The Snowblower Problem*

Esther M. Arkin† Michael A. Bender‡ Joseph S. B. Mitchell†
Valentin Polishchuk§

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.

1 Introduction

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 the maximum depth capacity of the snowblower. Our experience in using the snowblower 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 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

*The results in this paper were presented at the Seventh International Workshop on the Algorithmic Foundations of Robotics (WAFR 2006), New York City, July 16-18, 2006.

†Applied Mathematics and Statistics Department, Stony Brook University. estie,jsbm@ams.sunysb.edu

‡Computer Science Department, Stony Brook University and Tokutek, Inc. bender@cs.sunysb.edu

§Helsinki Institute for Information Technology, Computer Science Department, University of Helsinki. polishch@cs.helsinki.fi

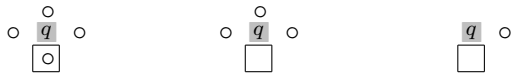

Figure 1: The throw models: left — the default, center — adjustable throw direction, right — fixed throw direction. The snowblower enters from the box; the circles mark the possible positions, where the snow from q may be thrown.

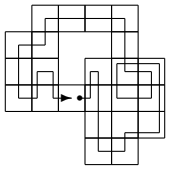

Figure 2: An example tour in the adjustable-throw model.

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

The SBP is related also to other problems on milling, vehicle routing, and traveling salesman tours. The two important new features of the SBP are: (a) material must be eventually removed, and (b) material may not pile up too high.

The objective in 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 (as if the driveway were 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. However, it can be cumbersome to change the throw direction too frequently during the course of clearing. Thus, we consider three *throw models* (Fig. 1). In the *default* model the snow can be thrown in any direction; even throwing backwards is allowed. In the *adjustable-throw* model the snow can be thrown only to the left, right, or forward (Fig. 2). In the *fixed-throw* model the snow is always thrown to the right (Fig. 3). Even though it seems silly to allow the throw direction to be back into one's face, we introduce the default model as the starting point for the analysis of other models. Another reason for considering the default model is that it is equivalent to the vacuum cleaner problem (discussed at the end of the paper).

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, 13]. The SBP is also

[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.

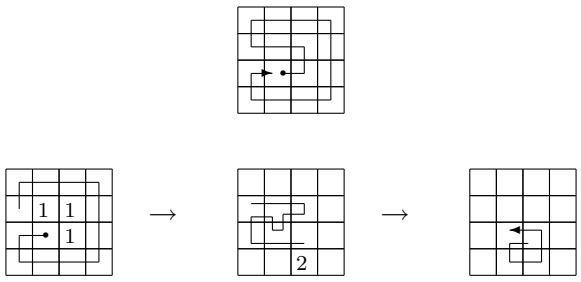

Figure 3: Top: A 4×4 square can be cleared in 16 moves in the adjustable-throw model. In the tour shown in the figure, one may throw to the right, except that when going along the boundary, the throw direction is adjusted so as to remove the snow from the square. Bottom: More moves are needed to clear the square when throwing only to the right. The numbers indicate the amount of snow on non-empty cells at certain points along the tour.

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 [9, 10, 16]. 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 in a grid environment every grid cell is visited only a constant number of times, whereas with material-handling problems, cells 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 [9, 10], 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 [8]. 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.

The title of this paper coincides with that of [11] but the problems considered appear to be totally unrelated.

Our Results

We introduce the snowblower problem, model its variants, and give the first algorithmic results for its solution. We observe that the problem is NP-complete for multiply connected domains. Our main result is an 8-approximation algorithm for clearing simple rectilinear polygons in the default throw model; when $D \in \{2, 3\}$, the approximation ratio drops to 6. We show how to reduce the other throw models to the default one; this leads to constant-factor approximations for the other models as well. The approximation factor increases as the throw direction becomes more restricted. We give extensions for clearing polygons with holes and nonrectilinear polygons.

Algorithms Overview. Our algorithms decompose the domain into Voronoi cells of the boundary pixels and proceed by clearing the domain cell-by-cell. The order of the boundary pixels along the boundary provides a natural order in which to clear the cells. We observe that each cell 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.

2 Preliminaries

The input is a polygonal domain, P. Since we are mainly concerned with constant-factor approximation algorithms, it suffices to consider distances measured according to the L_1 metric. We consider the snowblower to be a unit square that moves horizontally or vertically by unit steps. This justifies our assumption, in most of the discussion, that P is an *integral-orthogonal* simple polygon, comprised of a union of *pixels* — disjoint unit squares with integral vertex coordinates. In Section 5 we remark how our methods extend to general (nonrectilinear) regions and to polygonal domains with holes. Initially P is uniformly covered with snow of unit depth. One pixel g, the *garage*, on the boundary of the domain has no snow, and is occupied by the snowblower. The goal is to remove the snow from P and return the snowblower to the garage.

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 — the plane graph having a vertex in the center of each pixel of R and edges between adjacent pixels. Sometimes when we speak of the region R, we implicitly mean the dual graph G_R. We write $\mathsf{size}(R)$ for the number of pixels in R.

An *articulation vertex* of a graph is a vertex whose removal disconnects the graph. 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.

At any time let $\mathsf{snow}(R)$ be the set of pixels of R covered with snow and also, abusing notation, the number of these pixels. We say that regions R_1 and R_2 are *snow-disjoint* if $\mathsf{snow}(R_1) \cap \mathsf{snow}(R_2) = \emptyset$.

A pixel of degree less than four is a *boundary pixel*. For a boundary pixel, a side that is on the boundary of P is called a *boundary side*. The set of boundary sides, ∂P, forms the boundary of P. Note that we treat ∂P as a set of boundary sides, rather than just as a closed curve.

