# Cost-Algebraic Heuristic Search

**Stefan Edelkamp**[1*] **and Shahid Jabbar**[2*]
Fachbereich Informatik
Universität Dortmund
[1]stefan.edelkamp@cs.uni-dortmund.de
[2]shahid.jabbar@cs.uni-dortmund.de

**Alberto Lluch Lafuente**[†]
Dipartimento di Informatica
Università di Pisa
lafuente@di.unipi.it

**Abstract**

Heuristic search is used to efficiently solve the single-node shortest path problem in weighted graphs. In practice, however, one is not only interested in finding a short path, but an *optimal* path, according to a certain cost notion. We propose an algebraic formalism that captures many cost notions, like typical Quality of Service attributes. We thus generalize A*, the popular heuristic search algorithm, for solving optimal-path problem. The paper provides an answer to a fundamental question for AI search, namely to which general notion of cost, heuristic search algorithms can be applied. We proof correctness of the algorithms and provide experimental results that validate the feasibility of the approach.

## Introduction

Heuristic search (Pearl 1985) is an efficient solution to exploration problems in many fields, including action planning (Bonet & Geffner 2001), single-agent games (Korf 1985), computational biology (Zhou & Hansen 2003), and model checking (Edelkamp, Leue, & Lluch Lafuente 2003). Basically, the idea is to apply algorithms that exploit the information about the problem being solved in order to guide the exploration process. The benefits are twofold: the search effort is reduced and the solution quality is improved, which in many cases means that solution paths are shorter or cheaper.

In many practical domains, however, the quality of a path depends on the *nature* of costs associated with the underlying graph. For instance, in wide area networks costs are given by different Quality of Service (Xiao & Ni 1999) attributes. Some past works have proposed general notions of costs. Classical algorithms, see (Rote 1985) for instance, use different kinds of semirings to formalize and solve the algebraic path problem, a generalization of the all-pairs shortest path problem. More recently, algebraic structures have been provided in the domain of network routing with QoS (Sobrinho 2002), logics for wide area network applications with QoS (Lluch Lafuente & Montanari 2004; Ferrari & Lluch Lafuente 2004) and soft constraint programming (Bistarelli, Montanari, & Rossi 1997; 2002). However, to the best of our knowledge our work is the first attempt to define an abstract formalism for costs suited for heuristic search algorithms like A*.

We first define a suitable algebraic structure for graphs called *cost algebra*. Then, we generalize classical results of heuristic search specially regarding algorithm A*, according to a general notion of costs. The paper provides an answer to a fundamental question for AI search, namely to which general notion of cost, heuristic search algorithms can be applied. At the practical front, we provide experiments on different cost algebras explained in this paper along with a real-world application in route-planning (Jabbar 2003) where maps are constructed by bicycle tours. During a tour several parameters are registered: distance, speed, altitude, traffic density, hearth pulse, all of them formalized by the cost algebra. After some processing, maps are represented by weighted graphs over which optimal path queries are performed.

The paper is structured as follows. The next section defines the cost algebra. Then we turn to cost algebraic search in graphs, in particular for solving the optimality problem. Cost-algebraic versions of Dijkstra's algorithm and A* with consistent and admissible estimates are proposed. We discuss their correctness and highlight the differences with respect to the usual cost notion together with a note on the optimal efficiency of A*. Sometimes, there can be more than one type of cost attached with the edges of a graph, for example, a routing graph can have travel time and distance associated with each edge. A discussion on a method to combine our cost algebras is presented next in the paper. To validate the feasibility of our approach, we present two sets of experiments. Last, but not least, we draw conclusions, mainly on the level of generality that we have obtained.

## Cost Algebra on Graphs

We open this section recalling some algebraic concepts.

**Definition 1** *Let $A$ be a set and $\times : A \times A \to A$ be a binary operator. A* monoid *is a tuple $\langle A, \times, \mathbf{1} \rangle$ if $\mathbf{1} \in A$ and for all*

*Supported by the German Research Foundation (DFG) projects *Heuristic Search* Ed 74/3 and *Directed Model Checking* Ed 74/2.

†Supported by the European Research Training Network SEGRAVIS.

$a, b, c \in A$

(1) $a \times b \in A$ *(closeness)*
(2) $a \times (b \times c) = (a \times b) \times c$ *(associativity)*
(3) $a \times \mathbf{1} = \mathbf{1} \times a = a$ *(identity)*

**Definition 2** *Let $A$ be a set. A relation $\preceq \in A \times A$ is a* total order *whenever for all $a, b, c \in A$*

(1) $a \preceq a$ *(reflexivity)*
(2) $a \preceq b \wedge b \preceq a \Rightarrow a = b$ *(anti-symmetry)*
(3) $a \preceq b \wedge b \preceq c \Rightarrow a \preceq c$ *(transitivity)*
(4) $a \preceq b \vee b \preceq a$ *(total)*

In the rest of the paper $a \prec b$, $a \succeq b$ and $a \succ b$ abbreviate $a \preceq b \wedge a \neq b$, $b \preceq a$ and $a \succeq b \wedge a \neq b$, respectively.

