Yet another bidirectional algorithm for shortest paths

Wim Pijls[*] Henk Post[†]

Econometric Institute Report EI 2009-10

15 June, 2009

Abstract

For finding a shortest path in a network the bidirectional A* algorithm is a widely known algorithm. An A* instance requires a heuristic estimate, a real-valued function on the set of nodes. The version of bidirectional A* that is considered the most appropriate in literature hitherto, uses so-called balanced heuristic estimates. This means that the two estimates of the two directions are in balance, i.e., their sum is a constant value. In this paper, we do not restrict ourselves any longer to balanced heuristics. A generalized version of bidirectional A* is proposed, where the heuristic estimate does not need to be balanced. This new version turns out to be faster than the one with the balanced heuristic.
Keywords: Shortest path, Road network search, Bidirectional search.

1 Introduction

In the last decade we have seen a revival in shortest path algorithms. Because of the availability of large digital road maps, new sophisticated methods are needed. This paper deals with the point-to-point instance of the shortest path problem. The best-known algorithms in Operations Research literature are Bellman-Ford[1, 4] and Dijkstra[3]. In the literature on Artificial Intelligence one usually encounters the A* algorithm[9]. This algorithm assumes that a heuristic estimate is defined on the network. The point-to-point algorithms are nowadays applied in a bidirectional setting. This idea was introduced in [14]. Two simultaneous search processes starting from either endpoint meet somewhere in the middle between the endpoints.
For the A* algorithm one needs a heuristic function h estimating the remaining distance from a node to the target. When there are two processes, two estimates h and $\tilde{h}$ are used, each belonging to either process. The heuristic is called *balanced* if $h(v) + \tilde{h}(v)$ equals a constant value c for any node v.
The algorithm that is considered the most efficient of all previously known shortest path solutions is *bidirectional A** utilizing a balanced heuristic, cf. [6, 10, 11]. We discovered that this algorithm can be generalized to arbitrary (not necessarily balanced) heuristics. So a new bidirectional A* algorithm is proposed. It is an improved version with a shortened proof of the one in [13].

[*]Econometric Institute, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands, e-mail: pijls@ese.eur.nl

[†]Connexxion Taxi Services, The Netherlands, e-mail henk@ftonline.nl

Algorithm 1 NBA*, New Bidirectional A*

```
1: $S = \emptyset$;
2: $R = \emptyset$;                       // $S$ and $R$ are sets
3: $\mathcal{M} = V$;                     // $\mathcal{M}$ is a shared set
4: $\mathcal{L} = \infty$;                // $\mathcal{L}$ is a shared real value
5: for all $v \in V$ do
6:     $g(v) = \infty$;
7: end for
8: $g(s) = 0$;
9: $f = g(s) + h(s)$;
10: while any $v \in \mathcal{M}$ has $g(v) < \infty$ do
11:     $u_0 = \arg\min\{g(v) + h(v) \mid v \in \mathcal{M}\}$;          // $u_0$ is selected
12:     $\mathcal{M} = \mathcal{M} - \{u_0\}$;
13:     if $g(u_0) + h(u_0) - h(t) \geq \mathcal{L}$ or $g(u_0) + \tilde{f} - \tilde{h}(u_0) \geq \mathcal{L}$ then
14:         $R = R + \{u_0\}$;                                    // $u_0$ is rejected
15:     else
16:         $S = S + \{u_0\}$;                                    // $u_0$ is stabilized
17:         for all edges $(u_0, v) \in E$ with $v \in \mathcal{M}$ do
18:             $g(v) = \min(g(v), g(u_0) + d(u_0, v))$;
19:             $\mathcal{L} = \min(\mathcal{L}, g(v) + \tilde{g}(v))$;
20:         end for
21:     end if
22:     $f = \min\{g(v) + h(v) \mid v \in \mathcal{M}\}$;
23: end while
```

