Hierarchical Multiobjective Shortest Path Problems

Konstantin Slutsky*, Dmitry Yershov†,
Tichakorn Wongpiromsarn‡, and Emilio Frazzoli†

*CNRS/Université Paris Diderot, Paris, France
kslutsky@gmail.com

†Hyundai-Aptiv AD, Boston, USA
{dmitry,emilio}@nutonomy.com

‡University of Texas at Austin, USA
tichakorn@utexas.edu

Abstract. We consider the shortest path problem on graphs with weights taking values in Cartesian products of cost monoids. Such cost structures appear in multiobjective planning including, for instance, the minimum-violation planning framework. It is known that these products often do not satisfy the conditions of a cost monoid. Classical dynamic programming graph search algorithms may therefore fail to find an optimal solution.
We isolate the concept of a regular cost monoid and propose an iterative search algorithm that finds an optimal path in graphs weighted by products of such costs. Our algorithm allows this class of multiobjective planning problems to be solved in polynomial time.

1 Introduction

Computing the shortest path between two vertices (called the root and the goal) on a weighted graph is a central problem in robot motion planning [15], and it is of particular interest to autonomous vehicle navigation [7, 10, 16]. In the classical shortest path problem (SPP), edge weights are positive real numbers that determine the cost of traversing a given edge, and the path cost is the sum of costs of its constituting edges. A plethora of algorithms have been discovered for computing the shortest path with respect to the path cost objective (e.g., Dijkstra's algorithm [5]) and minimizing the number of operations needed to find a solution (e.g., A* algorithm [12]).

Konstantin Slutsky's research is partially supported by the ANR project AGRUME (ANR-17-CE40-0026).
Tichakorn Wongpiromsarn's research is partially supported by AFC Robotics Center of Excellence with award number W911NF1920333.

However, the desire to quantify complex robot behavior in a vast number of scenarios often requires considering a trade-off between many competing objectives. In the case of navigating autonomous cars in urban environments, for example, these objectives may capture traffic laws, ethics, liability, local driving culture, and ride quality. Previous multiobjective shortest path problem (MSPP) formulation distills the trade-off into a solution set of Pareto optimal paths that connect the root to the goal [9]. In [1,18,20], Dijkstra's and A* algorithms have been adapted for solving the MSPP. It was shown in [17] that memory requirements for storing Pareto optimal solution set can be polynomial in MSPP size. However, the time complexity of all known algorithms is exponential in the worst case. Two roadblocks remain for practical applications of the MSPP framework: i) the problem of finding a Pareto optimal set is known to be NP-hard [11], and ii) the trade-off remains in the solution set and must be resolved using a separate procedure.

We have previously considered the second of these difficulties and advocated for using a rulebook-based approach [3], in which a *rule* defines a real-valued functional on the set of vehicle paths, and a *rulebook* defines a hierarchy from the most to the least critical rule. In other words, the rulebook approach is a generalization of the minimum-violation planning framework [2,21,22] in that it produces a path with the least severe violation of rules (preferably no violation at all). More importantly, a rulebook-based solution is in a Pareto optimal set; hence, our approach can be used to address both practical limitations of the MSPP.

We consider a generalization of the SPP (or just GSPP for brevity), in which graph edge weights are elements of a monoid with a total order, and the path cost is defined as an accumulation of all costs of its edges with respect to the monoid operation. Since the set of positive real numbers ($\mathbb{R}^{\geq 0}$) is a totally ordered monoid with respect to the addition operation (which we denote $\mathbf{R}_+$), such an extension is indeed a generalization of the classical SPP. Moreover, this formulation generalizes the previous minimum-violation planning framework, in which edge cost can be represented by the lexicographically ordered monoid $(\mathbb{R}^{\geq 0})^n$ equipped with the coordinate-wise summation operation.

The set up of GSPP is closely related to the one considered in [6], where the important notion of a cost monoid[1] has been introduced (see also [14]). We are particularly interested in multiobjective weights, i.e., products of cost monoids, which we call multicosts. It is also worth mentioning that monoid-type cost structures are similar to those often considered in the so-called algebraic path problem (see, for instance, [23, Section 8], [13]).

In addition to classical SPP and minimum-violation framework, in which edge costs are accumulated along graph paths using the addition operation, GSPP enables using maximum operation $\mathbf{R}_{\mathrm{m}} = (\mathbb{R}^{\geq 0}, \max)$, as well as any other binary operations subject to monoid axioms. In [3], we found max particularly useful

[1] The term used in [6] is "cost algebra". We use a slightly different (but equivalent) presentation that requires only one binary operation, and therefore choose to speak of a "cost monoid" instead.

for accurate mathematical modeling of "nonadditive" rules, such as keeping a distance from other road users, for example. In the context of the MSPP, a single nonadditive cost component, a so-called "bottleneck", have been considered in [8], which expanded on the previous dynamic programming algorithm introduced in [19]. Furthermore, the MSPP with at least two bottlenecks has been studied in [4]. In this setting, our approach can be considered as an efficient strategy for finding a path with arbitrary many bottlenecks according to a set of prioritized rules.

It is known that traditional graph search algorithms may fail to find an optimal solution, for example, if the monoid operation is a product of max and sum (see, for instance, [6]). The main contribution of our paper is the notion of regularity for cost monoids (Section 5) together with an iterative algorithm (Section 6) that finds an optimal path in graphs weighted by regular multicosts (i.e., prioritized products of regular cost monoids, of which $\mathbf{R}_+$ and $\mathbf{R}_\mathrm{m}$ are examples).