The *least* and *greatest* operations are defined as usual, i.e., $\bigsqcup A = c$ such that $c \preceq a$ for all $a \in A$, and $\bigsqcap A = c$ such that $a \preceq c$ for all $a \in A$. We sometimes say that $\preceq$ is induced by $\sqcup$, if $\sqcup$ is the *least* operation of $\preceq$. We say that a set $A$ is *isotone* if $a \preceq b$ implies both $a \times c \preceq b \times c$ and $c \times a \preceq c \times b$ for all $a, b, c \in A$.

We finally define our cost formalism, which slightly extends the one of (Sobrinho 2002).

**Definition 3** *$A$* cost algebra *is defined as a 6-tuple $\langle A, \sqcup, \times, \preceq, \mathbf{0}, \mathbf{1}\rangle$, such that*

(1) $\langle A, \times, \mathbf{1}\rangle$ *is a monoid*
(2) $\preceq$ *is a total order induced by* $\sqcup$
(3) $\mathbf{0} = \bigsqcap A$ *and* $\mathbf{1} = \bigsqcup A$
(4) $A$ *is isotone*

Intuitively, $A$ is the domain set of cost values, $\times$ is the operation used to cumulate values and $\sqcup$ is the operation used to select the best (the least) amongst values. Consider for example, the following instances of cost algebras, typically used as Quality of Service formalisms:

- $\langle \{\mathit{true}, \mathit{false}\}, \vee, \wedge, \Rightarrow, \mathit{false}, \mathit{true}\rangle$ (boolean): Network/service availability.
- $\langle \mathbb{R}^+ \cup \{+\infty\}, \mathit{min}, +, \leq, +\infty, 0\rangle$ (optimization): Price, propagation delay.
- $\langle \mathbb{R}^+ \cup \{+\infty\}, \mathit{max}, \mathit{min}, \geq, 0, +\infty\rangle$ (max/min): Bandwidth.
- $\langle [0,1], \mathit{min}, \cdot, \geq, 0, 1\rangle$ (probabilistic): Performance and rates.
- $\langle [0,1], \mathit{max}, \mathit{min}, \geq, 0, 1\rangle$ (fuzzy): Performance and rates.

More specific cost structures do no longer cope for all the examples that we want to cover. For example, the slightly more restricted property of *strict isotonicity*, where one requires $a \prec b$ to imply both $a \times c \prec b \times c$ and $c \times a \prec c \times b$ for all $a, b, c \in A$, $c \neq \mathbf{0}$, is not satisfied in structure (max/min), since $\min\{3, 3\} = \min\{3, 5\}$, but $3 < 5$.

**Proposition 1** *All introduced algebras are cost algebras.*

**Proof:** The only non-trivial property to be checked is isotonicity.

$\langle \{\mathit{true}, \mathit{false}\}, \vee, \wedge, \Rightarrow, \mathit{false}, \mathit{true}\rangle$: We have to show that $a \Rightarrow b$ implies both $a \wedge c \Rightarrow b \wedge c$ and $c \wedge a \Rightarrow c \wedge b$ for all $a, b, c \in \{\mathit{true}, \mathit{false}\}$. W.l.o.g. take the first implication and to $c = \mathit{false}$. Condition $a \Rightarrow b$ implies $(\mathit{false} \Rightarrow \mathit{false}) = \mathit{true}$ in all cases.

$\langle \mathbb{R}^+ \cup \{+\infty\}, \mathit{min}, +, \leq, +\infty, 0\rangle$: Here we have to show that $a \leq b$ implies both $a + c \leq b + c$ and $c + a \leq c + b$ for all $a, b, c \in \mathbb{R}^+ \cup \{\infty\}$, which is certainly true.

$\langle \mathbb{R}^+ \cup \{+\infty\}, \mathit{max}, \mathit{min}, \geq, 0, +\infty\rangle$: $a \geq b$ implies $\min\{a, c\} \geq \min\{b, c\}$ and $\min\{c, a\} \geq \min\{c, b\}$, $a, b, c \in \mathbb{R}^+ \cup \{\infty\}$.

$\langle [0,1], \mathit{min}, \cdot, \geq, 0, 1\rangle$: $a \geq b$ implies $a \cdot c \geq b \cdot c$ and $c \cdot a \geq c \cdot b$, $a, b, c \in [0, 1]$.

$\langle [0,1], \mathit{max}, \mathit{min}, \geq, 0, 1\rangle$: similar proof as for (max/min) structure. □

Tackling with multiple optimization criteria is very interesting, since one is interested, in practice, in finding paths that are optimal according to more than one attribute. Classical examples from network routing are widest-shortest paths or shortest-widest path, whereas in the route planning domain we have shortest-quickest or quickest-shortest path queries.

To include multi-criteria into the search, one can define edge costs $\sum_i \lambda_i w_i$, with $\sum_i \lambda_i = 1$. Despite the success in many areas like game playing, the approach looses information (Müller 2001): all kinds of features are merged, even those, for which merging does not make sense. On the other hand, Multiobjective A* (Mandow & de la Cruz 2005) based on a partial ordering of cost vectors (Steward & White 2001) turns out to be a complex exploration procedure with respect to original A*. As a consequence, we consider the ordered combination of different criteria in a cross product cost algebra.

Unfortunately, it is not possible to define a meaningful composition operation of cost algebras, i.e., an operation that permits to combine two cost algebras into one cost algebra. The problem of Cartesian products and power constructions as defined for semirings (Bistarelli, Montanari, & Rossi 1997) is that they provide partial orders. On the other hand, Cartesian products that prioritize one criteria over the other have the problem to deliver non-isotone algebras in general (Sobrinho 2002). Indeed, the widest-shortest path can be formalized by a cost algebra, while the shortest-widest path cannot.

More formally, one can model a prioritized Cartesian product as follows.

**Definition 4** *The prioritized Cartesian product of two cost algebras $C_1 = \langle A_1, \sqcup_1, \times_1, \preceq_1, \mathbf{0}_1, \mathbf{1}_1\rangle$ and $C_2 = \langle A_2, \sqcup_2, \times_2, \preceq_2, \mathbf{0}_2, \mathbf{1}_2\rangle$, denoted by $C_1 \times_p C_2$ is a tuple $\langle A_1 \times A_2, \sqcup, \times, \preceq, (\mathbf{0}_1, \mathbf{0}_2), (\mathbf{1}_1, \mathbf{1}_2)\rangle$, where $(a_1, a_2) \times (b_1, b_2) = (a_1 \times b_1, a_2 \times b_2)$, $(a_1, a_2) \preceq (b_1, b_2)$ iff $a_1 \prec b_1 \vee (a_1 = b_1 \wedge a_2 \preceq b_2)$, and $a \sqcup b = a$ iff $a \preceq b$.*

**Proposition 2** *If $C_1, C_2$ are cost algebras and $C_1$ is strictly isotone then $C_1 \times_p C_2$ is a cost algebra.*

**Proof:** The only non-trivial part is isotonicity. If we have $(a_1, a_2) \preceq (b_1, b_2)$ then there are two cases. First, $a_1 \prec a_2$ in which case (by strict isotonicity) we have $a_1 \times c_1 \prec b_1 \times c_1$ and $c_1 \times a_1 \prec c_1 \times b_1$ which clearly implies $(a_1, a_2) \times (c_1, c_2) \preceq (b_1, b_2) \times (c_1, c_2)$ and $(c_1, c_2) \times (a_1, a_2) \preceq (c_1, c_2) \times (b_1, b_2)$.

The second case is $a_1 = b_1$ and $a_2 \preceq b_2$. This trivially implies $a_1 \times c_1 = b_1 \times c_1$ and $a_1 \times c_1 = b_1 \times c_1$ and, by isotonicity, $a_2 \times c_2 \preceq b_2 \times c_2$ and $c_2 \times a_2 \preceq c_2 \times b_2$. Clearly, we have $(a_1, a_2) \times (c_1, c_2) \preceq (b_1, b_2) \times (c_1, c_2)$ and $(c_1, c_2) \times (a_1, a_2) \preceq (c_1, c_2) \times (b_1, b_2)$. □

Similarly, one can prove that if $C_1$ and $C_2$ are strictly isotone, then $C_1 \times_p C_2$ is strictly isotone.

**Definition 5** *An* (edge-weighted) graph $G$ *is a tuple* $\langle V, E, in, out, \omega \rangle$ *where* $V$ *is a set of nodes,* $E$ *is a set of edges,* $in, out : E \to V$ *are source and target functions, and* $\omega : E \to A$ *is a weighting function.*

Graphs usually have a distinguished start node $s$, which we denote with $u_0^G$, or just $s$ if $G$ is clear from the context.

**Definition 6** *A* path *in a graph* $G$ *is an alternating sequence of nodes and edges* $u_0, e_0, u_1, \ldots$ *such that for each* $i \geq 0$ *we have* $u_i \in V$, $e_i \in E$, $in(e_i) = u_i$ *and* $out(e_i) = u_{i+1}$, *or, shortly* $u_i \xrightarrow{e_i} u_{i+1}$.

An initial path is a path starting at $s$. Finite paths are required to end at nodes. The length of a finite path $p$ is denoted by $|p|$. The concatenation of two paths $p, q$ is denoted by $pq$, where we require $p$ to be finite and end at the initial node of $q$. The cost of a path is given by the cumulative cost of its edges. Formally,

**Definition 7** *The* cost *of a finite path* $p$ *is denoted by* $\omega(p)$ *and defined as*

$$\begin{array}{ll} \omega(e) \times \omega(q) & \text{if } p = (u \xrightarrow{e} v)q \\ \mathbf{1} & \text{otherwise} \end{array}$$

Let $P(u)$ denote the set of all paths starting at node $u$. In the sequel we shall use $\delta(u, V)$ to denote the cost of the optimal path starting at a node $u$ and reaching a node $v$ in a set $V$. Formally, $\delta(u, V) = \bigsqcup_{p \in P(u) | (p \cap V) \neq \emptyset} \omega(p)$. For the ease of notation, we write $\delta(u, \{v\})$ as $\delta(u, v)$.

Next, we define *prefix optimality*, which formalizes the fact that for some paths all subpaths starting from the start node are optimal.

**Definition 8** *A path* $p = u_0 \xrightarrow{e_0} \ldots \xrightarrow{e_{k-1}} u_k$ *is* prefix-optimal, *if all of its prefixes, i.e., all paths* $u_0 \xrightarrow{e_0} \ldots \xrightarrow{e_{i-1}} u_i$ *with* $i < k$, *form an optimal path from* $u_0$ *to* $u_i$.

As an example consider the (max/min) cost algebra of a graph with nodes $v_1$, $v_2$, $v_3$, and $v_4$, and edges $v_1 \xrightarrow{e_1} v_2$, $v_2 \xrightarrow{e_2} v_3$, $v_1 \xrightarrow{e_3} v_3$ and $v_3 \xrightarrow{e_4} v_4$ with weights $\omega(e_1) = 4$, $\omega(e_2) = 4$, $\omega(e_3) = 2$, and $\omega(e_4) = 2$. Path $v_1 \xrightarrow{e_3} v_3 \xrightarrow{e_4} v_4$ and path $v_1 \xrightarrow{e_1} v_2 \xrightarrow{e_2} v_3 \xrightarrow{e_4} v_4$ are optimal with cost 2, but only the latter is prefix-optimal.

## Cost-Algebraic Analysis of Graphs

The *reachability problem* consists of finding a node $t$ such that a predicate *goal*$(t)$ is true. The *optimality problem* consists of finding an initial prefix-optimal path $p$ ending at a node $t$ such that *goal*$(t)$ is *true* and $\omega(p) = \delta(s, T)$, where $T = \{u \mid \textit{goal}(u) = \textit{true}\}$. Reachability and optimality problems can be solved with traditional graph exploration and shortest-path algorithms[1]. For the reachability problem, for instance, one can use, amongst others, depth-first search, hill climbing, greedy best-first search, Dijkstra's algorithm (and its simplest version breadth-first search) or A* (Pearl 1985). For the optimality problem, on the other hand, only the last two are suited. Nevertheless, Dijkstra's algorithm or A* are traditionally defined over a simple instance of our cost algebra, namely the optimization cost algebra $\langle \mathbb{R}^+ \cup \{+\infty\}, \mathbb{R} \cup \{+\infty\}, \textit{min}, +, \leq, +\infty, 0 \rangle$. We need thus to generalize the results that ensure the *admissibility* of the exploration algorithms, i.e., the fact that they correctly solve the optimality problem.

### Principle of Optimality

Admissibility depends on the *Principle of Optimality*, which intuitively means that the optimality problem can be decomposed.

**Definition 9** *The* Principle of Optimality *requires* $\delta(s, v) = \bigsqcup\{\delta(s, u) \times \omega(e) \mid u \xrightarrow{e} v\}$, *where* $s$ *is the start node in a given graph* $G$.

**Proposition 3** *Any cost algebra* $\langle A, +, \times, \preceq, \mathbf{0}, \mathbf{1} \rangle$ *satisfies the principle of optimality.*

**Proof:** Observe that $\bigsqcup\{\delta(s, u) \times \omega(e) | \ u \xrightarrow{e} v\}$ =

$$\begin{array}{ll} \stackrel{(1)}{=} & \bigsqcup\{\bigsqcup\{\omega(p) \mid p = (s, .., u)\} \times \omega(e) | u \xrightarrow{e} v\} \\ \stackrel{(2)}{=} & \bigsqcup\{\omega(p) \times \omega(e) \mid p = s \to \ldots \to u \xrightarrow{e} v\} \\ \stackrel{(3)}{=} & \bigsqcup\{\omega(p') \mid p' = s \to \ldots \to v\} \\ \stackrel{(4)}{=} & \delta(s, v), \end{array}$$

where (1) is by definition, (2) is by distributivity of $\times$, (3) is by isotonicity since $c \times b \preceq a \times b$ for all $a$ implies $\bigsqcup\{S\} \times b = \bigsqcup\{a \mid a \in S\} \times b = \bigsqcup\{a \times b \mid a \in S\}$, and (4) is by definition. □

### Cost-Algebraic Dijkstra's Algorithm

The core of our generalized Dijkstra's algorithm is sketched below. Basically, the algorithm maintains a set of *Open* nodes to be explored. It iteratively selects the most promising node (the with the currently optimal initial path which cost is represented by $f$) to explore from that set which is expanded, i.e., the set of nodes reachable via a single edge is computed and introduced in the set *Open*.

