A Coalgebraic Dijkstra Algorithm

Takahiro Sanada
Fukui Prefectural University, Japan

Yoàv Montacute
National Institute of Informatics, Tokyo, Japan

Kittiphon Phalakarn
National Institute of Informatics, Tokyo, Japan

Ichiro Hasuo
National Institute of Informatics, Tokyo, Japan
SOKENDAI (The Graduate University for Advanced Studies), Kanagawa, Japan
Imiron Co., Ltd., Tokyo, Japan

Abstract

The Dijkstra algorithm is a classical method for solving the shortest path problem on weighted graphs. There are several variations of the Dijkstra algorithm, including algorithms for the widest path problem and for two-player games. In this paper, we introduce the *coalgebraic shortest path problem* (CSPP), a unifying framework for a broad class of optimization problems on state-transition systems. This framework encompasses not only the aforementioned problems but also new ones such as the shortest binary tree problem. We further present a *coalgebraic Dijkstra algorithm* for solving the CSPP efficiently under a suitable condition. Our condition is necessary and sufficient for the algorithm to return correct solutions, thereby providing a precise criterion for when Dijkstra-style acceleration is possible. We also show that the proposed algorithm achieves asymptotic complexity comparable to that of the classical Dijkstra algorithm.

2012 ACM Subject Classification Theory of computation → Shortest paths

Keywords and phrases Coalgebra, Greatest fixed point, Dijkstra's algorithm, Shortest path

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.5

Category Invited Talk

Related Version *Full Version*: `https://arxiv.org/abs/2605.22149`

Funding The authors are supported by JST ASPIRE Grant No. JPMJAP2301.
Takahiro Sanada: Supported by JSPS KAKENHI Grant No. JP24K23867.
Yoàv Montacute: Supported by JST ACT-X Grant No. JPMJAX24CR.

1 Introduction

The shortest path problem (SPP) on a weighted directed graph is a classical problem. The Bellman–Ford algorithm [7,11] solves the SPP by iteratively updating the distances of vertices. This update can be viewed as an operator $\Phi\colon L \to L$ for a lattice L, and the solution to the SPP is the greatest fixed point $\nu\Phi$ of Φ [24]. The Dijkstra algorithm [10] is an improvement on the Bellman–Ford algorithm for weighted graphs without negative weights. The Dijkstra algorithm restricts updates to vertices belonging to a specific set and avoids updating vertices that are expected to have already achieved the minimum value.

There are several variations of the Dijkstra algorithm. Reachability is a special case of the SPP, thus amenable to the Dijkstra algorithm. The widest path problem, which seeks a path with maximum capacity, is solvable by a modified version of the Dijkstra algorithm [26]. A class of two-player games can also be solved by a Dijkstra-style algorithm [5].

It is important to note that, aside from these *positive* variations, there are many known *negative* variations – those optimization problems based on state transition systems that do *not* allow the application of the Dijkstra-style acceleration. Known examples of such negative variations include the SPP with negative edges, and the reachability probability in a probabilistic system such as a Markov chain.

An overview of such variations is given in Table 2, where the last column shows the applicability of the Dijkstra-style acceleration. While many previous works establish positive results [5, 23, 26, 30], we do see some negative entries.

Moreover, the table shows that the distinction between positive and negative can be subtle. For example (the sixth and seventh in Table 2), when the SPP comes with a (multiplicative) rate, 1) Dijkstra is applicable when the rate is ≥ 1 (thus the rate represents *interest*), while 2) it is not applicable when the rate is ≤ 1 (the rate represents *discount*). Another subtle difference is observed for dynamic games with a discount rate r (the second last in Table 2), where Dijkstra is applicable only when the stepwise reward is constant (with $\ell_0 = L$).

Therefore, in this paper, we are interested in the following questions.

1. What is the essence of the Dijkstra-style acceleration of fixed point computation?
2. What is the condition that makes a problem amenable to the Dijkstra-style acceleration?
3. What is a good mathematical level of abstraction for answering these questions?

Our answer to the last (meta-level) question is the language of *category theory* – the language of *coalgebras* [15, 27], in particular, as a categorical model of state-space transition systems. We focus on a coalgebraic signature functor of the form $F = \mathbb{B} \times \mathcal{P}_{\text{fin}}(G-)$, where $\mathbb{B} = \{\mathbf{t}, \mathbf{f}\}$ tells if a state is a target or not, and the finite power set functor $\mathcal{P}_{\text{fin}}$ represents a finite branching over G-transitions. Besides, we use a poset Ω equipped with a *modality* $\sigma\colon G\Omega \to \Omega$ – describing how weights are accumulated along G-transitions, much as in predicate transformer semantics for coalgebras [2, 14]. All these give us a categorical framework that accommodates various problems in Table 2 (§ 3).

Within this categorical framework, we formalize the *coalgebraic Dijkstra algorithm* (Algo. 2), embodying our understanding of "the essence of the Dijkstra algorithm" and addressing the first question above. Moreover, we identify a necessary and sufficient condition for the correctness of the coalgebraic Dijkstra algorithm, formulated in the language of category theory (Thm. 4.14), thus addressing the second question. In the course of this venture, we find that our categorical/coalgebraic abstraction level is the right one.

It turns out that the abstract categorical framework is concrete enough, too, so that we can conduct complexity analysis and discuss algorithmic improvements. Indeed, in § 4.2, we 1) establish the asymptotic complexity of the coalgebraic Dijkstra algorithm, 2) show that it coincides with the classical Dijkstra algorithm for the original problem of the SPP, and 3) propose an improvement by a Fibonacci heap, for the general coalgebraic algorithm.

We summarize our contributions as follows.

1. We formulate the *coalgebraic shortest path problem* (CSPP, § 3), using coalgebras of the type $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\text{fin}}(GX)$ and transition modalities $\sigma\colon G\Omega \to \Omega$, and formalizing the objective as the greatest fixed point of the *Bellman operator* $\Phi_\gamma^{G,\sigma}\colon [X, \Omega] \to [X, \Omega]$ suitably defined using σ and γ. This framework accommodates various problems (Table 2), covering both path-like and tree-like settings (such as *shortest binary tree* and some games).
2. We present the *coalgebraic Dijkstra algorithm* for the CSPP (§ 4), and identify a necessary and sufficient condition on a transition modality $\sigma\colon G\Omega \to \Omega$ for the algorithm to be correct. This explains the subtleties in the last "applicability" column of Table 2.
3. We establish the complexity of the coalgebraic Dijkstra algorithm (§ 4.2). We also present its improvement with Fibonacci heaps.

Related work

The Bellman–Ford algorithm [7, 11] solves the SPP of a single source or multiple sources. The Dijkstra algorithm [10] is an improvement of the Bellman–Ford algorithm. Misra [24] showed that the SPP is equivalent to finding the greatest fixed point of an operator. Our approach is a coalgebraic generalization of Misra's observation.

There are several approaches for generalizing path problems on weighted graphs [13, 20, 25, 29]. These approaches introduce algebraic structures, such as semirings, dioids and routing algebras, in order to capture the weights of paths. Typically, these algebraic structures have a carrier set and a binary operation on the carrier set to accumulate edge weights along a path. Instead of such a binary operation, we use a categorical algebra $\sigma\colon G\Omega \to \Omega$, where Ω is a carrier set and G is a functor describing the "arity" of σ. Owing to this generalization, our framework can extend paths to tree structures such as game trees or shortest trees.

Coalgebras are the dual notion of algebras and are widely used as a generalization of state-transition systems [15, 27]. Many algorithms for graph-like structures have been generalized to algorithms for coalgebras – for example, coalgebraic model checking [18, 19], coalgebraic automata learning [6, 16], and the computation of bisimilarity for coalgebras [9, 17, 28].

Organization of the paper

In § 2, we review the classical Dijkstra algorithm and provide a lattice-theoretic perspective on it. In § 3, we formulate the shortest path problem coalgebraically as a coalgebraic shortest path problem (CSPP), and exhibit its instances. In § 4, we introduce the coalgebraic Dijkstra algorithm, present its correctness condition (§ 4.1) and complexity studies (§ 4.2).

Notations

We write **Set** for the category of sets and maps. We write $\mathbb{N}$ for the set of natural numbers including zero, $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$, $\mathbb{R}$ for the set of real numbers, $\mathbb{R}^\infty = \mathbb{R} \cup \{\infty\}$, $\mathbb{R}^\infty_{\geq 0} = [0, \infty]$, $\mathbb{R}^{\pm\infty} = [-\infty, \infty]$ and $\mathbb{B} = \{\mathbf{t}, \mathbf{f}\}$. The identity functor, the finite power set functor, on **Set**, are denoted by $\mathrm{Id}\colon \mathbf{Set} \to \mathbf{Set}$, $\mathcal{P}_{\mathrm{fin}}\colon \mathbf{Set} \to \mathbf{Set}$, respectively. Given sets $X_0, X_1, \dots, X_n$, we denote the i-th projection by $\pi_i\colon X_0 \times \cdots \times X_n \to X_i$. For a set X, we write X^* for the set of finite strings over X. The empty string is denoted by ϵ. For a natural number $n \in \mathbb{N}$, the finite set $\{0, \dots, n-1\}$ of n elements is denoted by $[n]$. The *function space* $[X, Y]$ is the set of functions from X to Y.