The paper is structured as follows. Section 2 introduces notions and definitions used throughout the article. Section 3 shows how Dijkstra's algorithm may fail to find an optimal path when the weights do not satisfy the axioms of a cost monoid. The concept of an optimal subgraph (relative to the root and the goal vertices) is introduced in Section 4, where we also present an efficient algorithm of computing it. Section 5 is devoted to the notion of regularity of costs. We provide an algebraic characterization of regularity for cost monoids, show that the classical monoids $\mathbf{R}_+$ and $\mathbf{R}_\mathrm{m}$ are regular, give an example of a non-regular monoid, and argue that the property of being regular is closed under taking direct products with lexicographic order. Finally, in Section 6, we present the iterative algorithm of computing optimal paths in graphs weighted by regular multicosts.

2 Shortest Path Problem on Directed Graphs with Cumulative Algebraic Costs

We consider a generalization of the Shortest Path Problem (SPP) on directed graphs by replacing real-valued edge weights with elements of monoids with an order. Using this weight structure, we define cumulative path costs and the notion of path optimality. This formulation is similar to the one given in [6], with the notable difference being that we do not require the operation $*$ to be isotone.

Definition 1. *We use the following algebraic concepts.*

- *A* ***monoid*** $(S, *, \mathbf{1})$ *is a set* S *with an associative binary operation* $*$ *and an element* $\mathbf{1}$*, called the* ***unit****, such that* $a * \mathbf{1} = \mathbf{1} * a = a$ *for all* $a \in S$*.*
- *A quadruple* $(S, *, \mathbf{1}, \leq)$*, where* $(S, *, \mathbf{1})$ *is a monoid and* $\leq$ *is a total order on* S*, is a* ***monoid with an order****. We use the notation* $<$ *to mean the strict order induced by* $\leq$*, i.e.,* $a < b \iff a \neq b \wedge a \leq b$*. If operations* $*$ *and* $\leq$ *are unambiguous, we may simply use* S *instead of the tuple.*

- *The operation $*$ is **isotone** if $a \leq b$ implies $a * c \leq b * c$ and $c * a \leq c * b$ for all $a, b, c \in S$.*
- *A monoid with an order S is said to be a **cost monoid**[2] if the operation $*$ is isotone and $\mathbf{1}$ is the smallest element: $\mathbf{1} \leq a$ for all $a \in S$.*
- *A monoid S is said to be **cancellative**[3] if for all $a, b, c \in S$*

$$a * c = b * c \implies a = b \text{ and } c * a = c * b \implies a = b.$$

- *A **product of monoids with an order** $S_1, \ldots, S_n$ is defined to be the monoid $S_1 \times \cdots \times S_n$ equipped with the coordinatewise operation and lexicographic order.*

The two best known examples of cost monoids are $\mathbf{R}_+ = (\mathbb{R}^{\geq 0}, +, 0, \leq)$ and $\mathbf{R}_{\max} = (\mathbb{R}^{\geq 0}, \max, 0, \leq)$. As pointed out in [6], a product of cost monoids may not be a cost monoid, as the $*$ operation may fail to be isotone relative to the lexicographic order. Indeed, $\mathbf{R}_{\max} \times \mathbf{R}_+$ is not a cost monoid: for $a = (0, 1), b = (1, 0), c = (1, 1)$ one has $a < b$, but $a * c > b * c$. Nonetheless, if $S_1, \ldots, S_n$ are cost monoids and S_i are cancellative for all $1 \leq i \leq n-1$, then $S_1 \times \cdots \times S_n$ is a cost monoid. For instance, $\mathbf{R}_+$ is cancellative, while $\mathbf{R}_{\max}$ is not. As noted above, $\mathbf{R}_{\max} \times \mathbf{R}_+$ is not a cost monoid, while the product $\mathbf{R}_+ \times \mathbf{R}_{\max}$ is.

Definition 2. *A **multicost** is a monoid with an order represented as a product of cost monoids $S_1 \times \cdots \times S_n$.*

Note that a monoid with an order may, in general, be represented as a product of cost monoids in different ways. For instance

$$(\mathbf{R}_+ \times \mathbf{R}_+) \times \mathbf{R}_{\max} \text{ and } \mathbf{R}_+ \times (\mathbf{R}_+ \times \mathbf{R}_{\max})$$

are two different factorizations of the same monoid with an order. The definition of a multicost assumes that a particular factorization is chosen, and in particular, the monoids above will be viewed as two different multicosts depending on which factorization is considered.

A directed graph $\mathcal{G} = (V, E)$ is determined by a set of vertices V, and a set of directed edges $E \subseteq V \times V$. In addition, we equip directed graphs with two maps: the **origin** $\mathrm{o} : E \to V$ and the **target** $\mathrm{t} : E \to V$, which assign to an edge its first and second vertex, respectively.

An (edge) **path** in a graph is a finite sequence of edges $\{e_i\}_{i=1}^n$, such that $\mathrm{t}(e_i) = \mathrm{o}(e_{i+1})$ for all $1 \leq i < n$. We let $\mathcal{P}$ denote the set of all paths in $\mathcal{G}$.