**Initialization** [2] $f(u_0) \leftarrow \mathbf{1}$; *Open* $\leftarrow \{u_0\}$; **for each** $u \neq u_0 : f(u) \leftarrow \mathbf{0}$

[1] Abusing notation we refer here to a slight modification of the original algorithms, consisting of terminating the algorithm when a goal node is reached and returning the corresponding path.

[2] Note that the initialization of $f(u) \leftarrow \mathbf{0}$ for all $u \neq u_0$ can be avoided by keeping track of all nodes that have been expanded so far in a list *Closed*. This is essential for exploration domains, where the state space graph is not accessible prior to the search. For the whole algorithm we refer to (Cormen *et al.* 2001).

**Selection** Select $u \in$ *Open* with $f(u) = \bigsqcup\{f(v) \mid v \in$ *Open*$\}$

**Update** $f(v) \leftarrow \bigsqcup\{\{f(v)\} \cup \{f(u) \times \omega(e) \mid u \xrightarrow{e} v\}\}$

**Proposition 4** *Cost-algebraic Dijkstra's algorithm solves the optimality problem.*

**Proof:** Let $p = (s = u_0 \xrightarrow{e_0} \ldots \xrightarrow{e_{k-1}} u_k)$ be the first goal path found by the algorithm, with $u_k \in T$. It suffices to show, that if a node $u$ is selected we have $f(u) = \delta(s,u)$. Let $u$ be the first selected node with $f(u) \neq \delta(s,u)$, which clearly implies $f(u) \succ \delta(s,u)$. Let $s \ldots x \xrightarrow{e} y \ldots u$ be an optimal path for $u$ with $y$ being the first node that is not expanded. We are interested in the existence of at least one such path which is prefix-optimal. We have $f(x) = \delta(s,x)$ by the minimality of $u$ and $\delta(s,x) \times \omega(e) = \delta(s,y)$ by the principle of optimality, so that $f(y) \preceq f(x) \times \omega(e) = \delta(s,x) \times \omega(e) = \delta(s,y) \preceq \delta(s,T) \prec f(u)$, in contradiction to the selection of $u \in$ *Open* with $f(u) = \bigsqcup\{f(v) \mid v \in$ *Open*$\}$. For the inequality $\delta(s,y) \preceq \delta(s,u)$ we used the fact that $\omega(p) \succeq \mathbf{1}$, for all paths $p$. □

## Cost-Algebraic A*

Once stated the admissibility of the generalized Dijkstra's algorithm, we concentrate on A* (Hart, Nilsson, & Raphael 1968), an improvement that makes use of heuristic functions to accelerate the search process. We begin by adapting the usual notions of admissibility and consistency of heuristic functions (Pearl 1985).

**Definition 10** *A heuristic function $h : V \to A$ with $h(t) = \mathbf{1}$ for each goal node $t \in T$ is*

- admissible, *if for all $u \in V$ we have $h(u) \preceq \delta(u,T)$, i.e., $h$ is a lower bound.*
- consistent, *if for each $u, v \in V$ and $e \in E$ such that $u \xrightarrow{e} v$, we have $h(u) \preceq \omega(e) \times h(v)$.*

Next we generalize the fact that consistency implies admissibility.

**Proposition 5** *If $h : V \to A$ is consistent, then $h$ is admissible.*

**Proof:** Let $p = (u = u_0 \xrightarrow{e_0} u_1 \xrightarrow{e_1} \ldots u_{k-1} \xrightarrow{e_{k-1}} u_k = t)$ be any solution path with $\delta(u,T) = \omega(p)$. It is easy to see that $h(u) \preceq w(e_0) \times h(u_1) \preceq w(e_1) \times w(e_2) \times \ldots \times \omega(e_{k-1}) \times h(u_k) = \delta(u,T)$. □

Now we can define A* with the following (underlined) changes to Dijkstra's algorithm.