▶ **Definition 1.1.** *The* non-empty finite power set functor $\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}\colon \mathbf{Set} \to \mathbf{Set}$ *is defined by* $\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}(X) = \mathcal{P}_{\mathrm{fin}}(X) \setminus \{\emptyset\}$ *for* $X \in \mathbf{Set}$*. Its action on morphisms is given by direct images.*

The (discrete, finite-support) distribution functor $\mathcal{D}\colon \mathbf{Set} \to \mathbf{Set}$ *is the functor defined by*

$$\mathcal{D}(X) = \left\{\mu\colon X \to [0,1] \mid \#\{x \mid \mu(x) > 0\} < \infty \text{ and } \textstyle\sum_{x \in X} \mu(x) = 1\right\}$$

for $X \in \mathbf{Set}$*, where* $\#S$ *denotes the cardinality of a set* S*. Its action on morphisms is* $\mathcal{D}(f)(\mu)(y) = \sum_{x \in f^{-1}(y)} \mu(x)$ *for* $f\colon X \to Y$*,* $\mu \in \mathcal{D}(X)$ *and* $y \in Y$*.*

2 Preliminaries

We are interested in certain greatest fixed points; their existence is guaranteed by the following well-known result.

▶ **Lemma 2.1** (Cousot–Cousot [8, Cor. 3.3]). *Let $(L, \sqsubseteq)$ be a poset, and $\Psi\colon L \to L$ be a monotone function. Let L have infimum $\bigsqcap S$ of every $S \subseteq L$. We define a transfinite sequence*

$$\top \sqsupseteq \Psi(\top) \sqsupseteq \Psi^2(\top) \sqsupseteq \cdots \sqsupseteq \Psi^\alpha(\top) \sqsupseteq \cdots, \tag{1}$$

where α is an arbitrary ordinal, by the following transfinite induction: $\Psi^0(\top) = \top$, $\Psi^{\alpha+1}(\top) = \Psi(\Psi^\alpha(\top))$ and $\Psi^\alpha(\top) = \bigsqcap_{\beta<\alpha} \Psi^\beta(\top)$ where α is a limit ordinal. Then, the sequence (1) eventually stabilizes and its limit is the greatest fixed point (gfp) $\nu\Psi$ of Ψ. Precisely, there is an ordinal α such that $\Psi^\alpha(\top) = \Psi^{\alpha+1}(\top) = \cdots = \nu\Psi$.

While the sequence (1) is transfinite in general, our coalgebraic Dijkstra algorithm is guaranteed to terminate. See Prop. 4.5 and § 4.2 for complexity analysis.

The famous *Kleene theorem* can be thought of as a variant of Lem. 2.1, where 1) Ψ is assumed to be cocontinuous (preserving suitable infima) and 2) the stabilization is guaranteed at the ordinal ω. We use the Cousot–Cousot theorem because we do not require Ψ's cocontinuity in our general framework.

2.1 Dijkstra and Bellman–Ford via order-theoretic fixed points

We review the classical Dijkstra algorithm, and the Bellman–Ford algorithm as its prototype. We introduce necessary order-theoretic preliminaries, too.

Let (X, E, w, T) be a *weighted graph*, where X is the *state space*, $E \subseteq X \times X$ collects *edges*, and $w\colon E \to \mathbb{R}_{\geq 0}$ assigns *weights* to edges. Our notion of weighted graph additionally specifies the set $T \subseteq X$ of *target states*. We introduce some notation and terminology.

- $\Omega = \mathbb{R}^\infty_{\geq 0} = \mathbb{R}_{\geq 0} \cup \{\infty\}$ – we augment $\mathbb{R}_{\geq 0}$ with the maximum ∞ – is the *weight domain*. Its usual order $\leq$ is henceforth denoted by $\sqsubseteq$.
- A function $d\colon X \to \Omega$ – assigning, to each state x, a weight $d(x) \in \Omega$ – is an *Ω-valuation*.
- The set $[X, \Omega]$ of Ω-valuations inherits the order $\sqsubseteq$ from Ω in the pointwise manner ($d_1 \sqsubseteq d_2$ iff $d_1(x) \sqsubseteq d_2(x)$ for each $x \in X$).
- We define the *Bellman operator* $\Phi\colon [X, \Omega] \to [X, \Omega]$ by

$$\Phi(d)(x) = \begin{cases} 0 & \text{if } x \in T, \\ \min\limits_{(x,x') \in E} \big(w(x, x') + d(x')\big) & \text{if } x \notin T. \end{cases} \tag{2}$$

The following characterization is well-known. It is crucial for our theoretical development.

▶ **Proposition 2.2** ([24]). *The* shortest path valuation $d_{\mathrm{SP}}\colon X \to \Omega$, *which assigns to each state $x \in X$ the shortest path length $d_{\mathrm{SP}}(x)$ to one of the target states in T, coincides with the greatest fixed point $\nu\Phi$ of the Bellman operator (see (2)).*

▶ **Problem 2.3** (shortest path problem (SPP)). *The* shortest path problem *(SPP) is the problem of computing $d_{\mathrm{SP}}\colon X \to \Omega$ for a given weighted graph (X, E, w, T).*

The order-theoretic essence of the Bellman–Ford algorithm is the computation of $\nu\Phi$ (Prop. 2.2) via the iterative characterization in Lem. 2.1. Specifically, we let $L = [X, \Omega]$; it is easily seen to have all infima (including $\top = \bigsqcap \emptyset$) since $\Omega = \mathbb{R}^\infty_{\geq 0}$ does. Therefore, by iteratively applying the Bellman operator Φ (which is Ψ in Lem. 2.1), we eventually obtain $\nu\Phi$. (It is, of course, a different problem whether the sequence (1) stabilizes after finitely many steps. This is the case with the Bellman operator Φ for the shortest paths in (2).)

The Dijkstra algorithm (Algo. 1) accelerates this Bellman–Ford iteration; it does so by 1) maintaining the set S of *frozen* states, and 2) applying the Bellman update only selectively.

Algorithm 1 The (classical) Dijkstra algorithm for the SPP.

1: **procedure** DIJKSTRA(X, E, w, T)
2: $S \leftarrow T$
3: $Y \leftarrow S$
4: $d \leftarrow \lambda x \in X. \begin{cases} 0 & (x \in S) \\ \infty & (x \notin S) \end{cases} \in [X, \Omega]$
5: **while** $S \neq X$ **do**
6: $P \leftarrow$ the set of predecessors of Y
7: $d \leftarrow \lambda x. \begin{cases} \min_{(x,x') \in E}(w(x,x') + d(x')) & (x \in P \cap (X \setminus S)) \\ d(x) & (\text{otherwise}) \end{cases}$ ▷ Selective Bellman update
8: $Y \leftarrow \left\{ y \in X \setminus S \mid d(y) = \min_{z \in X \setminus S} d(z) \right\}$ ▷ Minimal freezing
9: $S \leftarrow S \cup Y$ ▷ Update the set of frozen states
10: **end while**
11: **return** d
12: **end procedure**

Specifically, the *minimal freezing* step (Line 8) is crucial in Algo. 1. There one selects, among the states that have not yet been frozen, states y with the minimal valuation and newly freezes them. The core argument of the correctness proof is that the valuation of these states y is already optimal: $d(y) = (\nu\Phi)(y)$ in the algorithm.

We shall develop a coalgebraic generalization of this argument, in § 4 onward. The classical correctness proof of the Dijkstra algorithm can be obtained as a special case of the general correctness proof there (Thm. 4.14).

▶ **Example 2.4.** Let $(\{0,1,2,3,4,5\}, E, w, \{0\})$ be the weighted graph in Fig. 1. The run of Algo. 1 for the weighted graph is summarized in Table 1. The returned valuation $d\colon \{0,1,2,3,4,5\} \to \mathbb{R}^{\infty}_{\geq 0}$ represents the shortest path from each vertex $x \in \{0,1,2,3,4,5\}$ to the target $0 \in T \subseteq X$.

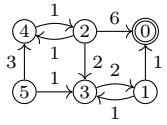


Figure 1 An example of weighted graph.