Lines and Combs

We define a "discrete" Voronoi diagram of ∂P, with cells called *lines* and *combs*. A *line* is a set of pixels $\mathcal{L}$ whose dual graph $G_{\mathcal{L}}$ is a path (Fig. 4). Let p be one of the "terminal" pixels of a line $\mathcal{L}$, i.e., one of the leaves of $G_{\mathcal{L}}$. We call p the *root* of $\mathcal{L}$. Let e be one of the sides of pixel p. We call e the *base* of $\mathcal{L}$. For any line we consider in the paper it will be understood from the context what are its root and base.

A *comb* is a set of pixels consisting of several vertically adjacent (horizontal) rows of pixels, with all of the rightmost pixels (or all of the leftmost pixels) in a common column (Fig. 5). A comb is a special type of *histogram* polygon [7]. The common vertical column of rightmost/leftmost pixels is called the *handle* of the comb, and each of the rows is called a *tooth*. The pixel of a tooth that is furthest from the handle is the *tip* of the tooth. Mixing up haberdasher and dental terms, we call

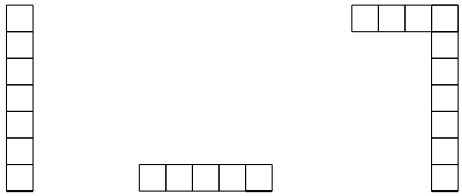

Figure 4: Lines of pixels. The bases are bold.

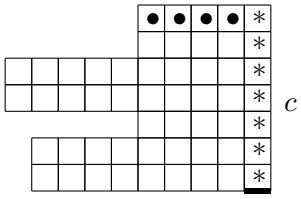

Figure 5: A comb. The base is bold. The pixels in the handle are marked with asterisks, the pixels in the wisdom tooth are marked with bullets.

the topmost row of the comb the *wisdom tooth*. The *root* pixel of the comb is either the bottommost or topmost pixel of the handle, and its bottom or top side, is the *base* of the comb. A *leftward* comb has its teeth extending leftwards from the handle; a *rightward* comb is defined similarly. The union of a leftward comb and a rightward comb having a common root pixel is called a *double-sided comb*.

Domain Decomposition

For a pixel $p \in P$ let $\mathsf{Vs}(p)$ 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 $\mathsf{Vs}(p)$ the *Voronoi side* of p. For a boundary side $e \in \partial P$ we let $\mathsf{Vor}(e)$ denote the (possibly, empty) set of pixels, having e is the Voronoi side: $\mathsf{Vor}(e) = \{p \in P \,|\, \mathsf{Vs}(p) = e\}$, Fig. 6. We call $\mathsf{Vor}(e)$ the *Voronoi cell* of e. The Voronoi cells of the elements of ∂P form a partition of P, called the *Voronoi decomposition* of P. This decomposition is a discrete version of the Voronoi diagram of the edges of P [6].

The rules for finding $\mathsf{Vs}(p)$ for a pixel p that is equidistant from two or more boundaries is based on the direction of the shortest path from p to $\mathsf{Vs}(p)$; vertical edges are preferred to horizontal, 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.

It is easy to see that for a side $e \in \partial P$, the Voronoi cell of e is either a line, or a comb, or a double-sided comb, with e as the base. By our tie-breaking rule, the combs may appear only as the Voronoi cells of horizontal sides. The double-sided combs may appear only as the Voronoi cells of (horizontal) edges of length 1.

The dual graph of a line is a tree (in fact, a path). The dual graph of a comb has a special

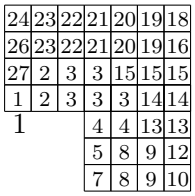

Figure 6: An example of the Voronoi decomposition. The sides of ∂P are numbered $1 \ldots 28$ counterclockwise. The pixels in the Voronoi cell of a side are marked with the corresponding number. Voronoi cell of side 3 is a comb; Voronoi cells of sides 6, 11, 17, 25, 28 are empty; cells of sides 1, 7, 10, 18, 24 are lines, comprised of just one pixel; cells of the other sides are lines with more than one pixel.

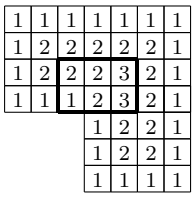

Figure 7: The numbers are $\mathsf{d}(p, \partial P)$ for pixels in the domain. The thick lines show a region R for which $6.5 = \mathit{dist}(R) > \mathit{snow}(R) = 6$ ($D = 2$).

spanning tree, consisting of the vertical path through the handle, and the horizontal paths through the teeth (the tree looks like a comb, hence the name). These trees are used by our algorithms to clear the domain. We will often identify the Voronoi cells with the trees.

Let p be a boundary pixel of P, let $e \in \partial P$ be the side of p such that $p \in \mathsf{Vor}(e)$. We denote $\mathsf{Vor}(e)$ 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 or e as the base.

2.1 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.

Let

$$\mathsf{dist}(R) \quad = \quad \frac{1}{D} \sum_{p \in \mathsf{snow}(R)} \mathsf{d}(p, \partial P) \quad ,$$

where $\mathsf{d}(p, \partial P)$ is 1 plus the shortest-path distance, in the dual graph of the domain, from the pixel p to a boundary pixel (Fig. 7).

Lemma 2.1. *Let R be a subset of P with the snowblower starting from a pixel outside R. Then $\mathsf{snow}(R)$ and $\mathsf{dist}(R)$ 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 $\mathsf{snow}(R)$ snowblower moves because each pixel of $\mathsf{snow}(R)$ needs to be visited.