**Initialization** $f'(u_0) \leftarrow \underline{h(u_0)}$, *Open* $\leftarrow \{u_0\}$; **for each** $u \neq u_0 : f'(u) \leftarrow \mathbf{0}$

**Selection** Select $u \in$ *Open* with $\underline{f'(u)} = \underline{\bigsqcup\{f'(v) \mid v \in \textit{Open}\}}$

**Update** $f(v) \leftarrow \bigsqcup\{\{f(v)\} \cup \{f(u) \times \omega(e) \mid u \xrightarrow{e} v\}\}$
$f'(v) \leftarrow f(v) \underline{\times h(v)}$

**Proposition 6** *Cost-algebraic A* for consistent estimates solves the optimality problem.*

**Proof:** Let $p = (u = u_0 \xrightarrow{e_0} u_1 \xrightarrow{e_1} \ldots u_{k-1} \xrightarrow{e_{k-1}} u_k = t)$ be a prefix-optimal least-cost solution path in the labeled edge-weighted graph as computed by A*. We have

$$f'(p) = \omega(p) \times h(t) = \omega(p) = f(p).$$

Therefore, the cost values at goal nodes of Dijkstra's algorithm and A* match. If the heuristic is consistent, goal paths in the algorithm of Dijkstra and A* are looked at in the same order. Since $f'(v) = f(v) \times h(v) = f(u) \times \omega(e) \times h(v) \succeq f(u) \times h(u) = f'(u)$ for all $e$ with $in(e) = u$ and $out(e) = v$, we have that $f'$ costs on solution paths are monotonic increasing with respect to $\preceq$. Thus, A* terminates with the optimal solution. □

For non-consistent heuristics, the above proof is invalid as the $f'$ costs are no longer monotonic increasing. This is dealt with *re-opening* (Pearl 1985), namely re-considering nodes that have already been expanded.

We can prove the following invariance result.

**Proposition 7** *Let $p = u_0 \to \ldots \to u_k$ be a prefix-optimal least-cost path from the start node $s = u_0$ to a goal node $u_k$. At each selection of a node $u$ from* Open, *there is a node $u_i$ in* Open *such that $f'(u_i) = \delta(s,u_i) \times h(u_i)$.*

**Proof:** At the start of the algorithm, we have the trivial path $p = (s)$ with $f'(s) = h(s)$, so that $u_i = s$ preserves the property. On the other hand any call to *Update* maintains the property as follows. Without loss of generality let $i$ be maximal among the nodes satisfying the property.

If node $u$ is not on $p$ or $f'(u) \succ \delta(s,u) \times h(u)$, node $u_i \neq u$ remains in *Open*. No successor $v$ of $u$ along $e$ on $p$ in *Open* with $f'(v) = \delta(s,v) \times h(v) \preceq f(u) \times \omega(e) \times h(v)$ is changed.

If node $u$ is on $p$ and $f'(u) = \delta(s,u) \times h(u)$ we distinguish the following cases. If $u = u_k$, the algorithm terminates and there is nothing to show.

If $u = u_i$, *Update* will be called for successor $v = u_{i+1}$ of $u$; for all other successor nodes, the above argument holds. If $v$ is already expanded then $f'(v) \succ \delta(s,v) \times h(u)$, and it will be re-inserted into *Open* with $f'(v) = f(u) \times \omega(e) \times h(v) = \delta(s,v) \times h(v)$. If $v$ is brand new, it is inserted into *Open* with this merit. Otherwise, *Update* sets it to $\delta(s,v) \times h(v)$. In either case, $v$ guarantees the invariance.

Now suppose $u \neq u_i$. By the maximality assumption of $i$ we have $u = u_l$ with $l < i$. If $v = u_i$, no call to *Update* can change it because $u_i$ already has optimal merit $f'(v) = f(u) \times \omega(e) \times h(v) = \delta(s,v) \times h(v)$. Otherwise, $u_i$ remains in *Open* with unchanged $f$-value, thus, $u_i$ still preserves the invariance. □

**Proposition 8** *Cost-algebraic A* with re-openings solves the optimality problem for admissible estimates.*

**Proof:** Assume the algorithm terminates at goal $v$ with $f'(v) \succ \delta(s,T)$. According to Proposition 7, there is a node $u$ with $f'(u) = \delta(s,u) \times h(u)$ in *Open*, which lies on a prefix-optimal least-cost solution path $p$ to $t$. We have $f'(v) \succ \delta(s,T) = \delta(s,u) \times \delta(u,T) \succeq \delta(s,u) \times h(u) = f'(u)$, in contradiction to the fact that $v$ is selected from *Open*. As $h$ is admissible we have used $h(u) \preceq \delta(u,T)$.

Since there is still one node $u_i$ in *Open* for which we have $f'(u_i) = \delta(s, u_i) \times h(u_i)$, the algorithm cannot terminate without eventually selecting this node, while extending a least-cost prefix-optimal solution path. □

It is often said that A* does not only yield an optimal solution, but that it expands the minimal number of nodes (up to tie-breaking) (Dechter & Pearl 1983). The result, however, holds only for *consistent* heuristics, as there are example graphs in which we have exponential re-openings for *admissible* heuristics (Pijls & Kolen 1992).

