A Note on the Connection between the Primal-Dual and the A* Algorithm

Xugang Ye, Johns Hopkins University, USA

Shih-Ping Han, Johns Hopkins University, USA

Anhua Lin, Middle Tennessee State University, USA

ABSTRACT

The primal-dual algorithm for linear programming is very effective for solving network flow problems. For the method to work, an initial feasible solution to the dual is required. In this paper, we show that, for the shortest path problem in a positively weighted graph equipped with a consistent heuristic function, the primal-dual algorithm will become the well-known A* algorithm if a special initial feasible solution to the dual is chosen. We also show how the improvements of the dual objective are related to the A* iterations.

Keywords: shortest path; primal-dual; A algorithm*

INTRODUCTION

The primal-dual algorithm (Dantzig et al., 1956; Bertsimas & Tsitsiklis, 1997; Papadimitriou & Steiglitz, 1998) for linear programming is very effective for solving network flow problems. Despite the name, the method that we discuss in this paper is not to be confused with the currently well-known primal-dual interior methods. The latter were originated by Karmarkar's seminal paper (Karmarkar, 1984) and they enjoy polynomial time-complexity for solving general linear programming problems (Wright, 1997) and have been generalized for some convex conic optimization problems such as second-order cone programming and semidefinite programming (Boyd & Vandenberghe, 2004). The primal-dual algorithm discussed in this paper starts from an initial dual feasible solution and iteratively improves the solution until a primal feasible solution, determined by the current dual solution, is found such that the pair satisfies the complementary slackness conditions (Bertsimas & Tsitsiklis, 1997). In this paper, we consider the primal-dual algorithm for the shortest path problem in a positively weighted graph equipped with extra information and study the consequence of choosing a proper initial feasible solution to the dual.

The key point of this paper is to utilize the extra information, or heuristic in the language of the field of artificial intelligence, of the graph to construct an initial feasible solution to the dual. We show that the well-known A* algorithm (Hart et al., 1968; Hart et al., 1972; Nilsson, 1980; Pearl, 1984) can be derived. This derivation actually goes directly from the primal-dual algorithm to the A* algorithm. Furthermore, we show how the improvements of the dual objective are related to the A* iterations.

This paper is organized as follows. We first set up the problem domain and introduce the A* algorithm and the primal-dual algorithm. We then use the heuristic to construct an initial feasible solution to the dual and propose a best-first search (Pearl, 1984) version of the primal-dual algorithm. We show that this version of the primal-dual algorithm behaves essentially as same as the A* algorithm that uses the same heuristic. Finally, we provide additional discussions.

BACKGROUND

We consider a directed, positively weighted simple graph denoted as $G = (V, A, W, \delta, b)$, where V is the set of nodes, A is the set of arcs, $W: A \to R$ is the weight function, $\delta > 0$ is a constant such that $\delta \leq W(a) < +\infty$ for all $a \in A$, and finally b is a constant integer such that $0 < b < |V|$ and $|\{v \mid (u, v) \in A \text{ or } (v, u) \in A\}| \leq b$ for all $u \in V$. Suppose we want to find a shortest *s-t* (directed) path in G, where $s \in V$ is a specified starting node and $t \in V$ is a specified terminal node. Further suppose that there exists a heuristic function $h: V \to R$ such that $h(v) \geq 0$ for all $v \in V$, $h(t) = 0$, and $W(u, v) + h(v) \geq h(u)$ for all $(u, v) \in A$. h is called consistent heuristic. According to Hart et al. (1968), Hart et al.(1972), Nilsson (1980), and Pearl (1984), the A* algorithm that uses such h is *complete*, that is, it can find a shortest *s-t* path in G as long as there exists an *s-t* path in G. The algorithm can be stated as follows. It searches from s to t.

The A Algorithm*

Notations:
h: heuristic
O: Open list
E: Closed list
d: distance label
f: node selection key
$pred$: predecessor

```
Steps:
Given $G$, $s$, $t$, and $h$
Step 1. Set $O = \{s\}$, $d(s) = 0$, and $E = \phi$.
Step 2. If $O = \phi$ and $t \notin E$, then stop (there is
        no s-t path); otherwise, continue.
Step 3. Find $u = \arg\min_{v \in O} f(v) = d(v) + h(v)$.
        Set $O = O \setminus \{u\}$ and $E = E \cup \{u\}$. If $t \in E$, then stop (a shortest s-t path is
        found); otherwise, continue.
Step 4. For each node $v \in V$ such that $(u, v) \in A$ and $v \notin E$,
        if $v \notin O$, then
           set $O = O \cup \{v\}$, $d(v) = d(u) + W(u, v)$, and $pred(v) = u$;
        otherwise,
           if $d(v) > d(u) + W(u, v)$, then
              set $d(v) = d(u) + W(u, v)$ and $pred(v) = u$.
        Go to Step 2.
```