For the distance lower bound, consider the path (in the dual graph of the domain) taken by the unit of snow residing on a pixel p; the length of the path is at least $\mathsf{d}(p, \partial P)$. Overlay these paths

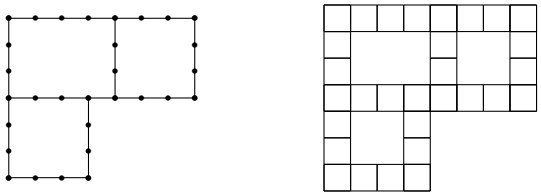

Figure 8: Left: A grid graph G of maximum degree 3. Right: The domain having G as the dual.

from all pixels. Now for each pixel edge, define the *thickness* of the overlay at the edge as the total number of the paths that cross the edge. Since the snowblower moves at most D units at a time, the cost of any clearing tour is at least the total thickness of the overlay at all pixel edges, divided by D. □

The snow lower bound is smaller than the distance bound for a "thin" region; e.g., for a set of boundary pixels. For a set of pixels "deep inside the domain", the distance lower bound is typically larger (see Fig. 7).

2.2 NP-Completeness

The *Hamiltonian cycle problem* [12] is to determine whether there exists a cylce in a graph that visits each vertex exactly once. It is known [14, 15] that the problem in is NP-hard even if restricted to grid graphs with maximum degree 3. We can reduce the problem to SBP in polygons with holes.

Specifically, let G be a grid graph with maximum degree 3. Construct a rectilinear domain P such that $G = G_P$ (Fig. 8); clearly, the construction can be done in polynomial time. Since the maximum degree of G_P is 3, each pixel $p \in P$ is a boundary pixel. That is, the snowblower can throw the snow away from p immediately upon entering the pixel. Hence, P can be cleared in N moves (where N is the number of pixels in P) if and only if G is Hamiltonian. This shows that SBP is NP-hard.

The algorithms proposed in this paper show that any domain can be cleared using a tour of length polynomial in the number of pixels in P. This means that SBP is in NP. An NP-hard problem that is also in NP, is NP-complete (refer to [12] for the definitions related to the complexity classes). Thus, we obtain

Theorem 2.2. *The SBP is NP-complete, both in the default model and in the adjustable throw model, for inputs that are polygonal domains with holes.*

The hardness of SBP in the fixed-throw model and in simple polygons is open. In fact, we do not even know what the optimal solutions are for simple cases like a square or rectangular domain.

3 Approximation Algorithm for the Default Model

In the default throw model the snowblower can throw the snow from a pixel onto any of its neighbors. We give an 8-approximation algorithm for the SBP in this case. 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 $\ell = \mathsf{size}(\mathcal{L})$, and suppose that the first J pixels of $\mathcal{L}$ counting from p are 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 (Fig. 9). Let $\ell - J = kD + r$, where $k, r \in \mathbb{N}_0, r < k$.[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.

Figure 9: The line $\mathcal{L}|4$. The snow is shown in light gray. $J = 4$ first pixels are clear. $l = 7$. $k = 2$, $r = 1$ if $D = 2$.

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 (Fig. 10) 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 3.1. *For arbitrary $D \geq 4$ the line-clearing cost is at most*

$$2\,\mathsf{snow}(\mathcal{L} \setminus p) \quad + \quad 4\,\mathsf{dist}(\mathcal{L}|J) \quad .$$

For $D = 2, 3$ the line-clearing cost is at most

$$2\,\mathsf{snow}(\mathcal{L} \setminus p) \quad + \quad 2\,\mathsf{dist}(\mathcal{L}|J) \quad .$$

If every pass is D-full, the cost is

$$4\,\mathsf{dist}(\mathcal{L}|J)$$

[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.

Figure 10: The back throw. s is the snowblower position. The snow is shown in light gray.

for $D \geq 4$ *and*

$$2\,\mathsf{dist}(\mathcal{L}|J)$$

for $D = 2, 3$.

Proof. The clearing cost is

$$\begin{aligned}
\mathsf{cost}(\mathcal{L}|J) &= \mathsf{cost}((\mathcal{L}|J)_D) + \mathsf{cost}(\mathcal{L}_r) = \\
&= \textstyle\sum_{i=1}^{k} 2(J-1+iD) + 2(\ell-1) = \\
&= 2kJ + Dk(k+1) - 2k + 2(\ell-1) \quad .
\end{aligned}$$

The *snow* lower bound of $\mathcal{L} \setminus p$ is

$$\mathsf{snow}(\mathcal{L} \setminus p) = \ell - 1 \quad .$$

The *distance* lower bound of $(\mathcal{L}|J)_D$ is

$$\mathsf{dist}((\mathcal{L}|J)_D) = \frac{1}{D}\sum_{i=1}^{kD}(J+i) = kJ + k(kD+1)/2 \quad .$$

Thus,

$$\mathsf{cost}(\mathcal{L}|J) = 2\,\mathsf{snow}(\mathcal{L} \setminus p) + \left(2 + \frac{D-3}{J + (Dk+1)/2}\right)\mathsf{dist}((\mathcal{L}|J)_D) \quad .$$

If every pass is a D-full pass, then $\mathsf{cost}(\mathcal{L}_r) = 0$. Therefore,

$$\mathsf{cost}(\mathcal{L}|J) = \mathsf{cost}((\mathcal{L}|J)_D) = \left(2 + \frac{D-3}{J + (Dk+1)/2}\right)\mathsf{dist}((\mathcal{L}|J)_D) \quad .$$

The lemma follows now from simple arithmetic. □

Clearing a Comb

Let $\mathcal{C}$ be a comb with the root p, base e, and handle $\mathcal{H}$ of length H. Let $\ell_1 \ldots \ell_H$ 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.