With the algorithm of Bellman and Ford there is a reopening strategy, which selects the element in *Open* that was inserted first, and re-insert all successors not in *Open*. It cannot terminate at the first goal node, as the first obtained solution will not necessarily be optimal. An extension is needed that improves a solution cost bound.

It is not difficult to see that the cost-algebraic version of the Bellman-Ford then solves the optimality problem. The algorithm terminates only when set *Open* becomes empty. According to Proposition 7, throughout the algorithm there is a node $u_i$ in *Open* for which we have $f'(u_i) = \delta(s, u_i) \times h(u_i)$. Selecting $u_i$ extends a prefix-optimal least-cost solution path $p$ with goal node $t$ by one edge. At the end, for the stored solution we have $f'(t) = \delta(s, T) = \omega(p)$. It is also obvious that the algorithm is of polynomial time.

## Experiments

We performed our experiments on an AMD Athlon, 1.3 GHz machine with 512 MB of main memory running Linux operating system. The experiments are ran on two different sets: random graphs to show the feasibility of our generalized cost-algebraic approach, and route-planning graphs, constructed from Global Positioning System (GPS) traces, to demonstrate the real world applicability of the Cartesian product approach. For the first set we generated different random graphs based on the $G(n, p)$ model, where $n$ is the number of nodes and $0 \leq p \leq 1$ determines the probability of the existence of an edge between two nodes. The start and goal nodes are also selected at random. The comparisons between two algorithms are done by using the same random nodes.

We have implemented our approach in C++. The generalized cost structure is made possible to be implemented by the use of *Templates* facility in C++. The results for different cost structures are shown in Tables 1. Columns $v$ and $t$ denote the number of visited edges, and the time taken by Dijkstra's algorithm, respectively.

In the result tables, the columns with subscripts $\phi$ corresponds to the results due to the use of abstraction heuristics. To generate heuristic estimates we applied graph abstractions which ensure that if there is an initial goal path in the concrete graph, there is one in the abstract system, and the cost of the optimal initial goal path in the concrete system is smaller (w.r.t. $\preceq$) than the cost of the one in the abstract system. The inverse graph of the abstract system is explored in order to create a pattern database (Culberson & Schaeffer 1998) that stores the exact distances from abstract states to the set of abstract goal states. The distance database is used as heuristic for analyzing the concrete graph. Therefore, the second set of experiments are performed by first generating an abstract graph by iteratively merging two randomly chosen nodes. For instance, if we merge nodes $v_1$, $v_2$ and there are edges $v_1 \xrightarrow{e_1} v_3$, $v_2 \xrightarrow{e_2} v_3$ or there are edges $v_3 \xrightarrow{e_1} v_1$, $v_3 \xrightarrow{e_2} v_2$ we merge them and set $\omega(\{e_1, e_2\}) = \omega(e_1) \sqcup \omega(e_1)$. Note that merging of edges may reduce the search effort but is not mandatory in multiple-edge graphs. Similarly, self-loops are eliminated as they do not contribute to a better solution. It is not difficult to see that a complete exploration with the cost-algebraic version of Dijkstra's algorithm on this abstract and inverse graph yields a consistent heuristic.

| Optimization | | | | | |
|---|---|---|---|---|---|
| $n$ | $e$ | $v$ | $t$ | $v_\phi$ | $t_\phi$ |
| 1,000 | 30,111 | 25,929 | 0.22s | 5,612 | 0.08s |
| 5,000 | 749,826 | 372,802 | 5.62s | 41,397 | 0.73s |
| 7,500 | 1,684,978 | 947,908 | 13.29s | 100,120 | 1.71s |
| 10,000 | 2,997,625 | 1,700,163 | 28.85s | 66,379 | 1.31s |
| Probabilistic | | | | | |
| $n$ | $e$ | $v$ | $t$ | $v_\phi$ | $t_\phi$ |
| 1,000 | 30,111 | 16,330 | 0.14s | 902 | 0.01s |
| 5,000 | 749,826 | 365,066 | 5.72s | 7,607 | 0.08s |
| 7,500 | 1,684,978 | 1,636,157 | 19.04s | 22,250 | 0.33s |
| 10,000 | 2,997,625 | 2,743,029 | 36.32s | 56,021 | 1.07s |
| Max/Min | | | | | |
| $n$ | $e$ | $v$ | $t$ | $v_\phi$ | $t_\phi$ |
| 1,000 | 30,111 | 24,226 | 0.24s | 23,570 | 0.27s |
| 5,000 | 749,826 | 600,615 | 7.69s | 264,523 | 4.13s |
| 7,500 | 1,684,978 | 233,162 | 4.57s | 159,229 | 3.1s |
| 10,000 | 2,997,625 | 1,109,862 | 23.62s | 1,028,962 | 19.59s |

Table 1: Results on Optimization, Probabilistic and Max/Min cost structures.