In particular, when $h = 0$, the A* algorithm stated above reduces to the Dijkstra's algorithm (Dijkstra, 1959; Ahuja et al., 1993; Papadimitriou & Steiglitz, 1998). For convenience, for any two nodes $u \in V$ and $v \in V$, let $dist(u, v)$ denote the distance from u to v in G. That is, if there is no *u-v* path in G, we define $dist(u, v) = +\infty$; otherwise, we define $dist(u, v)$ to be the length of a shortest *u-v* path in G. According to Hart et al. (1968) and Pearl (1984), a central property, called *strong optimality*, of the A* algorithm stated above is $d(u) = dist(s, u)$ when $u \in E$. If G is a *large* and *sparse* finite graph in the sense that $b << |V|$, then the algorithm can be efficiently implemented by storing G in the form of adjacency lists (Cormen et al., 2001) and maintaining the Open list O as a binary heap, or pairing heap, or Fibonacci heap (Ahuja et al., 1993; Cormen et al., 2001). For example, using a binary heap to maintain O yields a total number of operations bounded by $\mathrm{O}(|E_{\text{final}}|\cdot b\cdot\log_2|O_{\max}|)$ upon successful termination, where E_{final} is the final Closed list, $O_{\max}$ is the Open list of the largest size during the iterations, and "O" stands for the "big O" notation. In the worst case, this bound is just $\mathrm{O}(|A|\cdot\log_2|V|)$, which is the worst-case time-complexity of the Dijkstra's algorithm for the single-source shortest path problem using the binary heap implementation. The actual running time of the A* algorithm, however, is determined by $dist(s, t)$, h, and the data structure in implementation. One important result in the A* literature says that the better the heuristic function h, the smaller the size of E_{final}. We leave this to the discussions in the later sections.

Given a consistent heuristic h, we can define a new weight function W^h such that $W^h(u, v) = W(u, v) + h(v) - h(u)$ for all $(u, v) \in A$. This change of weights results in a new graph $G^h = (V, A, W^h, \delta, b)$. It has been known from Ahuja et al. (1993) that running the Dijkstra's algorithm to find a shortest s-t path in G^h is equivalent to running the A* algorithm stated above to find a shortest s-t path in G if the two algorithms apply the same tie-breaking rule. The equivalence is due to the fact that the two algorithms construct identical shortest path tree that is rooted at s although the distance labels of the same leaf are distinct. The equivalence tells that the two algorithms can be derived from each other.

Now consider modeling the shortest path problem as linear programming (LP). For convenience, we define $\tilde{G} = (V, \tilde{A}, \tilde{W}, \delta, b)$, where $\tilde{A} = \{(u, v) \mid (v, u) \in A\}$ and $\tilde{W}(u, v) = W(v, u)$ for all $(u, v) \in \tilde{A}$, that is, $\tilde{G}$ is formed by reversing the directions of all the arcs of G. Clearly, to find a shortest s-t path in G is equivalent to find a shortest t-s path in $\tilde{G}$. For each $(u, v) \in \tilde{A}$, let $x(u, v)$ denote the decision variable. A primal LP model for finding a shortest t-s path in $\tilde{G}$ is

$$\text{(P)}\quad \boxed{\begin{array}{ll}
\text{Min} \displaystyle\sum_{(u,v)\in\tilde{A}} \tilde{W}(u,v)\cdot x(u,v) & (2.1)\\
\text{Subject to} & \\
\displaystyle\sum_{v:(u,v)\in\tilde{A}} x(u,v) - \sum_{v:(v,u)\in\tilde{A}} x(v,u) & \\
= \begin{cases} 1 & \text{if } u = t;\\ -1 & \text{if } u = s;\\ 0 & \text{for all } u \in V \setminus \{s, t\},\end{cases} & (2.2)\\
x(u, v) \ge 0 \text{ for all } (u, v) \in \tilde{A}. & (2.3)
\end{array}}$$

As long as there exists an s-t path in G, it can be easily shown that a binary optimal solution to Model (2.1-2.3) exists. In fact, Model (2.1-2.3) is just to send a unit flow from a supplier t to a customer s in $\tilde{G}$ with least cost. The price of sending a unit flow along any $(u, v) \in \tilde{A}$ is $\tilde{W}(u, v)$. One option is to find a shortest t-s path in $\tilde{G}$ and send a unit flow along this path. The general option is to divide the unit flow into pieces. However, to minimize the cost, each piece must be sent along a shortest t-s path in $\tilde{G}$. This backward version of the primal LP model has a very nice dual, which can be expressed only with respect to G. In order to show this, we can first write down the dual of Model (2.1-2.3) with respect to $\tilde{G}$. It is

$$\text{(D)}\quad \boxed{\begin{array}{ll}
\text{Max } \pi(t) - \pi(s) & (2.4)\\
\text{Subject to} & \\
\pi(u) - \pi(v) \le \tilde{W}(u, v) & \\
\qquad \text{for all } (u, v) \in \tilde{A}, & (2.5)
\end{array}}$$

where for each $v \in V$, $\pi(v)$ is the decision variable, which is also known as the *potential* of v. Note that for each $(u, v) \in \tilde{A}$, $(v, u) \in A$ and vice versa. And also note that $\tilde{W}(u, v) = W(v, u)$. Hence Constraint (2.5) can be rewritten as

$\pi(u) - \pi(v) \le W(v, u)$ for all $(v, u) \in A$. (2.6)

By exchanging the arguments u and v, we can further rewrite (2.6) into

$\pi(v) - \pi(u) \le W(u, v)$ for all $(u, v) \in A$. (2.7)

Constraint (2.7) says that for each $(u, v) \in A$, a triangle inequality relative to s holds as the following Figure 1 shows. Hence if the primal is the model of searching from t to s in $\tilde{G}$, then the dual is the model of searching from s to t in G. Conversely, if the primal is the model of searching from s to t in G, then the dual is the model of searching from t to s in $\tilde{G}$.