[2] This definition is essentially equivalent to the notion of the cost algebra given in [6]. Since the operations $\sqcup$ and $\sqcap$ used therein are uniquely defined by the order, we choose to speak about monoids rather than algebras. Also, the definition of a cost algebra postulates the existence of a maximal element. It is notationally more convenient for us to avoid making this assumption, but these two approaches are equivalent, as any cost monoid can be enlarged by adding a maximal element.

[3] A cost monoid is cancellative if and only if the operation $*$ is strictly isotone: $a < b$ implies $a * c < b * c$ and $c * a < c * b$ for any element c.

Maps o and t extend to $\mathcal{P}$ by setting $\mathrm{o}(p) = \mathrm{o}(e_1)$ and $\mathrm{t}(p) = \mathrm{t}(e_n)$. Finally, for a pair of vertices $u, v \in V$, we define the set of paths between these vertices $\mathcal{P}(u, v) = \{p \in \mathcal{P} \mid \mathrm{o}(p) = u \wedge \mathrm{t}(p) = v\}$.

Definition 3. *We consider weighted graphs defined as follows.*

- *Given a monoid with an order S, a* ***weight*** *on $\mathcal{G}$ (or an S-**weight** if we need to be precise) is a function $w : E \to S$.*
- *The weight function w is extended to the set of all paths $w : \mathcal{P} \to S$ by setting for a path $p \in \mathcal{P}$, $p = (e_1, \ldots, e_n)$,*

$$w(p) = w(e_1) * w(e_2) * \cdots * w(e_n).$$

We refer to the value $w(p)$ as the ***cost*** *of the path.*

Note that $E \subseteq \mathcal{P}$, and moreover the definition of w on $\mathcal{P}$ extends its definition on E. Hence, edge weights can also be referred to as costs.

Finally, we are ready to define the notion of an optimal path and formulate the generalized Shortest Path Problem.

Definition 4. *A path p in a weighted graph $\mathcal{G}$ is said to be* ***optimal*** *if*

$$w(p) \leq w(q) \text{ for all } q \in \mathcal{P}(\mathrm{o}(p), \mathrm{t}(p)).$$

Problem 1 (Generalized Shortest Path Problem (GSPP)). Given an S-weighted graph $\mathcal{G}$, where S is a monoid with an order, and two dedicated vertices—the root v_r and the goal v_g—find an optimal path p in the set $\mathcal{P}(v_r, v_g)$ or determine that no such path exists.

Note that if we let the monoid S be the set of the nonnegative real numbers with addition $\left(\mathbb{R}^{\geq 0}, +, 0, \leq\right)$, then GSPP reduces to the classical SPP.

Remark 1. The setup of GSPP imposes very weak requirements on the operation that composes costs of individual edges, but it stays within the realm of totally ordered values. The concept of Pareto optimality and the related notion of Pareto front do not assume that the order on the costs is total, which is therefore an even broader set of problems than those captured by GSPP.

In this paper, we present an algorithm of solving GSPP when weights S constitute a regular multicost in the sense of Definitions 2 and 7.

3 Multicost GSPP and Failure of Classical Algorithms

We begin by presenting an example of a GSPP problem with multicost weights and elaborate on why classical algorithms, such as Dijkstra, may fail to find an optimal path in this situation. Consider a robot on a square grid with obstacles as in Figure 1. A robot in position R is going to its goal G. The robot can move to any of the four adjacent cells: left, right, up, and down. Visiting any cell bears

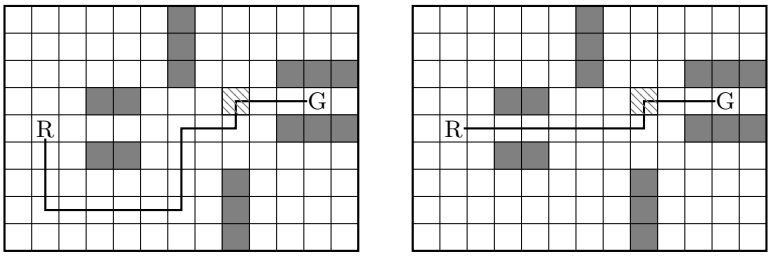

Fig. 1. A robot inside the cell (R) on a square grid moving to the goal cell (G). Left: a path computed using Dijkstra's algorithm; Right: the optimal path.

a cost $(a, 1)$, where a is the Euclidean distance to the nearest obstacle (including the boundary of the grid), and 1 is meant to represent the length of the cell.

Let $\mathbf{R}_{\min} = (\mathbb{R}^{\geq 0} \cup \{\infty\}, \min, \infty, \geq)$ denote a cost monoid on the non-negative reals together with the infinity, equipped with the min operation and $\geq$ order[4]. Let M be the multicost $\mathbf{R}_{\min} \times \mathbf{R}_{+}$. The unit element of M is given by $(\infty, 0)$.

The cost of a cell, as defined above, can naturally be viewed as an element of the monoid M. The robot needs to find an optimal path from the root to the goal, which amounts to finding a path that is as far as possible from all obstacles, and—among all such paths—choose the shortest one.

The problem is set up in such a way that the goal is close to an obstacle, and hence the first coordinate of an optimal path will necessarily be zero. Since M is not a cost monoid, Dijkstra's algorithm may fail to find an optimal path from the root to the goal. Indeed, the path found by the classical Dijkstra's algorithm is shown on the left diagram of Figure1, it has cost $(0, 17)$, while an optimal path (shown on the right diagram therein) has cost $(0, 11)$.

Being a dynamic programming algorithm, Dijkstra produces paths with optimal initial segments. In particular, the path from the root to the dashed cell is optimal and has cost $(1, 14)$. On the other hand, the optimal path to the goal also happens to pass through the dashed cell, and the cost of the initial segment to this cell is $(0, 8)$, which is worse according to the ordering of M. Nonetheless, suffixed with the remaining path to the goal, which has cost (0, 3), the order changes: $(1, 14) <_M (0, 8)$, but

$$(0, 17) = (1, 14) * (0, 3) >_M (0, 8) * (0, 3) = (0, 11).$$

The same type of the cost structure naturally appears in graphs that correspond to communication networks. Suppose, for instance, our primary aim is to maximize the throughput of a channel, and, among all channels with the same throughput, we prefer those with lower latency. Since the throughput of a channel is often modeled as the minimum throughput of its links, and the channel latency equals to the sum of its corresponding link latencies, the same monoid M is the appropriate cost structure to capture this optimization framework.

[4] The cost monoid $\mathbf{R}_{\min}$ is isomorphic to $\mathbf{R}_{\mathrm{m}}$ if we extend the latter to include ∞. An isomorphism is given by the map $\mathbb{R}^{\geq 0} \cup \{\infty\} \ni x \mapsto 1/x \in \mathbb{R}^{\geq 0} \cup \{\infty\}$.

The classical optimal path search algorithms may, therefore, fail to find an optimal path in a graph weighted by a multicost that is not a cost monoid itself. We present in Section 6 an algorithm that effectively finds such a path for regular multicosts.

4 Optimal Subgraph

Unless stated otherwise, throughout this section $\mathcal{G}$ denotes an S-weighted graph, where S is a cost monoid, and v_r, v_g denote the root and the goal of $\mathcal{G}$ respectively. In general, there may be many distinct optimal paths from v_r to v_g. We are interested in finding all optimal paths, and in having an efficient way to store these paths. This is achieved[5] through the concept of an optimal subgraph.

Definition 5. *Let $\mathcal{G}$ be a weighted graph with weights in some monoid with an order (which may not necessarily be a cost monoid). The* ***optimal subgraph*** *of $\mathcal{G}$ is a subgraph $\mathcal{OG} = (\mathcal{O}V, \mathcal{O}E)$ that consists of edges that lie on an optimal path:*

$$\mathcal{O}E = \{e \in E : \exists\, p \in \mathcal{P}(v_r, v_g) \text{ such that } e \in p \text{ and } p \text{ is optimal}\},$$

and corresponding vertices

$$\mathcal{O}V = \{v \in V : \exists\, e \in \mathcal{O}E \text{ such that } v = \mathrm{o}(e) \text{ or } v = \mathrm{t}(e)\}.$$

Example 1. To illustrate this concept, consider the graph that corresponds to Figure 1. It is an M-weighted graph, where $M = \mathbf{R}_{\min} \times \mathbf{R}_{+}$ is a multicost. Its optimal subgraph is shown in Figure 2. Note that all the paths from the root to the goal within the optimal subgraph have the same cost.

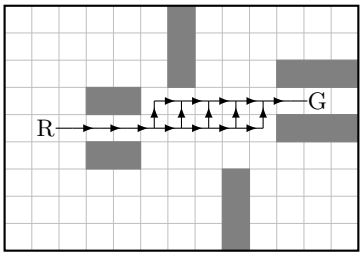

Fig. 2. Optimal subgraph corresponding to Figure 1.

[5] In Section 5, we show that under an additional assumption of regularity, the optimal subgraph is an efficient representation of all optimal paths.

We will also make use of the notions of the optimal *cost-to-come* from the root v_r and *cost-to-go* to the goal v_g at every vertex of $\mathcal{G}$, which we denote $v.come$ and $v.go$, respectively. Formally,

$$v.come = \min\{w(p) \mid p \in \mathcal{P}(v_r, v)\} \text{ and } v.go = \min\{w(p) \mid p \in \mathcal{P}(v, v_g)\}.$$

If either $\mathcal{P}(v_r, v)$ or $\mathcal{P}(v, v_g)$ is empty we set the corresponding value $v.come$ or $v.go$ to $\boldsymbol{\infty}$—some new symbol that is considered to be greater than any other weight.

The following lemma gives a local characterization of an optimal subgraph using the optimal cost-to-come and cost-to-go.

Lemma 1 (Local Optimal Subgraph Characterization). *Let S be a cost monoid, $\mathcal{G}$ be an S-weighted graph with the root v_r and the goal v_g, and let c be the optimal cost between v_r and v_g. If $c < \boldsymbol{\infty}$ then*

$$e \in \mathcal{O}E \iff \mathrm{o}(e).come * w(e) * \mathrm{t}(e).go \leq c.$$

Proof. We begin by noting that the inequality $\mathrm{o}(e).come * w(e) * \mathrm{t}(e).go \leq c$ can be replaced with the equality, as the left hand side corresponds to a path from the root to the goal.

