



Mitigation strategies against cascading failures within a project activity network

Christos Ellinas¹ · Christos Nicolaidis^{2,3} · Naoki Masuda^{4,5,6}

Received: 29 June 2020 / Accepted: 13 May 2021 / Published online: 18 June 2021
© The Author(s) 2021, corrected publication 2021

Abstract

Successful on-time delivery of projects is a key enabler in resolving major societal challenges, such as wasted resources and stagnated economic growth. However, projects are notoriously hard to deliver successfully, partly due to their interconnected and temporal complexity which makes them prone to cascading failures. Here, we develop a cascading failure model and test it on a temporal activity network, extracted from a large-scale engineering project. We evaluate the effectiveness of six mitigation strategies, in terms of the impact of task failure cascading throughout the project. In contrast to theoretical arguments, our results indicate that in the majority of cases, the temporal properties of the activities are more relevant than their structural properties in preventing large-scale cascading failures. In practice, these findings could stimulate new pathways for designing and scheduling projects that naturally limit the extent of cascading failures.

Keywords Project risk · Cascading failures · Mitigation strategies · Complex networks

✉ Christos Ellinas
christos@nodeslinks.com

✉ Naoki Masuda
naokimas@buffalo.edu

¹ Nodes and Links Ltd, Salisbury House, Station Road, Cambridge CB1 2LA, UK

² Department of Business and Public Administration, University of Cyprus, 1 Panepistimiou Av., 2109 Nicosia, Cyprus

³ Initiative on the Digital Economy, Massachusetts Institute of Technology, 100 Main St., Cambridge, MA 02142, USA

⁴ Department of Mathematics, State University of New York at Buffalo, Buffalo, NY 14260-2900, USA

⁵ Computational and Data-Enabled Science and Engineering Program, State University of New York at Buffalo, Buffalo, NY 14260-5030, USA

⁶ Faculty of Management and Economics, Dalian University of Technology, No. 2 Linggong Road, Ganjingzi District, Dalian 116024, Liaoning, China

Introduction

Project-based processes are central in resolving major societal challenges [35], from accelerating economic growth through infrastructure development [25, 41] to fostering public resilience through mobilizing emergency resources [38], [47]. World Bank data (2009) indicate that more than 22% of the world's gross domestic product—equivalent to approximately \$48 trillion—relies almost entirely on project-based delivery mechanisms [57]. Despite the importance of delivering projects successfully, many fail to meet their targets [69]. Delays, cost overruns and quality problems are regularly observed across all project domains, from software development [48] and construction [43] to infrastructure [23] and defence [12]. An industry survey reviewed 10,624 projects from 200 companies in 30 countries and across a variety of industries and concluded that only 2.5% of the companies delivered all of their projects successfully [50]. More recently, a review of 1417 IT projects reveals that 236 of them experienced cost overruns of at least 200% and the delivery of these projects was delayed by almost 70% in time [22]. The implications of project failure are expected to increase even further in the future due to their projected 1.5–2.5% annual growth in project value [21].

Research into understanding project failure can be broadly classified into two distinct, yet complementary, strands [52]. This first strand relies on qualitative methods, focusing on mapping the relationships between sociological factors that contribute to project failure, e.g., importance of leadership [59, 65], team communication [45] and corporate culture [55]. This line of research is central in identifying potential relationships that can control project failure (e.g., that quality of the initial planning is correlated with the project performance, contextual task features (i.e., technical complexity, novelty) is correlated with project success [56]. Whilst important, this research strand is generally associated with a multitude of biases such as recollection bias (i.e., information bias in which recalled information is inaccurate) and self-report bias (i.e., behavioral bias in which participants over-report positive results). As a result, these biases challenge the integration of their findings towards more general mitigation strategies against project failure [52].

A second research strand relies on computational methods [4] that model the condition of project failure, from the mechanism by which delay propagates [17] to the propensity of wastefully repeating certain tasks [58]. Under this view, projects are typically modeled as directed acyclic graph, often called activity network. This activity network corresponds to a set of distinct, yet interdependent activities that need to be scheduled and sequenced under a given set of constraints [66].

Using tools from operations research, the first surge of work on project failure focused on simulating an intuitive failure scenario—project-wide delays that arise from delays in completing certain critical tasks [62]. The criticality of these tasks arises from their inclusion in the critical path, which is defined as the sequence of tasks that determines the project duration. As such, a delay of

x days in completing a critical task—and assuming that task has zero buffer in relation to its immediate successor task(s)—the project will also be delayed by a maximum of x days [17, 30]. Note the linear nature of this failure scenario, i.e., the project delay can never be higher than the sum of the individual task delays. Prominent methods for evaluating the impact of delay propagation include critical path method [36], program evaluation and review technique [42], and their Monte-Carlo variants.

Subsequent work on project failure has focused on an alternative failure scenario in which changes in task specifications can trigger rework in subsequent, downstream tasks and similarly affect the timely delivery of the overall project. In this case, a relatively minor change in the specifications of a single task can propagate across an entire project, severely affecting the overall project performance. For example, Sosa [60] provides a case where a single, minor change in the specifications of a task impacted nearly a third of all tasks within a project. Similarly, Terwiesch and Loch [63] report a case where a similar change in task specifications resulted in a 20–40% increase in the overall project cost. Additional cases have also been reported by Mihm et al. [46]. In this case, a relatively minor change in the specifications of a single task can propagate across an entire project, severely affecting the overall project performance. This asymmetry between cause and effect suggests that nonlinear effects are in place [46], which is distinct from the linear effects of delay propagation as described in the previous paragraph.

Both failures scenarios—where a delay or a change in the task specification can be the cause of a failure propagating within the project—can be understood within the broader definition of an archetypal dynamical process called ‘cascading failure’. By ‘cascading failure’, we refer to iterative processes in which a single failure leads to subsequent failures, which can amplify the impact of the original failure, eventually leading to system-wide failure [9, 68]. Such cascade dynamics have been noted in a wide range of research domains, including epidemic spreading [49, 53], social contagion [2, 3], and traffic congestion in transportation systems [64], power grid blackouts [8] and financial systemic failure [28].

Work within the project space supports the relevance of this network-oriented view. For example, recent work has made links between project performance and the number of connections between tasks [5, 14, 34], the heterogeneity by which those connections are spread across tasks [33, 61], the necessity of these connections (i.e., “non-redundant” vs. “redundant”) [7] and the variety in the nature of these connections (e.g., functional dependency, information exchange) [5, 67].

Driven by the intersection of these lines of research (i.e., cascading failures and network-oriented project view) recent studies tackled long-lasting project management challenges using failure cascades as the central modeling framework. For example, Ellinas et al. [15] assessed the propensity of a project in promoting conflicts between subcontractors by assessing the different incentives generated by their respective involvement within different cascades of failures. Building on this work, Ellinas [13] and Guo et al. [26] developed broader modeling frameworks to identify certain project network features that influence the exposure of a project to cascading failures. However, this body of work does not provide any actionable mitigation strategies by which a decision maker can contain these failure cascades. They rather

assessed the extent to which different project features could contribute to project's robustness against failure cascades.