Preliminaries. Let a directed graph or network G be given by a pair (V, E) with V the set of nodes and E the set of edges. A path is a sequence of nodes without duplicate elements, such that two consecutive nodes are connected by an edge. We assume that two particular nodes are given, an origin node and a destination node. The shortest path from the origin to the destination is looked for. The weight or length of an edge (u, v) is denoted by $d(u, v)$, whereas $d^*(u, v)$ denotes the length of a shortest path from u to v.
As aforementioned, A* assumes a heuristic estimate h, defined as function from V into $\mathbb{R}$. An estimate h is called *consistent* if h obeys the inequality $h(u) - h(v) \leq d^*(u, v)$ for any two nodes $u, v \in V$. In some textbooks a different definition is found: $h(u) - h(v) \leq d(u, v)$ for any edge $(u, v) \in V$. The two definitions are equivalent, as can readily be shown. In this paper h is always assumed to be consistent.

2 A new algorithm

NBA* is a new search algorithm akin to A*. It is assumed that the code runs simultaneously on two sides. Each side has a start node s and an end node t. On one side the origin is chosen as s and the destination as t, on the other side s and t play inverse roles. In the process starting from the origin we use the original graph and the original distance function $d(u, v)$. In the alternate process the so-called reverse graph is used, where every original edge (u, v) is replaced with edge (v, u) of equal distance. Speaking on bidirectional search we refer to the two processes as the *primary* and the *opposite* process. Either side, whether it has the origin or the destination as start node, may be appointed as primary. In the opposite process every variable is denoted by a tilde. The execution of NBA* stops when one side stops.

The real value $g(v)$ for any $v \in V$ is called the *label* of a node v. As soon as this label has a finite value, $g(v)$ equals the length of path from s to v. (This is proved formally in [12]). The variables S and R denote sets, whereas f is a real-valued variable. A node u_0 which is inserted into S is said to be *stabilized*, because its label is stable, i.e., it will not be changed any more. A node u_0 which is inserted into R is said to be *rejected*, since u_0 will not be expanded. However, u_0 may lie in the desired shortest path.
The code utilizes multiple variables from the opposite search, see the denotations with a tilde. These variables are read-only variables and can only be set by the opposite search process. The algorithm has two shared variables. viz. $\mathcal{L}$ and $\mathcal{M}$, which are read/write variables for both sides. A meeting point of the two processes arises when a node v has a finite label on either side, cf. line 19. When $\mathcal{L}$ has a finite value, this value is equal to the length of a path from the origin to the destination. On termination this path is a shortest path. The set $\mathcal{M}$ is the set of nodes in the medium. The algorithm stops when this set is empty or just infinite labels are left (in which case the network is not connected).

NBA* running in an unidirectional setting is equivalent to the traditional A* algorithm. Then $\mathcal{L}$ keeps value ∞ and $g(t)$ provides the shortest path value. When moreover $h \equiv 0$, the algorithm is equivalent to Dijkstra's algorithm.

3 Correctness of the algorithm

The correctness of NBA* is proved by Theorem 1. Before, we need three lemmas. The NBA* algorithm contains one main loop. The lemmas present invariants of this main loop. An invariant is an assertion, which holds at the start as well as at the end of each iteration. Invariants are proved by mathematical induction. First, one shows that the invariant holds at the start of the first iteration. Second, under the assumption that the invariant holds at the start of an iteration, one proves that it holds at the end. In our proofs, we restrict ourselves to the second induction step, the first step being trivial.

Lemma 1 *NBA* has the following invariant:*

a) $g(u) + h(u) \le f \le g(v) + h(v)$ *for any* $u \in S$ *and* $v \in \mathcal{M}$.

b) for any $(u, v) \in E$ *with* $u \in S$ *and* $v \in S \cup \mathcal{M}, g(v) \le g(u) + d(u, v)$.

c) if $v \in \mathcal{M}$ *and the nodes of a shortest path from* s *to* v *(apart from* v*) belong to* S*, then* $g(v) = d^*(s, v)$.