Consider an arbitrary $e \in \mathcal{O}E$. On the one hand, it follows from the definition of the optimal subgraph that there exists an optimal $p = (e_1, \ldots, e_n)$, such that $w(p) = c$ and $e = e_i$ for some i. Let $p' = (e_1, \ldots, e_{i-1})$ and $p'' = (e_{i+1}, \ldots, e_n)$. Using the definition of cost-to-come, we find that $\mathrm{o}(e).come \leq w(p')$. Also from the definition of cost-to-go, it follows $\mathrm{t}(e).go \leq w(p'')$. Since by assumption weights form a cost monoid, we have

$$\mathrm{o}(e).come * w(e) * \mathrm{t}(e).go \leq w(p') * w(e) * w(p'') = w(p) = c.$$

On the other hand, let's assume e satisfies $\mathrm{o}(e).come * w(e) * \mathrm{t}(e).go \leq c$. We let $p' = (\ldots, e'_i, \ldots)$ be an optimal path in $\mathcal{P}(v_r, \mathrm{o}(e))$ and $p'' = (\ldots, e''_j, \ldots)$ be an optimal path in $\mathcal{P}(\mathrm{t}(e), v_g)$. The path

$$p = (\ldots, e'_i, \ldots, e, \ldots, e''_j, \ldots)$$

between v_r and v_g is optimal because

$$w(p) = w(p') * w(e) * w(p'') = \mathrm{o}(e).come * w(e) * \mathrm{t}(e).go \leq c.$$

Hence, e is in $\mathcal{O}E$. □

Let S be a monoid. The **opposite** monoid to S is the monoid S^{op} with the same underlying set S and the operation $*_{\mathrm{op}}$ defined by

$$x *_{\mathrm{op}} y = y * x.$$

Note that if S is a cost monoid, then S^{op} is also a cost monoid, when endowed with the same order.

Given a directed graph $\mathcal{G}$ with a root v_r and a goal v_g, the opposite graph $\mathcal{G}^{\mathrm{op}}$ has the same underlying set of vertices, but reverses all edges:

$$(u, v) \in E^{\mathrm{op}} \iff (v, u) \in E,$$

and changes the role of the root and the goal: $v_r^{\mathrm{op}} = v_g$ and $v_g^{\mathrm{op}} = v_r$.

Let `ForwardDijkstra` denote the classical Dijkstra's algorithm, which is run until all the vertices are visited (as opposed to stopping as soon as the goal is reached). It is proved in [6] that `ForwardDijkstra` finds an optimal spanning tree of the root whenever weights come from a cost monoid. Let also `BackwardDijkstra` denote the same algorithm that runs on the S^{op}-weighted opposite graph $\mathcal{G}^{\mathrm{op}}$. The `ForwardDijkstra` therefore computes optimal cost-to-come $v.come$ for each $v \in V$, and likewise `BackwardDijkstra` computes the optimal cost-to-go $v.go$—the minimal cost of a path from a vertex v to the goal v_g.

We propose the following algorithm for finding an optimal subgraph of a graph weighted in a cost monoid.

Algorithm 1: `OptimalSubgraph`

Data: Cost monoid S and a directed S-weighted graph $\mathcal{G}$ with a root and a goal vertices.
Result: The optimal subgraph $\mathcal{OG}$ of $\mathcal{G}$.

```
1  ForwardDijkstra(G);
2  BackwardDijkstra(G);
3  set c = v_r.go = v_g.come;
4  if c = ∞ then
5  |  return OG = (∅, ∅);
6  for e ∈ E do
7  |  if o(e).come * w(e) * t(e).go = c then
8  |  |  OE.insert(e);
9  |  |  OV.insert(o(e)) and OV.insert(t(e));
10 return OG = (OV, OE);
```

Proposition 1. *Let S be a cost monoid, and let $\mathcal{G}$ be an S-weighted graph. Algorithm 1 computes an optimal subgraph of $\mathcal{G}$ in time polynomial in the size of the vertex set $|V|$*[6]*.*

Proof. Forward and Backward Dijkstra's algorithms compute optimal costs from the root to a vertex and from a vertex to the goal. In view of Lemma 1, this gives us a characterization of when an edge lies on an optimal path. This characterization is used in line 7 of Algorithm 1.

The asymptotic complexity of this algorithm is the same as that of Dijkstra—it runs the Dijkstra's algorithm twice, followed by a linear pass over edges. □

[6] Our assumption is that comparing and multiplying elements from the cost monoid takes constant time.

5 Regular Cost Monoids

By construction, all optimal paths from v_r to v_g lie within $\mathcal{OG}$. However, the inverse may not be true—not every path from v_r to v_g within $\mathcal{OG}$ is optimal.

Definition 6. *A monoid with an order S is said to be **regular** if for all S-weighted graphs any path[7] from the root to the goal within $\mathcal{OG}$ is itself optimal.*

The definition above explains the use for the concept, but the following lemma provides an inner purely algebraic characterization that is valid for cost monoids.

Lemma 2 (Algebraic Characterization of Regularity). *A cost monoid S is regular if and only if for all $a, b, c, d \in S$ such that $a * c = a * d = b * c$ one has $b * d \leq a * c$.*

Proof. Suppose S satisfies the algebraic condition. We show that it is regular. Let $p = (e_1, \dots, e_m)$ be a path from the root to the goal within $\mathcal{OG}$. Note that $e_1 \in \mathcal{OG}$, and hence there is some optimal path that goes through e_1. Let $1 \leq k \leq m$ be maximal such that there exists an optimal path $\bar{p}$ that coincides with p on the first k edges

$$\bar{p} = (e_1, \dots, e_k, \bar{e}_{k+1}, \dots, \bar{e}_n).$$

We claim that $k = m$, and hence p itself is optimal. Suppose towards the contradiction that $k < m$.