Table 1 The run of Algo. 1 for the weighted graph in Fig. 1.

n	$d(0)$	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	S	Y
0	∞	∞	∞	∞	∞	∞		
1	0	∞	∞	∞	∞	∞	$\{0\}$	$\{0\}$
2	0	1	6	∞	∞	∞	$\{0,1\}$	$\{1\}$
3	0	1	6	3	∞	∞	$\{0,1,3\}$	$\{3\}$
4	0	1	5	3	∞	4	$\{0,1,3,5\}$	$\{5\}$
5	0	1	5	3	∞	4	$\{0,1,2,3,5\}$	$\{2\}$
6	0	1	5	3	6	4	$\{0,1,2,3,4,5\}$	$\{4\}$

For accommodating this *Dijkstra acceleration*, we present a technical adaptation of Lem. 2.1. For convenience, we focus on sequences of length ω.

▶ **Lemma 2.5** (selective Bellman update). *Let X be a set, Ω be a poset with the maximum $\top$, $\Phi\colon [X,\Omega] \to [X,\Omega]$ be a monotone map, and $(P_1, P_2, \dots)$ be a sequence of subsets of X (an element $x \in P_n$ is called an* active *state at the n-th iteration).*

We define maps $d_n\colon X \to \Omega$, for each $n \in \mathbb{N}$, by induction:

$$d_0 = \top_{[X,\Omega]} \quad \text{and} \quad d_{n+1}(x) = \begin{cases} \Phi(d_n)(x) & \text{if } x \in P_{n+1}, \\ d_n(x) & \text{if } x \notin P_{n+1}. \end{cases}$$

Note that the valuations are updated only in active states. Then, we have $d_n \sqsupseteq d_{n+1} \sqsupseteq \Phi(d_n)$ for each $n \in \mathbb{N}$. Moreover, for each n, we have $d_n \sqsupseteq \Phi^n(\top)$.

Obviously, convergence to $\nu\Phi$ – the main statement of Lem. 2.1 – is not guaranteed for an arbitrary choice of $(P_1, P_2, \dots)$. The Dijkstra algorithm achieves this convergence by the careful design of active states (which is by the design of frozen states).

3 The coalgebraic shortest path problem

In this section, we formalize our problem in the categorical language. It generalizes the SPP on weighted graphs (Prob. 2.3). The generalization is mainly in three directions.

- A *(signature) functor* F for coalgebras. Here coalgebras model state-based dynamics and generalize weighted graphs. We restrict to functors of the shape $F = \mathcal{W}_G = \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(G-)$, where G is a functor. Coalgebras for $\mathcal{W}_G$ are referred to as *weighted G-graphs*.
- A *weight domain* Ω that generalizes $\mathbb{R}^{\infty}_{\geq 0}$ in Prob. 2.3. (We use the notion of *pointed weight domain*; it additionally specifies a so-called *final weight*.)
- A *transition modality* $G\Omega \to \Omega$, formalized as a G-algebra over Ω. This specifies how weights are accumulated along a G-path (i.e. repeated G-transitions).

The generalized problem is called the *coalgebraic shortest path problem (CSPP)*. The original problem (Prob. 2.3) is its example, of course; other examples are given in § 3.5 (cf. Table 2).

3.1 Coalgebras as a generalization of weighted graphs

Coalgebras are categorical models of state transition systems [15, 27].

▶ **Definition 3.1** (coalgebra). *Let $F\colon \mathbf{Set} \to \mathbf{Set}$ be an endofunctor on $\mathbf{Set}$. An* F-coalgebra *is a morphism $\gamma\colon X \to FX$ in $\mathbf{Set}$.*

In this paper, we restrict the signature functor F to the following class.

▶ **Definition 3.2** (weighted G-graph). *Let $G\colon \mathbf{Set} \to \mathbf{Set}$ be an endofunctor on $\mathbf{Set}$. The endofunctor $\mathcal{W}_G\colon \mathbf{Set} \to \mathbf{Set}$ is defined by $\mathcal{W}_G := \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(G-)$ where $\mathbb{B} = \{\mathbf{t}, \mathbf{f}\}$ and $\mathcal{P}_{\mathrm{fin}}$ is the finite power set functor. A $\mathcal{W}_G$-coalgebra is called a* weighted G-graph.

In the weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(GX)$, the first component of $\gamma(x)$ (the Boolean truth value, $\mathbf{t}$ or $\mathbf{f}$) tells if a state x is a target or not. The second component of $\gamma(x)$ gives a finite branching (modeled by $\mathcal{P}_{\mathrm{fin}}$) over G-transitions. In the original SPP (Prob. 2.3), the functor G is given by $\mathbb{R}_{\geq 0} \times -$ and assigns a weight from $\mathbb{R}_{\geq 0}$ to each successor.

3.2 Weight domains

In the next definition, we impose additional conditions on the poset $(\Omega, \sqsubseteq)$. A *final weight* is the weight given to an immediately terminating path; it generalizes the weight $0 \in \mathbb{R}^{\infty}_{\geq 0}$ for the case $x \in T$ in (2).

▶ **Definition 3.3** ((pointed) weight domain). *A* weight domain *is a poset $\Omega = (\Omega, \sqsubseteq)$ such that 1) $\sqsubseteq$ is a total order, 2) it has the least element $\bot_\Omega$, and 3) it has the infimum of any subset of Ω (thus also the greatest element $\top_\Omega$). A* pointed weight domain *is a triple $(\Omega, \sqsubseteq, \xi)$ where $(\Omega, \sqsubseteq)$ is a weight domain and $\xi \in \Omega$ is a designated* final weight.

Recall that $[X, \Omega]$ denotes the function space from X to Ω, with its elements called Ω-valuations. The set $[X, \Omega]$ is equipped with the following pointwise order.

▶ **Definition 3.4.** *Let Ω be a weight domain, and X be a set. For maps $d, e \in [X, \Omega]$, we write $d \sqsubseteq e$ if $d(x) \sqsubseteq e(x)$ for every $x \in X$.*

The pair $([X, \Omega], \sqsubseteq)$ is a partially ordered set with the greatest element $\top_{[X,\Omega]} = \lambda x.\top_\Omega$ and the least element $\bot_{[X,\Omega]} = \lambda x.\bot_\Omega$.

3.3 Transition modalities and Bellman operators

In a weighted G-graph $\gamma\colon X \to \mathbb{B}\times\mathcal{P}_{\mathrm{fin}}(GX)$, at each state x, we pick one G-transition among those finitely many which are offered.[1] Repeating this exhibits a succession of G-transitions. Weights get accumulated in its course.

A transition modality specifies how this accumulation is defined. It is formalized as a categorical algebra, together with a suitable "continuity" condition that we need for correctness (Thm. 4.14).

▶ **Definition 3.5** (transition modality). *Let $G\colon \mathbf{Set}\to\mathbf{Set}$ be a functor. A* transition modality *σ for G is a G-algebra $\sigma\colon G\Omega\to\Omega$ such that, for any finite set X, the map*

$$\sigma\circ G(-)\colon \quad [X,\Omega] \longrightarrow [GX,\Omega], \qquad \big(X \xrightarrow{d} \Omega\big) \longmapsto \big(GX \xrightarrow{Gd} G\Omega \xrightarrow{\sigma} \Omega\big) \tag{3}$$

preserves the infimum of any subset D of $[X,\Omega]$, that is, $\sigma\circ G\left(\bigsqcap_{d\in D} d\right) = \bigsqcap_{d\in D}(\sigma\circ Gd)$.

Optimization problems on coalgebras are specified by Bellman operators, which are induced by a transition modality $\sigma\colon G\Omega\to\Omega$.

▶ **Definition 3.6** (Bellman operator). *Let $G\colon \mathbf{Set}\to\mathbf{Set}$ be a functor, $\Omega=(\Omega,\sqsubseteq,\xi)$ be a pointed weight domain, $\sigma\colon G\Omega\to\Omega$ be a transition modality and $\gamma\colon X\to\mathbb{B}\times\mathcal{P}_{\mathrm{fin}}(GX)$ be a weighted G-graph. The* Bellman operator *$\Phi_\gamma^{G,\sigma}\colon [X,\Omega]\to[X,\Omega]$ is defined as follows:*

$$\Phi_\gamma^{G,\sigma}(d)(x) = \begin{cases} \xi \sqcap \bigsqcap_{a\in A}\sigma\big((Gd)(a)\big) & \text{(if } \gamma(x)=(\mathbf{t},A)\text{)} \\ \bigsqcap_{a\in A}\sigma\big((Gd)(a)\big) & \text{(if } \gamma(x)=(\mathbf{f},A)\text{).} \end{cases}$$

The definition of the Bellman operator generalizes that for the SPP given in (2). By iteratively applying the Bellman operator $\Phi_\gamma^{G,\sigma}$ to $\top\in[X,\Omega]$, we obtain the greatest fixed point $\nu\Phi_\gamma^{G,\sigma}$. This procedure can be seen as a coalgebraic generalization of the original Bellman–Ford algorithm for the SPP.

