Received: 18 February 2023 | Revised: 12 May 2023 | Accepted: 22 June 2023

DOI: 10.1002/nme.7318

RESEARCH ARTICLE

WILEY

A Minkowski difference-based advancing front packing technique for generating convex noncircular particles in complex domains

Ming Xia[1] | **Xin Xu**[1] | **Fengqiang Gong**[2] | **Min Wang**[3] | **Y. T. Feng**[4]

[1]Hunan Key Laboratory of Geomechanics and Engineering Safety, Xiangtan University, Xiangtan, China

[2]School of Civil Engineering, Southeast University, Nanjing, China

[3]T-3 Fluid Dynamics and Solid Mechanics Group, Los Alamos National Laboratory, Los Alamos, New Mexico, USA

[4]Zienkiewicz Centre for Computational Engineering, Swansea University, Swansea, UK

Correspondence
Y. T. Feng, Zienkiewicz Centre for Computational Engineering, Swansea University, Swansea SA1 8EN, UK.
Email: y.feng@swansea.ac.uk

Funding information
National Natural Science Foundation of China, Grant/Award Number: 12072217; Natural Science Foundation of Hunan Province, Grant/Award Number: 2022JJ30567; Scientific Research Foundation of Education Department of Hunan Province, Grant/Award Number: 20B557

Abstract

In this work, a Minkowski difference-based advancing front approach is proposed to generate convex and non-circular particles in a predefined computational domain. Two specific algorithms are developed to handle the contact conformity of generated particles with the boundaries of the computational domain. The first, called the open form, is used to handle the smooth contact of generated particles with (external) boundaries, while the other, called the closed form, is proposed to handle the internal boundaries of a computational domain with a complex cavity. The Gilbert-Johnson-Keerthi (GJK) method is used to efficiently solve the contact detection between the newly generated particle at the front and existing particles. Furthermore, the problem of one-sided particle lifting, which can cause some defects in the packing structure in existing advancing front methods during packing generation, is highlighted and an effective solution is developed. Several examples of increasing complexity are used to demonstrate the efficiency and applicability of the proposed packing generation approach. The numerical results show that the generated packing is not only more uniform, but also achieves a higher packing density than existing advancing front methods.

KEYWORDS

advancing front approach, convex noncircular particles, discrete element method, Gilbert-Johnson-Keerthi (GJK) method, interior and external domain boundaries, Minkowski difference

1 | INTRODUCTION

Particle-based numerical methods, such as the discrete element method (DEM),[1–4] discontinuous deformation analysis (DDA)[5–7] and non-smooth contact dynamics (NSCD) method,[8–10] have been widely used to simulate complex particulate systems from atomic to crustal scale. The first and also an important step for such a simulation is to efficiently generate a high quality particle assemblage that satisfies some pre-requisites in terms of size, shape and volume fraction (or density).

In general, three different types of methods have been developed: the dynamical approach,[11–13] the constructive approach,[14–20] and the coupled constructive-dynamic approach.[21–23]

In the dynamical approach, particles are first generated in a domain and then their positions and/or sizes are dynamically changed under the applied compression/gravity force within the standard DEM framework. However, a large number of numerical steps are often required to obtain a satisfactory packing density, making this approach very time consuming. Additionally, there is a certain amount of overlap between particles due to external/internal forces during packing, resulting in a physically unrealistic pre-stress within the particle assembly.

In the constructive approach for generating particle packings, particles are added sequentially to the domain in accordance with certain geometric rules that are based on the existing particles already generated. This method ensures that there is no overlap between particles. The constructive approach is very efficient, typically several orders of magnitude faster than the dynamical approach. This is because the position of a particle is determined by only a few known geometric constraints. In the coupled constructive-dynamical approach, the packing initially generated by the constructive approach can be further densified by dynamic techniques to achieve a higher density. The efficiency of this coupled approach lies between that of the dynamic approach and that of the constructive approach.

Due to the high efficiency, several constructive techniques have been developed in the last two decades: regular arrangements, sequential inhibition, sedimentation techniques, mesh-based approach and advancing front approach.[24] Among them, the advancing front approach (AFA), which is originally proposed in the seminal work by Feng et al.[16] to pack discs in 2D domains, is very efficient with a linear complexity. For example, on a PC with a single 1 GHz processor, it takes only 3.77 s to pack 1 million discs. Another important advantage of AFA is that the particle size distribution can be user defined, which is very versatile.