The results show the effectiveness of cost algebraic heuristic search for all structures we studied. We achieved an exploration gain of up to two orders of magnitudes. The savings are considerably smaller in the fuzzy and (max/min) structures as there are more plateaus with the same $f$-value. The savings in explored edges increases with the size of the graph. Note that the tables should be read with the comparison of visited edges in generalized Dijkstra with that in generalized A*.

For the second set we utilized an already existing route-planning system called GPS-Route (Jabbar 2003) that when fed with a set of GPS points, can construct searchable maps. It is extended to work on Cartesian product of two different costs: travel distance and travel time, with both being instances of optimization cost structure discussed earlier. A shortest-quickest path between two nodes is sought for, which would give us the quickest path among all the shortest paths. Table 2 shows the results on some real-world maps from Southern Germany. The columns $v_\phi$ and $t_\phi$ denote the generated nodes and the time taken by A*. Another useful application of our Cartesian product approach would be to search for the shortest-straightest path, i.e., the shortest path involving least number of turns. This is possible

| $n$ | $e$ | $v$ | $t$ | $v_\phi$ | $t_\phi$ |
|---|---|---|---|---|---|
| 1,473 | 1,669 | 1,301 | 0.02s | 242 | 0.01s |
| 1,777 | 1,848 | 1,427 | 0.02s | 459 | 0.01s |
| 2,481 | 2,609 | 1,670 | 0.02s | 1,602 | 0.02s |
| 54,278 | 58,655 | 44,236 | 0.17s | 18,815 | 0.10s |

Table 2: Results on Shortest-quickest path search.

by taking a Cartesian product of optimization structure with $\langle \mathbb{N}, min, +, \leq, +\infty, 0 \rangle$ and initializing the new component of all edges to 1.

## Conclusion

We have proposed a general approach for finding optimal paths in graphs, where the notion of optimality is formalized by an algebraic structure to model costs. We generalized two algorithms to solve the generalized optimal path problem, Dijkstra and A*, and provided formal proofs of their correctness. Finally, we provided empirical evidence of the feasibility of our approach.

We expect that there is not much room for a more general cost structure than the one that we have chosen. Isotonicity appears to be mandatory, as it is easy to generate cases in which without istonicity the *only* optimal path is not prefix-optimal, as in the shortest-widest path.

In future we plan to extend our approach to other optimal path problems, for instance, by generalizing AO* algorithm used to solve the optimality problem in AND/OR graphs.

## References


Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *Journal of the ACM* 44(2):201–236.

Bistarelli, S.; Montanari, U.; and Rossi, F. 2002. Soft constraint logic programming and generalized shortest path problems. *Journal of Heuristics* 8(1):25–41.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. MIT Press.

Culberson, J. C., and Schaeffer, J. 1998. Pattern databases. *Computational Intelligence* 14(4):318–334.

Dechter, R., and Pearl, J. 1983. The optimality of A* revisited. In *AAAI*, 59–99.

Edelkamp, S.; Leue, S.; and Lluch Lafuente, A. 2003. Directed explicit-state model checking in the validation of communication protocols. *STTT* 5(2-3):247–267.

Ferrari, G., and Lluch Lafuente, A. 2004. A logic for graphs with QoS. In *1st Workshop on Views On Designing Complex Architecture*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for heuristic determination of minimum path cost. *IEEE Transactions on on Systems Science and Cybernetics* 4:100–107.

Jabbar, S. 2003. GPS-based navigation in static and dynamic environments. Master's thesis, University of Freiburg.

Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence* 27(1):97–109.

Lluch Lafuente, A., and Montanari, U. 2004. Quantitative mu-calculus and CTL based on constraint semirings. In *2nd Workshop on Quantitative Aspects of Programming Languages*, volume 112 of *ENTCS*.

Mandow, L., and de la Cruz, J. L. P. 2005. A new approach to multiobjective A* search. In *Proceedings of Nineteenth International Joint Conference on AI (IJCAI)*. to appear.

Müller, M. 2001. Partial order bounding: a new approach to evaluation in game tree search. *Artificial Intelligence* 129(1-2):279–311.

Pearl, J. 1985. *Heuristics*. Addison-Wesley.

Pijls, W., and Kolen, A. 1992. A general framework for shortest path algorithms. Technical Report 92-08, Erasmus Universiteit Rotterdam.

Rote, A. 1985. A systolic array algorithm for the algebraic path problem (shortest path; matrix inversion). *Computing* 34:191–219.

Sobrinho, J. L. 2002. Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet. *IEEE/ACM Tran. on Networking* 10(4):541–550.

Steward, B. S., and White, C. C. 2001. Multi-objective A*. *J. of ACM* 38(4):775–814.

Xiao, X., and Ni, L. 1999. Internet QoS: A big picture. *IEEE Network Magazine*.

Zhou, R., and Hansen, E. 2003. Sparse-memory graph search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1259–1268.