To fill this gap, we develop a simple failure cascade model and use it to evaluate the performance of six mitigation schemes. In terms of the cascade model, we build on a popular cascade model by integrating the buffering effect of float between pairs of tasks [24]. We do so by assuming that a large free float between two consecutive tasks lowers the probability that failure in a task impacts its immediate successors. We then use this model on an empirical activity network of 723 tasks and numerically evaluate the performance of six mitigation schemes. Each mitigation scheme relies on some properties of nodes, either structural or temporal. Our overall objective is to identify which node's property can provide the most effective way for prioritizing which task(s) to be mitigated first. Our results suggest that in a majority of cases, and in contrast to current theoretical arguments, the temporal (i.e., start and end date of each task) rather than the structural properties of the activities (e.g., task connectivity) provides the most efficient way for mitigating failure cascades. This result has implications for decision makers on how to prioritize task mitigation for improving project performance.

Experimental design

Data

We use real-world project data to evaluate the performance of six mitigation strategies. The data are from a large-scale engineering project in the defence domain, were human generated by a team of professional project planners and were used throughout the lifecycle of the project to drive delivery. Specifically, the data set corresponds to a set of planned activities ($N = 723$), which we refer to as tasks, that need to be completed to deliver a commercial defence product. The overall duration of the project is 745 days. Each task has a scheduled start and end date and the resolution of time is a day. The dependency between a pair of tasks is represented by a directed edge. There are 1220 directed links in total. The directed edge from task i to j , denoted by $e_{ij} \in E$, indicates that the output of task i , such as information or a physical artifact (i.e., product), is an input to task j . A directed edge from task i

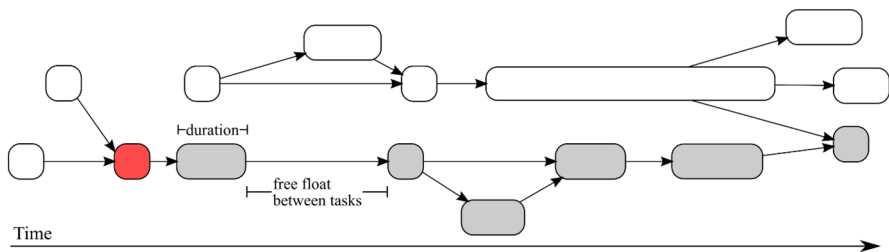


Fig. 1 Schematic of an activity network. A rounded rectangle represents a node (i.e., task). The gray rounded rectangles represent the tasks that may fail in response to a failure of the seed node shown in red

to task j implies that task i must be completed before task j starts. Therefore, task j can start only after all tasks that send a directed edge to task j have been completed. Similarly, a failure in task i may directly impact task j , and potentially all following (and reachable) tasks (see Fig. 1). The free float between task i and j is defined as the time difference between the completion τ_{ij} of task i and the start of task j [51]. The free float is equivalent to a widely used term, inter-event time [31, 32, 44]. We denote the free float between i and j as τ_{ij} .

The 723 activities (nodes) and the 1220 links define the activity network, which is a time-stamped directed acyclic graph. The number of immediate predecessors and successors of each task is equal to the task’s in-degree and out-degree, respectively. The mean in- and out-degrees of a task are equal to 1.69. The in-degree has standard deviation 4.45 and ranges from 0 to 90. A total of 111 nodes out of the 723 nodes have an in-degree of 0. Those tasks are located in the most upstream position in the network; and initiating any of these tasks does not need any other task to be completed beforehand. The out-degree has standard deviation 2.82 and ranges from 0 to 52. A total of 32 nodes have an out-degree of 0; these tasks are located in the most downstream position in the network, and failure of any of these tasks does not cause a cascading failure. The in- and out-degrees obey somewhat long-tailed distributions (Fig. 2a), as is evidenced by their relatively large standard deviations as compared to the mean. The inter-event time has the mean equal to 141.4 days, standard deviation

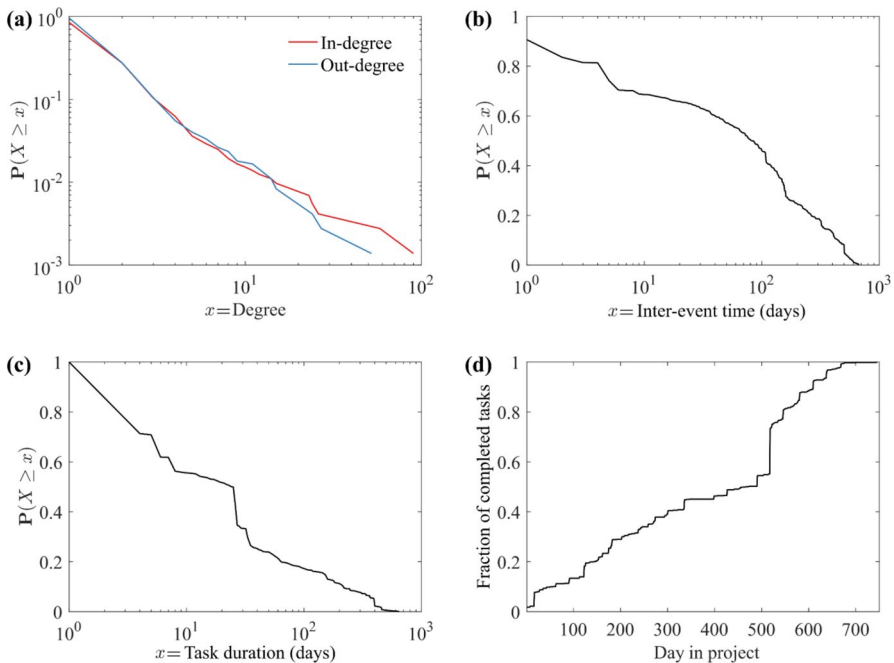


Fig. 2 Distributions of basic properties of the temporal network of tasks. **a** Survival probability (i.e., probability that the degree is larger than or equal to a specified value) of the in- and out-degrees of the node. **b** Survival probability of the inter-event time. **c** Survival probability of the task duration. **d** Fraction of tasks that have been completed by day t , plotted against t

169.5 days, and ranges from 0 to 670 days. The distribution of inter-event times is shown in Fig. 2b. The duration of task has the mean equal to 62.1 days, standard deviation 112.5 days, and ranges from 1 to 647 days. The distribution of the duration of tasks is shown in Fig. 2c. As time progresses, tasks are completed; the fraction of completed tasks by day is shown in Fig. 2d. The data set of the temporal network of tasks, including the start time and end time of each task, is provided as supplementary information and is available online (see Data availability section).

Modeling cascading failures of tasks

We introduce a discrete-time cascading failure model with binary states of the node, which is analogous to the Independent Cascade model [51] and other cascade-failure models [40, 52]. In our model, the probability that a failure propagates from an affected node i to a non-affected downstream neighbor node of node i , denoted by j , is a function of the free float between the two nodes and the values of the two parameters, as we explain in the following.

The final state of node j ($1 \leq j \leq N$) is denoted by $s_j \in [0, 1]$, where ‘0’ and ‘1’ correspond to the non-affected and affected state, respectively. We start the cascade dynamics from an initial condition, where one seed node (which can be any node) is in state 1 and all the other $N - 1$ nodes are in state 0. During the cascade dynamics, node j may irreversibly switch from state 0 to state 1 if node j has at least one upstream neighbour that is in state 1. Consequently, a node with no upstream neighbors can only be in a state of 1 if and only if it is a seed node.