Since $e_{k+1} \in \mathcal{OG}$, there exists an optimal path

$$q = (f_1, \dots, f_l, f_{l+1} = e_{k+1}, f_{l+2}, \dots, f_r).$$

Paths p, $\bar{p}$, and q are depicted in Figure 3. Consider the weights of the following paths:

$$\begin{aligned} a &= w(f_1) * \cdots * w(f_l) \\ b &= w(e_1) * \cdots * w(e_k) \\ c &= w(\bar{e}_{k+1}) * \cdots * w(\bar{e}_n) \\ d &= w(f_{l+1}) * \cdots * w(f_r) \end{aligned}$$

Note that $a * d = w(q)$ and $b * c = w(\bar{p})$, which are both optimal, and hence $a * d = b * c$.

If $b \leq a$, then by the definition of a cost monoid, $b * d \leq a * d$, and the path $(e_1, \dots, e_k, f_{l+1} = e_{k+1}, f_{l+2}, \dots, f_r)$ is therefore an optimal path that agrees with p on $k + 1$ many segments, contradicting the choice of k.

If, on the hand $a < b$, then $a * c \leq b * c$, but since $a * c$ is a weight of a path from the root to the goal and $b * c$ is optimal, we get $a * c = b * c$. Additionally, since $a * d$ is optimal, we get $b * c = a * d$. The algebraic condition applies, and

[7] Recall that $\mathcal{OG}$ is a directed graph. If the weights S come from a cancellative cost monoid, then all loops in $\mathcal{OG}$ will necessarily have weight $\mathbf{1}$. This is not necessarily the case for non-cancellative monoids.

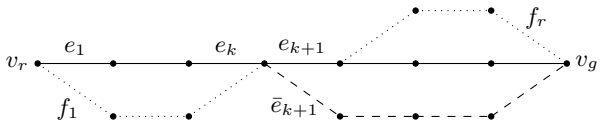

Fig. 3. Paths p, $\bar{p}$, and q.

guarantees that $b * d \leq a * c$, i.e., $b * d$ is optimal. However, $b * d$ is the weight of the path

$$(e_1, \ldots, e_k, f_{l+1}, \ldots, f_r)$$

which agrees with p on $k+1$ many initial segments contradicting the choice of k.

For the other direction, if S does not satisfy the algebraic condition, there are a, b, c, d such that $a * c = a * d = b * c < b * d$. The graph in Figure 4 coincides

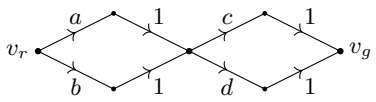

Fig. 4. Graph interpretation of the algebraic characterization.

with its own optimal subgraph, but it has a non-optimal path with the cost $b * d$. Thus, by definition, S is not regular. □

Example 2. Many monoids that are often considered in the context of GSPP are regular. This includes, for instance, all the monoids listed in [6, after Definition 3].

1. Cost monoid $\mathbf{R}_+$ is regular. More generally, any cancellative cost monoid is regular.
2. Cost monoid $\mathbf{R}_{\mathrm{m}}$ is regular, since for $a * b = \max\{a, b\}$ and has

$$a * c = a * d = b * c \implies a * c = \max\{a, b, c, d\} \geq \max\{b, d\} = b * d.$$

3. The cost monoid shown in Table 1, however, is not regular. It consists of five elements $1 < a < b < c < \infty$. It is straightforward to verify that it is indeed a cost monoid[8]. Nonetheless, $a * a = a * b = b * a < b * b$, and therefore it is not regular by Lemma 2.

Lemma 3. *Let S_1 and S_2 be regular monoids with an order. The product $S = S_1 \times S_2$ is a regular monoid with an order.*

[8] For example, to check associativity note that $x * (y * z) = \infty$ whenever neither of x, y, z is equal to 1; similarly for $(x * y) * z$.

Table 1. Non-regular cost monoid

	1	a	b	c	∞
1	1	a	b	c	∞
a	a	c	c	∞	∞
b	b	c	∞	∞	∞
c	c	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞

Proof. Let $\pi_i : S \to S_i$, $i = 1, 2$, be the projection onto the ith coordinate. Let $\mathcal{G}$ be an S-weighted graph with the weight function $w : E \to S$, and let $w_i = \pi_i \circ w$ be the S_i-weight obtained by considering the corresponding coordinate only. Pick a path p from v_r to v_g that lies within $\mathcal{OG}$. Our goal is to show that p is optimal.

We may view $\mathcal{G}$ as an S_1-weighted graph, so let $\mathcal{OG}_1$ denote the optimal subgraph of $\mathcal{G}$ relative to the weight function w_1. Note that by the definition of lexicographic order $\mathcal{OG}$ is a subgraph of $\mathcal{OG}_1$, and hence p lies within $\mathcal{OG}_1$. Since S_1 is regular by assumption, the first coordinate of the weight of p is optimal.

Consider now $\mathcal{OG}_1$ as an S_2-weighted graph with the weight w_2, and let $\mathcal{OG}_2$ be the corresponding optimal subgraph. Note that again $\mathcal{OG}$ is a subgraph[9] of $\mathcal{OG}_2$ and the costs $w(q)$ of all the paths q from v_r to v_g that lie within $\mathcal{OG}_2$ have the same first cost coordinate $w_1(q)$ (because such q lies in $\mathcal{OG}_1$, and S_1 is regular) and the same second cost coordinate $w_2(q)$ (by regularity of S_2). Thus any path within $\mathcal{OG}_2$ is optimal relative to $w = w_1 \times w_2$. We conclude that the path p is optimal as claimed. □

Lemma 4. *Let $S_1, \dots, S_n$ be regular monoids with an order. The product $S = S_1 \times \cdots \times S_n$ is a regular monoid with an order.*

Proof. We argue by induction on n—the number of monoids in the product. The base of induction $n = 1$ is trivial. For the step of induction we assume that the lemma has been proved for products of $(n-1)$-many monoids. Set $\tilde{S} = S_1 \times \cdots \times S_{n-1}$, which is regular by inductive assumption. Note that S is canonically isomorphic to $\tilde{S} \times S_n$, hence Lemma 3 applies and shows that S is regular as well. □

6 Iterated Algorithm for Regular Multicost GSPP

Definition 7. *A multicost $S_1 \times \cdots \times S_n$ is said to be regular if all the cost monoids S_i in the product are regular.*

Note that in view of Lemma 4, any regular multicost is a regular monoid with an order.

[9] In fact, $\mathcal{OG} = \mathcal{OG}_2$.

Let S be a regular multicost. The following Algorithm 2 can be used to compute the optimal subgraph of an S-weighted graph $\mathcal{G}$ by iteratively applying Algorithm 1.

Algorithm 2: Iterated Dijkstra Propagation Algorithm

Data: A regular multicost $S = S_1 \times \cdots \times S_n$ and an S-weighted graph $\mathcal{G}$.
Result: The optimal subgraph $\mathcal{OG}$ of $\mathcal{G}$.
1 set $\mathcal{G}_0 = \mathcal{G}$ as an S_1-weighted graph;
2 **for** $i \in \{1, \ldots, n\}$ **do**
3 | $\mathcal{G}_i = \texttt{OptimalSubgraph}(\mathcal{G}_{i-1})$ as an S_i-weighted graph;
4 return $\mathcal{OG} = \mathcal{G}_n$;

Proposition 2. *Let $\mathcal{G}$ be an S-weighted graph, where $S = S_1 \times \cdots \times S_n$ is a regular multicost. Let v_r and v_g be the root and the goal of $\mathcal{G}$. Algorithm 2 finds an optimal subgraph $\mathcal{OG}$ of $\mathcal{G}$. This algorithm runs in polynomial time in the number of vertices $|V|$.*

Proof. Showing that the output of the algorithm is an optimal subgraph follows an argument similar to the one in the proof of Lemma 3 and Lemma 4.

As for the complexity claim, Algorithm 2 consists of n-many runs of Algorithm 1, where n is the number of coordinates in the weights. Since Algorithm 1 is polynomial in view of Proposition 1, so is Algorithm 2. □

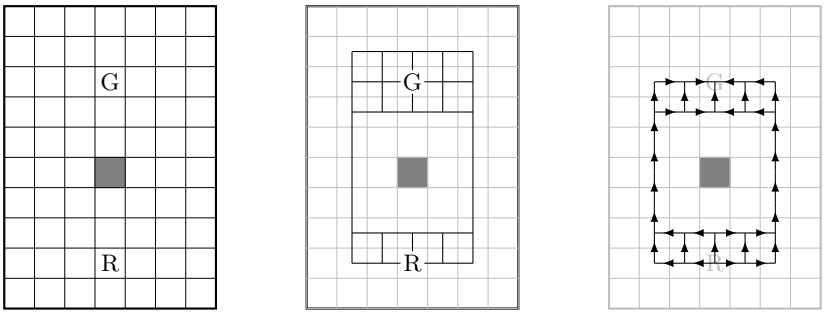

Fig. 5. Robot (R) on a square grid going to the goal (G); the dark gray square is an obstacle. Middle: light gray area represents an optimal subgraph after the first iteration; Right: an optimal subgraph after the second iteration.

We give two examples of M-weighted graphs, $M = \mathbf{R}_{\min} \times \mathbf{R}_{+}$, and show the output of Algorithm 2 on these graphs.

Example 3. Consider the graph corresponding to Figure 1. It is weighted by the regular multicost $\mathbf{R}_{\min} \times \mathbf{R}_{+}$. After the first iteration, $\mathcal{OG}_1$ consists of all edges that do not pass through any obstacles. Indeed, any path from the root to the goal that avoids the obstacles will have the same first coordinate of the cost. The graph $\mathcal{OG}_2$ obtained on the second iteration is shown in Figure 2.

Example 4. Figure 5 gives a different illustration of Algorithm 2. The setup is similar to the one in Section 3. A robot in the cell R needs to find a path to the goal G. The dark gray square in the center represents an obstacle. At each step, the robot may move by one cell in each of the four directions. The cost of an edge is a pair $(a, 1)$, where a is the minimal distance to the obstacle or boundary. Edge costs are viewed as elements of $\mathbf{R}_{\min} \times \mathbf{R}_{+}$ and are composed accordingly.