3.4 The coalgebraic shortest path problem

The problem we tackle is stated as follows. It is easy to verify that by taking $G=\mathbb{R}_{\geq 0}\times -$ and $\sigma\colon G\mathbb{R}^\infty_{\geq 0}\to\mathbb{R}^\infty_{\geq 0}$ defined by $\sigma(a,b)=a+b$, we obtain the original SPP. Note that some examples obtained by instantiating G and σ are not literally about shortest paths; however, for the sake of terminology, we call the general problem a *coalgebraic shortest path problem*.

▶ **Problem 3.7** (coalgebraic shortest path problem (CSPP)). *Let $G\colon\mathbf{Set}\to\mathbf{Set}$ be a functor, and $\sigma\colon G\Omega\to\Omega$ be a transition modality on a pointed weight domain $\Omega=(\Omega,\sqsubseteq,\xi)$. A* coalgebraic shortest path problem *(*CSPP*) on (G,σ) is the problem of computing the greatest fixed point $\nu\Phi_\gamma^{G,\sigma}$ of the Bellman operator $\Phi_\gamma^{G,\sigma}\colon[X,\Omega]\to[X,\Omega]$ for a given weighted G-graph $\gamma\colon X\to\mathbb{B}\times\mathcal{P}_{\mathrm{fin}}(GX)$.*

3.5 Instances of the CSPP

Many graph-based optimization problems are instances of the CSPP. See Table 2. What is notable there is that subtle differences in problems can change the applicability of the Dijkstra acceleration (see the last column). Such phenomena are well-known for shortest path with negative edges or longest path; we identify more instances, such as interest vs. discount rates.

[1] Specifically, by $(\pi_1\circ\gamma)(x)\in\mathcal{P}_{\mathrm{fin}}(GX)$ – recall our indexing convention from § 1.

Table 2 Instances of the CSPP on (G, σ). The "Dijkstra" column indicates if the coalgebraic Dijkstra algorithm is correct there. Known Dijkstra-like algorithms are noted with references.

Problem	$(\Omega, \sqsubseteq, \xi)$	G	$\sigma\colon G\Omega \to \Omega$	Dijkstra applies?
Reachability	$(\{0,\infty\}, \leq, 0)$	Id	$\sigma(a) = a$	Yes
Unweighted shortest path	$(\mathbb{N}^\infty, \leq, 0)$	Id	$\sigma(a) = 1 + a$	Yes
Unweighted longest path	$(\mathbb{N}^\infty, \geq, 0)$	Id	$\sigma(a) = 1 + a$	No
SPP [10]	$(\mathbb{R}^\infty_{\geq 0}, \leq, 0)$	$\mathbb{R}_{\geq 0} \times -$	$\sigma(a,b) = a + b$	Yes [10]
SPP with negative edges	$(\mathbb{R}^{\pm\infty}, \leq, 0)$	$\mathbb{R} \times -$	$\sigma(a,b) = a + b$	No
SPP with interest rate	$(\mathbb{R}^\infty_{\geq 0}, \leq, 0)$	$\mathbb{R}_{\geq 0} \times [1,\infty) \times -$	$\sigma(a,a',b) = a + a'b$	Yes
SPP with discount rate	$(\mathbb{R}^\infty_{\geq 0}, \leq, 0)$	$\mathbb{R}_{\geq 0} \times [0,1] \times -$	$\sigma(a,a',b) = a + a'b$	No
Widest path [26]	$(\mathbb{R}^\infty_{\geq 0}, \geq, \infty)$	$\mathbb{R}_{\geq 0} \times -$	$\sigma(a,b) = a \sqcup b = \min(a,b)$	Yes [26]
Most reliable path [30]	$([0,1], \geq, 1)$	$[0,1] \times -$	$\sigma(a,b) = ab$	Yes [30]
Shortest binary tree	$(\mathbb{R}^\infty_{\geq 0}, \leq, 0)$	$\mathbb{R}_{\geq 0} \times (-)^2$	$\sigma(a,b_0,b_1) = a + b_0 + b_1$	Yes
Binary reachability game	$(\{0,\infty\}, \leq, 0)$	$(-)^2$	$\sigma(a_0,a_1) = a_0 \sqcup a_1 = \max(a_0,a_1)$	Yes
Reachability game [23]	$(\{0,\infty\}, \leq, 0)$	$\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}$	$\sigma(A) = \bigsqcup A = \max A$	Yes [23]
Dynamic game [5]	$(\mathbb{R}^\infty_{\geq 0}, \leq, 0)$	$\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times -)$	$\sigma(A) = \max_{(a,b)\in A}(a + b)$	Yes [5]
Dynamic game with discount rate [5]	$(\mathbb{R}^\infty_{\geq 0}, \leq, 0)$	$\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}([\ell_0, L] \times -)$	$\sigma_r(A) = \max_{(a,b)\in A}(a + rb)$	No (unless $\ell_0 = L$ or $r = 1$, see Ex. 4.15) [5]
Max probabilistic reachability [3, §10.6.1]	$([0,1], \geq, 1)$	$\mathcal{D}$	$\sigma(\mu) = \sum_{a\in\Omega} \mu(a)a$	No

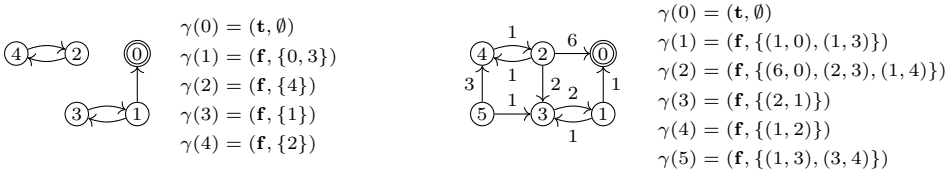


Figure 2 Examples of weighted G-graphs $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(X)$ and $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X)$.

▶ **Notations 3.8.** *We use* $\sqcap$ *and* $\sqcup$ *for the meet and join with respect to the order of* Ω*, respectively, and use* min *and* max *for the minimum and maximum with respect to the usual order of* $\mathbb{N}$ *or* $\mathbb{R}$*, respectively.*

▶ Remark 3.9. We formulated the CSPP using the greatest fixed point. A dual view is possible, reversing the order and computing the least fixed point. Our choice of the greatest fixed point is made to naturally match the usual fixed-point formulation of the SPP [24].

Here are instances of the CSPP (cf. Table 2).

▶ **Example 3.10** (SPP)**.** Let $\Omega = (\mathbb{R}^\infty_{\geq 0}, \leq, 0)$, $G\colon \mathbf{Set} \to \mathbf{Set}$ be a functor defined as $GX = \mathbb{R}_{\geq 0} \times X$ and $\sigma\colon G\Omega \to \Omega$ be a transition modality defined as $\sigma(a,b) = a + b$. For a finite set X, a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X)$ is a directed graph such that (1) X is a set of vertices, (2) a set of targets $Z = \{x \in X \mid \pi_0(\gamma(x)) = \mathbf{t}\}$ is specified, and (3) for $x, y \in X$ and $a \in \mathbb{R}_{\geq 0}$, there is an edge from x to y with length a if $(a, y) \in \pi_1(\gamma(x))$. An example of a coalgebra $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X)$ is shown on the right-hand side of Fig. 2, which is the coalgebraic representation of the weighted graph in Fig. 1. Given an Ω-valuation $d \in [X, \Omega]$, the Bellman update $\Phi^{G,\sigma}_\gamma(d)$ is given by

$$\Phi^{G,\sigma}_\gamma(d)(x) = \begin{cases} 0 & (\text{if } \gamma(x) = (\mathbf{t}, A)) \\ \min\{a + d(y) \mid (a,y) \in A\} & (\text{if } \gamma(x) = (\mathbf{f}, A)). \end{cases}$$

We can show that $\nu\Phi^{G,\sigma}_\gamma(x)$ is the value of the shortest path from x to one of the targets.

▶ **Example 3.11** (unweighted longest path)**.** Let $\Omega = (\mathbb{N}^\infty, \geq, 0)$, $G = \mathrm{Id}$ and $\sigma\colon \Omega \to \Omega$ be a transition modality defined as $\sigma(a) = 1 + a$. For a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(X)$, the Bellman operator $\Phi_\gamma^{G,\sigma}\colon [X,\Omega] \to [X,\Omega]$ is given by

$$\Phi_\gamma^{G,\sigma}(d)(x) = \begin{cases} \max(0, \max_{x\in A}(1 + d(x))) & (\text{if } \gamma(x) = (\mathbf{t}, A)) \\ \max_{x\in A}(1 + d(x)) & (\text{if } \gamma(x) = (\mathbf{f}, A)). \end{cases}$$

If γ is a directed acyclic graph (DAG), the greatest fixed point $\nu\Phi_\gamma^{G,\sigma}(x)$ is the length of the longest path from $x \in X$ to one of the targets. If γ is not a DAG, $\nu\Phi_\gamma^{G,\sigma}(x) = \infty$ holds for every x that can reach a vertex in a cycle.