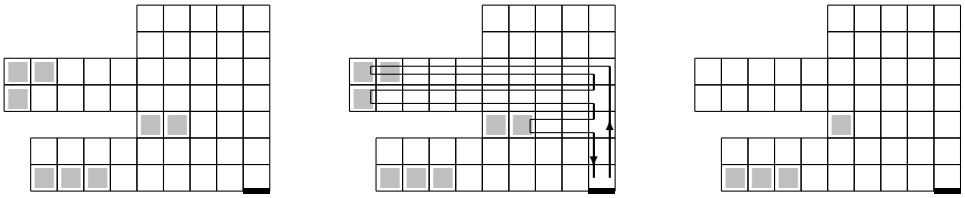

Figure 11: Left: a brush-ready comb. The snow is shown in light gray. Center: a brush, $D = 4$; the part of the brush, traveling through the handle, is bold. Right: the comb after the brush.

Our strategy for clearing $\mathcal{C}$ is as follows. While there exists a line $\mathcal{L} \subset \mathcal{C}$ rooted at p, such that $\mathsf{snow}(\mathcal{L}) \geq D$, 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. 11).

Lemma 3.2. *For arbitrary $D \geq 4$ the comb $\mathcal{C}$ can be cleared at a cost of at most*

$$4\,\mathsf{snow}(\mathcal{C} \setminus p) \quad + \quad 4\,\mathsf{dist}(\mathcal{C} \setminus p) \quad .$$

For $D = 2, 3$ the cost of clearing is at most

$$4\,\mathsf{snow}(\mathcal{C} \setminus p) \quad + \quad 2\,\mathsf{dist}(\mathcal{C} \setminus p) \quad .$$

Proof. If $\mathsf{snow}(\mathcal{C} \setminus p) < D$, then the cost of clearing is just $2\,\mathsf{snow}(\mathcal{C} \setminus p)$, so suppose, $\mathsf{snow}(\mathcal{C} \setminus p) \geq D$. Let B be the number of brushes used; let $\mathcal{B}$ be the set of pixels cleared by the brushes. For $b = 1 \ldots B$ let t_b and t'_b be the first and the last tooth visited during the bth brush. For $b \in \{1 \ldots B-1\}$ 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. 11, center). 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, $4\,\mathsf{size}(\mathcal{C} \setminus \mathcal{H})$. 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

$$\mathsf{cost}(\mathcal{B}) \quad \leq \quad 2\sum_{b=1}^{B}(t_b - 1) \,+\, 4\,\mathsf{size}(\mathcal{C} \setminus \mathcal{H}) \quad \leq \quad 2\sum_{b=2}^{B} t_b \,+\, 4\,\mathsf{snow}(\mathcal{C} \setminus p) - 2 \qquad (1)$$

since $t_1 \leq H$, and $H \geq 2$ (for otherwise $\mathcal{C}$ is a line).

There are exactly D pixels cleared during each brush $b \in \{0 \ldots B-1\}$, 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 $\mathcal{B}$

$$\mathsf{dist}(\mathcal{B}) \quad \geq \quad \sum_{b=1}^{B} t_{b'} \quad \geq \quad \sum_{b=1}^{B-1} t_{b+1} \quad = \quad \sum_{b=2}^{B} t_b \tag{2}$$

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

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 3.1, $\mathcal{P}$ can be cleared at a cost of at most $4\,\mathsf{dist}(\mathcal{P})$ (or $2\,\mathsf{dist}(\mathcal{P})$ 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. □

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 that allow us to clear efficiently lines and combs, we are ready to present the algorithm for clearing the domain.

Theorem 3.3. *When the snowblower can throw snow in any direction, an 8-approximate tour to clear a simple integral-orthogonal polygon can be found in polynomial time.*

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

Our algorithm clears P tree-by-tree starting with $\mathcal{T}(g)$. By Lemmas 3.1 and 3.2, $\mathcal{T}(p_i) \setminus p_i$ can be cleared at a cost of at most

$$4\,\mathsf{snow}(\mathcal{T}(p_i) \setminus p_i) \quad + \quad 4\,\mathsf{dist}(\mathcal{T}(p_i) \setminus p_i)$$

starting from p_i and returning to p_i. Since

$$\bigcup_{i=1}^{M} \mathcal{T}(p_i) \setminus p_i \quad = \quad P \setminus \{p_1 \ldots p_M\} \quad ,$$

the interior of P can be cleared at a cost of at most

$$\begin{array}{rcccccl} \mathsf{cost}(P \setminus \{p_1 \ldots p_M\}) & = & 4\,\mathsf{snow}(P \setminus \{p_1 \ldots p_M\}) & + & 4\,\mathsf{dist}(P \setminus \{p_1 \ldots p_M\}) & \leq \\ & \leq & 4\,\mathsf{snow}(P \setminus g) & + & 4\,\mathsf{dist}(P \setminus g) - 4M + 4 & . \end{array}$$

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 $\mathsf{snow}(P \setminus g)$ and is at least $\mathsf{dist}(P \setminus g)$, the theorem follows. □

For $D = 2, 3$ the bounds of Lemmas 3.1, 3.2 imply a better approximation ratio of 6.