We determine the final state of each node (and hence the final cascade size) by marking the nodes one by one as follows. Initially, the seed node is the only marked node (i.e., finalized to state 1) in the network. During the course of the following procedure, all nodes that are yet to be marked have state 0. Marked nodes have state either 0 or 1. In each round, we pick an unmarked node j whose all upstream neighbors have been marked. The first node to be marked after the seed node is a node that does not have any upstream neighbor (i.e., in-degree equal to 0) or a node that has the seed node as the only upstream neighbor. To determine the final state of node j (i.e., to mark node j), we assume that the failure of each upstream neighbor of node j , referred to as node i , independently causes node j to fail with probability p_{ij} . Then, we set the final state of node j to 1 with probability

$$P_j = 1 - \prod_{i: e_{ij} \in E} [(1 - s_i) + s_i(1 - p_{ij})], \quad (1)$$

where E is the set of links. Otherwise, we set the final state of node j to 0. In Eq. 1, the product term is the probability that node j does not fail, and each factor in the product is the probability that node i does not cause the failure of node j . If $s_i = 0$, this probability is equal to 1. If $s_i = 1$, this probability is equal to $1 - p_{ij}$. Once the state of node j is determined in this manner, we mark node j and select a next unmarked node such that all its upstream neighbors have been marked. Note that the results do not depend on the order of marking the nodes.

To set the value of p_{ij} , we consider the impact of time between the completion of task i and start of task j , which is called the free float in management literature and inter-event time in network science literature. We denote this quantity by τ_{ij} . We assume that the probability that the failure of node i causes the failure of node j decreases as τ_{ij} increases because a larger τ_{ij} indicates that more time is available for containing the effect of task i 's failure on its downstream neighbors [16, 39]. Reducing inter-event times has been suggested to reduce the risk of failure propagation as well [10, 29]. Therefore, we assume that

$$p_{ij} = q_0 \exp\left(-\frac{\tau_{ij}}{\tilde{\tau}}\right), \quad (2)$$

where $q_0 \in [0, 1]$ and $\tilde{\tau} (> 0)$ are parameters. Parameter q_0 is the probability that task j fails if task i does and there is no spare time (i.e., no free float) between the two tasks, i.e., $\tau_{ij} = 0$. Equation 2 indicates that if the two tasks are far apart in time, it is not likely that failure of one task triggers failure of a successor task. Parameter $\tilde{\tau}$ controls the impact of the free float, τ_{ij} , on the probability that the failure of node i causes the failure of node j . By definition, a large $\tilde{\tau}$ value yields a small probability that the failure of node i causes the failure of node j , and vice-versa.

Temporal mitigation of cascading failures

Robustness against cascading failures on networks can be engineered via structural or temporal mitigation schemes. Structural mitigation can be deployed when the structure of the network can be changed. For example, in power grids, one can modify the network structure to discourage the onset of large-scale cascades, e.g., by introducing network modules or purposefully fragmenting the network before a cascade happens [27]. However, some networks that are susceptible to cascading failures may not accommodate structural mitigation. In this situation, temporal mitigation, i.e., changing the timing of nodes or links without changing the static network structure, may be deployed without compromising the function of the system. In general, a temporal mitigation scheme can be implemented if nodes or links have timestamps that are relatively flexible. For example, in air traffic networks where nodes and time-stamped links are airports and flights, respectively, delaying flights is probably more feasible than changing the destination of the flights as a preventive measure against cascading failures [1, 20]. Similarly, in project management context, deploying structural mitigation in activity networks is not often practical because a directed edge from task i (e.g., designing a structural column for a building) to task j (e.g., manufacturing that column) indicates that task i 's output is necessary for starting task j , and therefore cannot be amended.

By the construction of our cascade model, increasing an inter-event time is a viable mechanism to reduce the probability that failure propagates from a task to another. Therefore, by postponing the start of a downstream task j , we reduce the probability of it being affected by a failure in its predecessor, task i . We utilize this mechanism to construct mitigation schemes, where we postpone some of the tasks located downstream to the seed node that has failed. Doing so increases some of the

inter-event times in the nodes belonging to the out-component of the seed node (i.e., the nodes downstream to the seed node). Therefore, a mitigation scheme is expected to reduce the overall probability that the failure propagates.

A mitigation scheme has to respect the end date of the entire project; no task can be postponed beyond the delivery date of the entire project. Furthermore, any downstream neighbor of task j is only allowed to start after task j has been completed. Therefore, the extent of postponing task j is further constrained by the start date of its downstream neighbours. Note that we allow the end date of task j to coincide with the start date of its downstream neighbor, in which case the inter-event time is equal to zero.

Precisely, the fraction of the nodes in the out-component of the seed node for which we postpone the start time is denoted by $\gamma \in [0, 1]$. For each mitigation scheme, we first consider the ranking of nodes in the out-component of the seed node. We then sequentially postpone a fraction γ of these nodes in descending order of the rank. When postponing each task i sequentially, we postpone it as much as possible under the following two conditions. First, adjacent tasks must not overlap [37]. In other words, the end date of task i must not exceed the start date of any task j that needs completion of task i . Second, the overall project duration must not be extended. In other words, the end date of task i must not exceed the original delivery date of the project.

We test six mitigation schemes, in which nodes to be mitigated are ranked based on either the (i) out-degree, (ii) size of out-component (i.e., the number of nodes that are reachable from the node in question), (iii) duration of the task, (iv) start date of the task, (v) end date of the task or (vi) at random. For example, consider the network shown in the upper part of Fig. 3a and assume that node v_1 fails. The subscript attached to the nodes in the figure represents the ranking in terms of the out-degree. The figure indicates that node v_3 is the first node to be mitigated (i.e., postponed). The amount of maximum postponement that can be applied to node v_3 is constrained by the start date of its immediate neighbor, node v_4 . Therefore, we postpone node v_3 such that its new end date is equal to the start date of node v_4 (the network shown in the lower part of Fig. 3a). Similarly, node v_5 is postponed such that its new end date is equal to the start date of node v_6 . The same procedure is applied to node v_2 and then to node v_6 . Note that postponing node v_5 makes the inter-event time between node v_5 and node v_6 equal to zero. However, postponing node v_6 subsequently increases the same inter-event time. We do not postpone the remaining two tasks with the lowest out-degrees, i.e., v_4 and v_7 , because the fraction of the mitigated nodes, denoted by γ , is set to 0.67 for illustration purposes, such that only four out of the six nodes downstream to node v_1 can be mitigated. Implementation of three other mitigation schemes on the same network and the same γ value is schematically shown in Fig. 3b–d.

The mitigation scheme is implemented as follows. Once seed node i fails, all nodes reachable from node i along a directed path (i.e., nodes belonging to the out-component of node i), which can fail, are rank ordered based on the node's score. The score of these nodes is equal to one of the following six quantities: out-degree, size of the out-component (i.e., the number of nodes that are reachable from the node to be scored), duration of the task, start date of the task, end date of the task,

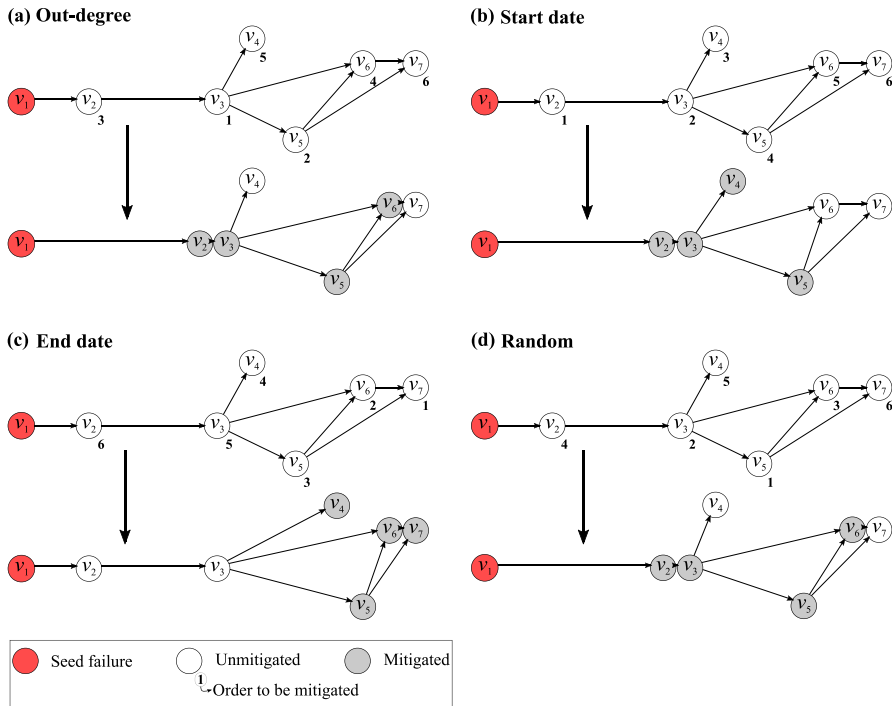


Fig. 3 An example illustrating the four mitigation schemes: **a** out-degree, **b** start date, **c** end date and **d** random. For each mitigation scheme, the top and bottom panels correspond to before and after the mitigation, respectively. Every node is ranked (subscript index) and postponed in that order. The tie is broken uniformly randomly. In all examples, we set $\gamma = 0.67$ such that four out of the six tasks are mitigated

or an entirely randomly drawn value. When multiple nodes have identical scores, we break the tie by ranking the nodes having the same score in a uniformly random order.

We denote by \tilde{V} the rank-ordered set of the nodes downstream to node i . In the example shown in Fig. 3a, in which the rank is determined according to the out-degree of the task, we obtain $\tilde{V} = \{v_3, v_5, v_2, v_6, v_4, v_7\}$. Parameter $\gamma \in [0, 1]$ specifies the fraction of nodes in \tilde{V} that are to be mitigated. In Fig. 3a, we set $\gamma = 0.67$. Therefore, the four highest-ranked nodes out of the six nodes, i.e., v_3, v_5, v_2 and v_6 , are mitigated. Node v_3 is first postponed until its end date coincides with the start date of its downstream neighbor v_4 . Next, the same postponement process is applied to node v_5 , node v_2 and then node v_6 . The temporal network after the mitigation is shown in the lower part of Fig. 3a.

Performance measures for mitigation schemes, R_1 and R_2

We evaluate the performance of the six mitigation schemes in terms of their ability of containing cascading failures. These mitigation schemes attempt to increase τ_{ij} for some i and j to reduce the probability that a failure cascade progresses. Our focus is

on the impact of the parameters that control the cascading dynamics (q_0 and $\bar{\tau}$) and the fraction of the tasks to be postponed (γ).

We measure the performance of each mitigation scheme in terms of two quantities. The first quantity, denoted by R_1 , is defined as the cascade size that stems from a seed node when the mitigation scheme is implemented, divided by the cascade size when there is no mitigation, averaged over all seed nodes. Quantity R_1 captures the relative impact of mitigation in the sense that the contribution of mitigating a large cascade is equivalent to that of mitigating a small cascade. The second quantity, denoted by R_2 , is defined as the cascade size averaged over all seed nodes when the mitigation is applied, which is then divided by the cascade size averaged over all seed nodes when no mitigation is applied. Quantity R_2 captures the absolute impact of mitigation in the sense that mitigating a large cascade is considered to be more valuable than mitigating a small cascade. A small R_1 or R_2 value indicates that the mitigation scheme is efficient.

For the given values of q_0 , $\bar{\tau}$, γ , and the given seed node, we ran the cascading dynamics 100 times (except for Supplementary Fig. 1, for which we ran the simulation 300 times). In the figures, we show the average values of the observables over all runs.

Results

We first focus on the unmitigated failure cascades to understand the effect of the free parameters of our model q_0 and $\bar{\tau}$ on the impact of failure. By definition, higher values of q_0 and $\bar{\tau}$ increase the probability of a single activity failing, resulting in a larger average cascade size (Fig. 4a). In addition, a higher value of $\bar{\tau}$ increases both the average cascade size and the probability of encountering a cascade of a given size (Fig. 4b). Activity networks are prone to large failure cascades, where the failure of a single activity can impact a disproportionately large number of subsequent activities. The heavy-tailed distribution of all cascade sizes highlights the disproportionate nature of these cascades, with a majority of failure cascades impacting a

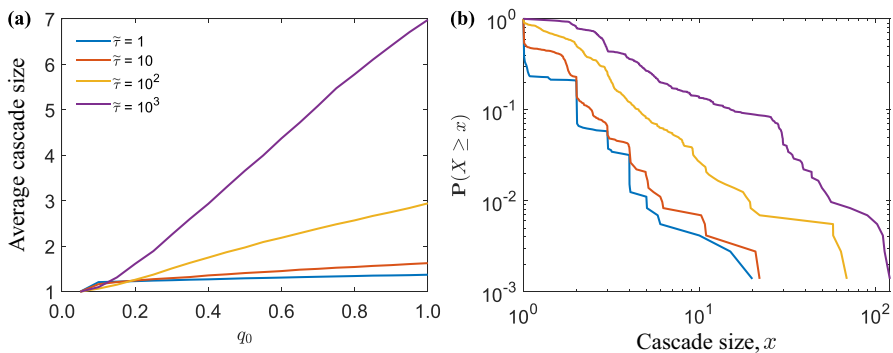


Fig. 4 **a** Average cascade size as a function of q_0 for four values of $\bar{\tau}$. **b** Survival probability of observing a cascade of size x , where we set $q_0 = 1$ (i.e., worst case scenario)

small number of tasks and a small number of failure cascades impacting many tasks. For example, when the probability that task j fails if task i does and there is no spare time between them (i.e., $q_0 = 1$) and time dependence is low (i.e., $\tilde{\tau} = 10^3$), the average cascade size is ~ 7 (Fig. 4a), whilst the largest cascade size is over 100 (Fig. 4b).

