

On The Complexity of the Shortest Path Problem in a Layered Directed Acyclic Graph with Consistency Constraints

Alexandre Dubray¹[0000-0002-3302-870X], Guillaume Derval², Siegfried
Nijssen¹, and Pierre Schaus¹

¹ Institute of Information and Communication Technologies, Electronics and Applied
Mathematics (ICTEAM), UCLouvain, Belgium
`{first.second}@uclouvain.be`

² Department of Electrical Engineering and Computer Science, ULiège, Belgium
`gderval@uliege.be`

Abstract. In this document, we explore the complexity of the shortest path problem in a layered directed acyclic graph under consistency constraints. In this context, a consistency constraint imposes that the optimal path select nodes with the same labels in some layers of the graph. We show that this problem is NP-Hard by reducing the 3-SAT problem to it.

1 Problem Definition

Let $G = (V, E)$ be a graph with V the set of nodes and E the set of edges. Each node $v \in V$ has a label, from a set $\mathcal{L}$, denoted l_v and V is divided into T layers $L_1, \dots, L_T$ such that $V = \bigcup_{j=1, \dots, T} L_j$ and $L_i \cap L_j = \emptyset \forall i \neq j$. In each layer, no two nodes have the same label. Thus, when clear from the context, a node can be identified by its label. We denote by $e = (l, l', t) \in E$ an edge from node with label l at layer L_t to node with label l' at layer L_{t+1} ($1 \leq t < T$) with weight w_e , where weights can be both positive and negative. For instance, the decoding problem in Hidden Markov Models can be encoded using such graph as shown in Figure 1. In this case, each layer has the same number of nodes, with the same labels representing the hidden states of the model.

A path in G from L_1 to L_T selects one node per layer and can be identified by the sequence of node labels on the path. More formally, let $P = \langle P_1, \dots, P_T \rangle \in \mathcal{L}^T$ be a path from L_1 to L_T such that $P_i \in L_i$. The cost of P , is the sum of the weights of the arcs in the path: $\sum_{t=1}^{T-1} w_{(P_t, P_{t+1}, t)}$.

One consistency constraint is specified by identifying a set of layers for which the same label must be selected in each layer of the path. More formally, $C = \{C_1, \dots, C_k\}$ are k consistency constraints with $C_i = \{c_1^i, \dots, c_{k_i}^i\} \subseteq \{1, \dots, T\}$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. A path P is said to be consistent if all the consistency constraints are respected. The problem of finding the shortest consistent path is

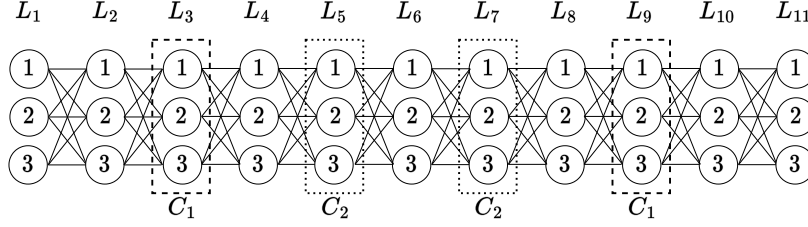


Fig. 1. Example of a layered DAG with two consistency constraints. The edges in the DAG are oriented from left to right and the labels on the nodes represent the hidden states or the source and target vertex.

thus formalized as follows:

$$P^* = \arg \min_{P \in \mathcal{L}^T} \sum_{t=1}^{T-1} w(P_t, P_{t+1}, t) \quad (1)$$

$$\text{s.t. } P_{c_1^i} = \dots = P_{c_{k_i}^i} \quad \forall C_i \in C \quad (2)$$

2 NP-Hardness of the Optimization Problem

In this section, we prove that finding the optimal solution for the problem defined by Equation (1)-(2) is NP-hard. To do so we reduce the 3-SAT problem to ours and show that this reduction is polynomial. Let L be a boolean formula in conjunctive normal form (CNF) with n clauses, where each clause of the formula contains exactly three literals. The 3-SAT problem is the one of deciding whether there exists an assignment to the literals in L to make it true (such an instance is said to be *satisfiable*). 3-SAT is NP-Complete [1].

We present a (polynomial size) encoding of any 3-SAT instance in an unweighted layered DAG with consistency constraints, where finding (resp. not finding) a path through the graph indicates that the instance is satisfiable (resp. not satisfiable). The graph is built from various parts, themselves containing layers. The connection between the parts is fully connected: the last layer of a part is fully connected to the first layer of the next part.

Each layer represents either a literal or a 2-clause (containing two literals that can be negated). A layer for a literal contains two nodes, T and F , representing its possible truth values. A layer for a 2-clause similarly contains four nodes, $(T, T), (T, F), (F, T), (F, F)$. Layers representing the same literal or 2-clause are bound by a consistency constraint. The parts are built as follows:

- First, a single node s is emitted.
- Every clause $a \vee b \vee c$ is then encoded, in its own part. A first layer representing a is created, and linked to a second layer for the 2-clause $b \vee c$. The nodes are linked only if they respect the relation $a \vee b \vee c$ (i.e. only the link between $a = F$ and $(b, c) = (F, F)$ doesn't exist). Three additional layers are added:

The links between the first and second layer enforce that the clause $a \vee b \vee c$ is respected. The links between the third, fourth and fifth layers ensure consistency between the literal layers and the 2-clause layer. With the addition of the consistency constraint between the second and fourth layer, both representing the 2-clause $b \vee c$, this implies that a valid path inside the part (from the first layer to the last) is equivalent to an assignment to the literals with the values represented by the nodes.

- Finally, a single node t is emitted.

Figure 2 shows the resulting graph for the 3-SAT instance $L = (l_1 \vee \neg l_3 \vee l_4) \wedge (l_2 \vee l_4 \vee l_3)$. Each layer has a label, on top of the graph, representing the literals included in the layer, and layers with the same label are bound by a consistency constraint.

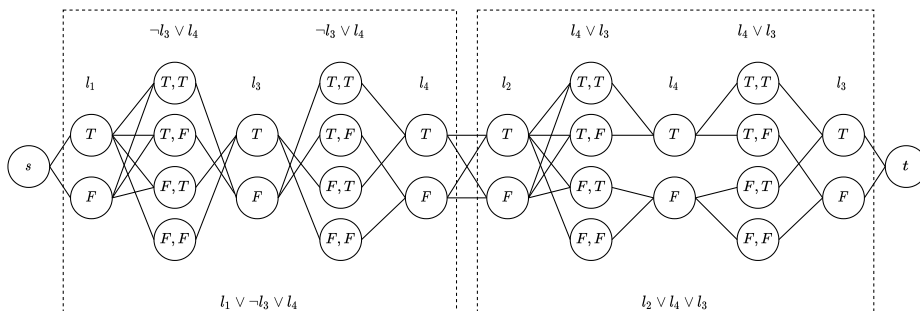


Fig. 2. Example of DAG to solve the 3-SAT problem for the expression $(l_1 \vee \neg l_3 \vee l_4) \wedge (l_2 \vee l_4 \vee l_3)$. The edges are oriented from left to right and have no cost. The labels in the node represent the truth values of the literals, and the label on top on the graph represents the literals concerned at the layer. A consistency constraint is imposed on all the layers with the same label.

Theorem 1. *Let L be a boolean formula in CNF with three literals per clause and G the graph produced by the algorithm above, which encodes L with the consistency constraints.*

A consistent $s - t$ path exists in G if and only if there exists an assignment of the literals that makes L true.

Proof. **Assignment $\Rightarrow$ consistent path.** Given an assignment making L true, let G' be the subgraph containing only the nodes associated to the truth value of the assignment. By construction G' is connected between s and t and the consistency constraints are respected.

Consistent path $\Rightarrow$ assignment. Given a valid path between s and t , we proved during the construction that having a valid path inside a part implies that the clause it represents is respected. With the addition of the consistency constraint between the layers representing the literals (implying that if a node $l = T$ is used in the path in some part, the node $l = F$ cannot be visited in another part), this shows that all the clauses are respected by the same literal/value pairs, hence giving a valid assignment.

We have shown that finding a path in such a graph, with consistency constraints, is equivalent to solving the 3-SAT problem. Moreover, the encoding is polynomial, as shown above: the number of layers/nodes is polynomial and the graph can be constructed in a single pass over the clauses. We thus conclude that finding whether any path exists in such a graph with consistency constraints is NP-Hard. This implies that finding the shortest path in a similar graph but with weights on the arcs is at least NP-Hard. Let us note that since we are concerned with DAG, this also proves that finding the longest path under consistency constraint is NP-Hard.

References

1. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on Theory of computing (1971)