CA-002

固定パスガソリンスタンド問題に対する両端キューを用いた線形時間アルゴリズム

A linear time algorithm for the fixed-path gas station problem using a double-ended queue

高橋 俊彦 †
Toshihiko Takahashi

概要

容量 U のガソリンタンクを持つ車が n か所のガソリンスタンド $v_1, v_2, \ldots, v_n$ を順に訪れる．固定パスガソリンスタンド問題とは, 隣り合うガソリンスタンド v_{i-1}, v_i 間の距離 $d(i-1, i)(2 \leq i \leq n)$, v_i でのガソリンの単価 $c(v_i)(1 \leq i \leq n)$ が与えられたとき, v_1 から v_n まで到達するための最小費用のガソリン給油スケジュールを求める問題である．

Khuller, Malekian, and Mestre は固定パスガソリンスタンド問題に対し, $O(n \lg n)$ 時間のアルゴリズムを与えた．本論文は Khuller らの手法を元に, 両端キューを用いた線形時間のアルゴリズムを与える．

固定パスガソリンスタンド問題に対しては Lin, Gertsch, and Russell も線形時間アルゴリズムを与えているが, 提案手法は上記の 2 つのアルゴリズムと異なり, ストリーミングアルゴリズムでもあるという特長を持っている．

1 はじめに

ガソリンスタンド問題 (gas station problem) は以下のように定義される最適経路問題の 1 つである [1, 2]. 点と辺にそれぞれ重みの付いた完全グラフ $G = (V, E)$, $c : V \to \mathbb{R}^+$, $d : E \to \mathbb{R}^+$, G 上の 2 頂点 $s, t \in V$, および正の実数 $U \in \mathbb{R}^+$ が入力として与えられる．点 $v \in V$ はガソリンスタンド (gas station), $c(v)$ は v で給油するガソリン 1 単位の価格をそれぞれ表す．辺 $e = (u, v) \in E$ に対し, $d(e) = d(u, v)$ は uv 間の距離, あるいは uv 間の移動で消費するガソリンの量とする[*1]. したがって, 点 u から v への移動はガソリンを $d(u, v)$ 単位消費し, もし点 u においてタンクのガソリンが $d(u, v)$ 未満であれば u から v へ移動することはできない．

問題はタンク容量 U の自動車が点 s から点 t まで最小費用で到達するための給油スケジュール, すなわち st パスとその上の各点での給油量, を与えることである．なお, 出発点 s においてタンクに $\mu(0 \leq \mu \leq U)$ 単位のガソリンがあるという設定をする場合もあるが, $\mu = 0$ と仮定しても問題は変わらない [2]．そのため本論文では後述の Algorithm 2 を除いて $\mu = 0$ を仮定する．

ガソリンスタンド問題にはいくつかのバリエーションが存在するが, 本論文では特定の st パスが指定されたときに, そのパス上での最小費用の給油スケジュールを求める**固定パスガソリンスタンド問題** (fixed-path gas station problem) を扱う．すなわち, 点と辺に重みのついた n 点のパス $P = \langle v_1, v_2, \ldots, v_n \rangle$, $c : V \to \mathbb{R}^+$, $d : E \to \mathbb{R}^+$, および正の実数 $U \in \mathbb{R}^+$ を入力とし, $s = v_1$ から $t = v_n$ までの最小費用の給油スケジュールを与える問題である．(その他のバリエーションについては [1] を参照.)

Khuller, Malekian, and Mestre は固定パスガソリンスタンド問題に対し, $O(n \lg n)$ 時間のアルゴリズムを与えた [1]．本論文では Khuller らの手法を元に, 両端キュー (double-ended queue) を用いた One-pass の線形時間アルゴリズムを与える．

† 新潟大学工学部 Faculty of Engineering of Niigata University

[*1] 走行距離/ガソリンの消費量=燃費は自動車固有の定数であるため d の単位を距離としてもガソリンの量としても定数倍しか変わらない．

一方, Lin, Gertsch, and Russell は生産計画問題の 1 種である inventory-capacitated lot-sizing problem (ICLSP)[4] の特別な場合として, 固定パスガソリンスタンド問題に対する線形時間アルゴリズムを与えている[*2].