There are two versions of the AFA: the closed form and the open form. In the closed form, the initial front is formed first by placing three touching discs in the centre of the domain. Then discs are generated in contact with the other two previous discs in an outward direction. The final particle assembly is obtained by deleting those discs generated outside the external boundary (EB) of the domain. Thus, in the closed form, there may be some gaps between the assembly and the boundaries. To overcome this problem, the open form is developed to include the EB in the generation of the discs. However, the gaps at the top of the domains are not addressed. Alternatively, Bagi[14] built the initial front near the EB and then generated discs inwards. This is called the inward-AFA based on the closed form. However, the central region cannot be completely filled. To reduce the gap near the EB, Dong et al.[25] used the Newton-Raphson iteration method to re-determine the radius and position of the new disc near the EB, which can guarantee that the disc is tangent to the boundary and the other two discs simultaneously. Recently, Xu and Xia[26] have extended the method to the open form to eliminate the gaps at the upper boundary. Overviews of AFA can be found in Löhner and Oñate[18] and Morfa et al.[27]

Due to the simple geometric description of circular particles, it has been widely used in particle-based numerical modelling. However, particle shape has some profound effects on the macro- and micromechanical behaviour of granular materials,[28,29] and particles are non-circular in reality. Therefore, how to efficiently generate convex non-circular particle assemblies has been a forefront topic in recent years. While the packing of convex non-circular particles has received considerable attention, the ellipse, being the simplest such particle, has been the focus of much research. However, non-elliptical particle packing remains challenging due to the complex geometry involved, making it difficult to achieve efficient handling. Extending previous packing approaches from circular to non-circular particles is not straightforward. Many existing approaches are based on either the dynamic approach or the coupled constructive-dynamic approach.

As mentioned above, these approaches are very time consuming. Therefore, some researchers have mainly used the constructive approach to generate non-circular particles. For example, Feng et al.[30] extended their closed-form AFA to generate convex polygons and ellipses based on Minkowski Sum. Subsequently, the AFA has been modified and extended to generate convex polygons.[27,31] However, there are three shortcomings: (1) relatively large gaps can be generated near EB, which reduces the quality of particle assemblies; (2) both interior boundary (IB) and EB cannot be well considered; and (3) *the one-sided lifting problem* during packing (as will be detailed in Section 4.1) using the open form is not reported.

In this work, a Minkowski difference-based advancing front approach is developed to overcome the aforementioned shortcomings of AFA in generating convex non-circular particle packings. Furthermore, careful consideration is given to IB/EB to solve the gap problem between particles and domain boundaries. The fundamentals of the Minkowski difference and its application are described in detail in Section 2. Then, the closed form and the open form are proposed in Sections 3 and 4, respectively. In particular, Section 4 highlights the one-sided lifting problem during packing using the open form, and the corresponding treatment is proposed. Section 5 presents several examples to illustrate the performance and effectiveness of the proposed approach. Finally, conclusions are drawn in Section 6.

It is worth noting that a convex polygon with a certain number of vertices can be used to approximate convex non-circular particles of any shape. Therefore, this work focuses on the packing of convex polygons. In the case of analytically defined convex shapes, such as ellipses and super-quadrics, they can be first discretised into convex polygons, and then the associated packing is reduced to the packing of convex polygons, which can be effectively handled by the proposed procedures.

2 | MINKOWSKI DIFFERENCE AND ITS APPLICATION

2.1 | Minkowski difference and contact state

A geometric shape (2D or 3D) is considered to be the set of all individual points contained in the shape.[4,32] Consider two blocks A and B. Their Minkowski sum, denoted as $A \oplus B$, is obtained by adding every point in A to every point in B:

$$A \oplus B = \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\} \tag{1}$$

The Minkowski difference of A and B, denoted as $A \ominus B$, is defined here to be the Minkowski sum of A and $(-B)$:

$$A \ominus B = \{\mathbf{a} - \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\} \tag{2}$$

where $(-B)$ can be viewed as the symmetric block of B with respect to the origin:

$$-B = \{-\mathbf{b} : \mathbf{b} \in B\} \tag{3}$$

The boundaries of $A \oplus B$ and $A \ominus B$ are denoted as $\partial(A \oplus B)$ and $\partial(A \ominus B)$, respectively.

Consider a hexagon A and a quadrilateral B as an example, and define the origin $\mathbf{o}$ to coincide with the centroid of B (i.e., $\mathbf{c}_b$). Figure 1A shows the corresponding Minkowski sum and difference of these two blocks. We will focus on the IB and EB in the present study. As a 2D domain can be considered as being formed by a number of connected lines, the Minkowski sum/difference of a line W and the quadrilateral B is also illustrated in Figure 1B as the result of sliding B along W. Both line and polygon vertices are stored in anti-clockwise order for easy visualisation. It can be seen that the