Proof
a) We show that the relation

$$g(u) + h(u) \le g(v) + h(v) \;\; \textit{for any } u \in S \;\; \textit{and } v \in \mathcal{M} \tag{1}$$

is preserved in each iteration of the main loop. Then the extended inequality (including f) holds trivially, since f equals the minimal value of $g(v) + h(v)$, $v \in \mathcal{M}$.
As $g(u_0) + h(u_0)$ is minimal outside S, (1) is preserved, when u_0 is inserted into S. Then $g(u_0) + h(u_0)$ becomes a maximal value inside S. For nodes v which do not take a new label in line 18, (1) is preserved.
For nodes v which do take a new label, we obtain using the consistency of h: $g(v) + h(v) = g(u_0) + d(u_0, v) + h(v) \ge g(u_0) + h(u_0)$. For those nodes v, relation (1) is preserved, since $g(u_0) + h(u_0)$ is maximal in S.

b) When a node u_0 is inserted into S, line 18 makes sure that the invariant holds for any (u_0, v) with $v \in M$. For an edge (u_0, v) with $v \in S$ the following holds: $g(v) + h(v) \leq g(u_0) + h(u_0)$ and hence, by the consistency of h, $g(v) \leq g(u_0) + h(u_0) - h(v) \leq g(u_0) + d(u_0, v)$.

c) This part is a result of part b). Let P be given by the series $(s = p_0, p_1, p_2, \ldots, p_n = v)$ with $p_i \in S$ for $0 \leq i < n$. Part b) states: $g(p_{i+1}) \leq g(p_i) + d(p_i, p_{i+1})$ for $0 \leq i < n$. Using the relation $g(s) = 0$, we conclude that $g(v) \leq \sum_{i=0}^{i=n-1} d(p_i, p_{i+1})$. This sum is equal to $d^*(s, v)$. Since $g(v)$ equals the length of path from s to v and hence $g(v) \geq d^*(s, v)$ we conclude $g(v) = d^*(s, v)$. $\Box$

For the next lemmas we need two definitions.

Definition 1 *An edge $(u, v) \in E$ is called a connection edge during execution of NBA*, if $u \in S$ and $v \in \tilde{S}$.*

Definition 2 *A path P given by $(s = p_0, p_1, p_2, \ldots, p_n = v)$ is called closed during execution of NBA*, if $p_i \in R \cup \tilde{R}$ for at least one i, $0 \leq i \leq n$, or an edge (p_i, p_{i+1}), $0 \leq i < n$, is a connection edge.*
If a path is not closed, it is called open.

Note that, if a path is closed in the primary process, it also obeys the definition of *closed* in the opposite process.
If a path is open, it contains nodes from S, $\tilde{S}$ or $\mathcal{M}$, but no two consecutive nodes out of S and $\tilde{S}$ respectively.

Lemma 2 *NBA* has the following invariant: for any $v \in S \cup R$, $g(v) = d^*(s, v)$ if at least one shortest path from s through v is open.*

Proof
Suppose a value for u_0 is established (cf. line 11) with an open shortest path P from s to u_0. Let p be the first node beyond S on P (maybe $p = u_0$). As P is open, $p \in \mathcal{M}$. The following schema holds::

$$\begin{aligned} g(u_0) + h(u_0) &\geq d^*(s, u_0) + h(u_0) \\ &= d^*(s, p) + d^*(p, u_0) + h(u_0) \\ &= g(p) + d^*(p, u_0) + h(u_0) \qquad \textit{(by Lemma 1c)} \qquad (2) \\ &\geq g(p) + h(p) \qquad \textit{(h is consistent)} \qquad (3) \end{aligned}$$

As $g(u_0) + h(u_0)$ is minimal in $\mathcal{M}$, the above inequalities are strict equalities and hence $g(u_0) = d^*(s, u_0)$. $\Box$

Lemma 3 *NBA* has the following invariant: the length of any closed path P from s to t is greater than or equal to $\mathcal{L}$.*

Proof
Suppose that u_0, included in P, is added to R (cf. line 14). Then $g(u_0) + h(u_0) - h(t) \geq \mathcal{L}$ or $g(u_0) + \tilde{f} - \tilde{h}(u_0) \geq \mathcal{L}$. Let path P' denote a path from s to t through u_0 with minimal length. To prove the invariant we prove that $L' \geq \mathcal{L}$ where L' equals the length of P'. This length L' has $L' = d^*(s, u_0) + d^*(u_0, t)$ and $L' = d^*(s, u_0) + \tilde{d}(t, u_0)$. We assume that P' is open. Otherwise, by the invariant itself, $L' \geq \mathcal{L}$. For this open P', Lemma 2 states that $g(u_0) = d^*(s, u_0)$.