ただし, 本論文で与える線形時間アルゴリズムは固定パスガソリンスタンド問題に対するストリーミングアルゴリズムでもあるという点が Khuller らや Lin らのアルゴリズムと大きく異なる. すなわち, ガソリンスタンド v_i に対するガソリン単価 $c(v_i)$ および前の点 v_{i-1} からの距離 $d(v_{i-1}, v_i)$ が逐次的に与えられる状況においても, 提案アルゴリズムは最適な給油スケジュールを与えることができる. すなわち, 提案アルゴリズムはガソリンスタンドあたり $O(1)$ 時間 (ならしコスト) のストリーミングアルゴリズムとなっている.

本論文の構成は以下の通り. 第 2 章では Khuller, Malekian, and Mestre の $O(n \lg n)$ 時間アルゴリズム, 第 3 章では Lin, Gertsch, and Russell の $O(n)$ 時間アルゴリズムを概説する. 第 4 章では両端キューを用いた $O(n)$ 時間アルゴリズムとその正当性を示す. 第 5 章では提案アルゴリズムの実行時間の線形性とストリーミングアルゴリズム化について述べる.

2 Khuller, Malekian, and Mestre の $O(n \lg n)$ 時間アルゴリズム

本章では [1] に基づき, Khuller, Malekian, and Mestre の $O(n \lg n)$ 時間アルゴリズムを概説する.

前述のように出発点 $s = v_1$ においてタンクが空 ($\mu = 0$) であると仮定する. また, [1] に倣い, 以下では各点 (ガソリンスタンド)v_i を単に i と表記する. また, 2 点 i, j の距離 (ij パスの長さ) を d_{ij} と表記する. なお, $d(i-1, i) > U (2 \leq i \leq n)$ となること, すなわち満タンにもかかわらず次のガソリンスタンドに到達できなくなることはないものとする. (入力として与えられる隣接する 2 点 $i-1, i$ の距離は $d(i-1, i)$, 単に 2 点 i, j の距離は d_{ij} と記法を使い分けている.)

ここで, 各点 (ガソリンスタンド)i に対し, 以下の値を定義する:

- $prev(i)$: $j \leq i$ かつ $d_{ji} \leq U$ を満たす最もガソリン単価の低い点 j.
- $next(i)$: $j > i$ かつ $d_{ij} \leq U$ を満たす最もガソリン単価の低い点 j.

ただし, 条件を満たす点が 2 つ以上ある場合は $t = n$ に近い方を選ぶものとする.

$prev(i) = i$ である点を *break point* と呼ぶ. break point について次の補題が成り立つ.

補題 1 (Khuller, Malekian, and Mestre[1])**.** すべての *break point* において到着時にタンクが空になるような最適給油スケジュールが存在する.

補題により, 固定パスガソリンスタンド問題は break point から break point までのパスに対する部分問題に分割される.

Khuller らのアルゴリズムはまず各点に対する $prev$ と $next$ を計算し, break point を特定する. 次に, 各 break point i から次の break point k に対して (すなわち, 各部分問題に対して), 以下の手続き DRIVE-TO-NEXT を適用する.

Algorithm 1 DRIVE-TO-NEXT(i, k)

1: $x \leftarrow i$
2: **if** $d_{xk} \leq U$ **then**
3: k に到達するために必要最小限の給油をし, k まで進む.(k に着いたときタンクは空になる.)
4: **else**
5: 満タンにして $next(x)$ まで進む. $x \leftarrow next(x)$ として Step2 へ.
6: **end if**

DRIVE-TO-NEXT 自体の実行時間は $O(n)$ であるが, アルゴリズム全体の実行時間は $prev$ と $next$ の計算時間に支配され, $O(n \lg n)$ となる [1].

$prev$ の計算は $s = 1$ から $t = n$ に向かって各点 i を走査することで行う. 点 i から s 側に距離 U 以内にある点を (ガソリン単価をキーとして) 優先度付きキュー Q に格納するようにすれば, i を挿入した直後の Q の最小キーが $prev(i)$ を与える. (もちろん i から距離 U より離れた点をそのつど Q から削除する必要がある.) Q への要素の挿入および削除は $O(\lg n)$ 時間で実行できるから, $prev$ 全体の計算時間は $O(n \lg n)$ となる. $next$ の計算も同様である.

[*2] Lin らの論文の方が先に発表されているが, 固定パスガソリンスタンド問題を fixed-path gas station problem ではなく fixed-route vehicle refueling problem と呼んでいることから, Khuller らが線形時間アルゴリズムの存在に気づいていなかったものと思われる.