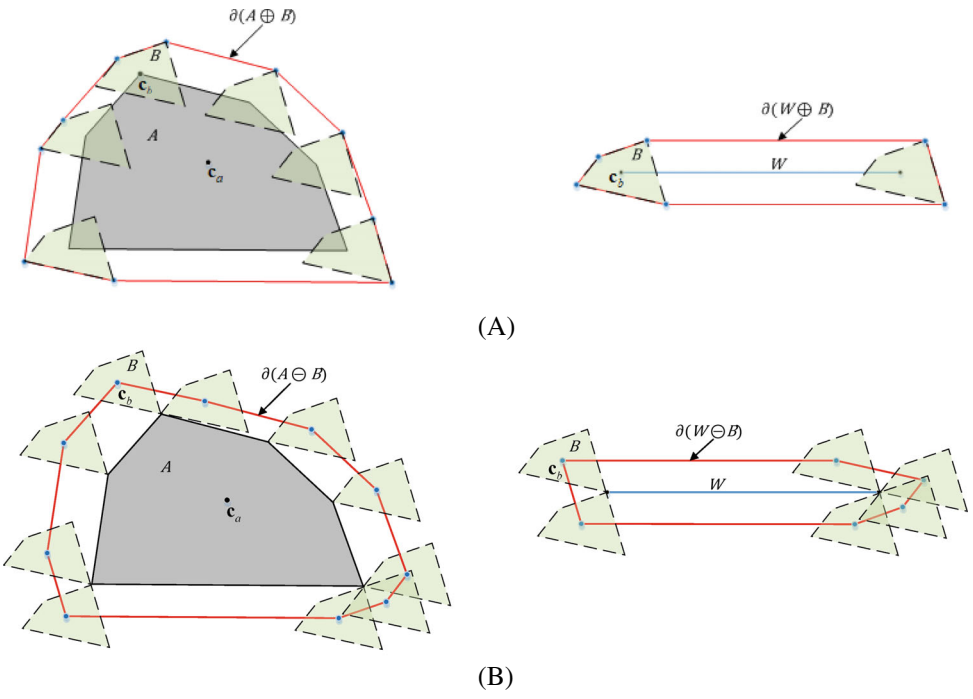

FIGURE 1 Two blocks: (A) Minkowski sum; (B) Minkowski difference.

geometric meaning of $\partial(A \ominus B)$ and $\partial(W \ominus B)$ are the locus of the centroid of block B (i.e., $\mathbf{c}_b$) sliding along the boundary of block A and line W, respectively, while the boundary of B is kept in contact with the boundary of A or W. The procedures to construct the Minkowski sum of two convex polygons or a polygon and a line are summarised in Algorithms 1 and 2, respectively. Note that in Algorithms 1 and 2, the vertices of a polygon or a line are ordered anti-clockwise with the first vertex having the lowest y coordinate.

Algorithm 1. Minkowski sum of two convex polygons

Input: Two convex polygons A and B with vertices $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and $\mathbf{w}_1, \ldots, \mathbf{w}_m$

Output: The Minkowski sum $A \oplus B$

1: Set $\mathbf{v}_{n+1} \leftarrow \mathbf{v}_1$, $\mathbf{w}_{m+1} \leftarrow \mathbf{w}_1$

2: Initialize i=1, j=1

3: **loop**

  a. Add $\mathbf{v}_i + \mathbf{w}_j$ as a vertex to $A \oplus B$

  b. **if** angle($\mathbf{v}_i\mathbf{v}_{i+1}$)<angle($\mathbf{w}_j\mathbf{w}_{j+1}$): $i \leftarrow i+1$

  else if angle($\mathbf{v}_i\mathbf{v}_{i+1}$)>angle($\mathbf{w}_j\mathbf{w}_{j+1}$): $j \leftarrow j+1$

  else: $i \leftarrow i+1$, $j \leftarrow j+1$

  end

until i=n+1 **and** j=m+1

Algorithm 2. Minkowski sum of a line and a convex polygon

Input: A line segment W with two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ and a convex polygon B with vertices $\mathbf{w}_1, \ldots, \mathbf{w}_m$

Output: The Minkowski sum $W \oplus B$

1: Set $\mathbf{v}_3 \leftarrow \mathbf{v}_1$, $\mathbf{w}_{m+1} \leftarrow \mathbf{w}_1$

2: Initialize i=1, j=1

3: **loop**

  a. Add $\mathbf{v}_i + \mathbf{w}_j$ as a vertex to $W \oplus B$

  b. **if** angle($\mathbf{v}_i\mathbf{v}_{i+1}$)<angle($\mathbf{w}_j\mathbf{w}_{j+1}$): $i \leftarrow i+1$

  else if angle($\mathbf{v}_i\mathbf{v}_{i+1}$)>angle($\mathbf{w}_j\mathbf{w}_{j+1}$): $j \leftarrow j+1$

  else: $i \leftarrow i+1$, $j \leftarrow j+1$

  end

until i=3 **and** j=m+1

When two blocks A and B are in contact (in touch or overlap), there must exist at least one point $\mathbf{p}$ that is shared by the two objects, $\mathbf{p} \in A$; $\mathbf{p} \in B$. Hence, the origin $\mathbf{o}$ must be enclosed in the Minkowski difference A $\ominus$ B:

$$\mathbf{o} \in A \ominus B, \quad \text{if A} \cap \text{B} \neq \emptyset \tag{4}$$

Consequently, the contact state of two blocks A and B is equivalent to the following statement:

$$\text{Contact state between two blocks}: \begin{cases} \mathbf{o} \notin A \ominus B & \equiv \text{ separation} \\ \mathbf{o} \in \partial(A \ominus B) & \equiv \text{ touch} \\ \mathbf{o} \in A \ominus B & \equiv \text{ overlap} \end{cases} \tag{5}$$

Figure 2 demonstrates that the relationship between the Minkowski difference of a fixed hexagon A and a moving quadrilateral B, and the origin of the Minkowski difference of the two shapes corresponds to the contact state of the two shapes.[4,33] Consequently, the contact state between the two blocks can be determined by whether or not the origin is enclosed in the corresponding Minkowski difference of the two blocks.

The Minkowski difference of a line and a polygon can also be utilised to evaluate their contact state. However, determining the contact state between a polygon and a line can be done in a simpler manner. For example, if we consider a convex polygon A with k vertices/edges and a line segment W with two vertices (as shown in Figure 3), there are three contact states that can occur between them: separation, touch, and overlap.

The distance $d(A, W)$ between A and W is defined as the minimum distance from the boundary of A to the line W:

$$d(A,W) = \min\left\{\left(\mathbf{v}_a^i - \mathbf{v}_W^s\right)\cdot \mathbf{n}_W\right\} \quad i \in (1, \ldots, k) \tag{6}$$

where $\mathbf{v}_a^i$ is the i^{th} vertex of the polygon A; $\mathbf{v}_W^s$ is the start point of the line W (it can be any point in the line W); $\mathbf{n}_W$ is the unit vector normal to the line W and points inward towards the packing domain as shown in Figure 3.

Consequently, the contact state between block A and line W is simplified to the following statement:

$$\text{Contact state between block and line}: \begin{cases} d(A,W) > 0 \equiv \text{separation} \\ d(A,W) = 0 \equiv \text{touch} \\ d(A,W) < 0 \equiv \text{overlap} \end{cases} \tag{7}$$

If the EB is convex, the contact state between block A and line W can be determined using Equations (6) and (7). However, if the EB is concave, the polygon should locate in the active area associated with the line before establishing the contact state between block A and line W. As shown in Figure 3B, the active area is defined using two lines (W_l and W_r) associated with the line W. Both W_l and W_r are vertical to W.

The distance $d(A, W_l)$ between A and W_l is defined as the minimum distance from the boundary of A to the left line W_l:

$$d\left(A,W_l\right) = \min\left\{\left(\mathbf{v}_a^i - \mathbf{v}_W^s\right)\cdot \mathbf{n}_{W_l}\right\} \quad i \in (1, \ldots, k) \tag{8}$$

where $\mathbf{n}_{W_l}$ is the unit vector normal to the line W_l formed by rotating $\mathbf{n}_W$ 90° anticlockwise as shown in Figure 3B.

The distance $d(A, W_r)$ between A and W_r is defined as the minimum distance from the boundary of A to the right line W_r:

$$d\left(A,W_r\right) = \min\left\{\left(\mathbf{v}_a^i - \mathbf{v}_W^e\right)\cdot \mathbf{n}_{W_r}\right\} \quad i \in (1, \ldots, k) \tag{9}$$

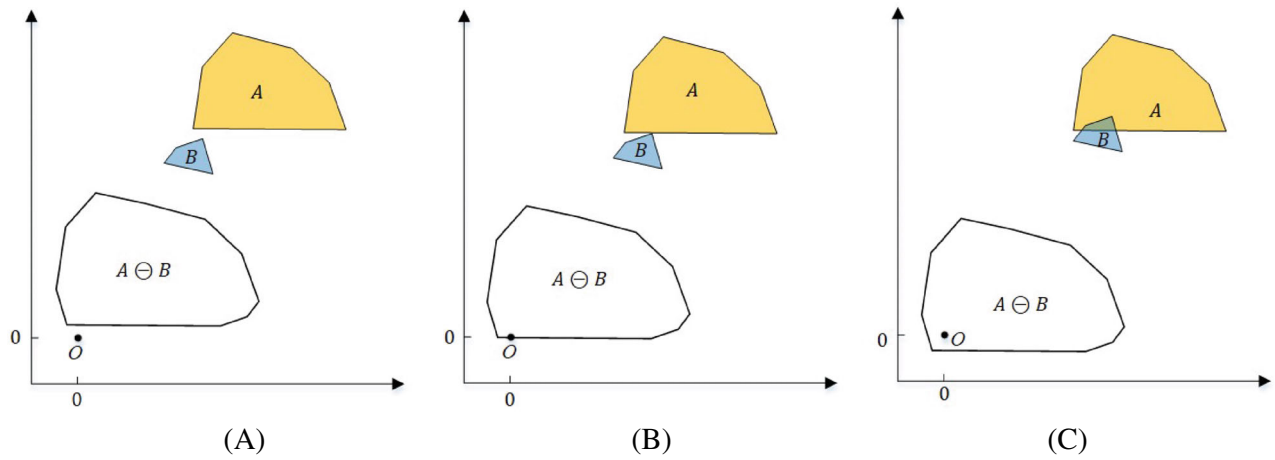

FIGURE 2 Contact states between hexagon A and quadrilateral B and their Minkowski differences: (A) separation; (B) touch; (C) overlap.

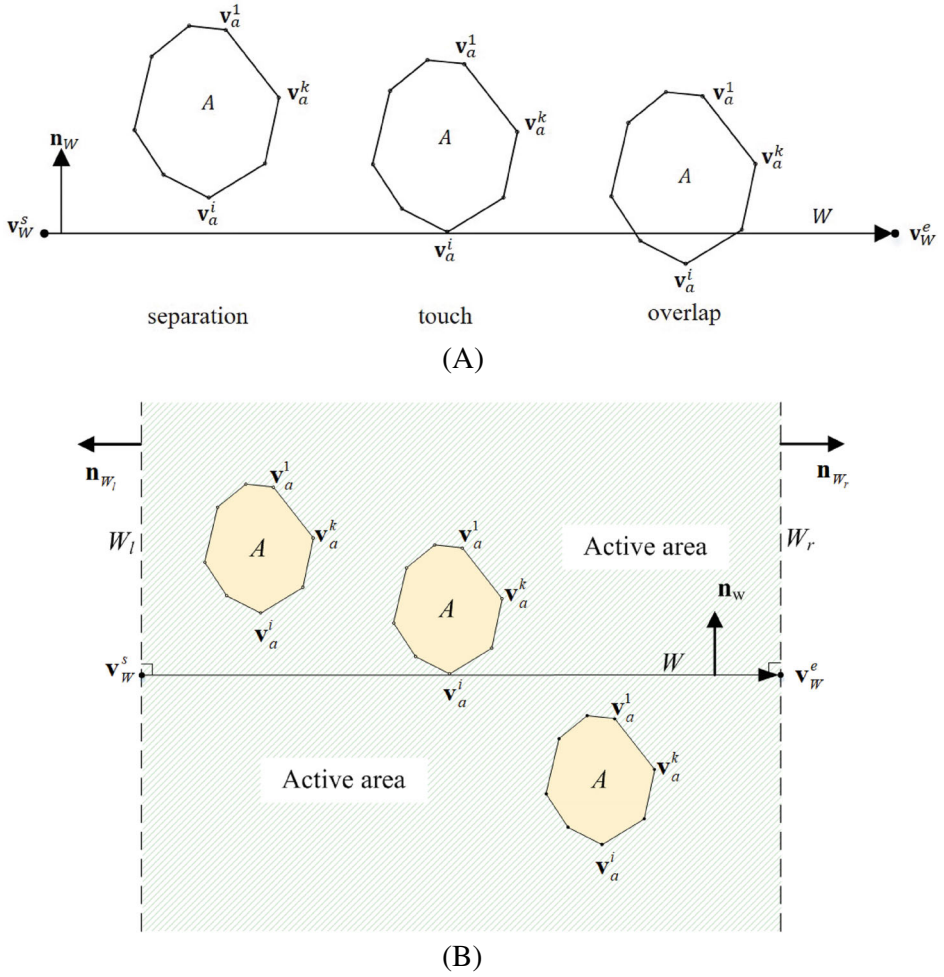

FIGURE 3 Three contact states between polygon A and line W: (A) EB is convex; (B) EB is concave.

where $\mathbf{v}_W^e$ is the end point of the line W; $\mathbf{n}_{W_l}$ is the unit vector normal to the line W_l formed by rotating $\mathbf{n}_W$ 90° clockwise as shown in Figure 3B.