Now, the relation $L' \geq \mathcal{L}$ is a result of two general inequalities: $d^*(u_0, t) \geq h(u_0) - h(t)$ and $\tilde{d}(t, u_0) \geq \tilde{f} - \tilde{h}(u_0)$. The former inequality holds by the definition of consistency. The latter inequality is derived as follows. The first node on P' beyond $\tilde{S}$ is called q. Since P' is open, $q \in \mathcal{M}$. The following schema holds:

$$\begin{array}{rcll} \tilde{d}^*(t, u_0) & = & \tilde{d}^*(t, q) + \tilde{d}^*(q, u_0) & \\ & = & \tilde{g}(q) + \tilde{d}^*(q, u_0) & \textit{(by Lemma 1c)} \\ & \geq & \tilde{g}(q) + \tilde{h}(q) - \tilde{h}(u_0) & \textit{(}\tilde{h}\textit{ is consistent)} \\ & \geq & \tilde{f} - \tilde{h}(u_0) & \textit{(by Lemma 1a)} \end{array}$$

Our conclusion is that the invariant is preserved when a node is added to R in line 14. When a node is added to $\tilde{R}$, the foregoing reasoning is applied in the opposite process.

Suppose u_0 included in P is added to S and a node v with $(u_0, v) \in E$ is already in $\tilde{S}$. When u_0 is selected, $\tilde{g}(u_0) \leq \tilde{g}(v) + \tilde{d}(v, u_0)$ by Lemma 1b. Let path P' with length L' denote the shortest path from s to t through edge (u_0, v). Again we may assume that P' is open. Lemma 2 says that $g(u_0) = d^*(s, u_0)$ and in the opposite process $\tilde{g}(v) = \tilde{d}^*(t, v)$. A trivial invariant of the algorithm is: $\mathcal{L} \leq g(u_0) + \tilde{g}(u_0)$. We have the following series of inequalities: $\mathcal{L} \leq g(u_0) + \tilde{g}(u_0) \leq g(u_0) + \tilde{g}(v) + \tilde{d}(v, u_0) = d^*(s, u_0) + \tilde{d}(v, u_0) + \tilde{d}^*(t, v) = L'$.
This reasoning is applied to the opposite process, if u_0 is already in S when v is inserted into $\tilde{S}$. $\square$

Theorem 1 *On termination $\mathcal{L}$ is equal to the length of the shortest path from the source to the destination.*

Proof
Let P be an arbitrary path from s to t with length L. Since $\mathcal{M}$ does not contain any finitely labeled node on termination, P must be a closed path (or otherwise the unidirectional version applies). Lemma 3 says $L \geq \mathcal{L}$. There is at least one path with length equal to $\mathcal{L}$. $\square$

4 Implementing and comparing NBA*

In an implementation of NBA* the sets R and S need not to be maintained, provided that the property $v \in \mathcal{M}$ can checked quickly for any node v. It is recommended to maintain two heaps, one with $g + h$ and one with $\tilde{g} + \tilde{h}$ values.
For the function h and $\tilde{h}$, one can choose $h(v) = \delta(v, t)$ and $\tilde{h}(v) = \tilde{\delta}(v, \tilde{t}) = \delta(s, v)$ where δ is an underestimate of the distance function d.
We have conducted experiments with NBA* on the road network of the Netherlands, Belgium, Luxembourg and Germany. This network is part of the Multinet version 2008 provided by TeleAtlas. The directed graph consists of 8,184,650 nodes and 17,474,810 directed edges. Our above function $\delta(v, w)$ is equal to the Euclidean distance between v and w. In our implementation, the two processes alternate during execution of the bidirectional algorithm. Each turn corresponds to one iteration of the main loop. Figure 1 displays the final configuration of one run. The bright (yellow) area is the set $S \cup \tilde{S}$ on termination, whereas the dark (blue) area is the final set $R \cup \tilde{R}$. The shortest path found is also shown. In line 13 of the code the criterion $g(u_0) + h(u_0) - h(t) \geq \mathcal{L}$ is checked first; if this check fails, the second check $g(u_0) + \tilde{f} - \tilde{h}(u_0) \geq \mathcal{L}$ is performed. It turns out that, when a node u_0 is rejected, in almost any case the criterion $g(u_0) + \tilde{f} - \tilde{h}(u_0) \geq \mathcal{L}$ applies.

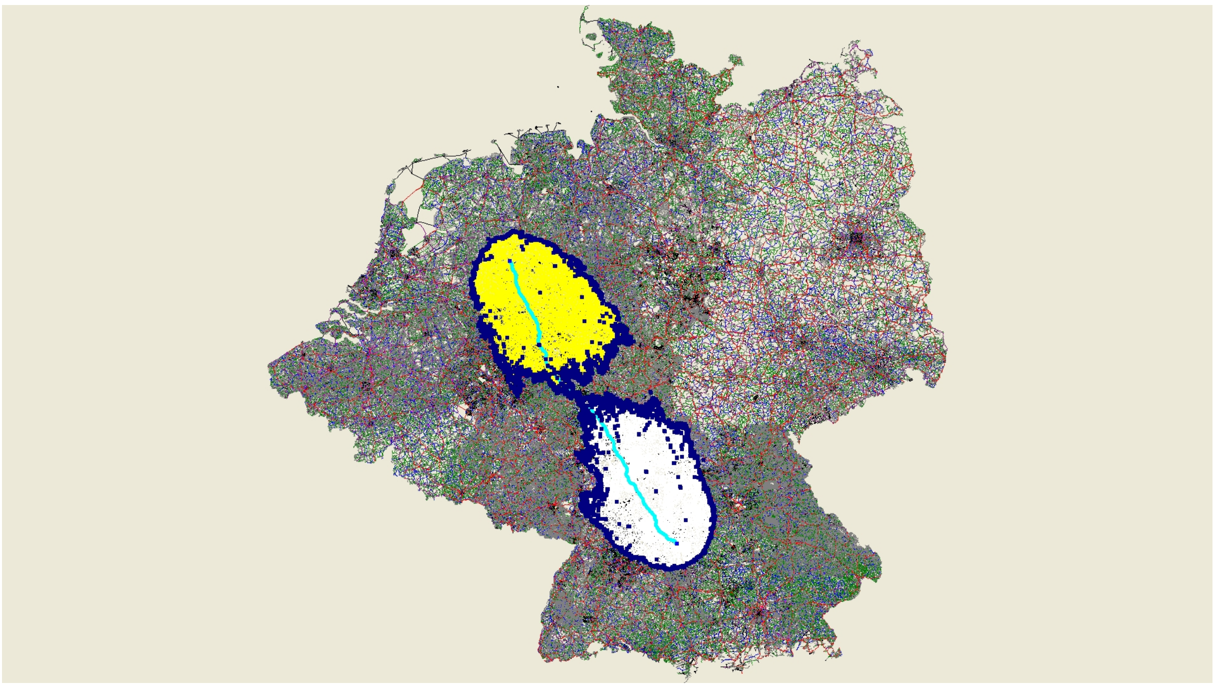

Figure 1: NBA* running on a real-world network.

The bidirectional versions of A* that were published previously[6, 10], utilized a balanced heuristic. As aforementioned, a balanced heuristic satisfies $h(v) + \tilde{h}(v) - c = 0$ for any v with c a constant value. A balanced heuristic is obtained for instance by making: $h(v) = (\delta(v,t) - \delta(s,v))/2$ and $h(v) = -\tilde{h}(v)$. In the algorithms in [14, 6, 10], a value $\mathcal{L} = g(u) + \tilde{g}(u)$ is established only when u is included in both S-sets. Then the algorithm enters the postprocessing stage. Lemma 1 is still valid. If any node v between the two S-sets is selected in the new stage, the following (in)equalities hold:

$$\begin{aligned}
\mathcal{L} &= g(u) + \tilde{g}(u) \\
&= g(u) + h(u) + \tilde{g}(u) + \tilde{h}(u) - c && \textit{(}h, \tilde{h} \textit{ is balanced)} \\
&\leq g(u) + h(u) + \tilde{f} - c && \textit{(by Lemma 1a in the opposite process)} \\
&\leq g(v) + h(v) + \tilde{f} - c && \textit{(by Lemma 1a)} \\
&= g(v) + \tilde{f} - \tilde{h}(v) && \textit{(}h, \tilde{h} \textit{ is balanced)}
\end{aligned}$$

Consequently any node $v \notin S \cup \tilde{S}$, when selected, would be inserted into R. Therefore in the postprocessing stage, the primary process selects only nodes that are stabilized in the opposite process, or equivalently, it inspects edges connecting S to $\tilde{S}$.

As mentioned earlier, NBA* is an improved version of a previous algorithm presented in [13]. The performance is similar. In [13] we compared our previous algorithm with the bidirectional algorithm described in [6] and [10], where a balanced heuristic is applied. It turned out that our algorithm clearly beats the other one in running time. This is due to the fact that our algorithm needs the computation of only one δ-value at each node v, whereas balanced heuristics compute two δ-values. NBA* shows that balanced heuristics are no longer needed. So we get rid of the somewhat awkward estimate including two functions δ. See [13] for detailed results and figures.

5 Concluding remarks

Another advantage of NBA* is that it quickly establishes a finite $\mathcal{L}$. We noticed that the first value $\mathcal{L}$ is updated only a few times. So the first value turns out to be close to the final value for most instances. Achieving soon a finite $\mathcal{L}$ is convenient when a quick computation of a (nearly) shortest path is wanted. This is of interest e.g. for taxi-drivers. Again we refer to [13] for figures.

The results in [13] are based upon the Euclidean heuristic. There are other heuristics, e.g. the heuristics in [5, 8, 7]. A promising instance is the so-called landmark heuristic[6]. Experimenting with NBA* using different heuristic estimates will be a subject in future research.

Appendix

The properties in this appendix are not relevant to the correctness of NBA*, but are important for getting a deeper insight into the working of the algorithm.
Lemma 2 suggests that the relation $g(v) \neq d^*(s, v)$ may occur for a selected node v. The network in Figure 2 will show us that this situation really happens. All edges in the primary process are directed from left to right. Two values for $h(b)$ are taken into account: $h(b) = 0$ and $h(b) = 4$. For any other node v $(v \neq b)$ we assume $h(v) = 0$. The algorithm may run according to the following steps:

1) First of all, s and $\tilde{s} = t$ are inserted in S and $\tilde{S}$ respectively. Then $g(a) = 5$, $\tilde{g}(a) = 6$ and $\mathcal{L} = 5 + 6 = 11$.
2) The process on the right selects node a (cf. line 11).
   If $h(b) = 0$, then $f = 3$ and $\tilde{g}(a) + f - h(a) = 6 + 3 - 0 = 9 < \mathcal{L}$.
   If $h(b) = 4$, then $f = 7$ and $\tilde{g}(a) + f - h(a) = 6 + 7 - 0 = 13 > \mathcal{L}$.
   In the former case a is added to $\tilde{S}$ and (s, a) becomes a connection edge, in the latter case it is added to $\tilde{R}$.
3) The process on the left inserts b into S. Next c is selected The g-label of c equals 7. The following holds: $g(c) = 7 > d^*(s, c) = 5 + 1$.

Apart from $g(c) > d^*(s, c)$, we notice another phenomenon. Since $\tilde{f}$ is equal to or larger than 6, we have $g(c) + \tilde{f} + \tilde{h}(c) \geq 7 + 6 + 0 = 13 \geq \mathcal{L}$ and hence c is rejected. In general, every node u_0 with a closed path between s and u_0 is rejected after being selected in line 11. This is proved in Lemma 4.

Lemma 4 *When u_0 is selected, it is added to R, if at least one shortest path P from s to u_0 is closed.*

Proof
In this proof, almost all inequalities exploit the consistency of h. If not, an additional explanation will be given.
a) Assume P contains a node $w \in R$ between s and u_0. We can choose w such that the part between s and w is open (otherwise part b) applies). Then $g(w) = d^*(s, w)$ by Lemma 2 and consequently $g(u_0) \geq d^*(s, u_0) = g(w) + d^*(w, u_0)$.
If w satisfies $g(w) + h(w) - h(t) \geq \mathcal{L}$, then the following inequalities are relevant: $g(u_0) + h(u_0) - h(t) \geq g(w) + d(w, u_0) + h(u_0) - h(t) \geq g(w) + h(w) - h(t)$.
If w was inserted into R due to the relation $g(w) - \tilde{h}(w) + \tilde{f} \geq \mathcal{L}$, then: $g(u_0) - \tilde{h}(u_0) \geq g(w) + d(w, u_0) - \tilde{h}(u_0) \geq g(w) - \tilde{h}(w)$. Since $\tilde{f}$ is a non-decreasing quantity, $g(u_0) + \tilde{f} - \tilde{h}(u_0) \geq \mathcal{L}$ and u_0 is inserted into R.

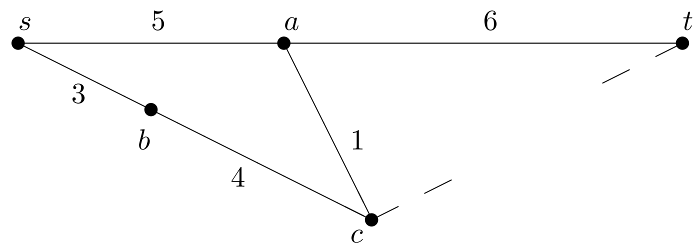

Figure 2: Node c is selected with $g(c) > d^*(s,c)$.

In both situations, u_0 is rejected.
b) Assume P contains a node $w \in \tilde{R}$ or a connection edge (w', w) with $w' \in S$ and $w \in \tilde{S}$. If $w \in \tilde{S}$, then $\tilde{f} \geq \tilde{g}(w) + \tilde{h}(w)$ by Lemma 1a. This lemma can be extended. The inequality in Lemma 1a also holds for any $u \in R$. This implies for $w \in \tilde{R}$ that $\tilde{f} \geq \tilde{g}(w) + \tilde{h}(w)$.

$$\begin{aligned} g(u_0) + \tilde{f} - \tilde{h}(u_0) &\geq g(u_0) + \tilde{g}(w) + \tilde{h}(w) - \tilde{h}(u_0) \\ &\geq g(u_0) + \tilde{g}(w) - d^*(w, u_0) \\ &\geq d^*(s, u_0) + d^*(w, t) - d^*(w, u_0) \\ &= d^*(s, w) + d^*(w, t) \\ &\geq \mathcal{L} \end{aligned}$$

The latest inequality holds by Lemma 3. It follows that u_0 is added to R.□

Corollary 1 *NBA* has the following invariant: for any $v \in S$, $g(v) = d^*(s, v)$.*

Proof
Lemma 4 implies that, when u_0 is added to S, every path from s to u_0 is open. Lemma 2 says $g(u_0) = d^*(s, u_0)$. □

Corollary 1 shows the optimality of NBA*. When a node v is stabilized, $g(v) = d^*(s, v)$ holds and $g(v) + \tilde{f} - \tilde{h}(v) < \mathcal{L}$ holds. Hence, such a node is a candidate for being a member of the shortest path. This result is the bidirectional counterpart of the results in [2].

Suppose a node u_0 is inserted into S and a shortest path from s to u_0 is called P. Although P cannot contain a connection edge (w', w) due to Lemma 4, it still may contain a node $w \in \tilde{S}$. This illustrated in Figure 3. Again, in the primary network, edges are directed from left to right. Further we have $h(a_i) = i$ and $\tilde{h}(b_i) = i$, $i = 1, 2, 3$. The other h- and $\tilde{h}$-values are equal to 0. The following steps may be executed:

1) s and $\tilde{s} = t$ are inserted in S and $\tilde{S}$ respectively.
2) $\tilde{g}(a_1) + \tilde{h}(a_1) = 4 + 0$ and $\tilde{g}(b_3) + \tilde{h}(b_3) = 1 + 3$. Assume that a_1 is selected in the right process and inserted into $\tilde{S}$.
3) Similarly, assume that the left process selects b_1. Although one shortest path to b_1 includes a node from $\tilde{S}$, b_1 is still stabilized.

Finally we discuss the following situation. Suppose a node u_0 is selected and every path between u_0 and t is closed. Then it is still possible that u_0 is not rejected, since it is not necessary that the rejection condition in line 13 is fulfilled.

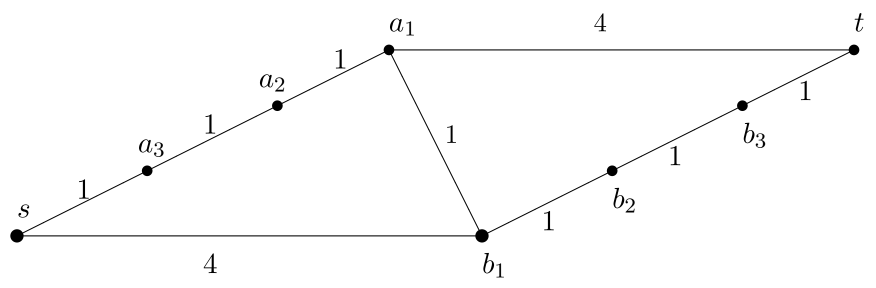

Figure 3: Stabilizing b_1 with a closed shortest path.

References

[1] R. Bellman, On a routing problem, Quaterly of Applied Mathematics 16(1) (1958) 87-90.

[2] R. Dechter, J Pearl, The optimality of A*, in: Search in Artificial Intelligence, Springer-Verlag, ISBN 0-387-96750-8 (1988), 166–199.

[3] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269-271.

[4] L.R. Ford, Network Flow Theory, Technical Report P-923 Rand Corporation, Santa Monica CA 1956.

[5] L. Fu, D. Sun, L.R. Rilett, Heuristic shortest path algorithms for transportation applications: State of the art. Computers and Operations Research 33 (2006) 3324-3343.

[6] A.V. Goldberg, C. Harrelson, Computing the Shortest Path: A* Search Meets Graph Theory, 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05) 2005.

[7] A. V. Goldberg, H. Kaplan, R. F. Werneck, Better Landmarks Within Reach, In: WEA 2007, LNCS 4525 (2007) 3851.

[8] R. Gutman, Reach-based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In: Proceedings 6th ALENEX (2004) 100111.

[9] E. P. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transaction, System Science and Cybernetics SSC(4)-2 (1968) 100-107.

[10] T. K. Ikeda, M. Hsu, H. Inai, S. Nishimura, H. Shimoura, T. Hashimoto, K. Tenmoku, K. Mitoh, A Fast Algorithm for Finding Better Routes by AI Search Techniques, Proceedings Vehicle Navigation and Information Systems Conference, IEEE 1994.

[11] G.A. Klunder, H.N Post, The Shortest Path Problem on Large Scale Real Road Networks, Networks, 48(4) (2006) 182-194.

[12] W. Pijls, Heuristic estimates in shortest paths, Statistica Neerlandica, 61(1) (2007) 61-74.

[13] W. Pijls and H. Post, A new bidirectional search algorithm with shortened postprocessing, European Journal of Operational Research, 198 (2009), pp. 363-369.

[14] I. Pohl, Bi-Directional Search, Machine Intelligence 6 (1971) 124-140.