3 Lin, Gertsch, and Russell の $O(n)$ 時間アルゴリズム

本章では [3] に基づき, Lin, Gertsch, and Russel の $O(n)$ 時間アルゴリズム Finding an optimal refueling policy を概説する．なお, [3] らアルゴリズムは $s = v_1$ においてタンクに $\mu(0 \leq \mu \leq U)$ のガソリンがあるという設定になっており, Algorithm 2 の記述はこれに従った．ただし, 分かりやすさのため, 同じものを表す記号や用語は適宜 [1] に合せて変更することとした．

Lin らのアルゴリズム (Algorithm 2) も Khuller のアルゴリズムと同様, 2 種類の値をあらかじめ計算する:

- $FAR(i)$: $d_{ij} \leq U$ である最大の点 $j \geq i$, すなわち, i で満タンにしたとき途中で給油せずに到達できる最も t 寄りの点.
- $LOW(i)$: i より後にありかつ $c(i)$ よりガソリン単価が低い点のうち最も i 寄りの点.

全点に対して FAR と LOW を計算するアルゴリズムおよびそれらの実行時間が $O(n)$ であることの証明は [3] で与えられている．

Algorithm 2 はこれらの値を用いて, s から t へ向かい, 給油を貪欲的に決定する車の動きを模倣する．なお, 変数 $fuel$ は現在の車のガソリンの量, 配列 $\mathbf{F} = \langle f_1, \ldots, f_n \rangle$ は点 $i(1 \leq i \leq n-1)$ における給油量を表す．

アルゴリズムの正当性および実行時間が $O(n)$ であることの証明は [3] を参照されたい．Algorithm 2 では $d(i-1, i) \leq U(2 \leq i \leq n)$ を仮定せず, 3 行目で $FAR(i) = i$ である i が存在すれば, i で満タンにしても $i+1$ に到達することはできず, 給油スケジュールは存在しないことの判定を行っている．

Algorithm 2 Finding an optimal refueling policy

1: $1 \leq i \leq n$ に対し, $FAR(i)$ および $LOW(i)$ を計算する．
   $\mathbf{F} = \langle f_1, \ldots, f_n \rangle \leftarrow \langle 0, \ldots, 0 \rangle$.
   $fuel \leftarrow \mu$, $i \leftarrow 1$.
2: **while** $i \leq n$ **do**
3: **if** $FAR(i) = i$ **then**
4: 給油スケジュールが存在しないとして停止.
5: **end if**
6: **if** $LOW(i) > FAR(i)$ **then**
7: $f_i \leftarrow U - fuel$
   $fuel \leftarrow U - d(i, i+1), i \leftarrow i+1$
8: **else**
9: **if** $d_{i,LOW(i)} > fuel$ **then**
10: $f_i \leftarrow d_{i,LOW(i)} - fuel$
11: **else**
12: $f_i \leftarrow 0$
13: **end if**
14: $fuel \leftarrow fuel + f_i - d_{i,LOW(i)}$
   $i \leftarrow LOW(i)$
15: **end if**
16: **end while**
17: **return F**

4 両端キューを用いた $O(n)$ 時間アルゴリズム

本章では両端キュー (double-ended queue) を用いた線形時間のアルゴリズムを提案する．基本的なアイディアは無駄な (給油を行わない) ガソリンスタンドを除きながら, 両端キュー D に s からの距離およびコストの昇順になるようにガソリンスタンドを保持することである．これにより, 自動的に D に含まれる末尾以外のガソリンスタンド x はその直後のガソリンスタンドは (Khuller らの定義した)$next(x)$ となる．

初めにアルゴリズム One-pass algorithm (Algorithm 3) と手続き NEW-DRIVE-TO-NEXT (Algorithm 4) を示す．なお, Khuller らのアルゴリズム同様, 一般性を失うことなく, 出発点 s においてタンクは空であること, および隣接する 2 点間の距離は U 以下である (t へ到達できる) ことを仮定する．また, $c(n) = 0$, すなわち, 目的地 $t = n$ でのガソリン単価は 0 とする．

アルゴリズムで用いる両端キューの操作は以下の通り:

- $back(D)$: 両端キュー D の末尾を参照する (末尾の要素は削除されない).
- $pop\_front(D)$: 両端キュー D の先頭を取り出す (先頭の要素が削除される).
- $pop\_back(D)$: 両端キュー D の末尾を取り出す (末尾の要素は削除される).
- $push\_back(D, x)$: 両端キュー D の末尾に点 x を追加 (挿入) する.

アルゴリズムで用いる変数および定数は登場順に以下の通り (前述のものも含む):

Algorithm 3 One-pass algorithm for gas station problem

1: $pos_1 \leftarrow 0, f_1 \leftarrow 0, D \leftarrow \langle 1 \rangle$
2: **for** $i = 2$ to n **do**
3: $f_i \leftarrow 0, pos_i \leftarrow pos_{i-1} + d(i-1, i)$
4: **while** $(D \neq \emptyset) \wedge (pos_i - pos_{back(D)} \leq U) \wedge (c(back(D)) \geq c(i))$ **do**
5: $x \leftarrow pop\_back(D)$
6: **end while**
7: **if** $(D \neq \emptyset) \wedge (pos_i - pos_{back(D)} \leq U) \wedge (c(back(D)) < c(i))$ **then**
8: $push\_back(D, i)$
9: **else**
10: $push\_back(D, x)$
11: $push\_back(D, i)$
12: NEW-DRIVE-TO-NEXT(D)
13: **end if**
14: **end for**

Algorithm 4 NEW-DRIVE-TO-NEXT

1: $x \leftarrow pop\_front(D), k \leftarrow back(D)$
2: **while** $pos_k - pos_x > U$ **do**
3: $f_x \leftarrow U - fuel$
4: $y \leftarrow pop\_front(D)$
5: $fuel \leftarrow U - (pos_y - pos_x)$
6: $x \leftarrow y$
7: **end while**
8: $f_x \leftarrow pos_k - pos_x - fuel$, $fuel \leftarrow 0$

- pos_i: 出発点 $s = v_1$ からの点 i の距離.
- f_i: 点 i でのガソリンの給油量.
- D: 両端キュー.
- $d(i-1, i)$: 辺 $e = (i-1, i)$ の重み. $i-1, i$ 間の距離 (ガソリン消費量) であり, 入力として与えられている.
- U: 自動車のタンク容量.
- $c(i)$: 点 i の重み. i でのガソリン単価であり, 入力として与えられている.
- $fuel$: 現在のガソリンの量.

まず Algorithm 3 の動作の詳細を説明する.

1 行目で初期化を行い, 2 行目以降で各点 (ガソリンスタンド) を $s = 1$ から $t = n$ へ向かい処理していく.

3 行目で点 i での給油量 f_i を 0 に初期化し, 出発点から点 i までの距離 pos_i を計算する.

4–6 行目の **while** ループは D が空でなく, 点 i が D の末尾から U 以内にあり, かつ i のガソリン単価が D の末尾のガソリン単価以下である間, 末尾を削除して x に保持する. **while** ループ終了時に x 以外に削除された点があれば, それらは無駄な点, すなわち, そこで給油を行う必要がない点であることに注意されたい.

次に示すように **while** ループ開始時に D の点はガソリン単価の昇順に並んでいる. x より先に削除された点 z があれば, $c(i) \leq c(x) \leq c(z)$ である. 一方, x は i から距離 U 以内に, z は x と i の間にそれぞれ位置する. したがって, z で $f_z > 0$ だけ給油し, i に到着した時点でタンクのガソリンの量が g となる給油スケジュールがあれば, x で i に到着するために必要な量 (0 以上) を給油し, z で給油をせず ($f_z = 0$), i に到着したときにタンクのガソリン量が g になるまで (0 以上) 給油するように変更することで元の費用以下の給油スケジュールが得られる. すなわち z は無駄な点であり, 取り除いて構わない.

7–8 行目では (**if** 文の条件が満たされるとき)D の末尾に i を追加する. このとき D の点はコストの昇順に並んでいる.

if 文の条件より追加された点 i はその直前の点から距離 U 以内にあり, そのガソリン単価 $c(i)$ は直前の点の単価より大きい. したがって, 元の D がコストの昇順であれば, 追加後の D もそうである. 一方, 初期化 (1 行目) および NEW-DRIVE-TO-NEXT(12 行目) 実行直後の D はいずれも 1 点のみからなるため (後者については後述) 自明にコストの昇順に並んでいる. これより, 8 行目終了時の D はコストの昇順に並んでいることがわかる.

9–12 行目が実行されるのは $(D = \emptyset) \vee (pos_i - pos_{back(D)} > 0)$ のときであり, x, i をこの順に D の末尾に加え, NEW-DRIVE-TO-NEXT を実行する.

アルゴリズムの正当性は両端キュー D の各点に対する給油スケジュールを与える NEW-DRIVE-TO-NEXT(Algorithm 4) が Khuller らの DRIVE-TO-NEXT と同様の動作をすることを示すことによる.

初めに, D の末尾 k が break point であることを確認する．(D の先頭も break point であることは後で確認する.)

> D において末尾 k から距離 U 以内にある点は k の直前の点 z のみであり, $c(z) \geq c(k)$ である．また, D に属さない点で k から ($s = 1$ 側に) 距離 U 以内の点があれば, それらはガソリン単価が $c(z)$ および $c(k)$ 以上であったため, Algorithm 3 の 4–6 行目で無駄であるとして削除された点である．したがって, k は自身を含めて $s = 1$ 側に距離 U 以内の点の中でも最もガソリン単価の低い点であり, break point である．

以下に NEW-DRIVE-TO-NEXT の動作の詳細を説明する．

1 行目で与えられる x と k はそれぞれ DRIVE-TO-NEXT (Algorithm 1) の i と k に対応する．すなわち, NEW-DRIVE-TO-NEXT は D 内の点に対する給油スケジュールを与える．

2–7 行目の **while** ループは DRIVE-TO-NEXT の 4–5 行目と同じである: 点 x でタンクを満タンにし (3 行目), D における x の直後の点 y へガソリン $U - (pos_y - pos_x)$ を消費して (5 行目) 移動する．ここで, D の末尾の 2 点以外の点 x に対して, その移動先は $y = next(x)$ となっている．

> D の末尾 k とその直前の点—z としよう—はそれぞれ Algorithm 3 の 11 行目, 10 行目で加えられた点である．したがって, 末尾 k から距離 U 以内にある点は z のみである．(もし z の直前の点も k から距離 U 以内にあれば, その点の方が z よりガソリン単価が低く, Algorithm 3 の 4—6 行目で z が削除されているはずである.) D は末尾 k を除いてガソリン単価の昇順になっているので, k,z 以外の点に対する $next$ は D における直後の点となる．

8 行目は $pos_k - pos_x \leq U$ となったとき (前述のように D において k から距離 U 以内にある点は k の直前の点のみであるから, この条件は D が k1 点のみになることと等価である), x で k に到達するするために必要最小限の給油をして, k まで進み, タンクを空にしている．これは DRIVE-TO-NEXT の 2–3 行目と同じ処理である．

NEW-DRIVE-TO-NEXT は $D = \langle k \rangle$ の状態で終了する．この k は break point であり, 次に NEW-DRIVE-TO-NEXT が呼び出されるときにも D の先頭に残っている．すなわち, NEW-DRIVE-TO-NEXT を呼び出したときには, D の先頭と末尾は break point である．

> 両端キュー D の先頭を x とする．もし, Algorithm 3 の 5 行目で x が削除されれば, $D = \emptyset$ となるため **while** ループを抜け, (7 行目の **if** 文の条件が成り立たないので)9 行目以下を実行する．このとき, x は 10 行目で空の D に追加され, 再び D の先頭となり, NEW-DRIVE-TO-NEXT が呼び出される．

以上から, NEW-DRIVE-TO-NEXT が DRIVE-TO-NEXT と同様の処理を行い, 最適な給油スケジュールを与えることがわかる[*3].

5 時間計算量とストリーミングアルゴリズム

初めに提案アルゴリズムの時間計算量を与える．

定理 1. *Algorithm 3* の実行時間は $O(n)$ である．

証明. ならし解析の手法である出納法を用いる．両端キュー D の末尾に (8, 10, 11 行目で) 点を追加する際, その点に 1 ドルの預金をする, ただし, ここでの 1 ドルは定数時間内の処理を行うために十分なコストであるものとする．

Algorithm 3 の **while** ループの 5 行目を実行するためのコストはそれぞれ *pop_back* で取り出される点の預金 1 ドルを用いて支払うことができる．同様に Algorithm 4 の **while** ループの 3–6 行目を実行するためのコストはそれぞれ 4 行目の pop_front で取り出される点の預金 1 ドルを用いて支払うことができる．

Algorithm 3 の **for** 文の繰り返し 1 回当たり, 高々 2 ドルを預金する (10, 11 行目)．したがって, **for** ループの繰り返し 1 回あたりのならしコストは定数で抑えられ, アルゴリズム全体の実行時間は $O(n)$ となる． □

Algorithm 3 はまたストリーミングアルゴリズムともなっている．すなわち, Khuller, Malekian, and Mestre の手法 [1] では *prev* および *next* を, Lin, Gertsch, and Russell のアルゴリズム [3] では FAR および LOW を, それぞれ n 個のガソリンスタンドに対してあらかじめ計算しておかねばならないのに対し, 提案手法の Algorithm 3 は逐次的に点 i の

[*3] もちろん, この証明は Khuller らの手法が正しいことを前提としている．

- ガソリン単価 $c(i)$, および
- 前の点 (ガソリンスタンド) からの距離 $d(i-1,i)$

が与えられるたびに, それらを両端キュー D に格納し, i が break point である場合, 直ちに i までの給油スケジュールを出力するストリーミングアルゴリズムとみなすことができる.

実際, Algorithm 3 をストリームデータ (点 i のガソリン単価と点 $i-1$ からの距離) が与えられるたびに, 3–13 行目を実行し, NEW-DRIVE-TO-NEXT において 3 行目, 8 行目で給油量 f_x を求めたとき, その値を出力するように変更することで, ストリーミングアルゴリズムが得られる.

ただし, このままでは, 例えばガソリン単価の昇順に点が入力される場合に, break point が現れず, 最後まですべての点が両端キュー D に蓄えられてしまうことになる. (もちろん, D に蓄えられてる点だけでなく, それらの点に対する c および d も記憶しておかなければならない.) 保持すべきデータのサイズが大きくなることはストリーミングアルゴリズムにとって望ましくないことであるが, この問題は以下のように解決できる.

Algorithm 3 で 8 行目で i を D の末尾に加えた後, D の先頭 z が i から距離 U よりも離れている点である間, z を NEW-DRIVE-TO-NEXT の **while** ループ内と同様の処理により D から取り出し, f_z の値を出力する.

i から U より離れた D 内の点 z は以後 Algorithm 3 の 5 行目で D から削除されることはなく, 直後の点が $next(z)$ であることが確定する. したがって, この時点で f_z を出力, すなわち NEW-DRIVE-TO-NEXT の処理を先取して適用することができる. これにより, ストリーミングアルゴリズムは D が i から距離 U 以内の点のみを格納すればよいように改良される.

定理 2. ストリーム版の固定パスガソリンスタンド問題に対し, ならし時間が定数のアルゴリズムが存在する. さらに, このアルゴリズムは距離がタンク容量 U 以内の点のガソリン単価と位置情報のみを記憶していればよい.

注意. Lin らの Algorithm 2 が s から t へ向かい, 給油を貪欲的に決定する車の動きを模倣していることから, 提案アルゴリズムのようにストリーミングアルゴリズムに修正できるように思えるかもしれないが, 以下のとおり非常に困難であることがわかる:

1. $FAR(i)$ および $LOW(i)$ はあらかじめ計算しておかねばならない. しかも, 両者とも i より先の点であるから, 元のアルゴリズムのように, 点 i までに与えられたデータから, 点 i での給油量 f_i を与えることはできない.
2. f_i の出力を, i が入力された時点でなく, 先延ばしすることで, 1 の問題を解決できるかもしれない (実際, Algorithm 3 はそのようにしている). 例えば, $FAR(i)$ は i が入力された後, i から距離 U 以上先 (t 寄り) の点が入力されれたときに確定する. しかし, オフラインアルゴリズムでは線形時間で計算可能であった FAR と LOW が, ストリーミングアルゴリズムにおいてどのくらいの時間で計算できるかは不明である.

6 おわりに

本論文では両端キューを用いた固定パスガソリンスタンド問題に対する線形時間のアルゴリズムを与えた. 同問題に対しては Lin, Gertsch, and Russell らがすでに線形時間アルゴリズムを与えているが, 提案アルゴリズムは距離 U 以内のガソリンスタンドの情報のみを保持するようなストリーミングアルゴリズムに修正することができる.

提案アルゴリズムは効率的なアルゴリズムの設計における両端キューの自明でない活用例としても興味深いものと考える.

参考文献

[1] S. Khuller, A. Malekian, and Julián Mestre, To fill or not to fill: The gas station problem, ACM Transactions on Algorithms, Volume 7, Issue 3, Article No.36, pp. 1—16, 2011.

[2] K. Papadopoulos and D. Christofides, A fast algorithm for the gas station problem, Information Processing Letters, Volume 131, pp. 55–59, 2018.

[3] S. H. Lin, N. Gertsch, and J. R. Russell, A linear-time algorithm for finding optimal vehicle refueling policies, Operations Research Letters, Volume 35, pp. 290–296, 2007.

[4] A. Atamtürk, S. Küçükyavuz, Lot sizing with inventory bounds and fixed costs: polyhedral study and computation, Operations Research, Volume 53, pp. 711—730, 2005.