▶ **Example 3.12** (SPP with negative edges)**.** We modify the SPP in Ex. 3.10 by allowing negative edge weights. Let $\Omega = (\mathbb{R}^{\pm\infty}, \leq, 0)$ and $G\colon \mathbf{Set} \to \mathbf{Set}$ be a functor defined as $GX = \mathbb{R} \times X$. Let $\sigma\colon G\Omega \to \Omega$ be a transition modality defined as $\sigma(a, b) = a + b$. For a finite set X, a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R} \times X)$ is a directed graph with targets $Z \subseteq X$. Each edge has a length, which may be negative. For $d \in [X,\Omega]$, the Bellman operator is

$$\Phi_\gamma^{G,\sigma}(d)(x) = \begin{cases} \min(0, \min\{\sigma(a, d(y)) \mid (a, y) \in A\}) & (\text{if } \gamma(x) = (\mathbf{t}, A)) \\ \min\{\sigma(a, d(y)) \mid (a, y) \in A\} & (\text{if } \gamma(x) = (\mathbf{f}, A)). \end{cases}$$

For $x \in X$, the value $\nu\Phi_\gamma^{G,\sigma}(x)$ of the greatest fixed point is the shortest path from x to one of the targets. For example, if $x \in X$ can reach a vertex in Z and is contained in a negative cycle, then $\nu\Phi(x) = -\infty$ holds.

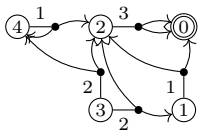
▶ **Example 3.13** (SPP with interest/discount rate)**.** We modify the SPP in Ex. 3.10 by introducing interest/discount rates. Let $\Omega = (\mathbb{R}^\infty_{\geq 0}, \leq, 0)$, as in Ex. 3.10. For a functor $GX = \mathbb{R}_{\geq 0} \times [1,\infty) \times X$, a transition modality $\sigma(a, a', b) = a + a'b$ and a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}^\infty_{\geq 0} \times [1,\infty) \times X)$, the greatest fixed point of the Bellman operator is the shortest path with interest rate. The "length" of a path $x_n \xrightarrow{(a_n, a'_n)} x_{n-1} \xrightarrow{(a_{n-1}, a'_{n-1})} \cdots \xrightarrow{(a_1, a'_1)} x_0$ is defined by $a_n + a'_n(a_{n-1} + a'_{n-1}(\cdots(a_1 + a'_1 0)\ldots))$, where a'_i are interest rates. For a functor $GX = \mathbb{R}_{\geq 0} \times [0,1] \times X$, a transition modality $\sigma(a, a', b) = a + a'b$ and a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times [0,1] \times X)$, the greatest fixed point of the Bellman operator is the shortest path with discount rate. In this case, the definition of the "length" of a path is similar to the shortest path with interest rate, but all rates a'_i satisfy $0 \leq a'_i \leq 1$.

▶ **Example 3.14** (the shortest binary tree problem)**.** We modify the SPP (Ex. 3.10) so that each edge has two successors. In this case, the problem becomes that of finding the shortest tree. Let $\Omega = (\mathbb{R}^\infty_{\geq 0}, \leq, 0)$ as in Ex. 3.10 and $G\colon \mathbf{Set} \to \mathbf{Set}$ be a functor defined as $GX = \mathbb{R}_{\geq 0} \times X^2$. Let $\sigma\colon G\Omega \to \Omega$ be a transition modality defined as $\sigma(a, b_0, b_1) = a + b_0 + b_1$. For a finite set X, a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X^2)$ is defined as follows: (1) A set of targets $Z = \{x \in X \mid \pi_0(\gamma(x)) = \mathbf{t}\}$ is specified. (2) For each vertex $x \in X$, there is a finite set $\{(a^j, y_0^j, y_1^j)\}_{j\in J_x} \subseteq \mathbb{R}_{\geq 0} \times X \times X$. An example of a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X^2)$ is shown in Fig. 3.

For an Ω-valuation $d \in [X,\Omega]$, the updated valuation $\Phi_\gamma^{G,\sigma}(d)$ is

$$\Phi_\gamma^{G,\sigma}(d)(x) = \begin{cases} 0 & (\text{if } \gamma(x) = (\mathbf{t}, A)) \\ \min_{(a,y_0,y_1)\in A}(a + d(y_0) + d(y_1)) & (\text{if } \gamma(x) = (\mathbf{f}, A)). \end{cases}$$

We define the notion of *run trees* of $\gamma\colon X \to \mathbb{B} \times \mathcal{P}(\mathbb{R}_{\geq 0} \times X^2)$: it is a rooted binary tree T of finite height such that (1) each node is labeled by a pair (x, a) of $x \in X$ and $a \in \mathbb{R}_{\geq 0}$, (2) for each leaf, its label (x, a) satisfies $x \in Z$ and $a = 0$, and (3) each node labeled by (x, a),



$\gamma(0) = (\mathbf{t}, \emptyset)$ $\quad \nu\Phi(0) = 0$
$\gamma(1) = (\mathbf{f}, \{(1,2,0)\})$ $\quad \nu\Phi(1) = 4$
$\gamma(2) = (\mathbf{f}, \{(3,0,0)\})$ $\quad \nu\Phi(2) = 3$
$\gamma(3) = (\mathbf{f}, \{(2,4,2),(2,2,1)\})$ $\quad \nu\Phi(3) = 9$
$\gamma(4) = (\mathbf{f}, \{(1,2,4)\})$ $\quad \nu\Phi(4) = \infty$

Figure 3 An example of a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X^2)$ for $X = \{0,1,2,3,4\}$.

with its children labeled by (y_i, b_i) $(i = 0, 1)$, satisfies $(a, y_0, y_1) \in \pi_1(\gamma(x))$. Given a run tree T, we define its length $\sigma(T)$ recursively as follows: (1) If T consists of a single node, then $\sigma(T) = 0$. (2) If the root of T is labeled by (x, a) and the subtrees of the root are T_0 and T_1, then $\sigma(T) = a + \sigma(T_0) + \sigma(T_1)$.

The value $\nu\Phi_\gamma^{G,\sigma}(x)$, the greatest fixed point, is the length of the shortest run tree T whose root is labeled by (x, a) for some $a \in \Omega$.

▶ **Example 3.15** (reachability games, dynamic games)**.** Let $\Omega = (\{0, \infty\}, \leq, 0)$ and $G\colon \mathbf{Set} \to \mathbf{Set}$ be the non-empty finite power set functor $\mathcal{P}_{\mathrm{fin}}^{\mathrm{ne}}$. Let $\sigma\colon G\Omega \to \Omega$ be a transition modality defined as $\sigma(A) = \bigsqcup A$. For a finite set X of vertices, a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathcal{P}_{\mathrm{fin}}^{\mathrm{ne}}(X))$ is given by the following structure: (1) A set $Z = \{x \in X \mid \pi_0(\gamma(x)) = \mathbf{t}\}$ of targets is specified. (2) For each vertex $x \in X$, there is a finite family $\{(y_i^j)_{i \in I_j}\}_{j \in J_x}$ of sets of vertices.

We define a *reachability game* on γ as follows: The game is played between two players – Alice and Bob. Starting with a pebble placed on a vertex $x \in X$, the game proceeds by iterating the following procedure.

1. If the vertex where the pebble is placed is in Z, Alice wins.
2. Otherwise, Alice chooses a label $j \in J_x$.
3. Bob chooses a label $i \in I_j$ and moves the pebble from x to y_i^j.

Starting with a pebble on a vertex $x \in X$, we ask whether Alice can always win in finitely many steps, regardless of Bob's moves. The answer is $\nu\Phi_\gamma^{G,\sigma}(x)$:

$$\nu\Phi_\gamma^{G,\sigma}(x) = \begin{cases} 0 & \text{if there is a winning strategy of Alice,} \\ \infty & \text{otherwise.} \end{cases}$$

By replacing Ω with $(\mathbb{R}_{\geq 0}^{\infty}, \leq, 0)$ and modifying G and σ accordingly (see Table 2), we obtain a game in which Alice aims to maximize her reward. The maximal reward achievable by Alice under the optimal strategy in this game coincides with the greatest fixed point.