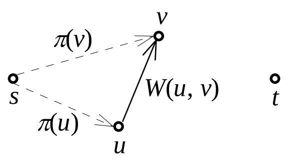

Figure 1: Triangle inequality relative to s for arc (u, v) in G

An obvious advantage of (D) is that a feasible solution is easy to find. At least, $\pi = 0$ is one. The key idea of the primal-dual algorithm for the shortest path problem, illustrated in Papadimitriou & Steiglitz (1998), is to start from a feasible solution π to (D), search for a feasible solution x to (P) such that for each $(u, v) \in A$, $x(u, v) = 0$ whenever $W(u, v) - \pi(v) + \pi(u) > 0$. If such x is found, then a shortest s-t path in G can be found. In fact, such x corresponds to an s-t path on which for each arc (u, v), the equality $W(u, v) - \pi(v) + \pi(u) = 0$ holds. If such x cannot be found, then some procedure is needed to update π such that Constraint (2.7) is still satisfied and Objective (2.4) is improved. An important feature of the primal-dual algorithm is that any equality in Constraint (2.7) still holds after π is updated. Another important feature is that after π is updated, one strict inequality in Constraint (2.7) may become equality. The primal-dual algorithm keeps attempting to construct an s-t path in G by using the arcs that correspond to the equalities in Constraint (2.7). According to Papadimitriou & Steiglitz (1998), given the initial feasible solution $\pi = 0$ to (D), the primal-dual algorithm behaves essentially as same as the Dijkstra's algorithm that searches from s to t in G. Hence the Dijkstra's algorithm can be derived from the primal-dual algorithm.

The two known results show that the A* algorithm with consistent heuristic h can be derived from the primal-dual algorithm. But the derivation needs the Dijkstra's algorithm as the bridge. It also involves the change of the weight function. In this paper we show that if we use h to construct an initial feasible solution to (D), then applying the primal-dual algorithm directly leads to the A* algorithm that searches from s to t in G.

DERIVATION

The key point of our derivation is to choose $\pi^{(0)} = -h$ as the initial feasible solution to (D). To justify the dual feasibility of $\pi^{(0)}$, we notice, by the consistency of h, that $W(u, v) + h(v) \ge h(u)$ for all $(u, v) \in A$. Hence $W(u, v) - \pi^{(0)}(v) \ge -\pi^{(0)}(u)$ for all $(u, v) \in A$. The inequality can be rewritten as $\pi^{(0)}(v) - \pi^{(0)}(u) \le W(u, v)$, which is exactly what the dual feasibility requires.

A nice property of (D) is that it does not require its solution to be nonnegative. Although $\pi^{(0)} = -h \le 0$, what really matters is $\pi^{(0)}(t) - \pi^{(0)}(s) = -h(t) + h(s) = h(s) \ge 0$. This means $\pi^{(0)} = -h$ is a better initial feasible solution to (D) than $\pi^{(0)} = 0$. But we still need to justify the validity of $\pi^{(0)} = -h$. That is, we still need to show that the primal-dual algorithm that starts from the solution $\pi^{(0)} =$

$-h$ to (D) can find a shortest s-t path in G as long as there exists an s-t path in G. It suffices to show the equivalence between the primal-dual algorithm that starts from $-h$ and the A* algorithm that uses h. We now give the description of the best-first search version of the primal-dual algorithm that starts from $-h$ as follows:

Algorithm 1

Notations:

h: heuristic

O: Open list

E: Closed list

π: potential

f_1: node selection key

$pred$: predecessor

θ: potential increment

Φ: cumulative potential increase

Steps:

```
Given $G, s, t$, and $h$
Step 1. Set $\Phi = 0$. Set $O = \{s\}$, $\pi(s) = -h(s)$,
        $pred(s) = s$, and $E = \phi$. Set $W(s, s) = 0$.
Step 2. If $O = \phi$ and $t \notin E$, then stop (there is
        no $s$-$t$ path); otherwise, continue.
Step 3. Find $u = \arg\min_{v \in O} f_1(v) = W(pred(v), v) - \pi(v) + \pi(pred(v))$.
        Set $\theta = W(pred(u), u) - \pi(u) + \pi(pred(u))$. Set
        $\Phi = \Phi + \theta$. Set $O = O \setminus \{u\}$ and $E = E \cup \{u\}$.
        Set $\pi(u) = -h(u) + \Phi$. If $t \in E$,
        then stop (a shortest $s$-$t$ path is found);
        otherwise, continue.
Step 4. For each $v \in O$, set $\pi(v) = -h(v) + \Phi$.
Step 5. For each $v \in V$ such that $(u, v) \in A$
        and $v \notin E$,
        if $v \notin O$, then
           set $O = O \cup \{v\}$, $pred(v) = u$, and
           $\pi(v) = -h(v) + \Phi$;
        otherwise,
           if $W(pred(v), v) + \pi(pred(v)) > W(u, v) + \pi(u)$, then
              set $pred(v) = u$.
        Go to Step 2.
```

For theoretical convenience, right after Step 5, for all $v \in V \setminus (E \cup O)$, we define $\pi(v) = -h(v) + \Phi$. We can show that Algorithm 1, just like the classical version (Papadimitriou & Steiglitz, 1998) of the primal-dual algorithm, maintains the dual feasibility throughout.

Proposition 1. The dual feasibility stated as Constraint (2.7) is maintained when running Algorithm 1.

