American Journal of Applied Mathematics
2025; 13(5): 339-343
http://www.sciencepublishinggroup.com/j/ajam
doi: 10.11648/j.ajam.20251305.13

Efficient Algorithms for Critical Node Detection on Path Graphs with Binary Connection Metrics

Syed Md Omar Faruk*, Kamrun Nahar Jabin

Department of Mathematics, Shahjalal University of Science and Technology, Sylhet, Bangladesh

Email address:
omarfaruk-mat@sust.edu (Syed Md Omar Faruk), arko.sust@gmail.com (Kamrun Nahar Jabin)
*Corresponding author

To cite this article:
Syed Md Omar Faruk, Kamrun Nahar Jabin. (2025). Efficient Algorithms for Critical Node Detection on Path Graphs with Binary Connection Metrics. *American Journal of Applied Mathematics, 13*(5), 339-343. https://doi.org/10.11648/j.ajam.20251305.13

Received: 9 August 2025; **Accepted:** 18 August 2025; **Published:** 25 September 2025

Abstract: This paper investigates the Critical Node Detection Problem (CNDP) on path graphs where connection costs between node pairs are binary (0 or 1). Two variants of the problem are explored based on whether node weights are uniform or arbitrary. For both cases, the objective is to identify a subset of nodes whose removal minimizes the number of important connections surviving. Efficient dynamic programming algorithms are proposed to solve these problems optimally. When node weights are uniform, the approach runs in $\mathcal{O}(n^3K)$ time, where n is the number of nodes and K is the maximum number of deletions allowed. For arbitrary node weights with a total removal budget W, the solution is derived in $\mathcal{O}(n^5)$ time.

Keywords: Critical Node Detection, Dynamic Programming, Polynomial Time Algorithm

1. Introduction

The analysis of complex networks has become central in a variety of disciplines such as cybersecurity, transportation, biological systems, and social media. A fundamental question in network science is how to identify a set of critical nodes whose removal significantly degrades the overall connectivity or functionality of the network. This problem, formally known as the Critical Node Detection Problem (CNDP), has been studied extensively under various connectivity measures and graph classes. Most existing literature focuses on the problem's complexity and solution strategies in general graphs or specific structures such as trees, where CNDP is known to be NP-hard even with unit node weights and binary connection costs.

In this work, we study the Critical Node Detection Problem on a path, which can be seen as a very special case of a tree [9]. One characterization of a path is that a path is a tree with two leaves. Path graphs, due to their linear and structured topology, serve as fundamental models in analyzing the Critical Node Detection Problem across various real-world domains. In communication and telecommunication networks, where nodes such as routers or switches are often arranged in linear topologies (e.g., fiber-optic lines or chain-connected routers), path graphs help identify critical points whose removal severely degrades connectivity or service quality [16]. Similarly, in transportation systems like metro lines, highways, or railways, CNDP on path graphs reveals key stations or intersections whose failure can cause significant disruption [17]. Power transmission infrastructures, particularly in rural or long-distance scenarios, can also be modeled as path graphs, where CNDP assists in pinpointing vital substations or transformers to ensure grid reliability and resilience [3]. In biological systems, path graphs represent linear chains of neurons, genes, or proteins, enabling CNDP to locate components critical to maintaining functional integrity in neurological or molecular networks [5]. Moreover, supply chains with sequential stages-ranging from raw materials to retailers-benefit from CNDP on path graphs for identifying vulnerable suppliers or intermediaries [11]. The same applies to distributed computing, where sequential processes and message-passing chains can be analyzed to detect failure-prone components [12]. Even in education, linear learning paths modeled as path graphs allow for the identification of foundational modules essential to curriculum continuity [14]. The simplicity of path graphs not only allows for efficient algorithmic solutions [10], but also

provides a meaningful abstraction in diverse application areas where the detection of structurally or functionally critical nodes is paramount. Despite their structural simplicity, path graphs allow polynomial-time solutions for various CNDP variants [4], making them not only useful for direct applications but also as a baseline for approximating solutions in more complex topologies.

Connectivity measures are central to the formulation of the CNDP, as they determine how the structural integrity of a network is evaluated after node removals. These measures quantify how well the network remains connected after the deletion of certain nodes, and their selection directly influences the problem's formulation and solution strategies. Commonly used connectivity metrics include the number of connected node pairs [1, 2, 4, 7, 8], the size of the largest connected component [13, 15], and the total number of surviving paths or communication efficiency [6]. In this study, we examine *pairwise connectivity* as a measure of network cohesion, as introduced in [4]. Given a graph $G = (V, E)$, the pairwise connectivity quantifies the number of node pairs that reside in the same connected component. For each pair $(u, v) \in V \times V$, the pairwise connectivity c_{uv} is defined as follows:

$$c_{uv} = \begin{cases} 1 & \text{if nodes } u \text{ and } v \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases}$$

Based on this metric, the *Critical Node Detection Problem (CNDP)* is formulated as follows: Given an undirected graph $G = (V, E)$ with $|V| = n$ and a positive integer K representing the maximum number of nodes that may be deleted, the objective is to find a subset $S \subseteq V$ with $|S| \leq K$ such that the total pairwise connectivity in the resulting graph $G[V \setminus S]$ is minimized. Formally, CNDP seeks to solve:

$$\text{(CNDP)} \qquad S = \operatorname*{argmin}_{S \subseteq V,\ |S| \leq K} \sum_{u,v \in V \setminus S} c_{uv}(G[V \setminus S]),$$

where

$$c_{uv} = \begin{cases} 1 & \text{if } u \text{ and } v \text{ are in the same connected} \\ & \text{component of } G[V \setminus S], \\ 0 & \text{otherwise.} \end{cases}$$

In this work, we investigate CNDP in the specific context of path graphs, focusing on the case where connection costs between node pairs are binary either a connection is considered important (cost 1) or not (cost 0). Although CNDP is known to be NP-hard on trees–even under uniform node weights and 0/1 connection costs as established by Di Summa et al. [7]. We demonstrate that the problem is tractable on path graphs.

We consider two natural variants of the problem: one with unit node weights, where a fixed number K of deletions is allowed, and the other with arbitrary node weights, where a total weight budget W restricts the removal set. In both cases, the goal is to remove a subset of nodes such that the number of important connections defined by a 0/1 connection cost matrix is minimized. We develop dynamic programming algorithms for each variant and prove their optimality. These results demonstrate that, unlike trees, CNDP on paths admits closed-form dynamic recurrences that can be solved efficiently, bridging a gap in the literature between trivial and NP-hard instances.

2. CNDP on Paths with Unit Node Weights and Binary Connections

In this section we illustrate an algorithm for solving CNDP on paths when we are given general $0/1$ connection costs c_{uv} for all u, v with unit weights w_v for all $v \in V$. For each $u, v \in V$, a connection is considered important if $c_{uv} = 1$. In this scenario, the objective is to minimize the number of connections that remain in the path $P(V, E)$ after the removal of up to K nodes.

Let $P(V, E)$ be a path with $|V| = n$ nodes and let the nodes are in order, i.e., $1, 2, \ldots, i, i+1, \ldots, n$. We are assuming that we have solved the sub-path problem optimally from node $i + 1$ to node n. To address the problem using dynamic programming, we introduce the following function:

> $F_i(k, t)$ = minimum number of connections surviving in the sub-path that begins at node i when k nodes are removed from the sub-path that starts at node i and ends at node n, with the minimum node that is removed at position t where $i \leq t \leq n + 1$. Note that when no node is removed ($k = 0$) in the sub-path, then we define $t = n + 1$ to indicate that there is no minimum node. Furthermore, if it is not possible to satisfy the above condition, then we define $F_i(k, t) = \infty$.

The values of F_i are calculated by traversing the path in postorder (from node n to node 1), by means of the following relations:

$$F_i(k,t) = \begin{cases} F_{i+1}(0, n+1) + \sum_{j=i+1}^{n} c_{ij} & \text{if } k = 0, \\ \min\limits_{i+1 \leq r \leq n+1} \{F_{i+1}(k-1, r)\} & \text{if } k > 0,\ t = i, \\ F_{i+1}(k, t) + \sum_{j=i+1}^{t-1} c_{ij} & \text{if } k > 0,\ t > i. \end{cases} \tag{1}$$

The initial conditions for the last node are the following:

$$F_n(k,t) = \begin{cases} 0 & \text{if } (k=0,\ t=n+1) \text{ or } (k=1,\ t=n), \\ \infty & \text{otherwise.} \end{cases} \tag{2}$$

Recursion (1) can be interpreted as follows:

When $k = 0$, it indicates that no nodes are removed from the sub-path starting at node i, i.e., we do not remove node i and we also don't remove any other node after node i. In this case we have to count all the important connections from node i to node n. Hence the number of paths that survive in the sub-path rooted at node i when we are not removing anything will be $F_i(0,0) = F_{i+1}(0, n+1) + \sum_{j=i+1}^{n} c_{ij}$.

If $k > 0$, then we have two options based on the node i has to be removed or not. The case $k > 0$, $t = i$ means that the node i has to be removed so that we are disconnecting all the connections that start at node i and therefore we have $k-1$ nodes left to remove in the sub-path. In this case we have to take $F_{i+1}(k-1, r)$ for every possible first node r varying from $i+1, \ldots, n+1$. Otherwise, $k > 0$, $t > i$ means that we are keeping the node i and t is the position of the minimum node that has to be removed and then we have to count all the important connections that go from node i to any node before node t.