The left hand side of Figure 5 shows the initial setup. The optimal subgraph $\mathcal{OG}_1$ computed after the first iteration is shown in the middle diagram. For every edge e in $\mathcal{OG}_1$, the opposite edge is also in $\mathcal{OG}_1$, so we depict it as an undirected graph. Note that any path from the root to the goal within $\mathcal{OG}_1$ is one unit of distance away from the obstacle and the boundary. Note also that the optimal subgraph in this case contains cycles. The right diagram shows the optimal subgraph computed after the second iteration. Unlike the previous iteration, for each edge only one direction is included in $\mathcal{OG}_2$. All paths from the root to the goal within this graph have the same length and are all optimal according to the chosen cost structure.

Algorithm 2 may have three qualitatively different outcomes: i) the optimal subgraph is empty, ii) the optimal subgraph consists of a single path between v_r and v_g, and iii) optimal subgraph consists of more than one path.

The first case is possible only when the goal is not reachable from the root, and it can be detected on the first pass of the algorithm. The second case is common when the solution is unique for at least one regular cost monoid in the product. In this case, the loop can be terminated at the iteration corresponding to this cost monoid. Also note that optimizations with respect to the remaining cost monoids become irrelevant. Finally, in the third case, we are allowed to choose any path from the optimal subgraph since all paths are optimal due to the regularity of $S_1, \dots, S_n$. Moreover, if one is only interested in finding an optimal path, the last iteration of Algorithm 2 can be substituted with the classical Dijkstra or the A* algorithm.

7 Summary

The focus of this paper is on extending the classical shortest path problem on directed graphs towards using abstract algebraic costs. This extension generalizes the minimum violation planning framework by considering additive and nonadditive characteristics of paths, for example, goal arrival time and obstacle clearance, respectively. Motivated by the fact that classical graph search algorithms may fail to find an optimal solution in the general settings of monoids

with an order, we introduced the class of regular cost monoids and presented an iterative planning algorithm that computes the optimal path for graphs weighted in regular multicosts.

Combined with a rulebooks framework [3], an implementation of our algorithm enables complex vehicle behaviors in unpredictable urban environments. Moreover, the established correctness proofs guarantee that the computed path violates the least critical rule by the smallest amount possible. We continue exploring extensions of the presented algorithms in the following directions: i) algorithms for replanning in unknown or dynamic environments and ii) further generalization of the cost structures.

References

1. H. Ben Ticha and N. Absi, "A solution method for the Multi-destination Bi-objectives Shortest Path Problem," no. July, 2017.
2. L. I. R. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 3217–3224.
3. A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8536–8542.
4. L. de Lima Pinto, C. T. Bornstein, and N. Maculan, "The tricriterion shortest path problem with at least two bottleneck objective functions," *European Journal of Operational Research*, vol. 198, no. 2, pp. 387–391, oct 2009.
5. E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, dec 1959.
6. S. Edelkamp, S. Jabbar, and A. Lluch-Lafuente, "Cost-algebraic heuristic search." *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, pp. 1362–1367, jan 2005.
7. D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Springer Tracts in Advanced Robotics*, vol. 56, no. 11-12, pp. 61–89, 2009.
8. X. Gandibleux, F. Beugnies, and S. Randriamasy, "Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function," *4or*, vol. 4, no. 1, pp. 47–59, mar 2006.
9. R. G. Garroppo, S. Giordano, and L. Tavanti, "A survey on multi-constrained optimal path computation: Exact and approximate algorithms," *Computer Networks*, vol. 54, no. 17, pp. 3081–3107, dec 2010.
10. D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.
11. P. Hansen, "Bicriterion Path Problems," 1980, pp. 109–127.
12. P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, feb 1968.
13. P. Höfner and B. Möller, "Dijkstra, Floyd and Warshall meet Kleene," *Form. Asp. Comput.*, vol. 24, no. 4-6, pp. 459–476, 2012.

14. R. Holte and S. Zilles, "On the optimal efficiency of cost-algebraic a*," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 2288–2295, 07 2019.
15. S. M. LaValle, *Planning algorithms*. Cambridge University Press, may 2006, vol. 9780521862.
16. M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.
17. L. Mandow and J. L. De La Cruz, "A memory-efficient search strategy for multiobjective shortest path problems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5803 LNAI, pp. 25–32.
18. L. Mandow and J. L. P. De La Cruz, "Multiobjective A* search with consistent heuristics," *Journal of the ACM*, vol. 57, no. 5, pp. 1–25, jun 2010.
19. E. Q. V. Martins, "On a multicriteria shortest path problem," *European Journal of Operational Research*, vol. 16, no. 2, pp. 236–245, 1984.
20. B. S. Stewart and C. C. White, "Multiobjective A*," *Journal of the ACM (JACM)*, vol. 38, no. 4, pp. 775–814, oct 1991.
21. J. Tumova, L. I. R. Castro, S. Karaman, E. Frazzoli, and D. Rus, "Minimum-violation LTL planning with conflicting specifications," in *2013 American Control Conference*, June 2013, pp. 200–205.
22. J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '13. New York, NY, USA: ACM, 2013, pp. 1–10.
23. U. Zimmermann, "Linear and combinatorial optimization in ordered algebraic structures," *Ann. Discrete Math.*, vol. 10, pp. viii+380, 1981.