The polygon A is located in the active area associated with the line W if the following condition is fulfilled:

$$\begin{cases} d(A, W_l) \leq 0 \\ d(A, W_r) \leq 0 \end{cases} \tag{10}$$

If the polygon A is located in the active area, the contact state between block A and line W can be determined using Equations (6) and (7). If a polygon is completely on the opposite side of the line, then this polygon is in overlap with the line. In this case, this polygon should be excluded as it lies outside the packing domain. Thus, this situation is considered in the present approach.

2.2 | Placing a polygon in contact with other polygons/lines based on Minkowski difference

Unlike discs, determining the position of a new polygon in contact with other polygons/lines is not straightforward. As illustrated in Figure 4, there are three different cases of placing a convex polygon in contact with other convex polygons or lines:

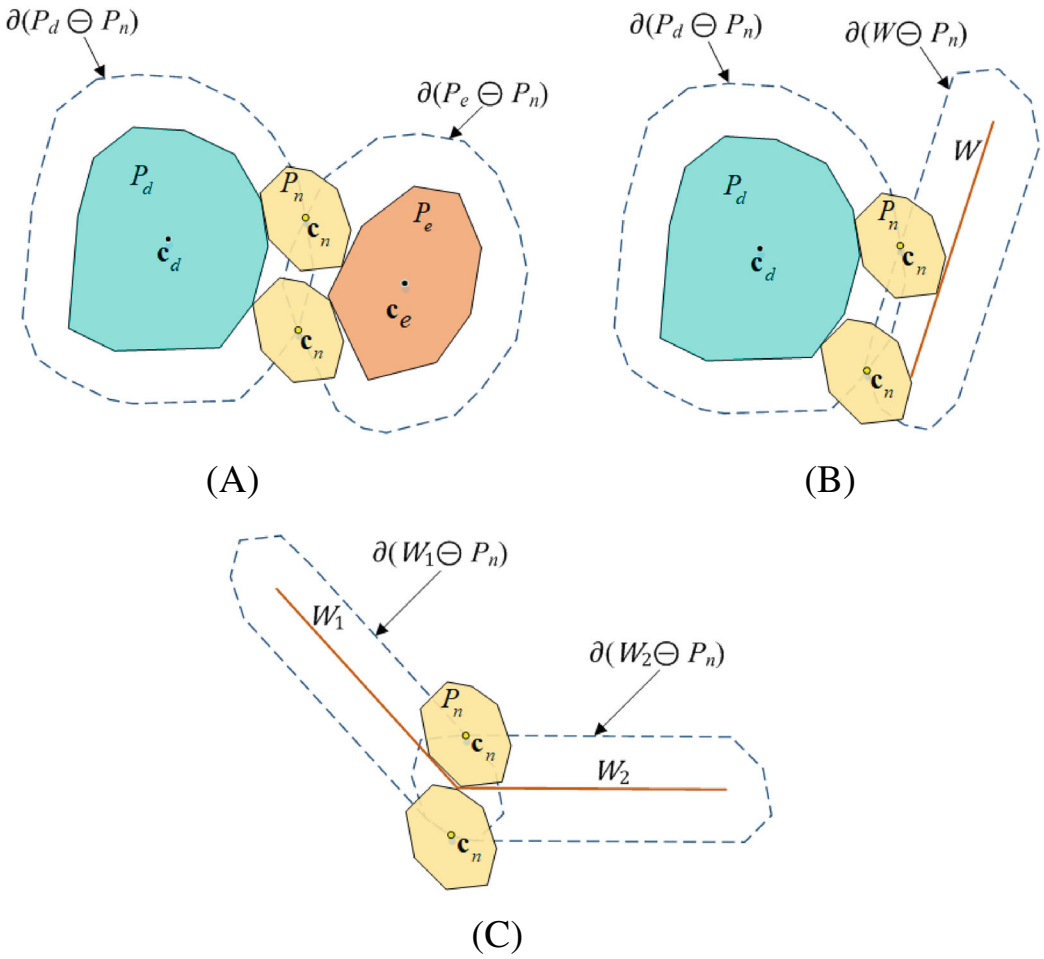

FIGURE 4 Three cases of placing polygon P_n: (A) in touch with two polygons P_d and P_e; (B) in touch with a polygon P_d and a line W; and (C) in touch with two lines W_1 and W_2.

(a) A convex polygon in contact with two convex polygons

In Figure 4A, P_d and P_e are two polygons and the position of a new polygon P_n needs to be determined so that it touches polygons P_d and P_e simultaneously. The two loci of the polygon P_n with polygons P_d and P_e are $\partial(P_d \ominus P_n)$ and $\partial(P_e \ominus P_n)$, respectively. The two possible centroids of polygon P_n are the intersection points of $\partial(P_d \ominus P_n)$ and $\partial(P_e \ominus P_n)$. This case is used in determining convex particles during outwards-direction packing as discussed in Section 3.3 and associated with the closed form.

(b) A convex polygon in contact with a convex polygon and a line

In Figure 4B, P_d and W are a polygon and a line respectively, and the position of a new polygon P_n needs to be found so that it touches polygon P_d and line W simultaneously. The two loci of the polygon P_n with P_d and W are $\partial(P_d \ominus P_n)$ and $\partial(W \ominus P_n)$, respectively. The intersection points of $\partial(P_d \ominus P_n)$ and $\partial(W \ominus P_n)$ are the potential centroids of polygon P_n. This case is often used for determining the second convex particle during the first layer packing or convex particles near IB/EB as discussed in Section 4 and associated with an open form.

(c) A convex polygon in contact with two lines

In Figure 4C, W_1 and W_2 are two lines and the position of a new polygon P_n needs to be found so that it touches lines W_1 and W_2 simultaneously. The two loci of the polygon P_n with W_1 and W_2 are $\partial(W_1 \ominus P_n)$ and $\partial(W_2 \ominus P_n)$, respectively. The intersection points of $\partial(W_1 \ominus P_n)$ and $\partial(W_2 \ominus P_n)$ are the possible centroids of polygon P_n. This case is often used for the determination of the first convex particle during the first layer packing in Section 4 and associated with the open form.

Note that convex polygons/lines (i.e., P_d, P_e, W_1 and W_2) are fixed, while the new polygon P_n is movable (but only its centroid is unknown). After the two loci of the polygon P_n with other polygons/lines are obtained, a linear algorithm proposed by Han et al.[34] is performed to search for convex polygon intersections. The determination of the centroid of a new polygon is given in Algorithm 3.

Algorithm 3. Determination of the position of a new polygon that touches two polygons/lines

Input: Given two polygons/lines P_d (or W_1) and P_e (or W_2); and a free polygon P_n generated from the reference polygon.
Output: The centroid of new polygon P_n (i.e. $\mathbf{c}_n$) that touches two polygons/lines P_d (or W_1) and P_e (or W_2)
1: Compute $P_d \ominus P_n$ and $P_e \ominus P_n$ as shown in Fig.4 using **Algorithm 1** or **2**
2: Compute the two intersections of $\partial(P_d \ominus P_n)$ and $\partial(P_e \ominus P_n)$
3: Choose the right intersection as the centroid of P_n (i.e. $\mathbf{c}_n$)

It should be noted that Du et al.[31] constructed the Minkowski difference between a polygon and a line or two lines in a more complex way. They first introduced an auxiliary point associated with the line to convert the line to a convex polygon, and then constructed the corresponding Minkowski difference. However, as stated in Section 2.1, the Minkowski difference between a polygon and a line or two lines can be easily and directly constructed. For this reason, the present approach is much more efficient to place new polygons in contact with lines.

3 | ADVANCING FRONT APPROACH: THE CLOSED FORM

3.1 | Geometric description of convex polygons

Two approaches can be used to define the template or reference shape for generating convex polygons in a geometric domain. The first approach involves random generation, while the second approach involves using realistic irregular particle images. In the latter approach, the boundary points are obtained from imaging and then made convex by extracting the convex hull. Alternatively, a predefined reference polygon can be translated, scaled, and rotated to generate all convex polygons, which can then be sequentially placed in a 2D domain.

Suppose that the reference polygon P_{ref} has k vertices/edges. The coordinates of the ith vertex of the nth generated polygon P_n (i.e., $\mathbf{v}_n^i$) ($n = 1, \ldots, \mathrm{NP}_{\max}$) are computed by:

$$\mathbf{v}_n^i = \mathbf{R}_n \mathbf{U}_n \left(\mathbf{v}_{ref}^i - \mathbf{c}_{ref} \right) \quad (i = 1, \ldots, k) \tag{11}$$

$$\mathbf{R}_n = \begin{bmatrix} \cos\alpha_n & -\sin\alpha_n \\ \sin\alpha_n & \cos\alpha_n \end{bmatrix}, \mathbf{U}_n = \begin{bmatrix} \lambda_{n1} & 0 \\ 0 & \lambda_{n2} \end{bmatrix} \tag{12}$$

where $\mathbf{R}_n$ is the rotation matrix for polygon P_n; α_n is the rotation angle in the interval $[0, 2\pi]$; $\mathbf{U}_n$ is the scaling matrix and λ_{n1} and λ_{n2} are two scaling factors; $\mathrm{NP}_{\max}$ is the maximum number of polygons need to be generated in the domain; $\mathbf{c}_{\mathrm{ref}}$ is the centroid of the reference polygon P_{ref}; and $\mathbf{v}_{\mathrm{ref}}^i$ is the coordinates of the ith vertex.

Initially, the centroids of all polygons P_n ($n = 1, \ldots, NP_{max}$) generated from the reference shape are at the origin for two reasons: (1) ($-P_n$) can be easily obtained for determining the corresponding Minkowski difference; (2) It is convenient for implementation and programming.

After the centroid of polygon P_n (i.e., $\mathbf{c}_n$) is determined using the approach proposed in Section 2.2, all vertices of polygon P_n (i.e., $\mathbf{v}_n^i$) can be updated as follows:

$$\mathbf{v}_n^i = \mathbf{c}_n + \mathbf{v}_n^i \quad (i = 1, \ldots, k) \tag{13}$$

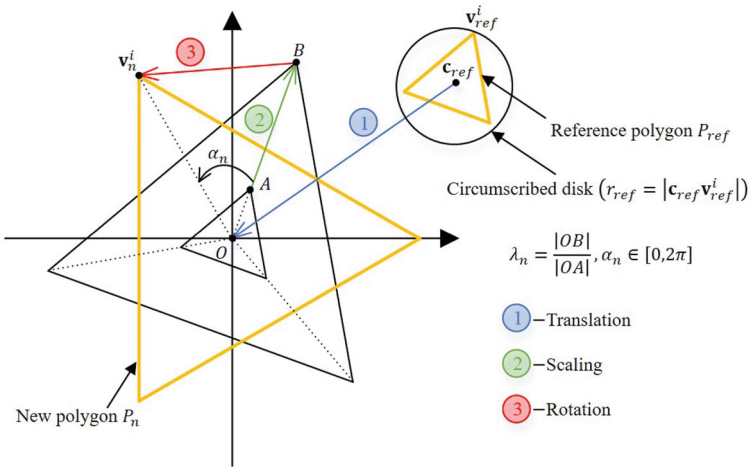

FIGURE 5 New polygon generation based on a reference polygon.

In this study, the two scaling coefficients for the length and width of the generated polygons are set to be the same within the interval $[\lambda_{\min}, \lambda_{\max}]$, that is, $\lambda_{n1} = \lambda_{n2} = \lambda_n \in [\lambda_{\min}, \lambda_{\max}]$. As a result, all polygons have an identical aspect ratio. Also assume that α_n and λ_n can be determined from a desired probability distribution function (PDF), such as the uniform, Gaussian or lognormal distribution.[19]

The process of generating a new polygon is illustrated using a triangle in Figure 5, where a circumscribed disc is also assigned to the polygon (i.e., triangle). The radii of the circumscribed discs for the reference polygon P_{ref} and the generated polygon P_n are r_{ref} and $\lambda_n r_{\text{ref}}$, respectively.

Equations (11)–(13) show that a polygon P_n, denoted as $P_n\left(\mathbf{c}_n, \mathbf{c}_{\text{ref}}, \mathbf{v}^i_{\text{ref}}, \alpha_n, \lambda_n\right)$, can be described by three spatial position descriptors $(\mathbf{c}_n, \mathbf{c}_{\text{ref}}, \mathbf{v}^i_{\text{ref}})$ and two shape descriptors (α_n, λ_n). As $\mathbf{c}_{\text{ref}}$ and $\mathbf{v}^i_{\text{ref}}$ are pre-defined for the reference P_{ref}, and the rotation angle (α_n) and scaling factor (λ_n) of the polygon P_n also follow a pre-defined PDF, only $\mathbf{c}_n$ is unknown. Therefore the polygon can be simply denoted as $P_n(\mathbf{c}_n)$.

3.2 | The initial front generation

The first step of AFA is to form an initial front, which is a group of polygons or lines in the 2D domain. Then, an active segment is selected, and a new polygon is placed to make contact with the polygons or lines associated with the active front, aiming to achieve a high packing density. After placing the new polygon, the front is advanced, and the process is repeated by selecting a new active front. This procedure continues until the entire 2D domain is filled with non-overlapping polygons.[16]

In the present study, both IB and EB are considered (Figure 6), and the corresponding initial fronts are formed in different ways. The domain boundary is first decomposed into straight lines composed of one or more segments and arranged in an anti-clockwise sense. When the shape of IB or EB is circular, elliptical or super-quadric, it can be approximated by a convex polygon through an adaptive sampling algorithm proposed by Han et al.[34] For an IB with super-quadric shape, its surface is expressed as:

$$\left(\frac{x}{a}\right)^m + \left(\frac{y}{b}\right)^m - 1 = 0 \tag{14}$$

where a, b and m are three positive numbers, which determine the shape and size of the super-quadric. An ellipse is obtained when $m = 2$. