The Snowblower Problem*

Esther M. Arkin[1], Michael A. Bender[2], Joseph S. B. Mitchell[1], and Valentin Polishchuk[1]

[1] Department of Applied Mathematics and Statistics, Stony Brook University. {estie,jsbm,kotya}@ams.stonybrook.edu §
[2] Department of Computer Science, Stony Brook University. bender@cs.stonybrook.edu ¶

Abstract: We introduce the *snowblower problem* (*SBP*), a new optimization problem that is closely related to milling problems and to some material-handling problems. The objective in the SBP is to compute a short tour for the snowblower to follow to remove all the snow from a domain (driveway, sidewalk, etc.). When a snowblower passes over each region along the tour, it displaces snow into a nearby region. The constraint is that if the snow is piled too high, then the snowblower cannot clear the pile.

We give an algorithmic study of the SBP. We show that in general, the problem is NP-complete, and we present polynomial-time approximation algorithms for removing snow under various assumptions about the operation of the snowblower. Most commercially available snowblowers allow the user to control the direction in which the snow is thrown. We differentiate between the cases in which the snow can be thrown in any direction, in any direction except backwards, and only to the right. For all cases, we give constant-factor approximation algorithms; the constants increase as the throw direction becomes more restricted.

Our results are also applicable to robotic vacuuming (or lawnmowing) with bounded capacity dust bin and to some versions of material-handling problems, in which the goal is to rearrange cartons on the floor of a warehouse.

1 Introduction

During a recent major snowstorm in the northeastern USA, one of the authors used a snowblower to clear an expansive driveway. A snowblower is a "material shifting machine," which lifts snow and deposits it nearby. The goal is to dispose of all the snow, moving it outside the driveway. There is a skill in making sure that the deposited piles of snow do not grow higher than

§ Partially supported by the U.S.-Israel Binational Science Foundation (2000160), NASA (NAG2-1620), NSF (CCR-0098172, ACI-0328930, CCF-0431030), and Metron Aviation.

¶ Partially supported by NSF Grants EIA-0112849 and CCR-0208670.

* Full version is available from http://arxiv.org/abs/cs/0603026.

the maximum depth capacity of the snowblower. This experience crystallized into an algorithmic question, which we have called the *Snowblower Problem* (*SBP*): **How does one optimally use a snowblower to clear a given polygonal region?**

The SBP shows up in other contexts: Consider a mobile robot that is equipped with a device that allows it to pick up a carton and then place the carton down again in a location just next to it, possibly on a stack of cartons. With each such operation, the robot shifts a unit of "material". The SBP models the problem in which the robot is to move a set of boxes to a specified destination in the most efficient manner, subject to the constraint that it cannot stack boxes higher than a capacity bound.

In a third motivating application, consider a robotic lawnmower or vacuum cleaner that has a catch basin for the clippings, leaves, dust, or other debris. The goal is to remove the debris from a region, with the constraint that the catch basin must be emptied (e.g., in the compost pile) whenever it gets full.

The SBP is related to other problems on milling, vehicle routing, and traveling salesman tours, but there are two important new features: (a) material must be moved (snow must be thrown), and (b) material may not pile up too high.

While the SBP arises naturally in these other application domains, for the rest of the paper, we use the terminology of snow removal.

The objective of the SBP is to find the shortest snowblower tour that clears a domain P, assumed to be initially covered with snow at uniform depth 1. An important parameter of the problem is the maximum snow depth $D > 1$ through which the snowblower can move. At all times no point of P should have snow of greater depth than D. The snow is to be moved to points outside of P. We assume that each point outside P is able to receive arbitrarily much snow (i.e., that the driveway is surrounded by a "cliff" over which we can toss as much snow as we want).[1]

Snowblowers offer the user the ability to control the direction in which the snow is thrown. Some throw directions are preferable over others; e.g., throwing the snow back into the user's face is undesirable. However, it can be cumbersome to change the throw direction too frequently during the course of clearing. Thus, we consider three *throw models*. In the *default* model throwing the snow backwards is allowed. In the *adjustable-throw* model the snow can be thrown only to the left, right, or forward. In the *fixed-throw* model the snow is always thrown to the right. Even though it seems silly to allow the throw direction to be back into one's face, the default model is the starting point for the analysis of other models and is equivalent to the vacuum cleaner problem (discussed later).

[1] The "cliff" assumption accurately models the capacitated-vacuum-cleaner problem for which there is a (central) "dustpan vac" in the baseboard, where a robotic vacuum cleaner may empty its load [1] and applies also to urban snow removal using snow melters [2] or disposing off the snow into a river.

Results. In this paper we introduce the snowblower problem, model its variants, and give the first algorithmic results for its solution. We show that the SBP is NP-complete for multiply connected domains P. Our main results are constant-factor approximation algorithms for each of the three throw models, assuming $D \geq 2$; refer to Table 1. The approximation ratio of our algorithms increases as the throw direction becomes more restricted. We give extensions for clearing polygons with holes, both where the holes are obstacles and cliffs. Then we discuss how to adapt our algorithms for clearing nonrectilinear polygons and polygons with uneven initial distributions of snow. We conclude by giving a succinct representation of the snowblower tour, in which the tour specification is polynomial in the complexity of the input polygon.

Default model, Thm. 2		
D	2 or 3	any $D \geq 4$
Apx.	6	8

Adjustable throw, Thm. 3	
D	any $D \geq 2$
Apx.	$4 + 3D/\lfloor D/2 \rfloor$

Fixed throw, Thm. 4	
D	any $D \geq 2$
Apx.	$34 + 24D/\lfloor D/2 \rfloor$

Table 1: Approximation factors of our algorithms.

Related Work. The SBP is closely related to milling and lawn-mowing problems, which have been studied extensively in the NC-machining and computational-geometry literatures; see e.g., [4,5,11]. The SBP is also closely related to material-handling problems, in which the goal is to rearrange a set of objects (e.g., cartons) within a storage facility; see [8,9,14]. The SBP may be considered as an intermediate point between the TSP/lawnmowing/milling problems and material-handling problems. Indeed, for $D = \infty$, the SBP is that of optimal milling. Unlike most material-handling problems, the SBP formulation allows the material (snow) to pile up on a single pixel of the domain, and it is this compressibility of the material that distinguishes the SBP from previously studied material-handling problems. With TSP and related problems, every pixel is visited only a constant number of times, whereas with material-handling problems, pixels may have to be visited a number of times exponential in the input size. For this reason, material-handling problems are not even known to be in NP [8,9], in contrast with the SBP. Note that in material handling problems the objective is to minimize *workload* (distance traveled while loaded), while in the SBP (as in the milling/mowing problems) the objective is to minimize total travel distance (loaded or not).

The SBP is also related to the earth-mover's distance (EMD), which is the minimum amount of work needed to rearrange one distribution (of earth, snow, etc.) to another; see [7]. In the EMD literature, the question is explored mostly from an existential point of view, rather than planning the actual process of rearrangement. In the SBP, we are interested in optimizing the length of the tour, and we do not necessarily know in advance the final distribution of the snow after it has been removed from P.

The title of this paper coincides with that of [10] but the problems considered appear to be totally unrelated.

Notation. The input is a polygonal domain, P. Since we are mainly concerned with proving constant factor approximation algorithms, it suffices to consider distances measured according to the L_1 metric. We consider the snowblower to be an (axis-parallel) unit square that moves horizontally or vertically by unit steps. This justifies the assumption in most of our discussion that P is an integral-orthogonal simple polygon, which is comprised of a union of *pixels* – (closed) unit squares with disjoint interiors and integral coordinates. In Section 5 we remark how our methods extend to general (nonrectilinear) regions.

We say that two pixels are *adjacent* or *neighbors* if they share a side; the *degree* of a pixel is the number of its neighbors. For a region $R \subseteq P$ (subset of pixels), let G_R denote the *dual graph* of R, having a vertex in the center of each pixel of R and edges between adjacent pixels. A pixel of degree less than four is a *boundary pixel*. For a boundary pixel, a side that is also on the boundary of P is called a *boundary side*. The set of boundary sides, ∂P, forms the boundary of P. We assume that the elements of ∂P are ordered as they are encountered when the boundary of P is traversed counterclockwise.

An *articulation vertex* of a graph G is a vertex whose removal disconnects G. We assume that G_P has no articulation vertices. (Our algorithms can be adapted to regions having articulation vertices, at a possible increase in approximation ratio.)

Algorithms Overview. Our algorithms proceed by clearing the polygon Voronoi-cell-by-Voronoi-cell, starting from the Voronoi cell of the garage g — the pixel on the boundary of P at which the snowblower tour starts and ends. The order of the boundary sides in ∂P provides a natural order in which to clear the cells. We observe that the Voronoi cell of each boundary side is a tree of one of two special types, which we call *lines* and *combs*. We show how to clear the trees efficiently in each of the throw models. We prove that our algorithms give constant-factor approximations by charging the lengths of the tours produced by the algorithms to two lower bounds, described in the next section.

2 Preliminaries

Voronoi Decomposition. For a pixel $p \in P$ let $V(p)$ denote the element of ∂P closest to p. In case of ties, the tie-breaking rule (see below) is applied. Inspired by computational-geometry terminology, we call $V(p)$ the *Voronoi side* of p. We let $\delta(p)$ denote the length of the path from p to the pixel having $V(p)$ as a side. For a boundary side $e \in \partial P$ we let Voronoi(e) denote the (possibly, empty) set of pixels, having e is the Voronoi side: Voronoi(e) $= \{p \in P | \, V(p) = e\}$. We call Voronoi($e$) the *Voronoi cell* of e. The Voronoi cells of the elements of ∂P form a partition of P, called the *Voronoi decomposition* of P.

A set of pixels $\mathcal{L}$ whose dual graph $G_{\mathcal{L}}$ is a straight path or a path with one bend, is called a *line*. Each line $\mathcal{L}$ has a *root* pixel p, which corresponds to one of the two leaves of $G_{\mathcal{L}}$, and a *base*, $e \in \partial P$, which is a side of p.

A *(horizontal) comb* $\mathcal{C}$ is a union of pixels consisting of a set of vertically adjacent (horizontal) rows of pixels, with all of the rightmost pixels (or all of the leftmost pixels) in a common column. (A vertical comb is defined similarly; however, by our tie breaking rules, we need consider only horizontal combs.) A comb is a special type of *histogram* polygon [6]. The common vertical column of rightmost/leftmost pixels is called the *handle* of comb $\mathcal{C}$, and each of the rows is called a *tooth*. A *leftward* comb has its teeth extending leftwards from the handle; a *rightward* comb is defined similarly. The pixel of a tooth that is furthest from the handle is the *tip* of the tooth. The topmost row is the *wisdom tooth* of the comb. The *root* pixel p of the comb is either the bottommost or topmost pixel of the handle, and its bottom or top side, $e \in \partial P$, is the *base* of the comb. See Fig. 1, left. The union of a leftward comb and a rightward comb having a common root pixel is called a *double-sided comb*.

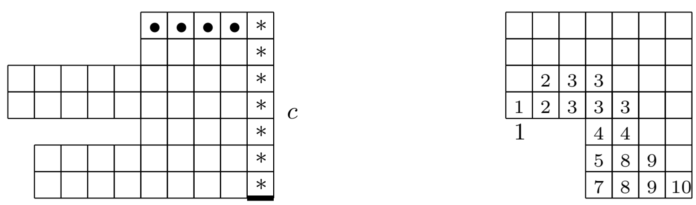

Fig. 1: Left: a comb. The base is bold. The pixels in the handle are marked with asterisks, the pixels in the wisdom tooth are marked with bullets. Right: Voronoi cells. The sides of ∂P are numbered $1 \ldots 28$ counterclockwise. The pixels in the Voronoi cell of a side are marked with the corresponding number. Voronoi cell of side 3 is a comb; Voronoi cells of sides 6, 11, 17, 25, 28 are empty; cells of sides 1, 7, 10, 18, 24 are lines, comprised of just one pixel; cells of the other edges are lines with more than one pixel.

Tie Breaking. Our rules for finding $V(p)$ for a pixel p that is equidistant between two or more boundaries is based on the direction of the shortest path from p to $V(p)$; horizontal edges are preferred to vertical, going down has higher priority than going up, going to the right — than going left. In fact, any tie-breaking rule can be applied as long as it is applied consistently. The particular choice of the rule only affects the orientation of the combs.

Voronoi Cell Structure. An analysis of the structure of the Voronoi partition under our tie breaking rules gives:

Lemma 1. *For a side $e \in \partial P$, the Voronoi cell of e is either a line (whose dual graph is a straight path), or a comb, or a double-sided comb. By our tie-breaking rule, the combs may appear only as the Voronoi cells of horizontal edges. The double-sided combs may appear only as the Voronoi cells of (horizontal) edges of length 1.*

Let p be a boundary pixel of P, let $e \in \partial P$ be the side of p such that $p \in \text{Voronoi}(e)$. We denote Voronoi(e) by $\mathcal{T}(p)$ or $\mathcal{T}(e)$, indicating that it is a unique tree (a line or a comb) that has p as the root and e as the base.

Lower Bounds. We exhibit two lower bounds on the cost of an optimal tour, the *snow lower bound*, based on the number of pixels, and the *distance lower bound*, based on the Voronoi decomposition of the domain. At any time let $s(R)$ be the set of pixels of R covered with snow and also, abusing notation, the number of these pixels. Let $d(R) = \frac{1}{D}\sum_{p \in s(R)} \delta(p)$.

Lemma 2. *Let R be a subset of P with the snowblower starting from a pixel outside R. Then $s(R)$ and $d(R)$ are lower bounds on the cost to clear R.*

Proof. For the snow lower bound, observe that region R cannot be cleared with fewer than $s(R)$ snowblower moves because each pixel of $s(R)$ needs to be visited.

For the distance lower bound, observe that, in order to clear the snow initially residing on a pixel p, the snowblower has to make at least $\delta(p)$ moves. When the snow from p is carried to the boundary of P and thrown away, the snow from at most $D - 1$ other pixels can be thrown away simultaneously. Thus, a region R cannot be cleared with fewer than $d(R)$ moves. □

NP-Completeness. It is known [12,13] that the Hamiltonian path problem in cubic grid graphs is NP-complete. The problem can be straightforwardly reduced to SBP. If G is a cubic grid graph, construct an (integral orthohedral) domain P such that $G = G_P$. Since G_P is cubic, each pixel $p \in P$ is a boundary pixel, thus, the snowblower can throw the snow away from p upon entering it. Hence, SBP on P is equivalent to TSP on G, which has optimum less than $n + 1$ iff G is Hamiltonian (where n is the number of nodes in G). The reduction works for any $D \geq 1$.

The algorithms proposed in this paper show that any domain can be cleared using a set of moves of cardinality polynomial in the number of pixels in the domain, assuming $D \geq 2$. Thus, we obtain

Theorem 1. *If $D \geq 2$, the SBP is NP-complete, both in the default model and in the adjustable throw model, for inputs that are domains with holes.*

3 Approximation Algorithm for the Default Model

In this section we give an 8-approximation algorithm for the case when the snow can be thrown in *all four* directions. We first show how to clear a line efficiently with the operation called *line-clearing*. We then introduce another operation, the *brush*, and show how to clear a comb efficiently with a sequence of line-clearings and brushes. Finally, we splice the subtours through each line and comb into a larger tour, clearing the entire domain. The algorithm for the default model, developed in this section, serves as a basis for the algorithms in the other models.

Clearing a Line. Let $\mathcal{L}$ be a line of pixels; let p and e be its root and the base. We are interested in clearing lines for which the base is a boundary side, i.e., $e \in \partial P$. Let $\ell = s(\mathcal{L})$; let the first J pixels of $\mathcal{L}$ counting from p be clear. We assume that p is already clear ($J > 0$); the snow from it was thrown away through the side e as the snowblower first entered pixel p. Let $\mathcal{L}|J$ denote $\mathcal{L}$ with the J pixels clear; let $\ell - J = kD + r$.[2] Denote by $(\mathcal{L}|J)_D$ the first kD pixels of $\mathcal{L}|J$ covered with snow; denote by $\mathcal{L}_r$ the last r pixels on $\mathcal{L}|J$. The idea of decomposing $\mathcal{L}|J$ into $(\mathcal{L}|J)_D$ and $\mathcal{L}_r$ is that the snow from $(\mathcal{L}|J)_D$ is thrown away with k "fully-loaded" throws, and the snow from $\mathcal{L}_r$ is thrown away with (at most one) additional "under-loaded" throw.

We clear line $\mathcal{L}$ starting at p by moving all the snow through the base e and returning back to p. The basic clearing operation is a back throw. In a back throw the snowblower, entering a pixel u from pixel v, throws u's snow backward onto v. Starting from p, the snowblower moves along $\mathcal{L}$ away from p until either the snowblower moves through D pixels covered with snow or the snowblower reaches the other end of $\mathcal{L}$; this is called the *forward pass*. Next, the snowblower makes a U-turn and moves back to p, pushing all the snow in front of it and over e; this is called the *backward pass*. A forward and backward pass that clears exactly D units of snow is called a *D-full pass*.

Lemma 3. *For arbitrary $D \geq 4$ the line-clearing cost is at most $2s(\mathcal{L} \setminus p) + 4d(\mathcal{L}|J)$. For $D = 2, 3$ the line-clearing cost is at most $2s(\mathcal{L} \setminus p) + 2d(\mathcal{L}|J)$. If every pass is D-full, the cost is $4d(\mathcal{L}|J)$ for $D \geq 4$ and $2d(\mathcal{L}|J)$ for $D = 2, 3$.*

Proof. The clearing cost is $c(\mathcal{L}|J) = c((\mathcal{L}|J)_D) + c(\mathcal{L}_r) = \sum_{i=1}^{k} 2(J - 1 + iD) + 2(\ell - 1) = 2kJ + Dk(k+1) - 2k + 2(\ell - 1)$. The *snow* lower bound of $\mathcal{L} \setminus p$ is $s(\mathcal{L} \setminus p) = \ell - 1$. The *distance* lower bound of $(\mathcal{L}|J)_D$ is $d((\mathcal{L}|J)_D) = \frac{1}{D}\sum_{i=1}^{kD}(J + i) = kJ + k(kD+1)/2$.

Thus,

$$c(\mathcal{L}|J) = 2s(\mathcal{L} \setminus p) + \left(2 + \frac{D-3}{J + (Dk+1)/2}\right) d((\mathcal{L}|J)_D)$$

If every pass is a D-full pass, then $c(\mathcal{L}_r) = 0$. Therefore, $c(\mathcal{L}|J) = c((\mathcal{L}|J)_D) = \left(2 + \frac{D-3}{J+(Dk+1)/2}\right) d((\mathcal{L}|J)_D)$. □

[2] For ease of presentation, we adapt the following convention. For $d \in \{D, \lfloor D/2 \rfloor\}$ and an integer w we understand the equality $w = ad + b$ as follows: b and a are the remainder and the quotient, respectively, of w divided by d.

Clearing a Comb. Let $\mathcal{C}$ be a comb with the root p, base e, and handle $\mathcal{H}$ of length H. Let $\ell_1 \dots \ell_H$ be the lengths of the teeth of the comb. Since we are interested in clearing combs for which the base e is a boundary side ($e \in \partial P$), we assume that pixel p is already clear — the snow from it was thrown away through e as the snowblower first entered p.

Our strategy for clearing $\mathcal{C}$ is as follows. While there exists a line $\mathcal{L} \subset \mathcal{C}$ rooted at p, such that $s(\mathcal{L}) \geq D$, we perform as many D-full passes on $\mathcal{L}$ as we can. When no such $\mathcal{L}$ remains, we call the comb *brush-ready* and we use another clearing operation, the *brush*, to finish the clearing.

A brush, essentially, is a "capacitated" depth-first-search. Among the teeth of a brush-ready comb that are not fully cleared, let t be the tooth, furthest from the base. In a brush, we move the snowblower from p through the handle, turn into t, reach its tip, U-turn, come back to the handle (pushing the pile of snow), turn onto the handle, move by the handle back towards p until we reach the next not fully cleared tooth, turn onto the tooth, and so on. We continue clearing the teeth one-by-one in this manner until D units of snow have been moved (or all the snow on the comb has been moved). Then we push the snow to p through the handle and across e. This tour is called a *brush* (Fig. 2).

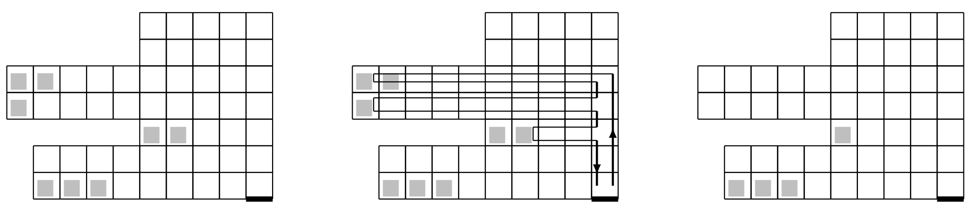

Fig. 2: Left: a brush-ready comb. The snow is shown in light gray. Center: a brush, $D = 4$; the part of the brush, traveling through the handle, is bold. Right: the comb after the brush.

Lemma 4. *For arbitrary $D \geq 4$ the comb $\mathcal{C}$ can be cleared at a cost of at most $4s(\mathcal{C} \setminus p) + 4d(\mathcal{C} \setminus p)$ (at most $4s(\mathcal{C} \setminus p) + 2d(\mathcal{C} \setminus p)$ for $D = 2, 3$).*

Proof. If $s(\mathcal{C} \setminus p) < D$, then the cost of clearing is just $2s(\mathcal{C} \setminus p)$, so suppose, $s(\mathcal{C} \setminus p) \geq D$. Let B be the number of brushes used; let $\mathcal{B}$ be the set of pixels cleared by the brushes. For $b = 1 \dots B$ let t_b and t'_b be the first and the last tooth visited during the bth brush. For $b \in \{1 \dots B-1\}$ the bth brush enters at least 2 teeth, so $t_b > t'_b \geq t_{b+1}$.

Each brush can be decomposed into two parts: the part traveling through the teeth and the part traveling through the handle (Fig. 2). Since each tooth is visited during at most 2 brushes, the length of the first part is at most 4 times the size of all teeth, that is, $4s(\mathcal{C} \setminus \mathcal{H})$. The total length of the second part of all brushes is $2\sum_{b=1}^{B}(t_b - 1)$. Thus, the cost of the "brushing" is

$$c(\mathcal{B}) \le 2\sum_{b=1}^{B}(t_b - 1) + 4s(\mathcal{C} \setminus \mathcal{H}) \le 2\sum_{b=2}^{B} t_b + 4s(\mathcal{C} \setminus p) - 2 \qquad (1)$$

since $t_1 \le H$ and $H \ge 2$ (for otherwise $\mathcal{C}$ is a line).

There are exactly D pixels cleared during each brush $b \in \{0 \dots B-1\}$, and each of these pixels is at distance at least $t_{b'}$ from the base of the comb. Thus, the *distance* lower bound of the pixels, cleared during brush b, is at least $t_{b'}$. Consequently, the *distance* lower bound of $\mathcal{B}$

$$d(\mathcal{B}) \ge \sum_{b=1}^{B} t_{b'} \ge \sum_{b=1}^{B-1} t_{b+1} = \sum_{b=2}^{B} t_b \qquad (2)$$

From (1) and (2), $\mathcal{B}$ can be cleared at a cost of at most $2d(\mathcal{B}) + 4s(\mathcal{C} \setminus p)$.

Let $\mathcal{P} \subseteq \mathcal{C}$ be the pixels, cleared during the line-clearings. By our strategy, during each line-clearing, every pass is D-full; thus, by Lemma 3, $\mathcal{P}$ can be cleared at a cost of at most $4d(\mathcal{P})$ (or $2d(\mathcal{P})$ if $D = 2, 3$). Since $\mathcal{P}$ and $\mathcal{B}$ are snow-disjoint and $\mathcal{P} \cup \mathcal{B} = \mathcal{C} \setminus p$, the lemma follows. $\square$

The above analysis is also valid in the case when the handle is initially clear. This is the case when the second side of a double-sided comb is being cleared. Thus, a double-sided comb can be cleared within the same bounds on the cost of clearing.

Clearing the Domain. Now that we have defined the operations which allow us to clear efficiently lines and combs, we are ready to present the algorithm for clearing the domain.

Theorem 2. *For arbitrary $D \ge 4$ (resp., $D = 2, 3$) an 8-approximate (resp., 6-approximate) tour can be found in polynomial time.*

Proof. Let $p_1, \dots, p_M$ be the boundary pixels of P as they are encountered when going around the boundary of P counterclockwise starting from $g = p_1$; let $e_1, \dots, e_M \in \partial P$ be the boundary sides of $p_1, \dots, p_M$ such that $e_i = Ve(p_i)$, $i = 1 \dots M$. The polygon P can be decomposed into disjoint trees $\mathcal{T}(p_1), \dots, \mathcal{T}(p_M) = \mathcal{T}(e_1), \dots, \mathcal{T}(e_M)$ with the bases $e_1 \dots, e_M$, where each tree $\mathcal{T}(e_i)$ is either a line or a comb.

Our algorithm clears P tree-by-tree starting with $\mathcal{T}(e_1) = \mathcal{T}(g)$. By Lemmas 3 and 4, for $i = 1 \dots M$, the tree $\mathcal{T}(p_i) \setminus p_i$ can be cleared at a cost of at most $4s(\mathcal{T}(p_i) \setminus p_i) + 4d(\mathcal{T}(p_i) \setminus p_i)$ starting from p_i and returning to p_i. Since $\bigcup_1^M \mathcal{T}(p_i) \setminus p_i = P \setminus \{p_1 \dots p_M\}$, the interior of P can be cleared at a cost of at most $c(P \setminus \{p_1 \dots p_M\}) = 4s(P \setminus \{p_1 \dots p_M\}) + 4d(P \setminus \{p_1 \dots p_M\}) \le 4s(P \setminus g) + 4d(P \setminus g) - 4M + 4$. Finally, the tours clearing the interior of P can be spliced into a tour, clearing P at a cost of at most $2M$. Since the optimum is at least $s(P \setminus g)$ and is at least $d(P \setminus g)$, the theorem follows. $\square$

4 Other Models

In this section we give approximation algorithms for the case when the throw direction is restricted. Specifically, we first consider the adjustable-throw-direction formulation. This is a convenient case for the snowblower operator who does not want the snow thrown in his face. We then consider the fixed-throw-direction formulation, which assumes that the snow is always thrown to the right.

We remark that the relatively low approximation factors of the algorithms for the default model, presented in the previous section, were due to a very conservative clearing: the snow from *every* pixel $p \in P$ was thrown through the Voronoi side $V(p)$. Unfortunately, it seems hard to preserve this appealing property if throwing back is forbidden. The reason is that the comb in the Voronoi cell Voronoi(e) of a boundary side $e \in \partial P$ often has a "staircase"-shaped boundary; clearing the first "stair" in the staircase cannot be done without throwing the snow onto a pixel of Voronoi(e'), where $e' \neq e$ is another boundary side. This is why the approximation factors of the algorithms in this section are higher than those in the previous one.

Adjustable Throw Direction

In the adjustable-throw model the snow cannot be thrown backward but can be thrown in the three other directions. To give a constant-factor approximation algorithm for this case, we show how to emulate line-clearings and brushes avoiding back throws (Fig. 3). The approximation ratios increase slightly in comparison with the default model.

Line-clearing. We can emulate a (half of a) pass by a sequence of moves, each with throwing the snow to the left, forward or to the right (Fig. 3, left and center). Thus, the line-clearing may be executed in the same way as it was done if the back throws were allowed. The only difference is that now the snow is moved to the base when the snow from only $\lfloor D/2 \rfloor$ pixels (as opposed to D pixels) of the line is gathered.

Lemma 5. *The line-clearing cost is at most* $3D/\lfloor D/2 \rfloor d(\mathcal{L}|J) + 2s(\mathcal{L} \setminus p)$*. If every pass is* $\lfloor D/2 \rfloor$*-full, the cost is* $3D/\lfloor D/2 \rfloor d(\mathcal{L}|J)$*.*

Proof. Let $\ell - J = k' \lfloor D/2 \rfloor + r'$. Let $(\mathcal{L}|J)_{\lfloor D/2 \rfloor}$ be the first $k'\lfloor D/2 \rfloor$ pixels of $\mathcal{L}|J$, let $\mathcal{L}_{r'}$ be its last r' pixels. Then the cost of the clearing of $\mathcal{L}|J$ is $c(\mathcal{L}|J) = c((\mathcal{L}|J)_{\lfloor D/2 \rfloor}) + c(\mathcal{L}_{r'}) = \sum_{i=1}^{k'} 2(J + i\lfloor D/2 \rfloor) + 2\ell = 2k'J + \lfloor D/2 \rfloor k'(k'+1) + 2\ell$. The lower bounds are given by $s(\mathcal{L} \setminus p) = \ell - 1$ and

$$d((\mathcal{L}|J)_{\lfloor \frac{D}{2} \rfloor}) = \frac{1}{D} \sum_{i=1}^{k'\lfloor \frac{D}{2} \rfloor} (J + i) = \frac{\lfloor \frac{D}{2} \rfloor}{D} \left[k'J + \frac{k'(k'\lfloor \frac{D}{2} \rfloor + 1)}{2} \right] \tag{3}$$

Thus,

$$c(\mathcal{L}|J) \leq \frac{D}{\lfloor \frac{D}{2} \rfloor} \left(2 + \frac{2 + \lfloor D/2 \rfloor k' - k'}{k'J + \frac{\lfloor D/2 \rfloor}{2} k'^2 + \frac{k'}{2}} \right) d(\mathcal{L} \setminus p) + 2s(\mathcal{L} \setminus p) \qquad \square$$

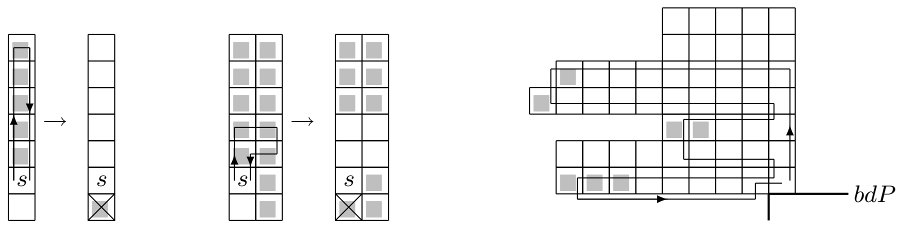

Fig. 3: Emulating line-clearing and brush. The (possible) snow locations are in light gray; s is the snowblower. Left: forward and backward passes in the default model; there are D units of snow on the checked pixel. Center: the passes emulation; there is (at most) $2\lfloor D/2 \rfloor$ units of snow on the checked pixel. Right: the snow to be cleared during a brush is in light gray; there are $\lfloor D/2 \rfloor$ light gray pixels.

Brush. Brush also does not change too much from the default case. The difference is the same as with the line-clearing: now, instead of clearing D pixels with a brush, we prepare to clear only $\lfloor D/2 \rfloor$ pixels (Fig. 3, right). Consequently, the definition of a brush-ready comb is changed — now we require that there is less than $\lfloor D/2 \rfloor$ pixels covered with snow on each tooth of such a comb. Observe that together with each unit of snow, the snow from at most 1 other pixel is moved — thus (although the brush may go outside the comb, as, e.g., in Fig. 3), the brush is feasible.

Lemma 6. *A comb can be cleared at a cost of* $3D/\lfloor D/2 \rfloor d(\mathcal{C} \setminus p) + 4s(\mathcal{C} \setminus p)$.

Proof. In comparison with the default model (Lemma 4) several observations are in place. The number of brushes may go up; we still denote it by B. The cost of the brushes $1 \ldots B-1$ does not change. If the Bth brush has to enter the first tooth, there may be 2 more moves needed to return to the root of the comb (see Fig. 3, right); hence, the total cost of the brushing (1) may go up by 2. The *distance* lower bound (2) goes down by $D/\lfloor D/2 \rfloor$. The rest of the proof is identical to the proof of Lemma 4 (with Lemma 5 used in place of Lemma 3). $\square$

Observe that in fact the snow can be removed from *more than* $\lfloor D/2 \rfloor$ pixels during a brush; we just ignore it for now in our analysis. Note that a double-sided comb can also be cleared in the described way.

Clearing the Domain. As in the default case (Theorem 2),

Theorem 3. *A* $(4+3D/\lfloor D/2 \rfloor)$*-approximate tour can be found in polynomial time.*

Comment on the Parity of D. We remark that if D is even, the cost of the clearing is the same as it would be if the snowblower were able to move through snow of depth $D+1$ (the slight increase of $6/(D-1)$ in the approximation factor would be due to the decrease of the *distance* lower bound).

Fixed Throw Direction

In reality, changing the throw direction requires some effort. In particular, a snow *plow* does not change the direction of snow displacement at all. In this section we consider the fixed throw direction model, i.e., the case of the snowblower which can only throw the snow to the right. We exploit the same idea as in the previous subsection — reducing the problem in the fixed throw direction model to the problem in the default model. All we need is to show how to emulate line-clearing and brush.

In what follows we retain the notation from the previous section.

Lemma 7. *The line-clearing cost is at most* $24D/\lfloor D/2 \rfloor d(\mathcal{L}|J) + 25s(\mathcal{L} \setminus p)$. *If every pass is* $\lfloor D/2 \rfloor$*-full, the cost is* $24D/\lfloor D/2 \rfloor d(\mathcal{L}|J)$.

Proof. We first consider clearing a line whose dual graph is embedded as a single straight line segment and whose base is perpendicular to the segment; we describe the line-clearing, assuming that the line is vertical. Next, we extend the solution to the case when the base is parallel to the edges of the dual graph; this can only be a horizontal line — the first tooth in a (double-)comb. Finally, we consider clearing an L-shaped line; this can only by a tooth together with the (part of the) handle.

A Line $\mathcal{L}$ with $G_{\mathcal{L}} \perp e$. As in the adjustable-throw case (see Fig. 3, left and center), to clear $\mathcal{L}$ we will need to use the pixels to the right of $\mathcal{L}$ to throw the snow onto. Let p' be the boundary pixel, following p counterclockwise around the boundary of P. Before the line-clearing is begun, it will be convenient to have p' clear. Thus, the first thing we do upon entering $\mathcal{L}$ (through p) is clearing p'. Together with returning the snowblower to p it takes 2 or 4 moves (Fig. 4, left); we call these moves *the double-base setup.*

Then, the following invariant is maintained during line-clearing. If the snowblower is at a pixel $q \in \mathcal{L}$ before starting the forward pass, all pixels on $\mathcal{L}$ from p to q are clear, along with the pixels to the right of them (Fig. 4, right). The invariant holds in the beginning of the line-clearing and our line-clearing strategy respects it.

Each back throw is emulated with 5 moves (Fig. 5, left). After moving up by $\lfloor D/2 \rfloor$ pixels (and thus, gathering $2\lfloor D/2 \rfloor$ units of snow on these $\lfloor D/2 \rfloor$ pixels), the snowblower U-turns and moves towards p "pushing" the snow in front of it; a push is emulated with 11 moves (Fig. 6).

The above observations already show that the cost of line-clearing increases only by a multiplicative constant in comparison with the adjustable-throw

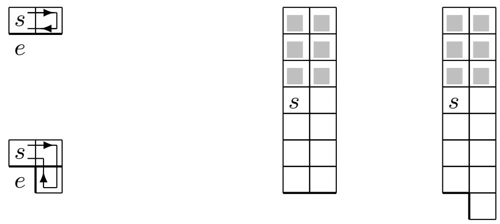

Fig. 4: Left: the double-base setup. Right: before the forward pass the snow below the snowblower is cleared on both lines.

case. A more careful look at the Figs. 5, 6 reveals that: (1) in the push emulation the first two moves are the opposites of the last two, thus, all 4 moves may be omitted – consequently, a push may be emulated by a sequence of only 7 moves; (2) if the boundary side, following e, is vertical, the last push, throwing the snow away from P, may require 9 moves (Fig. 5, right); and, (3) when emulating the last back throw in a forward pass, the last 2 of the 5 moves (the move up and the move to the right in Fig. 5, left) can be omitted – indeed, during the push emulation, the snowblower may as well start to the right of the snow (see Fig. 6). Thus, a line $\mathcal{L}|J$ can be cleared at a cost of $c(\mathcal{L}|J) \leq 4 \quad + \sum_{i=1}^{k'} (J - 1 + (i-1)\lfloor D/2 \rfloor + 5\lfloor D/2 \rfloor + 7(J + i\lfloor D/2 \rfloor - 1)) + J + 5r' + 7(\ell - 1)$.

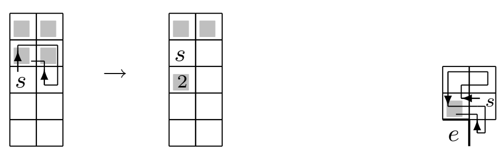

Fig. 5: Left: emulating back throw. Right: pushing the $2\lfloor D/2 \rfloor$ units of snow away from P and returning the snowblower to p may require 9 moves.

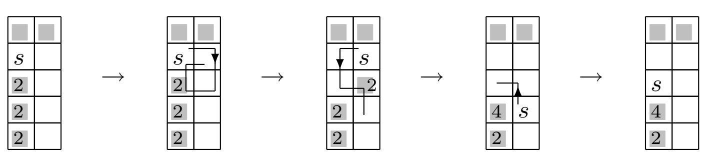

Fig. 6: Emulating pushing the snow in front of the snowblower.

A Line $\mathcal{L}$ with $G_{\mathcal{L}}||e$. Consider a horizontal line, extending to the *left* of the base; such a line may represent the first tooth of a comb. The double-base can be cleared with 8 or 12 moves, the root can be cleared with 3 moves, left) instead of 9 moves; the rest of the clearing does not change. Consider now a horizontal line extending to the *right* of the base; such a line may appear as the first tooth in a double-sided comb. The double-base for such a line can be cleared with 3 moves; the rest of the clearing is the same as for the vertical line.

L-shaped Line. An L-shaped line $\mathcal{L}$ consists of a vertical and a horizontal segment. Each of the segments can be cleared as described above.

Thus, *any* line $\mathcal{L}|J$ can be cleared at a cost of at most $c(\mathcal{L}|J) \leq 12 + \sum_{i=1}^{k'} (J - 1 + (i-1)\lfloor D/2 \rfloor + 5\lfloor D/2 \rfloor + 7(J + i\lfloor D/2 \rfloor - 1)) + J + 5r' + 7(\ell - 1)$. Since the *snow* and *distance* (3) lower bounds do not change, the lemma follows. □

Lemma 8. *A comb can be cleared at a cost of* $34s(\mathcal{C} \setminus p) + 24D/\lfloor D/2 \rfloor d(\mathcal{C} \setminus p)$.

Proof. Brush in the fixed throw direction model can be described easily using analogy with: a) brush in default and adjustable-throw models and b) line-clearing in fixed-throw model. As in the adjustable-throw model, we prepare to clear $\lfloor D/2 \rfloor$ pixels during each brush. Same as with line-clearing, we setup the double-base for the comb with at most 12 moves; also, 9 moves per brush may be needed to push the snow away from P through the base. Back throw and push can be emulated with 5 and 7 moves (Fig. 5, left and Fig. 6). Thus, if the cost of a brush (1) in the default model was, say, c, the cost of the brush in the fixed-throw model is at most $7c + 9$. Since any brush starts with the double-base setup, $c \geq 6$; this, in turn, implies $7c + 9 \leq (51/6)c$. Hence, the cost of clearing $\mathcal{B}$ increases by at most a factor of 51/6.

By Lemma 7, the cost of clearing $\mathcal{P}$, $c(\mathcal{P}) \leq 24D/\lfloor D/2 \rfloor d(\mathcal{P})$. The *snow* and *distance* lower bounds do not change in comparison with the adjustable-throw case. The lemma now follows from simple arithmetic. □

As in the default and adjustable-throw models (Theorems 2, 3),

Theorem 4. *A* $(34 + \frac{24D}{\lfloor D/2 \rfloor})$*-approximate tour can be found in polynomial time.*

5 Extensions

Polygons with Holes. Our methods extend to the case in which P is a polygonal domain with holes. There are two natural ways that holes may arise in the model.

First, the holes may represent obstacles (e.g., walls of buildings that border the driveway). No snow can be thrown onto such holes; the holes' boundaries serve as walls for the motion of the snowblower and for the deposition of snow.

Our algorithm for the default model extends immediately to this variation. The SBP in restricted-throw models, however, may become infeasible.

In the second variation, the holes' boundaries are assumed to be the same "cliffs" as the polygon's outer boundary. It is in fact this version of the problem that we proved to be NP-complete. With some modifications our algorithms work for this variation as well; see the full paper for details.

Nonrectilinear Polygonal Domains. If P is rectilinear, but not integral, we proceed as in [4]: first, the boundary of P is traversed once, and then our algorithms are applied to the remaining part, P', of the domain. Every time the snow is thrown away from P', a certain length (which depends on the throw model) may need to be added to the cost of the tour; thus, the approximation factors of our algorithms may increase by an additive constant.

We can also extend our methods to general nonrectilinear domains. Since the snowblower is not allowed to move outside the domain, care must be taken about specifying which portion of the domain is actually clearable. This portion can be found by traversing the boundary of the domain; then, the accessible portion can be cleared as described above.

Vacuum-Cleaner Problem. Consider the following problem. The floor — a polygonal domain, possibly with holes — is covered with dust and debris. The house is equipped with a central vacuum system, and certain places on the boundary of the floor (the baseboard) are connected to the "dustpan vac" — a dust dump location of infinite capacity [1]. The robotic vacuum cleaner has a dust/debris capacity D and must be emptied to a dump location whenever full. The described problem is equivalent to the SBP in the default throw model and provided the motivation to study the SBP with throwing backwards allowed.

Nonuniform Depth of Snow. Our algorithms generalize easily to the case in which some pixels of the domain initially contain more than one unit of snow. For a problem instance to be feasible it is required that there is less than D (less than $\lfloor D/2 \rfloor$ in restricted-throw direction models) units of snow on each pixel. The approximation ratios in this case depend (linearly) on D (or, in general, on the ratio of D to the minimum initial depth of snow on P).

Capacitated Disposal Region. If instead of "cliffs" at the boundary of P, there is a finite capacity (maximum depth) associated with each point in the complement of P, the SBP more accurately models some material handling problems, but also becomes considerably more difficult. The *snow* lower bound still applies, the *distance* lower bound transforms to a lower bound based on a minimum-cost matching between the pixels in P and the pixels in the complement of P. This problem represents a computational problem related to "earth-mover distance" [7] and is beyond the scope of this paper.

Possible Improvements. We opted for higher approximation factors in favor of more easily described algorithms. For instance, in the adjustable-throw case, the line-clearing cost could be reduced by going up for $D-3$ pixels, making a small detour, and going back; in the fixed-throw model, instead of emulating each and every back throw with 5 moves, we could emulate a whole ($\lfloor D/2 \rfloor$-full) pass at once.

Open Problems. The complexity of the SBP in simple polygons and the complexity of the SBP in the fixed-throw model are open. We also do not have an algorithm for the case of holes as obstacles with restricted throw direction; the hardness of this version is also open.

One factor we did not address is the difficulty in *turning* a snowblower (see [3] for the discussion of the TSP-like problems with turn costs). Another factor is that a snowblower can throw much further than one cell away.

Acknowledgements. We would like to thank the anonymous referees for their helpful comments.

References

1. http://www.centralvacuumstores.com/vacpan.htm.
2. http://www.plowsunlimited.com/snow_melters.htm.
3. E. Arkin, M. Bender, E. Demaine, S. Fekete, J. Mitchell, and S. Sethia. Optimal covering tours with turn costs. *SIAM J. on Computing*, 35(3):531–566, 2005.
4. E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comput. Geom. Theory Appl.*, 17:25–50, 2000.
5. E. M. Arkin, M. Held, and C. L. Smith. Optimization problems related to zigzag pocket machining. *Algorithmica*, 26(2):197–236, 2000.
6. F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete Comput. Geom.*, 21(3):405–420, 1999.
7. S. Cohen and L. Guibas. The earth mover's distance: Lower bounds and invariance under translation. Technical Report CS-TR-97-1597, 1997.
8. J. Culberson. Sokoban is PSPACE-complete. In *Proc. Int. Conf. Fun with Algorithms*, pages 65–76, Elba, Italy, June 1998.
9. E. D. Demaine, M. L. Demaine, M. Hoffmann, and J. O'Rourke. Pushing blocks is hard. *Comput. Geom. Theory Appl.*, 26(1).
10. I. Eliazar. The snowblower problem. *Que. Syst. Th. Appl.*, 45(4):357–380, 2003.
11. M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes Comput. Sci.* Springer-Verlag, June 1991.
12. A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.
13. C. H. Papadimitriou and U. V. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms*, 5:231–246, 1984.
14. V. Polishchuk. The box mover problem. In *Proc. CCCG'04.*, pages 36–39, 2004.