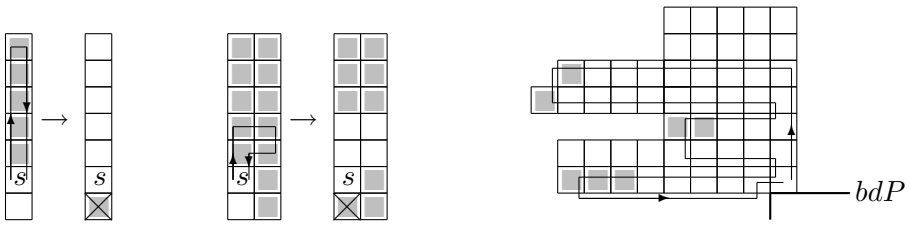

Figure 12: Emulating line-clearing and brush. The snow locations are in light gray; s is the snowblower. Left: after a forward and a backward pass in the default model, there are D units of snow on the checked pixel. Center: the passes emulation; there is (at most) $2\lfloor D/2 \rfloor$ units of snow 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.

4 Other Models

In this section we give approximation algorithms for the case when the throw direction is restricted. Note that the relatively low approximation factor of the algorithm for the default model, presented in the previous section, was due to a very conservative clearing: the snow from *every* pixel $p \in P$ was thrown through the Voronoi side $\mathsf{Vs}(p)$. Unfortunately, it seems hard to preserve this appealing property if throwing back is forbidden. (The reason is that the comb in the Voronoi cell $\mathsf{Vor}(e)$ 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 $\mathsf{Vor}(e')$, where $e' \neq e$ is some other boundary side.)

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. The approximation ratio increases in comparison with the default model, but remains constant.

Line-clearing. The pass in the default model consisted from a forward pass (D steps of throwing the snow back), a U-turn, and a backward pass (D steps of throwing the snow forward); see Fig. 12, left. 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. 12, center). Specifically, we first move forward for $\lfloor D/2 \rfloor$ steps throwing the snow to the right, onto the adjacent line $\mathcal{L}'$ (which increases the depth of the snow on $\mathcal{L}'$ by 1). We then turn right onto $\mathcal{L}'$, throwing the snow to the right. Then, turn right again, and move forward throwing the snow forward. On $\lfloor D/2 \rfloor$'s step, throw the snow to the right. Then move to the right (to arrive in the pixel adjacent to the initial position of the snowblower s) throwing the snow to the left. Finally, move left throwing the snow forward. We end up at the initial snowblower position (before the emulation), with a pile of snow of depth $2\lfloor D/2 \rfloor$ in front of the snowblower (Fig. 12, center) – just like in the default model (Fig. 12, left).

Because in our algorithm the lines are processed in order, pixels on the line $\mathcal{L}'$ to the right of the line currently being cleared, never contain more than 2 units of snow. Thus, the line-clearing

may be executed in the same way as it was done when 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.) Hence, just as in the default model, line-clearing cost may be charged to constant times the distance and the snow lower bounds.

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. 12, 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. 12), the brush is feasible.

A double-sided comb can be cleared in the same way. Overall, just as an the default model, cost of brushing may be charged to constant times the lower bounds.

Clearing the Domain. Since both line-clearing and brushing can be done with a cost within constant times the lower bounds, we have

Theorem 4.1. *When the snowblower can throw snow left, right, or forward, a constant-factor approximation to the optimal snowblower tour can be found in polynomial time.*

We defer the precise calculation of the constants to Theorem A.3 in the appendix.

Fixed Throw Direction

We exploit the same idea as in the previous subsection — reducing the problem in the fixed-throw model to the problem in the default model. With more involved patterns, we can emulate line-clearing and brush while throwing snow only to the right; the emulation cost is only a constant factor away from the cost in the default model. Thus, we obatain

Theorem 4.2. *When the snowblower can throw snow only to the right, a constant-factor approximation to the optimal snowblower tour can be found in polynomial time.*

The emulation patterns, proofs of their correctness and precise calculation of the approximation ratio may be found in Appendix B. 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 (Fig. 13); 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, etc.

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 for the snowblower (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

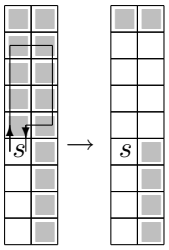

Figure 13: For $D \geq 3$ the line-clearing cost in the adjustable-throw model may be reduced.

infeasible; it is not always possible for the snowblower to enter a thin (one-pixel-width) "channel" between two holes.

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; we sketch here the necessary changes. The boundary, ∂P, of the domain now consists of the boundary pixel sides both on the *outer boundary* of the polygon, denoted $\partial^o P$, and on the boundaries of the holes. As in the simple polygon case, we build the Voronoi decomposition of the domain and prepare to clear it Voronoi-cell-by-Voronoi-cell. The only problem with it now is that there is no readily available Hamiltonian cycle through the elements of ∂P. To build a suitable cycle we find a spanning tree, $\mathbb{T}$, in the graph having a node for each hole and a node for $\partial^o P$ and having an edge between two holes H_1 and H_2 if there are boundary pixel sides, $e_1 \in \partial H_1$ and $e_2 \in \partial H_2$, such that there exist adjacent pixels $p_1 \in \mathsf{Vor}(e_1)$ and $p_2 \in \mathsf{Vor}(e_2)$. Then, after $\mathsf{Vor}(e_1)$ is cleared, we direct the snowblower to start clearing $\mathsf{Vor}(e_2)$.[3] After that, the snowblower continues to clear the Voronoi cells of the boundary pixel sides of H_2 until another edge of $\mathbb{T}$, connecting H_2 to a hole H_3, is encountered (if H_2 is a leaf of $\mathbb{T}$, $H_3 = H_1$), and so on. The cost of clearing the Voronoi cells of the sides in ∂P does not change, and the cost of traversing $\mathbb{T}$ is at most twice the size of P; thus, the proposed algorithm remains a constant-factor approximation algorithm.

Nonrectilinear Polygonal Domains. Our discussion so far has assumed that P is integral-orthogonal and that the snowblower makes axis-parallel movements. If P is rectilinear, but not necessarily 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

[3]The clearing starts from creating a path of width 2 from p_2 to e_2; the path "bridges" the holes H_1 and H_2.

system, and certain places on the boundary of the floor (the baseboard) are connected to the "dustpan vac" — a dust dump location of essentially 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 straightforwardly 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. Another generalization of the problem is to consider the dump locations to have finite capacities. 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" [8] and is beyond the scope of this paper.

Implicit Representation of the Tour. As in [4, 5], we make the distinction between *explicit* and *implicit* representations of the domains and snowblower tours. As mentioned in the introduction, we assume that a domain is given as the union of pixels; this way the size of the input to the problem is $O(N)$, where N is the number of pixels in the domain. The size of the description of the snowblower tour produced by our algorithm is polynomial in N, i.e., polynomial in the input size. Instead, the domain may be given in polygonal representation, as a list of coordinates of its n vertices; the size of such a representation is $O(n \log W)$, where W is the largest coordinate in the input. In principle, N may be $\Omega(W)$, and hence the length of the description of the snowblower tour may appear to be exponential in the size of the input. Below we give a succinct (polynomial in $n \log W$) representation of the tour.

Our algorithms produce tours, comprised of line-clearings and brushes. Our Voronoi decomposition of the domain is the discretized version of the Voronoi diagram of the edges of the domain; the latter is $O(n)$ in size and can be found efficiently [6]. Given the diagram, it is easy to constrain the (axis-parallel) motion of the snowblower to stay within a Voronoi cell of an edge: when "in doubt", i.e., when the snowblower is about to enter a pixel, intersected by an edge of the Voronoi diagram, it can be decided "in place", in constant time (based on the tie breaking rules), whether entering the pixel will place the snowblower into the Voronoi cell of another side of P.

The clearing of a Voronoi face in the Voronoi diagram is done tree-by-tree: first a (double-)comb is cleared (if present), then a set of lines (if present), then the other comb (if present). Thus, finding a short representation of a tour boils down to exhibiting succinct representations of the tours through a comb and through a set of lines with adjacent bases comprising a boundary edge of P. The descriptions of these tours given in Sections 3 and 4 provide such representations.

Open Problems. The complexity of the SBP in simple polygons and the complexity of the SBP in the fixed-throw model are 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. Finally, our approximation ratios are likely not the best possible. We were not able to come up with examples where our lower bounds are close to the optimum. One difficulty here is actually computing the optimum, even for small examples. In trivial cases like a domain consisting of just one line, the snow and distance lower bounds are far from the optimum cost.

Acknowledgements

We thank Joseph O'Rourke and anonymous reviewers for their helpful comments. We thank the referees and the participants of WAFR 2006 for stimulating discussions and suggestions. EMA and JSBM have been partially supported by the U.S.-Israel Binational Science Foundation (2000160), NASA (NAG2-1620), NSF (CCF-0528209,CCF-0729019), and Metron Aviation. MAB has been partially supported by NSF Grants EIA-0112849 and CCR-0208670. VP is supported by the Academy of Finland grant 138520.

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. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, Sept. 1991.

[7] 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.

[8] S. Cohen and L. Guibas. The earth mover's distance: Lower bounds and invariance under translation. Technical Report CS-TR-97-1597, Stanford Univ. Dept Of Computer Science, 1997.

[9] J. Culberson. Sokoban is PSPACE-complete. In *Proc. Int. Conf. Fun with Algorithms*, pages 65–76, Elba, Italy, June 1998.

[10] E. D. Demaine, M. L. Demaine, M. Hoffmann, and J. O'Rourke. Pushing blocks is hard. *Comput. Geom. Theory Appl.*, 26(1):21–36, August 2003. Special issue of selected papers from the 13th Canadian Conference on Computational Geometry, 2001.

[11] I. Eliazar. The snowblower problem. *Queueing Systems Theory and Applications*, 45(4):357–380, 2003.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, USA, 1979.

[13] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes Comput. Sci.* Springer-Verlag, June 1991.

[14] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.

[15] C. H. Papadimitriou and U. V. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms*, 5:231–246, 1984.

[16] V. Polishchuk. The box mover problem. In *Proceedings of 16th Canadian Conference on Computational Geometry*, pages 36–39, 2004.

A Adjustable-throw model

Lemma A.1. *The line-clearing cost is at most* $3D/\lfloor D/2\rfloor\,\mathsf{dist}(\mathcal{L}|J) + 2\,\mathsf{snow}(\mathcal{L}\setminus p)$. *If every pass is* $\lfloor D/2\rfloor$*-full, the cost is at most* $3D/\lfloor D/2\rfloor\,\mathsf{dist}(\mathcal{L}|J)$.

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

$$\begin{aligned}
\mathsf{cost}(\mathcal{L}|J) &= \mathsf{cost}((\mathcal{L}|J)_{\lfloor D/2\rfloor}) + \mathsf{cost}(\mathcal{L}_{r'}) = \\
&= \textstyle\sum_{i=1}^{k'} 2(J + i\lfloor D/2\rfloor) + 2\ell = \\
&= 2k'J + \lfloor D/2\rfloor k'(k'+1) + 2\ell \quad .
\end{aligned}$$

The lower bounds are given by

$$\mathsf{snow}(\mathcal{L}\setminus p) = \ell - 1$$

and

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

Thus,

$$\mathsf{cost}(\mathcal{L}|J) \le \frac{D}{\lfloor \frac{D}{2}\rfloor}\left(2 + \frac{2 + \lfloor D/2\rfloor\, k' - k'}{k'J + \frac{\lfloor D/2\rfloor}{2}k'^2 + \frac{k'}{2}}\right)\mathsf{dist}(\mathcal{L}\setminus p) + 2\,\mathsf{snow}(\mathcal{L}\setminus p) \quad .$$

□

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

Proof. In comparison with the default model (Lemma 3.2) several observations are in order. The number of brushes may go up; we still denote it by B. We also retain the other notation, introduced in the default case. 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. 12, 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 3.2 (with Lemma A.1 used in place of Lemma 3.1). □

Similarly to the default case (Theorem 3.3), we obtain

Theorem A.3. *When the snowblower can throw snow in left, right, or forward, a* $(4 + 3D/\lfloor D/2\rfloor)$*-approximate tour to clear a simple integral-orthogonal polygon can be found in polynomial time.*

B Fixed throw model

Lemma B.1. *The line-clearing cost is at most* $24D/\lfloor D/2\rfloor\,\mathsf{dist}(\mathcal{L}|J) + 25\,\mathsf{snow}(\mathcal{L}\setminus p)$. *If every pass is* $\lfloor D/2\rfloor$*-full, the cost is* $24D/\lfloor D/2\rfloor\,\mathsf{dist}(\mathcal{L}|J)$.

Figure 14: The double-base setup. The boundary sides are bold.

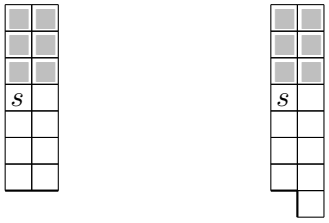

Figure 15: Before the forward pass the snow below the snowblower is cleared on both lines.

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 (like, e.g., the line in Fig. 9); 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 be 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. 12, 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. 14); 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. 15). 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. 16). 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. 17).

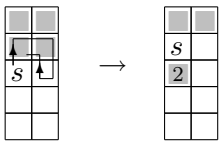

Figure 16: Emulating back throw.

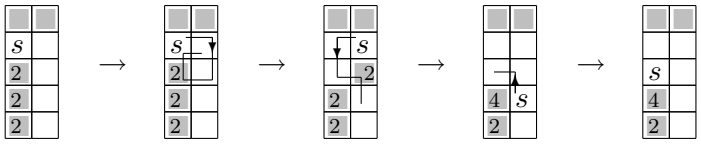

Figure 17: Emulating pushing the snow in front of the snowblower.

Figure 18: Pushing the $2\lfloor D/2 \rfloor$ units of snow away from P and returning the snowblower to p may require 9 moves.

Overall $\mathcal{L}|J$ can be cleared at a cost of at most

$$\begin{aligned} \mathsf{cost}(\mathcal{L}|J) &= \mathsf{cost}(\textit{double-base}) + \mathsf{cost}((\mathcal{L}|J)_{\lfloor D/2 \rfloor}) + \textit{cost}(\mathcal{L}_{r'}) = \\ &= 4 + \\ &\quad + \textstyle\sum_{i=1}^{k'} (J - 1 + (i-1)\lfloor D/2 \rfloor + 5\lfloor D/2 \rfloor + 11(J + i\lfloor D/2 \rfloor - 1)) + \\ &\quad + J + 5r' + 11(\ell - 1) \quad . \end{aligned} \tag{4}$$

This already shows that the cost of line-clearing increases only by a multiplicative constant in comparison with the adjustable-throw case. We can further reduce the cost by observing the following:

- In the push emulation (Fig. 17), the first two moves are the opposites of the last two moves. Thus, all 4 of these moves can be omitted. Consequently, a push can be emulated by a sequence of 7 moves, and the 11 in (4) can be changed to 7.
- The last push, throwing the snow away from P, may require 9 moves if the boundary side, following e, is vertical (Fig. 18). Thus, a 2 may be added to each summand in the second and third line of (4).
- When emulating the last back throw in a forward pass, the last 2 of the 5 moves emulating the throw (the move up and the move to the right in Fig. 16) may be omitted. Indeed, during the emulation of the push, the snowblower may as well start to the right of the snow (see Fig. 17). Thus, a 2 may be subtracted from each summand in the second and third lines in (4).

Overall for a line $\mathcal{L}$ with $G_{\mathcal{L}} \perp e$ we have

$$\begin{aligned} \mathsf{cost}(\mathcal{L}|J) \leq\ & 4 + \\ & + \textstyle\sum_{i=1}^{k'} (J - 1 + (i-1)\lfloor D/2 \rfloor + 5\lfloor D/2 \rfloor + 7(J + i\lfloor D/2 \rfloor - 1)) + \\ & + J + 5r' + 7(\ell - 1) \end{aligned} \tag{5}$$

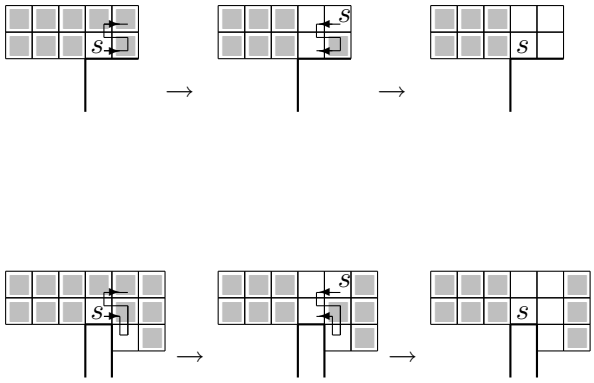

Figure 19: Setting up the double-base for clearing a horizontal line extending to the left of its base. Depending on the direction of the edge adjacent to the base from the right, there are 8 (above) or 12 (below) moves necessary.

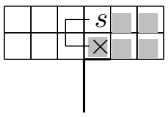

Figure 20: Clearing the root of a horizontal, extending to the left, line with 3 moves. There is $2\lfloor D/2 \rfloor$ units of snow on the checked pixel.

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 (Fig. 19), the root can be cleared with 3 moves (Fig. 20) instead of 9 moves (see Fig. 18); the rest of the clearing does not change.

Thus, a horizontal line $\mathcal{L}|J$, extending to the *left* of its base can be cleared at a cost of at most

$$\begin{aligned} \mathsf{cost}(\mathcal{L}|J) \quad = \quad & 12 \quad + \\ & + \textstyle\sum_{i=1}^{k'} \left(J - 1 + (i-1)\lfloor D/2 \rfloor + 5\lfloor D/2 \rfloor + 7(J + i\lfloor D/2 \rfloor - 1) - 6\right) + \\ & + \quad J + 5r' + 7(\ell - 1) - 6 \end{aligned} \tag{6}$$

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 (Fig. 21); the rest of the clearing is the same as for the vertical line.

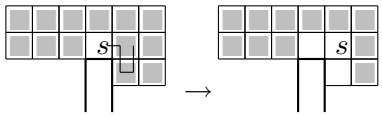

Figure 21: Setting up the double-base for clearing a horizontal line extending to the right of its base.

Thus, a horizontal line $\mathcal{L}|J$, extending to the *right* of its base can be cleared at a cost of at most

$$\begin{aligned}\mathsf{cost}(\mathcal{L}|J) \quad \leq \quad & 3 \quad + \\ & + \textstyle\sum_{i=1}^{k'} \left(J - 1 + (i-1)\left\lfloor D/2 \right\rfloor + 5\left\lfloor D/2 \right\rfloor + 7(J + i\left\lfloor D/2 \right\rfloor - 1)\right) + \\ & + J + 5r' + 7(\ell - 1)\end{aligned} \tag{7}$$

An 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 the cost of clearing an L-shaped line $\mathcal{L}|J$ is maximum of the setup and clearing costs in (5)–(7):

$$\begin{aligned}\mathsf{cost}(\mathcal{L}|J) \quad \leq \quad & 12 \quad + \\ & + \textstyle\sum_{i=1}^{k'} \left(J - 1 + (i-1)\left\lfloor D/2 \right\rfloor + 5\left\lfloor D/2 \right\rfloor + 7(J + i\left\lfloor D/2 \right\rfloor - 1)\right) + \\ & + J + 5r' + 7(\ell - 1)\end{aligned}$$

Of course, *any* line can be cleared at the above cost.

The *snow* lower bound is still given by

$$\mathsf{snow}(\mathcal{L} \setminus p) \quad = \quad \ell - 1$$

The *distance* lower bound is still given by (3)

$$\mathsf{dist}((\mathcal{L}|J)_{\lfloor D/2 \rfloor}) \quad = \quad \frac{\lfloor D/2 \rfloor}{D}\left[k'J + \frac{k'(k'\lfloor \frac{D}{2} \rfloor + 1)}{2}\right]$$

and the lemma follows. □

Lemma B.2. *A comb can be cleared at a cost of* $34\,\mathsf{snow}(\mathcal{C} \setminus p) + 24D/\lfloor D/2 \rfloor\,\mathsf{dist}(\mathcal{C} \setminus p)$.

Proof. Brush in the fixed throw direction model can be described easily using analogy with: a) brush in the default and the adjustable-throw models and b) line-clearing in the 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. 16 and Fig. 17). 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 brushing increases by at most a factor of 51/6.

The *snow* and *distance* lower bounds do not change in comparison with the adjustable-throw case, so, by Lemma A.2 the cost of brushing is

$$\mathsf{cost}(\mathcal{B}) \quad \leq \quad \frac{51}{6}\left[\frac{2D}{\lfloor \frac{D}{2} \rfloor}\,\mathsf{dist}(\mathcal{B}) \quad + \quad 4\,\mathsf{snow}(\mathcal{C} \setminus p)\right] \quad = \quad \frac{17D}{\lfloor \frac{D}{2} \rfloor}\,\mathsf{dist}(\mathcal{B}) \quad + \quad 34\,\mathsf{snow}(\mathcal{C} \setminus p) \quad .$$

By Lemma B.1, the cost of clearing $\mathcal{P}$—the part of the comb cleared with line-clearings—is at most

$$\mathsf{cost}(\mathcal{P}) \quad \leq \quad \frac{24D}{\lfloor D/2 \rfloor}\,\mathsf{dist}(\mathcal{P})$$

Since $\mathcal{B}$ and $\mathcal{P}$ are snow-disjoint and $\mathcal{B} \cup \mathcal{P} \subseteq \mathcal{C} \setminus p$,

$$\frac{17D}{\lfloor \frac{D}{2} \rfloor} \mathsf{dist}(\mathcal{B}) + \frac{24D}{\lfloor D/2 \rfloor} \mathsf{dist}(\mathcal{P}) \quad \leq \quad \frac{24D}{\lfloor D/2 \rfloor} \mathsf{dist}(\mathcal{C} \setminus p) \quad ,$$

and the lemma follows. □

Identically to the default and adjustable-throw models, from Lemmas B.1, B.2, we have:

Theorem B.3. *When the snowblower can throw snow only to the right, a* $(34 + \frac{24D}{\lfloor D/2 \rfloor})$*-approximate tour to clear a simple integral-orthogonal polygon can be found in polynomial time.*