A special case of *dynamic games* studied by Bardi and López [5] can also be modeled in our framework. Let $\Omega = (\mathbb{R}_{\geq 0}^{\infty}, \leq, \xi)$ be a weight domain, ℓ_0 and L be positive real numbers with $\ell_0 \leq L$, $G = \mathcal{P}_{\mathrm{fin}}^{\mathrm{ne}}([\ell_0, L] \times X)$ and $\sigma\colon G\Omega \to \Omega$ be a transition modality defined as $\sigma_r(A) = \max_{(a,b) \in A}(a + rb)$ for some fixed discount rate $r \in (0, 1]$. For a coalgebra $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}^{\mathrm{ne}}([\ell_0, L] \times X)$, the value $\nu\Phi_\gamma^{G,\sigma}(x)$ is the maximum reward that Alice can obtain starting from x. The CSPP on (G, σ_r) is an alternating case of the dynamic game [5].

Whether the Dijkstra acceleration works in this setting is a subtle issue. It is discussed later in Ex. 4.15.

▶ **Example 3.16** (the maximum probabilistic reachability)**.** Let $\Omega = ([0,1], \geq, 1)$ and $G\colon \mathbf{Set} \to \mathbf{Set}$ be the distribution functor $G = \mathcal{D}$. Let $\sigma\colon \mathcal{D}\Omega \to \Omega$ be a transition modality defined as $\sigma(\mu) = \sum_{a \in \Omega} \mu(a)a$. For a finite set X, a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathcal{D}X)$ is given by the following structure: (1) A set $Z = \{x \in X \mid \pi_0(\gamma(x)) = \mathbf{t}\}$ is specified. (2) For each vertex $x \in X$, there is a finite set $\{\mu_j\}_{j \in J_x}$ of probabilistic distributions on X.

For an Ω-valuation $d \in [X, \Omega]$, the updated valuation by the Bellman operator $\Phi_\gamma^{G,\sigma}(d)$ is

$$\Phi_\gamma^{G,\sigma}(d)(x) = \begin{cases} 1 & (\text{if } \gamma(x) = (\mathbf{t}, A)) \\ \max_{\mu \in A} \sum_{y \in X} \mu(y) d(y) & (\text{if } \gamma(x) = (\mathbf{f}, A)). \end{cases}$$

We define a one-player pebble game on γ. Starting with a pebble placed on a vertex $x \in X$, the game proceeds by iterating the following procedure:

1. If the vertex where the pebble is placed is in Z, the player wins.
2. Otherwise, the player chooses a label $j \in J_x$.
3. The pebble moves from x to $y \in X$ with probability $\mu_j(y)$.

The value $\nu\Phi_\gamma^{G,\sigma}(x)$ is the maximum winning probability of the player.

4 The coalgebraic Dijkstra algorithm for the CSPP

We now introduce our coalgebraic generalization of the Dijkstra algorithm (Algo. 1). Its correctness relies delicately on precise problem settings (Table 2); we present general axiomatics that capture the difference. We introduce the notion of *finitely supported functor* here.

As discussed in § 1, here we present the most general axiomatics, where G is an arbitrary finitely supported functor.

In our algorithm, we use the following coalgebraic notion of *predecessors*.

▶ **Definition 4.1** (predecessor and successor [17, Def. 3.4]). *Let $F\colon \mathbf{Set} \to \mathbf{Set}$ be a functor, $\gamma\colon X \to FX$ be a coalgebra, and $Y \subseteq X$. An element $x \in X$ is a* predecessor *of Y if*

$$x \in X \setminus \{z \in X \mid \gamma(z) \in F(X \setminus Y)\}, \quad \textit{that is,} \quad x \in \{x \in X \mid \gamma(x) \notin F(X \setminus Y)\}.$$

We define $\mathrm{Pred}(Y) := \{x \in X \mid x \text{ is a predecessor of } Y\}$. *For $x, y \in X$, y is a* successor *of x if x is a predecessor of $\{y\}$. We define* $\mathrm{Succ}(x) := \{y \in X \mid y \text{ is a successor of } x\}$.

▶ **Definition 4.2** ($\textsc{CoalgDijkstra}_{(G,\sigma)}$). *Let $\Omega = (\Omega, \sqsubseteq, \xi)$ be a pointed weight domain, $G\colon \mathbf{Set} \to \mathbf{Set}$ a functor, and $\sigma\colon G\Omega \to \Omega$ a transition modality. The* coalgebraic Dijkstra algorithm *$\textsc{CoalgDijkstra}_{(G,\sigma)}$ for the CSPP on (G, σ) (cf. Prob. 3.7) is Algo. 2.*

It generalizes the (classical) Dijkstra algorithm for the SPP (Algo. 1). It is almost the same as Algo. 1, sharing the two key features (frozen states S and selective Bellman update). The difference is almost solely in Line 7, where Algo. 2 uses a generic Bellman operator $\Phi_\gamma^{G,\sigma}$ – parametrized in a transition modality σ – as opposed to a specific one in Algo. 1.

▶ **Notations 4.3.** *We use the following notational convention for Algo. 2. For d, S, and Y at the beginning of the n-th iteration ($n = 1, 2, \dots$) of the main loop in Algo. 2, we write d_n, S_n, and Y_n, respectively. We also set $d_0 = \top_{[X,\Omega]}$.*

The following example illustrates how Algo. 2 operates on a tree-like structure; it is to be compared with a path-like example in Ex. 2.4.

▶ **Example 4.4** (shortest binary tree, continued from Ex. 3.14). Let $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{R}_{\geq 0} \times X^2)$ be the weighted G-graph in Fig. 3. The execution of Algo. 2 on γ is in Table 3.

It is easy to see the termination of Algo. 2 for finite X: S grows strictly in every iteration.

▶ **Proposition 4.5.** *If the state space X of the weighted G-graph is finite, Algo. 2 terminates.*

Correctness requires delicate conditions on (G, σ). Nevertheless, it holds in general that the sequence $(d_i)_i$ of valuations, constructed iteratively in Algo. 2, is descending.

▶ **Lemma 4.6.** *For a finite set X and a weighted G-graph $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(GX)$, we have $\top = d_0 \sqsupseteq d_1 \sqsupseteq d_2 \sqsupseteq \cdots \sqsupseteq \nu\Phi_\gamma^{G,\sigma}$.*

Algorithm 2 The coalgebraic Dijkstra algorithm for the CSPP on $(G, \sigma\colon G\Omega \to \Omega)$.

procedure $\textsc{CoalgDijkstra}_{(G,\sigma)}(\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(GX))$
  $S \leftarrow \{x \in X \mid \pi_0(\gamma(x)) = \mathbf{t}\}$
  $Y \leftarrow S$
  $d \leftarrow \lambda x \in X. \begin{cases} \xi & (x \in S) \\ \top_\Omega & (x \notin S) \end{cases} \in [X, \Omega]$ } Initialization
  while $S \neq X$ **do**
    $P \leftarrow \mathrm{Pred}(Y)$
    $d \leftarrow \lambda x. \begin{cases} \Phi_\gamma^{G,\sigma}(d)(x) & (x \in P \cap (X \setminus S)) \\ d(x) & (\text{otherwise}) \end{cases}$ ▷ Selective Bellman update
    $Y \leftarrow \left\{ y \in X \setminus S \;\middle|\; d(y) = \prod_{z \in X \setminus S} d(z) \right\}$ ▷ Minimal freezing
    $S \leftarrow S \cup Y$ ▷ Update the set of frozen states
  end while
  return d
end procedure

Table 3 The run of Algo. 2 for the weighted G-graph γ in Fig. 3.

n	$d_n(0)$	$d_n(1)$	$d_n(2)$	$d_n(3)$	$d_n(4)$	S_n	Y_n
0	∞	∞	∞	∞	∞		
1	0	∞	∞	∞	∞	$\{0\}$	$\{0\}$
2	0	∞	3	∞	∞	$\{0, 2\}$	$\{2\}$
3	0	4	3	∞	∞	$\{0, 1\}$	$\{1\}$
4	0	4	3	9	∞	$\{0, 1, 3\}$	$\{3\}$
5	0	4	3	9	∞	$\{0, 1, 3, 4\}$	$\{4\}$

4.1 Correctness for finitely supported functors

We discuss correctness of the coalgebraic Dijkstra algorithm, presenting an abstract necessary and sufficient condition (Thm. 4.14) that captures the delicate difference in Table 2.

▶ **Definition 4.7.** *We say that* $\textsc{CoalgDijkstra}_{(G,\sigma)}$ *in Algo. 2 is* correct *if it returns the solution to the CSPP on* (G, σ)*, that is,* $d = \nu\Phi_\gamma^{G,\sigma}$.

We use the following technical notion in the discussions below. Intuitively, the subset $\Omega^\sigma \subseteq \Omega$ collects all the elements of Ω that can possibly appear in Algo. 2.

▶ **Definition 4.8.** *We define the subset* $\Omega^\sigma \subseteq \Omega$ *using the recursion* $\Omega_0^\sigma = \{\xi, \top_\Omega\}$, $\Omega_{n+1}^\sigma = \Omega_n^\sigma \cup \{\sigma(a) \mid a \in G(\Omega_n^\sigma)\}$*, for all* $n \in \mathbb{N}$*, such that* $\Omega^\sigma = \bigcup_{n=0}^\infty \Omega_n^\sigma$*. Here* ξ *is from Def. 3.3.*

We introduce the following categorical axiom for a functor G.

▶ **Definition 4.9** (finitely supported functor)**.** *Let* $G\colon \mathbf{Set} \to \mathbf{Set}$ *be a functor. A* finite support *for* G *is a natural transformation* $\theta\colon G \to \mathcal{P}_{\mathrm{fin}}$*, to the finite power set functor* $\mathcal{P}_{\mathrm{fin}}$*, such that*

for every set X *and* $t \in GX$*,* $\theta_X(t)$ *is the smallest subset* $S \subseteq X$ *such that* $t \in GS$.

It is obvious that a finite support θ *for* G*, if such exists, is unique.*

A functor G *is called* finitely supported *if it admits a finite support. A finite support* θ *is called* non-empty *if* $\theta_X(t)$ *is not empty for every set* X *and* $t \in GX$.

The class of finitely supported functors is broad. It contains the identity functor, the (non-empty) finite powerset functor $\mathcal{P}_{\mathrm{fin}}, \mathcal{P}_{\mathrm{fin}}^{\mathrm{ne}}$, the distribution functor $\mathcal{D}$, and constant functors (Lem. 4.10). The class is closed under sum, product, and composition (Lem. 4.11). In particular, every polynomial functor is finitely supported. Concretely, we define the following.

1. $\theta^{\mathrm{Id}}_X \colon X \to \mathcal{P}_{\mathrm{fin}}(X)$ is defined as $\theta^{\mathrm{Id}}_X(x) = \{x\}$.
2. $\theta^{\mathcal{P}_{\mathrm{fin}}}_X \colon \mathcal{P}_{\mathrm{fin}}(X) \to \mathcal{P}_{\mathrm{fin}}(X)$ is defined as $\theta^{\mathcal{P}_{\mathrm{fin}}}_X(A) = A$.
3. $\theta^{\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}}_X \colon \mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}(X) \to \mathcal{P}_{\mathrm{fin}}(X)$ is defined as $\theta^{\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}}_X(A) = A$.
4. $\theta^{\mathcal{D}}_X \colon \mathcal{D}(X) \to \mathcal{P}_{\mathrm{fin}}(X)$ is defined as $\theta^{\mathcal{D}}_X(\rho) = \{x \in X \mid \rho(x) > 0\}$.
5. For a set V, $\theta^{V}_X \colon V \to \mathcal{P}_{\mathrm{fin}}(X)$ is defined as $\theta^{V}_X(v) = \emptyset$.

▶ **Lemma 4.10.** *The natural transformations θ^{Id}, $\theta^{\mathcal{P}_{\mathrm{fin}}}$, $\theta^{\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}}$, $\theta^{\mathcal{D}}$ and θ^{V} defined above are finite supports for* Id, $\mathcal{P}_{\mathrm{fin}}$, $\mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}$, $\mathcal{D}$ *and* V, *respectively.*

▶ **Lemma 4.11.** *If G_0 and G_1 are finitely supported functors, then so are $G_0 + G_1$, $G_0 \times G_1$ and $G_1 \circ G_0$.*

One may wonder about the relationship between Def. 4.9 and another well-established notion with a similar intuition, namely *finitariness* (see e.g. [1]). We have the following result.

▶ **Proposition 4.12.** *If a functor $F \colon \mathbf{Set} \to \mathbf{Set}$ is finitary and taut, then F is finitely supported.*

The proof is based on Manes's results on finitary and taut functors [21, 22]. The tautness assumption is necessary: the naturality of θ fails without it; and it is known that a finitary functor may not be taut [21]. The converse does not hold: there is a finitely supported functor that is not taut. Lem. 4.10 is an immediate consequence of Prop. 4.12.

We now establish the correctness of the coalgebraic Dijkstra algorithm. The core of our categorical axiomatics is the notion of *expansiveness* of G-algebra $\sigma \colon G\Omega \to \Omega$ (as part of a transition modality, Def. 3.5). Intuitively, expansiveness means that the accumulation of weights along a path does not decrease. For example, in the SPP (Ex. 3.10), the transition modality is given by $\sigma(a, b) = a + b$, and expansiveness amounts to the condition $b \leq a + b$ for all $a \in \mathbb{R}_{\geq 0}$ and $b \in \mathbb{R}^{\infty}_{\geq 0}$.

▶ **Definition 4.13** (expansiveness). *Let Ω be a pointed weight domain (Def. 3.3), $G \colon \mathbf{Set} \to \mathbf{Set}$ be a functor and θ be a finite support of G. A G-algebra $\sigma \colon G\Omega \to \Omega$ is called* expansive *if $b \sqsubseteq \sigma(t)$ for every $t \in G\Omega^{\sigma}$ and $b \in \theta(t)$.*

We now state our main theorem. To show (2⇒1), we assume that σ is not expansive and construct a counterexample on which Algo. 2 is not correct.

▶ **Theorem 4.14** (correctness). *Let $G \colon \mathbf{Set} \to \mathbf{Set}$ be a weak-pullback-preserving, non-empty finitely supported functor. Let $\sigma \colon G\Omega \to \Omega$ be a transition modality on a pointed weight domain Ω. The following statements are equivalent:*

1. *The transition modality σ is expansive.*
2. *For every weighted G-graph $\gamma \colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(GX)$, $\textsc{CoalgDijkstra}_{(G,\sigma)}$ is correct.*

▶ **Example 4.15** (the dynamic game, continued from Ex. 3.15). Recall Ex. 3.15, where the weight domain is $(\mathbb{R}^{\infty}_{\geq 0}, \leq, \xi)$, the functor G is $GX = \mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}([\ell_0, L] \times X)$, and the transition modality $\sigma_r \colon \mathcal{P}^{\mathrm{ne}}_{\mathrm{fin}}([\ell_0, L] \times \Omega) \to \Omega$ is $\sigma_r(A) = \max_{(a,b)\in A}(a + rb)$ for a fixed discount rate $r \in (0, 1]$. The CSPP on (G, σ_r) is a special case[2] of the dynamic game studied by Bardi and López [5]. The paper has the following condition [5, Condition 1] for their Dijkstra acceleration to work when $r < 1$:

$$L \sum_{j=0}^{n-1} r^j + r^n \xi \;\leq\; \frac{\ell_0}{1 - r} \quad \text{for all } n < N, \tag{4}$$

[2] In our setting, two players make moves alternately, an (inessential) restriction not present in [5].

where N is the number of steps in which their Dijkstra-style algorithm terminates. This condition uses universal quantification over n, which in general makes its verification harder.

We note that, in the earlier version [4] of the same paper, the following condition is given:

$$L + r\xi \;\leq\; \frac{\ell_0}{1-r}. \tag{5}$$

This condition (5) in the earlier version [4] has apparently been corrected to (4) in the published version [5]. The condition (4) is stricter, and harder to check (due to universal quantification).

This example demonstrates that conditions for the Dijkstra acceleration to work can be subtle, and that our necessary and sufficient condition – categorically formulated in Thm. 4.14 – is useful. It is easy to see that the earlier condition (5) does not imply the expansiveness of σ, meaning our theory can raise a red flag about the conjectured condition (5).

It can be seen that the corrected condition (4) – to be precise, its uniform version which requires (4) for any $n \in \mathbb{N}$ – is essentially equivalent to our condition of expansiveness. Indeed, the uniform version of (4) implies $\ell_0 = L$, forcing all the moves to have the same stepwise reward. This severely restricts the expressivity of the studied game-like formalism.

4.2 Complexity

We fix a pointed weight domain Ω, a functor $G\colon \mathbf{Set} \to \mathbf{Set}$, and a transition modality $\sigma\colon G\Omega \to \Omega$. Let X be a finite set and $\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(GX)$ be a weighted G-graph. We assume that it takes $\mathcal{O}(T)$ time to compute $\Phi_\gamma^{G,\sigma}(d)(x)$ for $d\colon X \to \Omega$, $x \in X$. We define $V := \#X$ and $E := \sum_{x\in X} \#\mathrm{Succ}(x)$. Furthermore, assume that for each $y \in X$, the set of predecessors $\mathrm{Pred}(y)$ is precomputed. We have the following complexity result.

▶ **Proposition 4.16.** *The time complexity of Algo. 2 is $\mathcal{O}(TE + V^2)$.*

It is well known that the complexity of the classical Dijkstra algorithm can be improved by the use of a Fibonacci heap [12], a data structure implementing a priority queue. For Algo. 2, we can use a Fibonacci heap to improve the complexity of the computation of Y.

