/opre.33.4.862. URL: https://doi.org/10.1287/opre.33.4.862.

[14] Shimon Even. *Graph Algorithms*. Cambridge University Press, Sept. 2011.

[15] Michael Figurnov, Shakir Mohamed, and Andriy Mnih. *Implicit Reparameterization Gradients*. 2019. arXiv: 1805.08498 [cs.LG]. URL: https://arxiv.org/abs/1805.08498.

[16] Michael B. Giles. "Multilevel Monte Carlo methods". In: *Acta Numerica* 24 (2015), pp. 259–328. DOI: 10.1017/S096249291500001X.

[17] Hugo Hernandez. *Leibniz's Rule and other Properties of Integrals of Randomistic Variables*. July 2019. DOI: 10.13140/RG.2.2.16616.01285.

[18] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions, Volume 2*. Wiley Series in Probability and Statistics. Wiley, 1995. ISBN: 9780471584940. URL: https://books.google.com.fj/books?id=BTANEAAAQBAJ.

[19] A. B. Kahn. "Topological sorting of large networks". In: *Commun. ACM* 5.11 (Nov. 1962), pp. 558–562. ISSN: 0001-0782. DOI: 10.1145/368996.369025. URL: https://doi.org/10.1145/368996.369025.

[20] James E. Kelley. "Critical-Path Planning and Scheduling: Mathematical Basis". In: *Operations Research* 9.3 (June 1961), pp. 296–320. DOI: 10.1287/opre.9.3.296. URL: https://ideas.repec.org/a/inm/oropre/v9y1961i3p296-320.html.

[21] George B. Kleindorfer. "Bounding Distributions for a Stochastic Acyclic Network". In: *Operations Research* 19.7 (1971), pp. 1586–1601. DOI: 10.1287/opre.19.7.1586. URL: https://doi.org/10.1287/opre.19.7.1586.

[22] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Books on Mathematics Series. Dover Publications, 2001. ISBN: 9780486414539. URL: https://books.google.nl/books?id=m4MvtFenVjEC.

[23] Runze Li, Jintao Xu, and Wenxun Xing. *Stable gradient-adjusted root mean square propagation on least squares problem*. 2025. arXiv: 2411.15877 [math.OC]. URL: https://arxiv.org/abs/2411.15877.

[24] D. G. Malcolm et al. "Application of a Technique for Research and Development Program Evaluation". In: *Operations Research* 7.5 (1959), pp. 646–669. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/167013 (visited on 03/20/2026).

[25] J. J. Martin. "Distribution of the Time Through a Directed, Acyclic Network". In: *Operations Research* 13.1 (1965), pp. 46–66. DOI: 10.1287/opre.13.1.46. URL: https://doi.org/10.1287/opre.13.1.46.

[26] Viet Anh Nguyen et al. *Mean-Covariance Robust Risk Measurement*. 2025. arXiv: 2112.09959 [q-fin.PM]. URL: https://arxiv.org/abs/2112.09959.

[27] Martin L. Puterman and Wiley InterScience (Online service). *Markov decision processes : discrete stochastic dynamic programming*. New York: Wiley, 1994. ISBN: 9780470316887. URL: https://onlinelibrary.wiley.com/book/10.1002/9780470316887.

[28] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics v. 1. Springer, 2003. ISBN: 9783540443896. URL: https://books.google.nl/books?id=mqGeSQ6dJycC.

[29] K. Thulasiraman and M.N.S. Swamy. *Graphs: Theory and Algorithms*. A Wiley interscience publication. Wiley, 1992. ISBN: 9780471513568. URL: https://books.google.nl/books?id=4kHN6uSinQoC.

[30] R Torishnyi and V Sobol. "Smooth approximation of probability and quantile functions: vector generalization and its applications". In: *Journal of Physics: Conference Series* 1925.1 (May 2021), p. 012034. DOI: 10.1088/1742-6596/1925/1/012034. URL: https://doi.org/10.1088/1742-6596/1925/1/012034.

[31] R. Vershynin. *High-Dimensional Probability, 2nd Edition*. Cambridge University Press, 2025. URL: https://www.math.uci.edu/~rvershyn/papers/HDP-book/HDP-2.pdf.