Proof. The proof is inductive. The base case is upon the completion of the first iteration. At this moment, $\pi = \pi^{(0)}$, it's trivially true. Suppose right before the k-th iteration ($k > 1$), the potential of any $v \in V$ is $\pi(v)$ and π satisfies the dual feasibility. We need to show that right after the k-th iteration, the dual feasibility is still maintained. Right before the k-th iteration, let $[E, O]$ denote the E-O cut, which is the set of arcs from E to O. We only need to show that the node selection rule in Step 3 of Algorithm 1 is equivalent to finding an arc $(u, v) \in [E, O]$ such that $W(u, v) - \pi(v) + \pi(u)$ is the minimum. In fact,

$$\begin{aligned}
&\min_{(u,v)\in[E,O]} [W(u, v) - \pi(v) + \pi(u)] \\
&= \min_{v\in O} \min_{\substack{u\in E \\ (u,v)\in[E,O]}} [W(u, v) - \pi(v) + \pi(u)] \\
&= \min_{v\in O} \left[\left[\min_{\substack{u\in E \\ (u,v)\in[E,O]}} [W(u,v) + \pi(u)] \right] - \pi(v) \right] \\
&= \min_{v\in O} [W(pred(v), v) + \pi(pred(v)) - \pi(v)].
\end{aligned}$$

Hence, the node selection rule in Step 3 of Algorithm 1 is equivalent to the arc selection rule in the classical version of the primal-dual algorithm. Note that the later one guarantees the satisfaction of the dual feasibility right after the k-th iteration, therefore, the proposition is true. Q.E.D.

From Algorithm 1 we can see that the potentials of the nodes that have entered the Closed list E become permanent.

Furthermore, we can show that the potential difference between any $u \in E$ and s is actually the length of the shortest s-u path in G.

Proposition 2. After each iteration of Algorithm 1, $\pi(u) - \pi(s) = dist(s, u)$ for any node $u \in E$.

Proof. When node u enters E, an s-u pointer path, say P: v_1 (= s) ~ v_2 ~ ⋯ ~ v_k (= u), is determined. Denote $L(P)$ as the length of P. Note that $\pi(v_2) = W(v_1, v_2) + \pi(v_1)$, ⋯, $\pi(v_k) = W(v_{k-1}, v_k) + \pi(v_{k-1})$. By telescoping, we have $\pi(v_k) = L(P) + \pi(v_1)$, i.e. $\pi(u) - \pi(s) = L(P)$. Since there is an s-u path in G, there must be a shortest s-u path in G. This is because any s-u path in G with length no longer than $L(P)$ has only finite number of arcs, hence the number of s-u paths in G with length no longer than $L(P)$ is finite. Let $\widehat{P}: \widehat{v}_1$ (= s) ~ $\widehat{v}_2$ ~ ⋯ ~ $\widehat{v}_k$ (= u) be a shortest s-u path in G and denote $L(\widehat{P})$ as the length of $\widehat{P}$. By Proposition 1, Algorithm 1 maintains the dual feasibility stated as Constraint (2.7). Hence $\pi(\widehat{v}_2) \le W(\widehat{v}_1, \widehat{v}_2) + \pi(\widehat{v}_1)$, ⋯, $\pi(\widehat{v}_k) \le W(\widehat{v}_{k-1}, \widehat{v}_k) + \pi(\widehat{v}_{k-1})$. By telescoping, we have $\pi(\widehat{v}_k) \le L(\widehat{P}) + \pi(\widehat{v}_1)$, i.e. $\pi(u) - \pi(s) \le L(\widehat{P})$. The two arguments jointly imply that P is in fact a shortest s-u path and $\pi(u) - \pi(s) = dist(s, u)$. Q.E.D.

We now show the equivalence between Algorithm 1 and the A* algorithm we listed at the beginning.

Proposition 3. Under the same tie-breaking rule, Algorithm 1 is equivalent to the A* algorithm that uses h, searching from s to t.

Proof. The proof is inductive. We need to show that right after each same iteration, the two algorithms have the same Open list and Closed list, and for each node in the Open list, the two algorithms assign the same predecessor in the Closed list. The base case is upon the completion of the first iteration of the two algorithms, respectively. Note that in the base case, s is the only node "closed" by the two algorithms. Hence the base case is trivially true. Suppose (inductive hypothesis) right before the k-th iteration ($k > 1$) of the two algorithms, the claim above is true. We now show that the claim still holds right after the k-th iteration.

Firstly, we need to show that the node selection rule in Step 3 of Algorithm 1 is equivalent to the node selection rule in the A* algorithm. Consider the moment Algorithm 1 is about to enter its Step 3. At this moment, (arbitrarily) consider a node $v \in O$. Note that v must have a predecessor, say $u \in E$. The selection key of v is $W(u, v) - \pi(v) + \pi(u)$. Note that

$$\begin{aligned} & W(u, v) - \pi(v) + \pi(u) \\ = & W(u, v) - (-h(v) + \Phi) + \pi(u) \\ = & W(u, v) + \pi(u) - \pi(s) + h(v) - \Phi + \pi(s). \end{aligned}$$

By Proposition 2, we have $\pi(u) - \pi(s) = dist(s, u)$. Hence

$$\begin{aligned} & W(u, v) - \pi(v) + \pi(u) \\ = & W(u, v) + dist(s, u) + h(v) - \Phi + \pi(s). \end{aligned}$$

We can see that $W(u, v) + dist(s, u) = W(u, v) + d(u) = d(v)$ and $d(v) + h(v) = f(v)$. Also Note that both Φ and $\pi(s)$ remain the same when different nodes in O are considered. Hence by inductive hypothesis, under the same tie-breaking rule, Algorithm 1 selects the same node from O as the A* algorithm.