Since $n \in V$ is the last node of the path, equation (2) says that either we remove the node n ($k = 1$) or we keep it ($k = 0$) and in both cases the number of important connections surviving is 0.

Assuming the path P is rooted at node 1, the optimal value of the problem can be expressed as

$$\text{OPT} = \min\{F_1(K,t) : t = 1, \ldots, n+1\}. \tag{3}$$

As is common in dynamic programming approaches, the optimal solution can subsequently be reconstructed through backtracking.

Proposition 2.1. The CNDP on a path with binary (0/1) connection costs and uniform node weights can be solved using the recursion (1)–(2) within $\mathcal{O}(n^3K)$ time.

Proof. For each of the n nodes, there can be up to $\mathcal{O}(n)$ possible values of t and $\mathcal{O}(K)$ possible values of k, resulting in a total of $\mathcal{O}(n^2K)$ values of F to be computed. The most computationally demanding part is the recurrence in equation (1), which involves at most $\mathcal{O}(n)$ operations. Therefore, the worst-case total time complexity is bounded by $\mathcal{O}(n^3K)$.

3. CNDP on Paths with Arbitrary Node Weights and Binary Connections

This section addresses the same problem as the preceding one, but with general node weights rather than unit node weights.

Given the path $P = (V, E)$ with general 0/1 connection costs and a budget W with weights w_v on the vertices, the objective is to determine a subset $S \subseteq V$ such that the total weight $w(S) \leq W$, and the number of surviving connected pairs in the resulting subgraph $P[V \setminus S]$ is minimized.

This version of the problem can be tackled using a dynamic programming approach analogous to the one used for the unit weight case. The recursive formulation depends on two main parameters: k, representing the number of surviving important connections in the sub-path, and t, indicating the position of the first node that is deleted.

Following the same notation as in the earlier section, we define the function below to construct the recursive solution.

> $F_i(k,t)$ denotes the minimum cumulative weight of nodes that must be deleted from the sub-path starting at node i, and t is the position of the minimum node that is removed where $i \leq t \leq n+1$ and k important connections survive in the sub-path. Note that when no node is removed in the sub-path, then we define $t = n+1$ to indicate that there is no minimum node. Furthermore, if it is not possible to satisfy the above condition, then we define $F_i(k,t) = \infty$.

We compute the values of F recursively for all $i \in V$, $k = 0, \ldots, n(n-1)/2$, $i \leq t \leq n+1$, as follows.

$$F_i(k,t) = \begin{cases} w_i + \min\limits_{i+1 \leq r \leq n+1} \{F_{i+1}(k,r)\} & \text{if } t = i, \\ F_{i+1}(k - \sum_{j=i+1}^{t-1} c_{ij}, t) & \text{if } t > i. \end{cases} \tag{4}$$

The initial conditions for the last node are the following:

$$F_n(k,t) = \begin{cases} w_n & \text{if } k = 0, t = n, \\ 0 & \text{if } k = 0, t = n+1, \\ \infty & \text{in all other cases.} \end{cases} \tag{5}$$

Equation (4) can be interpreted as follows:

The case $t = i$ means that the minimum node which has to be removed is at position i. Since we are removing node i, we have to look at the sub-path starting at the node $i+1$ to get k connections. Therefore, according to the definition of F, if the root node i is deleted, the minimum total weight of nodes to be removed from the sub-path rooted at i is given by $F_i(k,t) = w_i + \min_r\{F_{i+1}(k,r)\}$ where w_i corresponds to the weight of the node i. On the other hand, if the root i of the sub-path is not removed, then all the nodes we have to remove is from the sub-path rooted at node $i+1$ and the position of the first node that has to be removed is at position t. Since we are keeping node i and the first node that is removed is at position t, the number of connections starting from node i to before node t is $\sum_{j=i+1}^{t-1} c_{ij}$ and the connections surviving in the sub-path rooted at node i will be $(k - \sum_{j=i+1}^{t-1} c_{ij})$. Thus by definition of F, $F_i(k,t) = F_{i+1}(k - \sum_{j=i+1}^{t-1} c_{ij}, t)$.

Equation (5) says that we get a connectivity of 0 by removing the last node n of the path as the first node. Otherwise, we get also a connectivity of 0 by removing nothing.

Assuming that the path is rooted at node 1, the optimal value can be determined as

$$\mathrm{OPT} = \min\left\{k \;\middle|\; F_1(k,t) \le W,\ k = 0,\ldots,\frac{n(n-1)}{2},\ \ t = 1,\ldots,n+1.\right. \tag{6}$$

As is standard in dynamic programming, the corresponding optimal solution can be reconstructed through backtracking.

Proposition 3.1. The CNDP on a path with binary (0/1) connection costs and arbitrary node weights can be solved using the recurrence relations (4)–(5) within a time complexity of $\mathcal{O}(n^5)$.

Proof. For each node $i \in V$, there are at most $\mathcal{O}(n)$ values for t and $\frac{n(n-1)}{2} + 1 = \mathcal{O}(n^2)$ values for k, resulting in $\mathcal{O}(n^4)$ total entries of F to compute. The most computationally intensive step occurs in equation (4), which requires evaluating up to $\mathcal{O}(n)$ values of r. Consequently, the total computational effort in the worst case is bounded by $\mathcal{O}(n^5)$.

4. Conclusion

This study addresses the Critical Node Detection Problem on path graphs under binary (0/1) connection cost models, contributing exact polynomial-time algorithms for two important variants: with unit node weights and with arbitrary node weights under a total weight budget. Despite the known NP-hardness of CNDP on trees, we demonstrate that its restriction to paths permits efficient and optimal solutions. By exploiting the linear structure of path graphs, we design dynamic programming formulations that run in $O(n^3K)$ time for the unit-weight case and $O(n^5)$ time for the arbitrary-weight case. These findings not only fill a gap in the CNDP literature by characterizing tractable instances on basic graph classes but also have practical implications for various domains, including communication networks, transportation systems, power grids, and biological chains, where linear topologies frequently arise.

ORCID

0009-0004-0591-6648 (Syed Md Omar Faruk)

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

[1] Addis, B., Aringhieri, R., Grosso, A., Hosteins, P.: *Hybrid constructive heuristics for the critical node problem*, Annals of Operations Research, 238(1-2), 637-649 (2016). https://doi.org/10.1007/s10479-016-2110-y

[2] Addis, B., Di Summa, M., Grosso, A.: *Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth*, Discrete Applied Mathematics, 161(16), 2349-2360 (2013). https://doi.org/10.1016/j.dam.2013.03.021

[3] R. Ali and Y. Chen, "Power grid robustness via critical node identification in line-structured systems," *Electric Power Systems Research*, vol. 187, p. 106455, 2020.

[4] Arulselvan A., Commander C. W., Elefteriadou L., Pardalos P. M.: *Detecting critical nodes in sparse graphs*, Computers & Operations Research, 36, 2193-2200 (2009). https://doi.org/10.1016/j.cor.2008.08.016

[5] D. Banerjee and S. Rao, "Molecular pathway disruption analysis using path graph models," *Bioinformatics*, vol. 34, no. 5, pp. 876-883, 2018.

[6] Costa, Luciano da Fontoura and Rodrigues, Francisco A and Travieso, Gonzalo and Boas, Pedro RV: *Characterization of complex networks: A survey of measurements*, Advances in physics, 56 (1), 167-252 (2007).

[7] Di Summa, M., Grosso, A., Locatelli, M.: *Complexity of the critical node problem over trees*, Computers & Operations Research, 38(12), 1766-1774 (2011).

[8] Di Summa, M., and Faruk, SMO.: *Critical node/edge detection problems on trees*, 4OR, 21: 439-455 (2022).

[9] Faruk, S. M. O., and Jabin, K. N. (2022). *Critical Node Detection Problem over a Path*, Applied Mathematics, 12(2), 25-31. https://doi.org/10.5923/j.am.20221202.01

[10] M. Fernandes and R. Bianchini, "Polynomial-time solutions for CNDP in special graph classes," *Theoretical Computer Science*, vol. 823, pp. 59-73, 2020.

[11] N. Gupta and S. Jain, "Risk-aware supply chain modeling and analysis using CNDP," *Journal of Operations Management*, vol. 68, no. 1, pp. 40-55, 2022.

[12] H. Kim and W. Tan, "Reliability optimization in sequential distributed computing systems," *ACM Transactions on Computer Systems*, vol. 37, no. 3, pp. 1-26, 2019.

[13] Oosten, M., Rutten, J. H. G. C., Spieksma, F. C. R.: *Disconnecting graphs by removing vertices: a polyhedral approach*, Statistica Neerlandica 61(1), 35-60 (2007).

[14] M. O' Reilly and A. Singh, "Adaptive curriculum design using critical path analysis," *Computers & Education*, vol. 164, p. 104117, 2021.

[15] Shen, S., Smith, J.: *Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs*, Networks, 60(2), 103-119 (2012). https://doi.org/10.1002/net.20464

[16] J. Smith and X. Liu, "Critical node detection in linear communication networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 230-242, 2019.

[17] L. Zhang and P. Kumar, "Vulnerability analysis of urban transit networks using path-based criticality," *Transportation Research Part C*, vol. 128, p. 103202, 2021.