Furthermore, for a more precise complexity result, we replace Line 7 in Algo. 2 with

$$d \leftarrow \lambda x. \begin{cases} d(x) \sqcap \prod_{\substack{a\in\pi_1(\gamma(x)) \\ \theta^G_X(a)\cap Y\neq\emptyset}} \sigma(Gd(a)) & \text{if } x \in P \cap (X \setminus S) \\ d(x) & \text{otherwise.} \end{cases}$$

Intuitively, it is unnecessary to recompute the entire $\Phi_\gamma^{G,\sigma}(d)(x)$ each time; only the portion related to the set Y needs to be updated.

Now, Algo. 3 is an improvement of Algo. 2 using a Fibonacci heap and the above replacement. For a refined complexity analysis, we assume it takes $\mathcal{O}(T')$ time to compute $\sigma((Gd)(a))$, where $d\colon X \to \Omega$, $x \in X$ and $a \in GX$.

▶ **Theorem 4.17.** *When using the Fibonacci heap as a priority queue Q, the time complexity of Algo. 3 is $\mathcal{O}(T'E + V \log V)$.*

This complexity coincides with the classical complexity result for the (classical) Dijkstra (namely $\mathcal{O}(E + V \log V)$ – note that $T' = \mathcal{O}(1)$ for the SPP).

Algorithm 3 The coalgebraic Dijkstra algorithm with a Fibonacci heap.

procedure $\textsc{CoalgDijkstra}^{\mathrm{F}}_{(G,\sigma)}(\gamma\colon X \to \mathbb{B} \times \mathcal{P}_{\mathrm{fin}}(GX))$
  $S \leftarrow \{x \in X \mid \pi_0(\gamma(x)) = \mathbf{t}\}$
  $Y \leftarrow S$
  $d \leftarrow \lambda x \in X. \begin{cases} \xi & (x \in S) \\ \top_\Omega & (x \notin S) \end{cases} \in [X, \Omega]$
  Let Q be the empty Fibonacci heap.
  while $S \neq X$ **do**
    $P \leftarrow \mathrm{Pred}(Y)$
    $d \leftarrow \lambda x. \begin{cases} d(x) \sqcap \bigsqcap_{a \in \pi_1(\gamma(x)),\ \theta^G_X(a) \cap Y \neq \emptyset} \sigma(Gd(a)) & \text{if } x \in P \cap (X \setminus S) \\ d(x) & \text{otherwise.} \end{cases}$ ▷ Selective Bellman update
    for $x \in P \cap (X \setminus S)$ **do**
      if the key x is in Q **then**
        Update the value of the key x to $d(x)$ in Q
      else
        Insert $(d(x), x)$ to Q
      end if
    end for
    Get $\{(a, y_0), \ldots, (a, y_m)\}$ from Q such that a is the minimum in Q
    Delete $(a, y_0), \ldots, (a, y_m)$ from Q
    $Y \leftarrow \{y_0, \ldots, y_m\}$ ▷ Minimal freezing
    $S \leftarrow S \cup Y$ ▷ Update the set of frozen states
  end while
  return d
end procedure

5 Conclusion and Future Work

We presented a categorical generalization of the Dijkstra-style acceleration in various graph-based optimization problems. Whether Dijkstra is correct relies delicately on problem settings (Table 2); we presented a categorical necessary and sufficient condition, formulated on the notions of finitely supported functor and expansive transition modality. Our general complexity analysis generalizes the classical one, too.

We used **Set** as the space of collections of states. One direction for future work would be to determine whether it is possible to generalize **Set** to arbitrary categories $\mathbb{C}$, in order to obtain the coalgebraic Dijkstra algorithm over topological spaces or nominal sets.

Further, the background of the definition of the Bellman operator $\Phi^{G,\sigma}_\gamma$ lies in the theory of fibrations. In this paper, we did not explicitly use the notion of fibrations. Another direction for future work is to examine whether the fibrational viewpoint provides useful applications or generalizations.

References

1 Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. Cambridge University Press, 1994.

2 Alejandro Aguirre, Shin-ya Katsumata, and Satoshi Kura. Weakest preconditions in fibrations. *Math. Struct. Comput. Sci.*, 32(4):472–510, 2022. doi:10.1017/S0960129522000330.

3 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

4 Martino Bardi and Juan Pablo Maldonado López. A Dijkstra-type algorithm for dynamic games. *HAL*, hal-00881218v2, 2015.

5 Martino Bardi and Juan Pablo Maldonado López. A Dijkstra-type algorithm for dynamic games. *Dyn. Games Appl.*, 6(3):263–276, 2016. doi:10.1007/S13235-015-0156-0.

6 Simone Barlocco, Clemens Kupke, and Jurriaan Rot. Coalgebra learning via duality. In *FoSSaCS*, volume 11425 of *LNCS*, pages 62–79. Springer, 2019. doi:10.1007/978-3-030-17127-8_4.

7 Richard Bellman. On a routing problem. *Q. Appl. Math.*, 16(1):87–90, 1958.
8 Patrick Cousot and Radhia Cousot. Constructive versions of Tarski's fixed point theorems. *Pac. J. Math.*, 82(1):43–57, 1979.
9 Hans-Peter Deifel, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Generic partition refinement and weighted tree automata. In *FM*, volume 11800 of *LNCS*, pages 280–297. Springer, 2019. doi:10.1007/978-3-030-30942-8_18.
10 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959. doi:10.1007/BF01386390.
11 Lester R. Ford Jr. Network flow theory. Technical report, Rand Corporation Paper, Santa Monica, 1956.
12 Michael L. Fredman and Robert E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.
13 Michel Gondran and Michel Minoux. *Graphs, Dioids and Semirings*. Springer, 2008.
14 Ichiro Hasuo. Generic weakest precondition semantics from monads enriched with order. *Theor. Comput. Sci.*, 604:2–29, 2015. doi:10.1016/J.TCS.2015.03.047.
15 Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge University Press, 2016. doi:10.1017/CBO9781316823187.
16 Bart Jacobs and Alexandra Silva. *Automata Learning: A Categorical Perspective*, volume 8464 of *LNCS*, pages 384–406. Springer, 2014. doi:10.1007/978-3-319-06880-0_20.
17 Jules Jacobs and Thorsten Wißmann. Fast coalgebraic bisimilarity minimization. *Proc. ACM Program. Lang.*, 7(POPL):1514–1541, 2023. doi:10.1145/3571245.
18 Ryota Kojima and Corina Cîrstea. Continuation semantics for fixpoint modal logic and computation tree logics. In *MFPS*, volume 5 of *ENTICS*. EpiSciences, 2025. doi:10.46298/ENTICS.16653.
19 Mayuko Kori and Kazuki Watanabe. On coalgebraic product constructions for Markov chains and automata. *CoRR*, abs/2504.06592, 2025. doi:10.48550/arXiv.2504.06592.
20 Daniel J. Lehmann. Algebraic structures for transitive closure. *Theor. Comput. Sci.*, 4(1):59–76, 1977. doi:10.1016/0304-3975(77)90056-1.
21 Ernest G. Manes. Implementing collection classes with monads. *Math. Struct. Comput. Sci.*, 8(3):231–276, 1998. URL: http://journals.cambridge.org/action/displayAbstract?aid=44747.
22 Ernest G. Manes. Taut monads and T0-spaces. *Theor. Comput. Sci.*, 275(1-2):79–109, 2002. doi:10.1016/S0304-3975(00)00415-1.
23 René Mazala. *Infinite Games*, volume 2500 of *LNCS*, pages 23–42. Springer, 2001. doi:10.1007/3-540-36387-4_2.
24 Jayadev Misra. A walk over the shortest path: Dijkstra's algorithm viewed as fixed-point computation. *Inf. Process. Lett.*, 77(2-4):197–200, 2001. doi:10.1016/S0020-0190(00)00202-7.
25 Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3):321–350, 2002. doi:10.25596/JALC-2002-321.
26 Maurice Pollack. Letter to the editor—the maximum capacity through a network. *Oper. Res.*, 8(5):733–736, 1960.
27 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.
28 Takahiro Sanada, Ryota Kojima, Yuichi Komorida, Koko Muroya, and Ichiro Hasuo. Explicit Hopcroft's trick in categorical partition refinement. In *CMCS*, volume 14617 of *LNCS*, pages 135–155. Springer, 2024. doi:10.1007/978-3-031-66438-0_7.
29 João L. Sobrinho. Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet. *IEEE/ACM Trans. Netw.*, 10(4):541–550, 2002. doi:10.1109/TNET.2002.801397.
30 Omar Wing. Algorithms to find the most reliable path in a network. *IRE Trans. Circuit Theory*, 8(1):78–79, 1961.