In the case where we mitigate all downstream tasks (i.e., $\gamma = 1$), the mitigation scheme based on the end date of the task outperforms the other five mitigation schemes. This is the case in terms of both performance measures R_1 (Fig. 5) and R_2 (Fig. 6). Quantities R_1 and R_2 measure the relative and absolute reduction in the cascade size by a mitigation scheme. These figures also show that apart from the mitigation scheme based on the end date of the task, the random mitigation scheme outperforms the other four mitigation schemes. The relative ranking of the six mitigation schemes is consistent in the whole range of q_0 and $\tilde{\tau} \in \{1, 10, 10^2, 10^3\}$, except for $\tilde{\tau} = 10^3$, where there are some rank changes presumably due to random fluctuations. Note that as $\tilde{\tau}$ tends large ($\tilde{\tau} \geq 10^3$), p_{ij} is approximately equal to q_0 regardless of the size of τ_{ij} and regardless of the mitigation scheme. Therefore, R_1 and R_2 converge to 1 for any q_0 as $\tilde{\tau}$ increases (see Supplementary Fig. 1 for numerical results with $\tilde{\tau} = 10^4$ and $\tilde{\tau} = 10^5$).

To investigate the entire parameter space, we identified the mitigation scheme that was the most efficient, i.e., yielding the smallest value of R_1 and R_2 , when we varied q_0 , $\tilde{\tau}$ and γ . The results in terms of R_1 are shown in Fig. 7. When there is little variation between the best and worst performing schemes ($< 1\%$; arbitrarily chosen; white

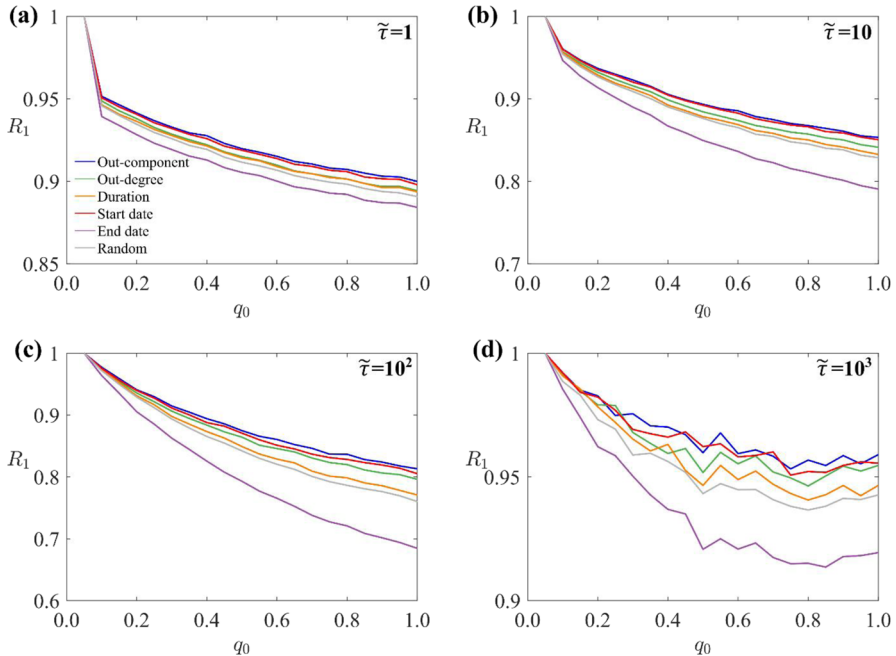


Fig. 5 Performance of the six mitigation schemes in terms of R_1 , as a function of q_0 . **a** $\tilde{\tau} = 1$. **b** $\tilde{\tau} = 10$. **c** $\tilde{\tau} = 10^2$. **d** $\tilde{\tau} = 10^3$. We set $\gamma = 1$

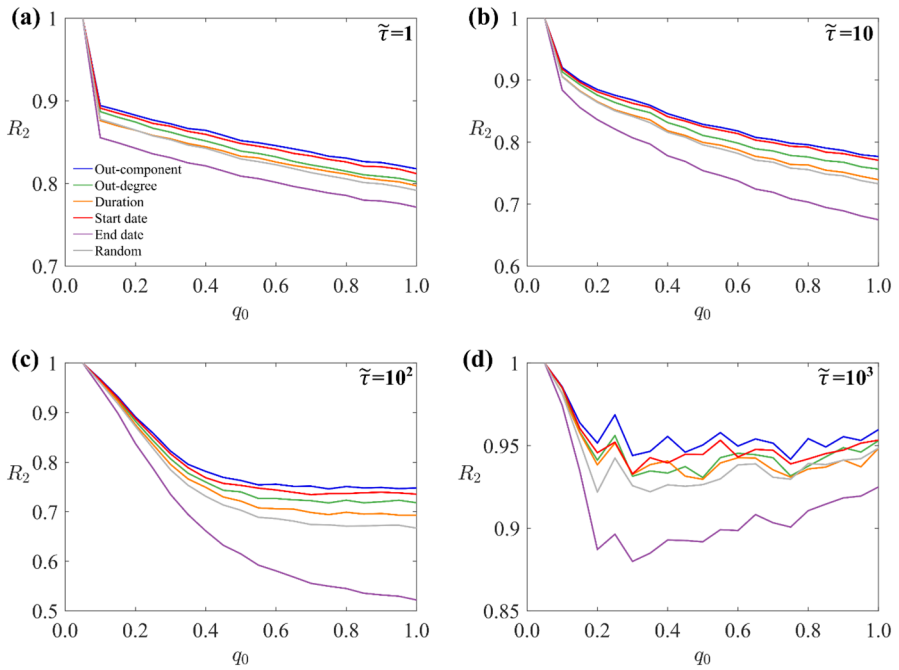


Fig. 6 Performance of the six mitigation schemes in terms of R_2 , as a function of q_0 . **a** $\tilde{\tau} = 1$. **b** $\tilde{\tau} = 10$. **c** $\tilde{\tau} = 10^2$. **d** $\tilde{\tau} = 10^3$. We set $\gamma = 1$

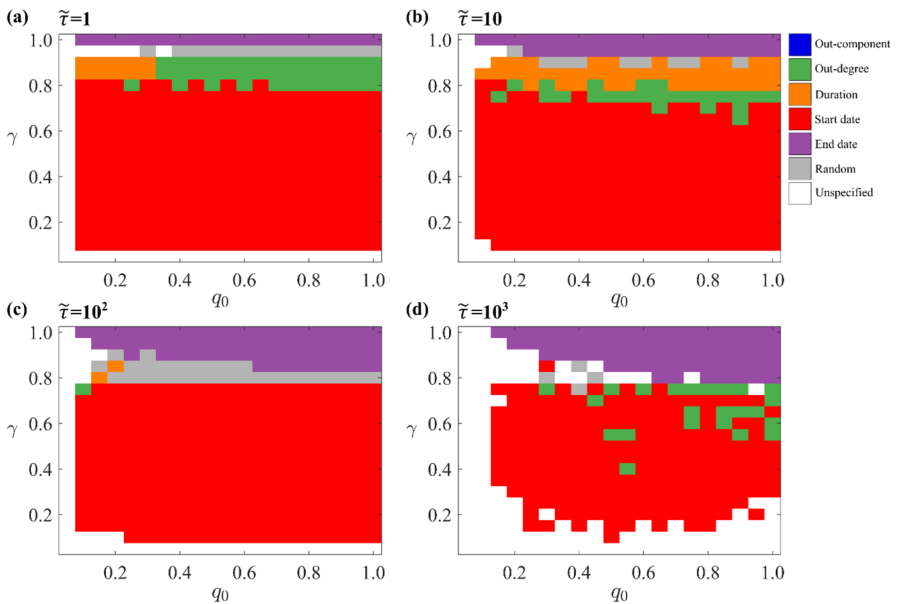


Fig. 7 Best performing mitigation scheme in terms of R_1 in the parameter space spanned by q_0 and γ . **a** $\tilde{\tau} = 1$. **b** $\tilde{\tau} = 10$. **c** $\tilde{\tau} = 10^2$. **d** $\tilde{\tau} = 10^3$

regions labeled ‘Unspecified’ in Fig. 7), we argue that no best mitigation scheme exists. Figure 7 reveals two parameter regimes. First, when $\gamma \geq 0.8$, the mitigation scheme based on either the out-degree, duration, end date or at random performs the best, depending on the specific combination of γ and q_0 values. As $\tilde{\tau}$ increases from 1 to 10^3 , the mitigation scheme based on the end date tends to be consistently the best in this parameter regime (Fig. 7d). Second, when $\gamma < 0.8$, the mitigation scheme based on the start date tends to be the best performing mitigation scheme across the entire range of q_0 and $\tilde{\tau}$. The results in terms of R_2 (Supplementary Fig. 2) are similar to those in terms of R_1 (Fig. 7).

Discussion

We modeled project failures as cascading failures on networks composed of tasks constituting the project. The model incorporates both structural and temporal features of activity networks of projects. We implemented six mitigation schemes by postponing a fraction γ of tasks downstream to the task that has failed. When one was allowed to postpone all the tasks downstream to the task that has failed, our numerical results indicated that it was more efficient to prioritize task mitigation according to the end date of each task than the other five mitigation schemes. When one was allowed to postpone a relatively small fraction of tasks, it was generally more efficient to postpone tasks based on their start date. Some additional cases existed where the mitigation scheme based on the out-degree or duration of the task was the most efficient. Specifically, when γ is large, either the mitigation scheme based on the out-degree, that based on the duration, or that based on the end date was the best. These numerical results suggest that, in a majority of the parameter region that we have explored, temporal features of the tasks, such as the duration, start and end date of the task, may be more important than structural features, such as the out-degree of the task, for preventing large-scale cascading failures of projects.

The present results suggest that the importance of tasks should not only be ranked based on the impact (i.e., the size of the cascade failure) that the failure of a single task can cause (e.g., project delay) but also based on the extent to which the impact can be mitigated. We provided proxies for identifying important tasks in the sense of mitigation using a task’s start and end date. Using these proxies, decision makers can focus on proactively managing these tasks. One way for incorporating this in practice is to relax the typically strict constraints in terms of start (and end) date for some tasks by, for example, removing monetary penalties for such delays [19]. Relaxing such penalties can introduce planning flexibility for decision makers to purposefully postpone certain tasks, reducing the overall exposure of the project to cascading failures.

This work is aligned with other, domain-specific research strands that focus on the broader objective of improving project performance. For example, Eppinger et al. [18] focused on modular and decomposable projects, where links between activities can be modified to some extent [18]. By doing so, the authors were able to re-sequence certain activities to reduce risk and improve overall project performance

[18]. Based on this idea, Baldwin and Clark [6] deployed task re-sequencing for optimizing project modularity, which was argued to be able to reduce project risk [6]. In these scenarios, structural mitigation is deployed because there are no hard constraints assumed on the links between activities, so that the links can be rewired. In our case, we assumed temporal mitigation due to the constraint that the network structure, which represents precise inter-dependency among the tasks, is not allowed to be changed. Despite the contextual differences, their work and ours have the common thread of utilizing network structure and complex systems thinking for improving project performance, which we believe is a promising research direction.

Our modeling framework has some limitations. First, our analysis has focused only on the benefits of deploying mitigation in the form of postponing the start date of tasks. However, postponing tasks may increase the number of active tasks on particular days, which is generally associated with poor project performance due to an increased cost or decreased quality [11, 37, 58]. In addition, postponing tasks is impossible when a sequence of tasks has no float between any consecutive pair of tasks, which is a type of critical path [54]. Future work should consider this drawback in conjunction with the benefits potentially gained through the mitigation mechanisms proposed in the present study. Second, from a methodological standpoint, our approach is limited by the single pass in which mitigation is applied to tasks. Consider the example shown in Fig. 3a, in which node v_5 was postponed before node v_6 was. In this case, the amount of postponement is constrained by the start date of node v_6 . Postponing node v_6 at a later stage opens up the opportunity for node v_5 to be further postponed, which is currently not exploited. One can exploit this opportunity to explore further improvements in mitigation efficiency. Third, a mitigation scheme can be classified into passive and active. In a passive mitigation scheme, one modifies the structure or time stamps of the activity network before a cascade is possibly seeded. In contrast, in an active mitigation scheme, one modifies the activity network while a cascade is progressing. In the present study, we focused on active mitigation schemes. Carefully planning the start time of each task, given, for example, the network structure and the possibility of different tasks to fail with different probabilities, may consist in a plausible passive mitigation scheme on the activity network. This topic also warrants future work. Fourth, we do not know have mechanistic understanding of why one mitigation scheme works better than another. To clarify this requires a more systematic investigation, possibly involving multiple data sets, which is beyond the scope of the present study.

In addition, our methodology hints to the possibility of mitigation having unintended negative effects. In the example shown in Fig. 3a, the probability that the failure of node v_1 propagates to node v_2 has been reduced because $\tau_{v_1 v_2}$ has been increased. This is a positive effect of mitigation that we have intended. However, in the same example, the probability that the failure of node v_2 propagates to node v_3 has been increased compared to the case of the unmitigated activity network, because the mitigation has decreased $\tau_{v_2 v_3}$. This is a negative effect of mitigation that we have not intended.

Despite these and other possible limitations, the present modeling framework serves as a stepping stone for future work. It opens new pathways of exploring

whether causal relationships exist between structural and/or temporal features of temporal networks of tasks and mitigation effectiveness and efficiency.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s42001-021-00123-x>.

Data availability The datasets generated during and/or analyzed during the current study are available in the GitHub repository https://github.com/naokimas/project_network.

Declarations

Conflict of interest The authors declare no competing interests. CE is the co-founder of Nodes & Links. Nodes & Links provided support in the form of salaries for CE, but did not have any additional role in the study design, analysis, decision to publish, or preparation of the manuscript.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. AhmadBeygi, S., Cohn, A., Guan, Y., & Belobaba, P. (2008). Analysis of the potential for delay propagation in passenger airline networks. *Journal of Air Transport Management*, 14(5), 221–236. <https://doi.org/10.1016/j.jairtraman.2008.04.010>
2. Aral, S., & Nicolaides, C. (2017). Exercise contagion in a global social network. *Nature Communications*, 8, 14753. <https://doi.org/10.1038/ncomms14753>
3. Aral, S., & Walker, D. (2011). Viral product design: A randomized trial of peer influence in networks. *Management Science*, 57(9), 1623–1639. <https://doi.org/10.1287/mnsc.1110.1421>
4. Ashworth, M. J., & Carley, K. M. (2007). Can tools help unify organization theory? Perspectives on the state of computational modeling. *Computational and Mathematical Organization Theory*, 13(1), 89–111. <https://doi.org/10.1007/s10588-006-9000-9>
5. Baccarini, D. (1996). The concept of project complexity—a review. *International Journal of Project Management*, 14(4), 201–204. [https://doi.org/10.1016/0263-7863\(95\)00093-3](https://doi.org/10.1016/0263-7863(95)00093-3)
6. Baldwin, C. Y., & Clark, K. B. (2006). Modularity in the design of complex engineering systems. In: D. Braha, A. Minai, & Y. Bar-Yam (Eds.), *Complex Engineered Systems* (pp. 175–205). Springer, Berlin, Heidelberg: Understanding Complex Systems. https://doi.org/10.1007/3-540-32834-3_9
7. Browning, T. R., & Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, 126(2), 212–228. <https://doi.org/10.1016/j.ijpe.2010.03.009>
8. Brummitt, C. D., D'Souza, R. M., & Leicht, E. A. (2012). Suppressing cascades of load in interdependent networks. *Proceedings of the National Academy of Sciences*, 109(12), 680–689. <https://doi.org/10.1073/pnas.1110586109>
9. Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E., & Havlin, S. (2010). Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291), 1025–1028. <https://doi.org/10.1038/nature08932>

10. Buzna, L., Peters, K., Ammoser, H., Kühnert, C., & Helbing, D. (2007). Efficient response to cascading disaster spreading. *Physical Review E*, 75(5), 056107. <https://doi.org/10.1103/PhysRevE.75.056107>
11. Cho, S. H., & Eppinger, S. D. (2005). A simulation-based process model for managing complex design projects. *IEEE Transactions on Engineering Management*, 52(3), 316–328. <https://doi.org/10.1109/TEM.2005.850722>
12. Drezner, J. A., Jarvaise, J. M., Hess, R. W., Norton, D. M., & Hough, P. G. (1993). *An analysis of weapon system cost growth*, (RAND Corporation MR-291-AF). Santa Monica, CA: RAND Corporation.
13. Ellinas, C. (2019). The domino effect: An empirical exposition of systemic risk across project networks. *Production and Operations Management*, 28, 63–81. <https://doi.org/10.1111/poms.12890>
14. Ellinas, C., Allan, N., & Johansson, A. (2018). Towards project complexity evaluation: a structural perspective. *IEEE Systems Journal*, 12, 228–239. <https://doi.org/10.1109/JSYST.2016.2562358>
15. Ellinas, C., Allan, N., & Johansson, A. (2016). Project systemic risk: Application examples of a network model. *International Journal of Production Economics*, 182, 50–62. <https://doi.org/10.1016/j.ijpe.2016.08.011>
16. Ellinas, C., Allan, N., Durugbo, C., & Johansson, A. (2015). How robust is your project? From local failures to global catastrophes: A complex networks approach to project systemic risk. *PLoS ONE*, 10, e0142469. <https://doi.org/10.1371/journal.pone.0142469>
17. Elmaghraby, S. E. (1995). Activity nets: A guided tour through some recent developments. *European Journal of Operational Research*, 82(3), 383–408. [https://doi.org/10.1016/0377-2217\(94\)00184-e](https://doi.org/10.1016/0377-2217(94)00184-e)
18. Eppinger, S. D., Whitney, D. E., Smith, R. P., & Gebala, D. A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1), 1–13. <https://doi.org/10.1007/BF01588087>
19. Estévez-Fernández, A. (2012). A game theoretical approach to sharing penalties and rewards in projects. *European Journal of Operational Research*, 216(3), 647–657. <https://doi.org/10.1016/j.ejor.2011.08.015>
20. Fleurquin, P., Ramasco, J. J., & Eguiluz, V. M. (2013). Systemic delay propagation in the US airport network. *Scientific Reports*, 3, 1159. <https://doi.org/10.1038/srep01159>
21. Flyvbjerg, B. (2014). What you should know about megaprojects and why: An overview. *Project Management Journal*, 45(2), 6–19. <https://doi.org/10.1002/pm.j.21409>
22. Flyvbjerg, B., & Budzier, A. (2011). Why your it project may be riskier than you think. *Harvard Business Review*, 89, 601–603.
23. Flyvbjerg, B., Holm, M. K. S., & Buhl, S. L. (2003). How common and how large are cost overruns in transport infrastructure projects? *Transport Reviews*, 23(1), 71–88. <https://doi.org/10.1080/01441640309904>
24. Goldenberg, J., Libai, B., & Muller, E. (2001). Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3), 211–223. <https://doi.org/10.1023/A:1011122126881>
25. Grimsey, D., & Lewis, M. K. (2002). Evaluating the risks of public private partnerships for infrastructure projects. *International Journal of Project Management*, 20(2), 107–118. [https://doi.org/10.1016/S0263-7863\(00\)00040-5](https://doi.org/10.1016/S0263-7863(00)00040-5)
26. Guo, N., Guo, P., Dong, H., Zhao, J., & Han, Q. (2019). Modeling and analysis of cascading failures in projects: A complex network approach. *Computers and Industrial Engineering*, 127, 1–7. <https://doi.org/10.1016/j.cie.2018.11.051>
27. Guo, H., Zheng, C., Iu, H. H. C., & Fernando, T. (2017). A critical review of cascading failure analysis and modeling of power system. *Renewable and Sustainable Energy Reviews*, 80, 9–22. <https://doi.org/10.1016/j.rser.2017.05.206>
28. Haldane, A. G., & May, R. M. (2011). Systemic risk in banking ecosystems. *Nature*, 469(7330), 351–355. <https://doi.org/10.1038/nature09659>
29. Helbing, D. (2013). Globally networked risks and how to respond. *Nature*, 497(7447), 51–59. <https://doi.org/10.1038/nature12047>
30. Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research*, 25(4), 279–302. [https://doi.org/10.1016/S0305-0548\(97\)00055-5](https://doi.org/10.1016/S0305-0548(97)00055-5)

31. Holme, P. (2015). Modern temporal network theory: A colloquium. *European Physical Journal B*, 88, 234. <https://doi.org/10.1140/epjb/e2015-60657-4>
32. Holme, P., & Saramäki, J. (2012). Temporal networks. *Physics Reports*, 519(3), 97–125. <https://doi.org/10.1016/j.physrep.2012.03.001>
33. Homer-Dixon, T. (2010). *The ingenuity gap: Can we solve the problems of the future?* Vintage, Canada.
34. Jacobs, M. A., & Swink, M. (2011). Product portfolio architectural complexity and operational performance: Incorporating the roles of learning and fixed assets. *Journal of Operations Management*, 29(7–8), 677–691. <https://doi.org/10.1016/j.jom.2011.03.002>
35. Jensen, A., Thuesen, C., & Geraldi, J. (2016). The projectification of everything: Projects as a human condition. *Project Management Journal*, 47(3), 21–34. <https://doi.org/10.1177/875697281604700303>
36. Kelley, J. E., & Walker, M. R. (1959). Critical-path planning and scheduling. Proceedings of Eastern Joint IRE-AIEE-ACM Computer Conference, 160–173. <https://doi.org/10.1145/1460299.1460318>
37. Krishnan, V., Eppinger, S. D., & Whitney, D. E. (2008). A model-based framework to overlap product development activities. *Management Science*, 43(4), 437–451. <https://doi.org/10.1287/mnsc.43.4.437>
38. Kusumasari, B., Alam, Q., & Siddiqui, K. (2010). Resource capability for local government in managing disaster. *Disaster Prevention and Management*, 19(4), 438–451. <https://doi.org/10.1108/09653561011070367>
39. Lawson, M. B. (2001). In praise of slack: time is of the essence. *Academy of Management Perspectives*, 15(3), 125–135. <https://doi.org/10.5465/ame.2001.5229658>
40. Lorenz, J., Battiston, S., & Schweitzer, F. (2009). Systemic risk in a unifying framework for cascading processes on networks. *European Physical Journal B*, 71, 441–460. <https://doi.org/10.1140/epjb/e2009-00347-4>
41. Love, P. E. D., Davis, P. R., Chevis, R., & Edwards, D. J. (2010). Risk/reward compensation model for civil engineering infrastructure alliance projects. *Journal of Construction Engineering and Management*, 137(2), 127–136. [https://doi.org/10.1061/\(asce\)co.1943-7862.0000263](https://doi.org/10.1061/(asce)co.1943-7862.0000263)
42. Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations Research*, 7(5), 646–669. <https://doi.org/10.1287/opre.7.5.646>
43. Mansfield, N. R., Ugwu, O. O., & Doran, T. (1994). Causes of delay and cost overruns in Nigerian construction projects. *International Journal of Project Management*, 12(4), 254–260. [https://doi.org/10.1016/0263-7863\(94\)90050-7](https://doi.org/10.1016/0263-7863(94)90050-7)
44. Masuda, N., & Lambiotte, R. (2016). *A guide to temporal networks*. World Scientific.
45. Matta, N. F., & Ashkenas, R. N. (2003). Why good projects fail anyway. *Harvard Business Review*, 81(9), 109–116.
46. Mihm, J., Loch, C., & Huchzermeier, A. (2003). Problem-solving oscillations in complex engineering projects. *Management Science*, 49(6), 733–750. <https://doi.org/10.1287/mnsc.49.6.733.16021>
47. Moe, T. L., & Pathranarakul, P. (2006). An integrated approach to natural disaster management: public project management and its critical success factors. *Disaster Prevention and Management*, 15(3), 396–413. <https://doi.org/10.1108/09653560610669882>
48. Moløkken, K., & Jørgensen, M. (2003). A review of software surveys on software effort estimation. In: 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings, pp. 223–230. <https://doi.org/10.1109/ISESE.2003.1237981>
49. Nicolaides, C., Cueto-Felgueroso, L., González, M. C., & Juanes, R. (2012). A metric of influential spreading during contagion dynamics through the air transportation network. *PLoS ONE*, 7(7), e40961. <https://doi.org/10.1371/journal.pone.0040961>
50. Nieto-Rodríguez, A., & Evrard, D. (2004). *Boosting Business Performance through Programme and Project Management*. London, UK: PriceWaterhouseCoopers.
51. Pagani, G. A., & Aiello, M. (2013). The power grid as a complex network: A survey. *Physica A*, 392(11), 2688–2700. <https://doi.org/10.1016/j.physa.2013.01.023>
52. Parvan, K., Rahmandad, H., & Haghani, A. (2015). Inter-phase feedbacks in construction projects. *Journal of Operations Management*, 40(1), 48–62. <https://doi.org/10.1016/j.jom.2015.07.005>
53. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., & Vespignani, A. (2015). Epidemic processes in complex networks. *Reviews of Modern Physics*, 87(3), 925. <https://doi.org/10.1103/RevModPhys.87.925>

54. Project Management Institute (2017). Project management body of knowledge: A guide to the project management body of knowledge. In Newtown Square, Pennsylvania: Project Management Institute.
55. Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. *Journal of Management Information Systems*, 17(4), 5–36. <https://doi.org/10.1080/07421222.2001.11045662>
56. Scott-Young, C., & Samson, D. (2008). Project success and project team management: Evidence from capital projects in the process industries. *Journal of Operations Management*, 26(6), 749–766. <https://doi.org/10.1016/j.jom.2007.10.006>
57. Scranton, P. (2014). Projects as a focus for historical analysis: Surveying the landscape. *History and Technology*, 30(4), 354–373. <https://doi.org/10.1080/07341512.2014.1003164>
58. Smith, R. P., & Eppinger, S. D. (1997). Identifying controlling features of engineering design iteration. *Management Science*, 43(3), 276–293. <https://doi.org/10.1287/mnsc.43.3.276>
59. Snowden, D. J., & Boone, M. E. (2007). A leader's framework for decision making. *Harvard Business Review*, 85, 68–76.
60. Sosa, M. E. (2014). Realizing the need for rework: From task interdependence to social networks. *Production and Operations Management*, 23(8), 1312–1331. <https://doi.org/10.1111/poms.12005>
61. Sosa, M., Mihm, J., & Browning, T. (2011). Degree distribution and quality in complex engineered systems. *Journal of Mechanical Design*, 133(10), 101008. <https://doi.org/10.1115/1.4004973>
62. Tavares, L. V. (2002). A review of the contribution of operational research to project management. *European Journal of Operational Research*, 136(1), 1–18. [https://doi.org/10.1016/S0377-2217\(01\)00097-2](https://doi.org/10.1016/S0377-2217(01)00097-2)
63. Terwiesch, C., & Loch, C. H. (1999). Managing the process of engineering change orders: The case of the climate control system in automobile development. *Journal of Product Innovation Management*, 16(2), 160–172. [https://doi.org/10.1016/S0737-6782\(98\)00041-1](https://doi.org/10.1016/S0737-6782(98)00041-1)
64. Toole, J. L., Colak, S., Sturt, B., Alexander, L. P., Evsukoff, A., & González, M. C. (2015). The path most traveled: Travel demand estimation using big data resources. *Transportation Research Part C*, 58, 162–177. <https://doi.org/10.1016/j.trc.2015.04.022>
65. Turner, J. R., & Müller, R. (2005). The project manager's leadership style as a success factor on projects: A literature review. *Project Management Journal*, 36(2), 49–61. <https://doi.org/10.1177/875697280503600206>
66. Valadares Tavares, L., & Weglarz, J. (1990). Project management and scheduling: A permanent challenge for OR. *European Journal of Operational Research*, 49(1), 1–2. [https://doi.org/10.1016/0377-2217\(90\)90115-R](https://doi.org/10.1016/0377-2217(90)90115-R)
67. Vidal, L. A., Marle, F., & Bocquet, J. C. (2011). Measuring project complexity using the Analytic Hierarchy Process. *International Journal of Project Management*, 29(6), 718–727. <https://doi.org/10.1016/j.ijproman.2010.07.005>
68. Watts, D. J. (2002). A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(9), 5766–5771. <https://doi.org/10.1073/pnas.082090499>
69. Williams, T. (1995). A classified bibliography of recent research relating to project risk management. *European Journal of Operational Research*, 85(1), 18–38. [https://doi.org/10.1016/0377-2217\(93\)E0363-3](https://doi.org/10.1016/0377-2217(93)E0363-3)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.