Secondly, we need to show that, under the same tie-breaking rule, after a node, say u, is removed from O and put into E in Step 3 of Algorithm 1, the predecessor update on any node $v \in O$ is as same as that in the A* algorithm. In fact, if there is no arc $(u, v) \in A$, there won't be any predecessor update on v. If there is an arc $(u, v) \in A$, then in Algorithm 1, the update is based on comparing $W(pred(v), v) + \pi(pred(v))$ with $W(u, v) +$

$\pi(u)$. By Proposition 2 again,

$$\begin{aligned}&W(pred(v), v) + \pi(pred(v))\\ &= W(pred(v), v) + \pi(pred(v)) - \pi(s) + \pi(s)\\ &= W(pred(v), v) + dist(s, pred(v)) + \pi(s)\end{aligned}$$

and

$$\begin{aligned}&W(u, v) + \pi(u)\\ &= W(u, v) + \pi(u) - \pi(s) + \pi(s)\\ &= W(u, v) + dist(s, u) + \pi(s).\end{aligned}$$

Note that $\pi(s)$ is common, hence the comparison is actually between $W(pred(v), v) + dist(s, pred(v)) = d(v)$ and $W(u, v) + dist(s, u) = W(u, v) + d(u)$. Under the same tie-breaking rule, this is just the predecessor update rule in the A* algorithm.

Finally, note that if there is a node $v \in V \setminus (E \cup O)$ such that $(u, v) \in A$, then Algorithm 1 will put it into O and assign it a predecessor u. Hence $W(pred(v), v) + \pi(pred(v)) = W(u, v) + \pi(u) = W(u, v) + dist(s, u) + \pi(s)$, which implies that v receives a distance label $d(v) = W(u, v) + dist(s, u)$. This is just what the A* algorithm does.

Combine the three arguments above, we have shown that the two algorithms maintain the same Open list and Closed list, and for each node in the Open list, they assign the same predecessor in the Closed list. Q.E.D.

The original version of the primal-dual in Papadimitriou & Steiglitz (1998) selects an arc in E-O cut and adds the head of this arc into E. Other than manipulating the arc selection, Algorithm 1 manipulates the node selection in each iteration. Although the A* algorithm is algorithmically different from Algorithm 1, they behave the same. Those equivalences imply that the time-complexity of the A* algorithm should also be valid for the primal-dual algorithm. And indeed, the A* algorithm is a good implementation of the primal-dual algorithm that starts from $\pi^{(0)} = -h$.

DUALITY

There is a nice property of the A* algorithm that uses consistent heuristic. For the A* algorithm we listed at the beginning, suppose a node u_1 is closed no later than another node u_2, then according to Hart et al. (1968) and Pearl (1984), $f(u_1) \le f(u_2)$. This property is called *monotonicity*. By monotonicity, before t is closed, for any closed node u, $f(u) \le dist(s, t)$. By combining the monotonicity property of the A* algorithm and Algorithm 1, we have the following interesting result.

Proposition 4. After each iteration of Algorithm 1, $\pi(t) - \pi(s) = \max_{u \in E} f(u) \le dist(s, t)$.

Proof. The inequality directly follows from the monotonicity property. We now show the equality. Consider any iteration of Algorithm 1. Suppose node u is selected in Step 3 during this iteration. Upon the completion of this iteration, by the proof of Proposition 3, we see that $\Phi = f(u) + \pi(s)$; also note that $\pi(t) = -h(t) + \Phi = \Phi$, hence $\pi(t) - \pi(s) = f(u)$. By monotonicity property, upon the completion of this iteration, $f(u) = \max_{u' \in E} f(u')$. Q.E.D.

If we define $d(t) = +\infty$ when $t \notin E \cup O$, we then have that $\pi(t) - \pi(s) \le d(t)$ always holds. This is because $d(t) \ge dist(s, t)$ always hold. The inequality $\pi(t) - \pi(s) \le d(t)$ can be viewed as the *weak duality*. The duality gap is $d(t) - (\pi(t) - \pi(s)) = d(t) - \max_{u \in E} f(u)$. As the primal objective, $d(t)$ is monotonically nonincreasing; as the dual objective, $\pi(t) - \pi(s) = \max_{u \in E} f(u)$ is monotonically increasing. By completeness, if there exists an s-t path in G, then t will eventually be closed. Upon the moment t is closed, we have $\max_{u \in E} f(u) = f(t) = d(t)$. Hence both $\pi(t) - \pi(s)$ and $d(t)$ reach optimal. This means the duality gap is eliminated. The final equality is just the so-called *strong duality*.

Just like the A* algorithm, Algorithm 1 may terminate at its Step 2. Simple analysis (Pearl, 1984) of the A* algorithm tells that termination at Step 2 implies there is no s-t path in G. An explanation within the primal-dual framework is that when Algorithm 1 terminates at its Step 2, the potentials of all nodes outside E can be arbitrarily raised the same value without violating Constraint (2.7). But the objective (2.4) will be unbounded. This just implies the infeasibility of (P). Just like the A* algorithm, Algorithm 1 may not terminate at all. This happens when there is no s-t path in G, but there are infinitely many nodes that are connected with s via paths. Under this circumstance, the objective (2.4) is also unbounded. We conclude this section with a numerical example that demonstrates how the A* iterations improve the dual objective (2.4).

As shown in Figure 2, a 257×257 2-D terrain is fractally generated using the diamond square algorithm ((Miller, 1986) as the test graph. The graph is a grid with 8-adjacency with blue-color coded nodes as reachable and red-color coded nodes as forbidden. The starting node is $s = (0, 0)$ and the target node is $t = (256, 256)$. The subfigures (b) and (c) display the search results associated with $\pi^{(0)} = 0$ (the Dijkstra's algorithm) and $\pi^{(0)} = -d^{Eu,t}$ (the A* algorithm), respectively. Both algorithms were coded with Matlab 7.1 and tested in a PC with Intel dual core CPU T2050 at 1.6 GHz and 1.0 G RAM. The notation $d^{Eu,t}$ stands for the Euclidean distance with respect to the target node t, that is, for any reachable node v, $d^{Eu,t}(v)$ denotes the Euclidean distance between v and t. Obviously, $d^{Eu,t}$ is a consistent heuristic in the A* search. In both (b) and (c), the green-color coded nodes represent those that have been closed when t is closed. The black curves are the shortest s-t paths that are found by the two algorithms. The subfigure (d) displays how the dual objective is iteratively improved by the two algorithms. It can be seen that, in this numerical instance, the A* algorithm needs much less iterations than the Dijkstra's algorithm. Although the computational cost of evaluating the heuristic in the A* search is involved, substantially less iterations lead to the much better overall efficiency.

INITIAL FEASIBLE SOLUTIONS

Our analysis so far tells that the selection of $\pi^{(0)} = -h$ is sound. Actually, $-h$ defines a class of initial feasible solutions to (D). Suppose we have two consistent heuristics h_1 and h_2. Denote PD$_i$ as the primal-dual algorithm starting from $-h_i$, $i = 1, 2$. By completeness, both PD$_1$ and PD$_2$ will successfully terminate as long as there exists an s-t path in G. If $h_1(v) > h_2(v)$ for all $v \in V \setminus \{t\}$ and there exists an s-t path in G, then A dominance theorem (Hart et al., 1968; Nilsson, 1980; Pearl, 1984) on the A* algorithm says that $E_1 \subseteq E_2$, where E_i denotes the final Closed list of PD$_i$, $i = 1, 2$.

There are two interesting extreme cases. One is that $\pi^{(0)} = 0$. We have seen, in this case, that Algorithm 1 reduces to the Dijkstra's algorithm. This just indicates the derivation of the Dijkstra's algorithm from the primal-dual algorithm. The other is that $\pi^{(0)}(v) = -\, dist(v, t)$ for all $v \in V$. In this case, Algorithm 1 only closes the nodes that lie on a shortest s-t path in G, hence the initial feasible solution to (D) is perfect.

Sometimes, there exits some metric function H: $V \times V \to R$ such that $H(u, v) \geq 0$ for all $u, v \in V$, $H(v, v) = 0$ for all $v \in V$, and $W(u, v) + H(v, w) \geq H(u, w)$ for all $(u, v) \in A$. We immediately have a consistent heuristic, say h^H, defined as $h^H(v) = H(v, t)$ for all $v \in V$. Under some condition, we can also find another consistent heuristic. Suppose we already have a partial solution that is represented by a shortest path tree T of $\tilde{G}$

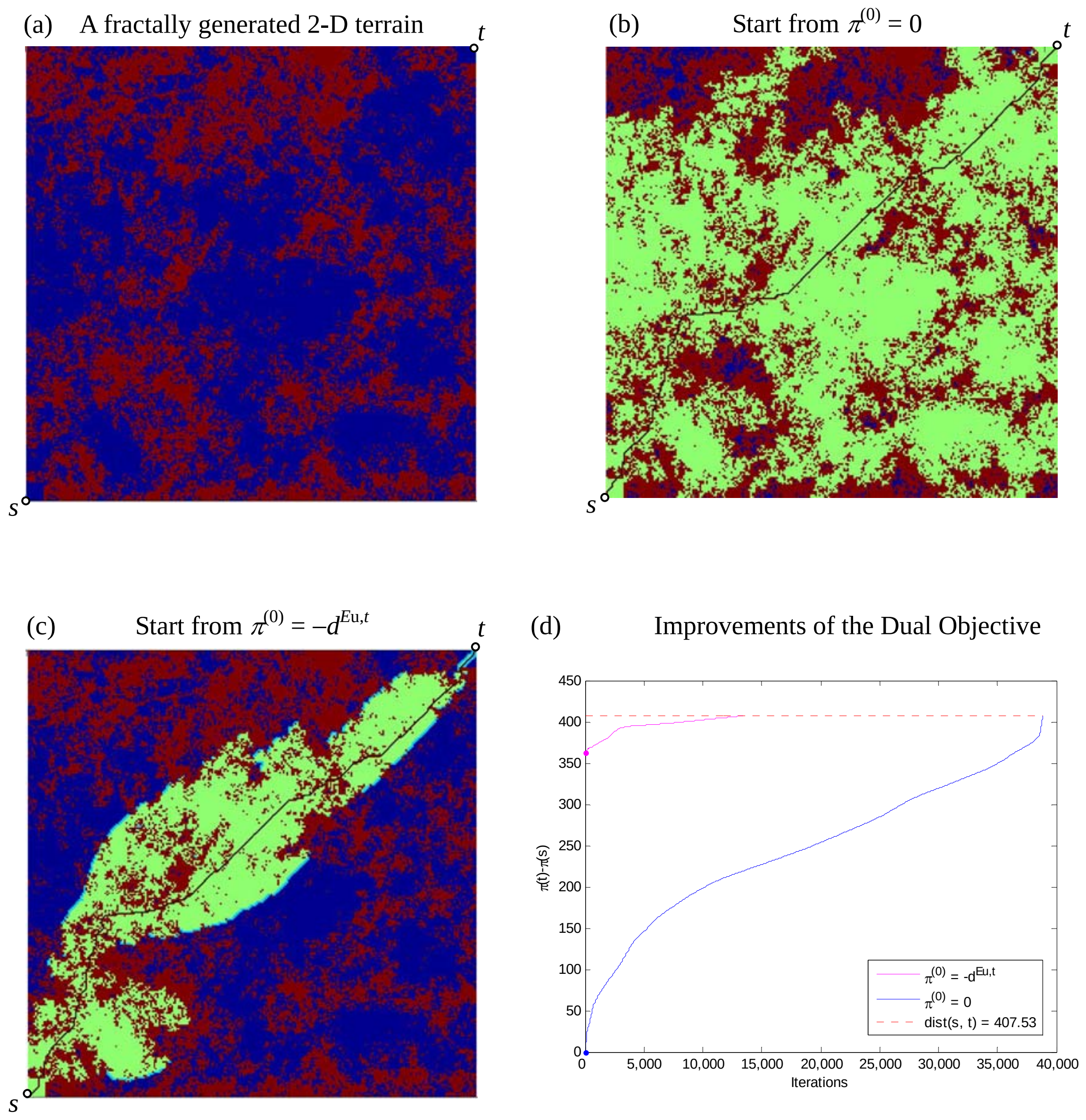

Figure 2: A numerical demonstration that how the A* iterations improve the dual objective. (a): A 257×257 fractally generated 2-D terrain is used as the test graph. The graph has 8-adjacency with blue-color coded nodes as reachable and red-color coded nodes as forbidden. The starting node is s = (0, 0) and the target node is t = (256, 256). (b): Start from $\pi^{(0)} = 0$, that is, the Dijkstra's algorithm. The CPU time is 47.31 sec.. (c): Start from $\pi^{(0)} = -d^{Eu,t}$, that is, the A* algorithm. The CPU time is 27.42 sec.. (d): Improvements of the dual objective along the iterations of the two algorithms.

rooted at t. Suppose this shortest path tree is found by the Dijkstra's algorithm that searches from t to s in $\tilde{G}$. Let E^T and O^T denote the Closed list and Open list associated with T. We define

$$h^{H,T}(v) = \begin{cases} \min\limits_{\tau \in O^T} [H(v, \tau) + dist(\tau, t)] & \text{for all } v \in V \setminus E^T; \\ dist(v, t) & \text{for all } v \in E^T. \end{cases} \quad (5.1)$$

It can be easily shown that $h^{H,T}$ is a consistent heuristic, and for any $v \in V$, as the two estimates of $dist(v, t)$, $h^{H,T}(v)$ is at least as good as $h^H(v)$, that is, $h^H(v) \leq h^{H,T}(v) \leq dist(v, t)$. The primal-dual algorithm can start from $\pi^{(0)} = -h^H$ if H is available. The primal-dual algorithm can also start from $\pi^{(0)} = -h^{H,T}$ if both H and T are available. In the case that both H and T are available, to start from $-h^{H,T}$ may result in less closed nodes upon closing t than to start from $-h^H$, but it doesn't mean that the former has better efficiency since to evaluate $h^{H,T}$ by (5.1) is more costly than to evaluate h^H. However, an issue that should be addressed is that the primal-dual algorithm that starts from $-h^{H,T}$ may be able to successfully find a solution earlier than the moment t is closed. Hence, a different termination condition might apply. This is actually related to the bidirectional search that we will discuss later.

Sometimes there also exists another type of heuristic h' such that $h'(v) \geq 0$ for all $v \in V$, $h'(s) = 0$, and $W(u, v) + h'(u) \geq h'(v)$ for all $(u, v) \in A$. h' is called consistent relative to s. It's obvious that $\pi^{(0)} = h'$ is a feasible solution to (D). The corresponding objective value of (D) is $\pi^{(0)}(t) - \pi^{(0)}(s) = h'(t) - h'(s) = h'(t) \geq 0$. It seems that $\pi^{(0)} = h'$ is also a better choice of initial feasible solution to (D) than $\pi^{(0)} = 0$, however, the following simple example shows that the primal-dual algorithm that starts from h' may not terminate at all even if there exists an s-t path in G.

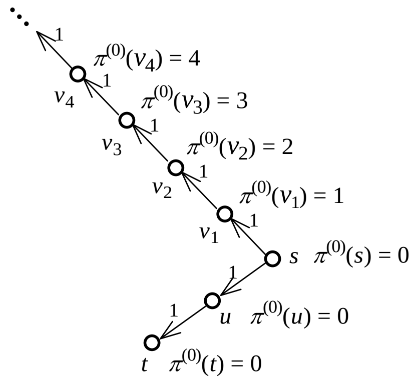

Figure 3: A example that the primal-dual algorithm starting from some $\pi^{(0)} = h'$ does not terminate, where the length of any arc is 1 and the heuristic function h' is $\{h'(s) = 0, h'(u) = 0, h'(t) = 0$, and $h'(v_i) = i$ for $i = 1, 2, \cdots\}$.

Figure 3 shows a simple infinite graph. We want to find a shortest s-t path. When applying the primal-dual algorithm with $\pi^{(0)} = \{h'(s) = 0, h'(u) = 0, h'(t) = 0$, and $h'(v_i) = i$ for $i = 1, 2, \cdots\}$ as the initial feasible solution to (D), the algorithm (Algorithm 1 with initial node potential function set as this $\pi^{(0)}$) will close s at first, then v_1, then v_2, $\cdots$. Note that u will never be closed, let alone t. Hence the algorithm won't be able to successfully terminate. Even if we restrict this graph to be finite and the algorithm is able to terminate, the efficiency is bad because it traverses the wrong way before heading toward the right way.

BIDIRECTIONAL SEARCH

Although h' is not a proper choice of the initial feasible solution to (D) for Algorithm 1 to start from, it can be used for bidirectional search. Consider the primal LP model with respect to G, in which for each $(u, v) \in A$, we still use $x(u, v)$ to denote the decision variable:

(P′)

$$\text{Min} \sum_{(u,v)\in A} W(u,v)\cdot x(u,v) \tag{6.1}$$

Subject to

$$\sum_{v:(u,v)\in A} x(u,v) - \sum_{v:(v,u)\in A} x(v,u) = \begin{cases} 1 & \text{if } u = s; \\ -1 & \text{if } u = t; \\ 0 & \text{for all } u \in V \setminus \{s, t\} \end{cases} \tag{6.2}$$

$$x(u, v) \geq 0 \text{ for all } (u, v) \in A. \tag{6.3}$$

This forward version of primal LP model stands for sending a unit flow from a supplier s to a customer t in G with least cost. It has the following dual:

(D′)

$$\text{Max } \pi(s) - \pi(t) \tag{6.4}$$

Subject to

$$\pi(u) - \pi(v) \leq W(u, v) \text{ for all } (u, v) \in A. \tag{6.5}$$

Similar analysis can show that the primal-dual algorithm that uses $-h'$ as the initial feasible solution to (D′) is essentially the A* algorithm that searches from t to s in $\tilde{G}$ using the heuristic h'. This version of the primal-dual algorithm is exactly the backward version of Algorithm 1. If both algorithms are used, searching toward each other, then a bidirectional A* search can be established. For the backward version of Algorithm 1, let O', π' and $pred'$ denote the corresponding Open list, potential function, and predecessor function, respectively. When the two search fronts (Open lists) meet, an s-t path is found, and its length, denoted as L, can be expressed as

$$L = [W(pred(v), v) + \pi(pred(v)) - \pi(s)] + [W(v, pred'(v)) + \pi'(pred'(v)) - \pi'(t)],$$

where $v \in O \cup O'$ is the meeting node. When the search continues, a sequence of lengths, say $L_1, L_2, \ldots$, is generated.

Let $\widehat{L} = \min\{L_1, L_2, \ldots\}$, which is the length of the shortest s-t path in G found so far. Since $\pi(t) - \pi(s) \leq dist(s, t) \leq \widehat{L}$ and $\pi'(s) - \pi'(t) \leq dist(s, t) \leq \widehat{L}$, we have a termination condition for the bidirectional A* search, expressed via π and π', as

$$\max\{\pi(t) - \pi(s), \pi'(s) - \pi'(t)\} = \widehat{L}. \tag{6.6}$$

This condition is essentially as same as the one that appears in Pohl (1971). It can eventually be satisfied.

Again, the bidirectional A* search reduces to the bidirectional Dijkstra's search when $h = h' = 0$. An alternative termination condition, according to Korf & Zhang (2005) and Proposition 2, that is expressed via π and π', is

$$\min_{v\in O} [W(pred(v), v) + \pi(pred(v)) - \pi(s)] + \min_{v'\in O'} [W(v', pred'(v')) + \pi'(pred'(v')) - \pi'(t)] = \widehat{L}, \tag{6.7}$$

which can also eventually be satisfied.

CONCLUSION

We have shown that for the shortest path problem in a positively weighted graph equipped with a consistent heuristic function, choosing the initial feasible solution to the dual in the primal-dual algorithm is equivalent to choosing the consistent heuristic in the A* algorithm. Although the equivalence between the primal-dual algorithm and the A* algorithm is a known result, there are still questions to answer. One is that how the A* iterations relate to the improvements of the dual objective. Our new derivation of the equivalence is not only a direct and simpler way but also answers this question as stated in Proposition 4. Compared to the Dijkstra's algorithm in improving the dual objective, the A* search enjoys an initial "leap" of $h(s)$ and the continual role played by the heuristic h during the iterations. Moreover, the completeness of the A* search guarantees that the duality gap can be eventually eliminated if there exists a solution.

Through both the theoretical and numerical investigations, we not only gain a further understanding of the A* algorithm from the primal-dual perspective but also like to pose new questions. For example, it's still not quite clear on how the primal-dual and the bidirectional search relate to each other. A central issue of the later is to find a good termination condition. Another issue is how to alternatively switch from one direction to the other. Although those two issues have been extensively studied, the idea of looking at the questions from the primal-dual perspective seems to be novel. We suggest a further investigation as a possible extension of the work in this paper.

ACKNOWLEDGEMENTS

The authors would like to thank the editor-in-chief of the journal, Professor John Wang, and the anonymous reviewers for their valuable comments and suggestions that improved the paper significantly.

REFERENCES

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Theory, algorithms, and applications*. Englewood Cliffs, NJ: Prentice Hall.

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization*. Belmont, Massachusetts: Athena Scientific.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms* (2nd ed.). Cambridge MA: the MIT Press.

Dantzig, G. B., Ford, L. R., & Fulkerson, D. R. (1956). A primal-dual algorithm for linear programs. In Kuhn, H. W. & Tucker, A. W. (Eds.), *Linear inequalities and related systems* (pp. 171 - 181). Princeton, NJ: Princeton University Press.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerical Mathematics, 1*, 269 - 271.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1972). Correction to "A formal basis for the heuristic determination of minimum cost paths". *SIGART Newsletter, 37*, 28 - 29.

Hart, P. E., Nilsson, N. J., & Rapheal, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of System Sciences and Cybernetics, SSC-4, 2*(July), 100 - 107.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica, 4*, 373 - 395.

Korf, R. E., & Zhang, W. (2005). Frontier search. *Journal of the Association for Computing Machinery, 52*(5), 715 - 748.

Miller, G. (1986). The definition and rendering of terrain maps. *Computer Graphics, 20*(4), 39 - 48.

Nilsson, N. J. (1980). *Principles of artificial intelligence*. San Mateo, California: Morgan Kaufmann.

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Dover Publications.

Pearl, J. (1984). *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley.

Pohl, I. (1971). Bidirectional search. In Meltzer, B., & Michie, D. (Eds.), *Machine intelligence 6* (pp. 127 - 140). New York: American Elsevier.

Wright, S. J. (1997). *Primal-dual interior-point methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics.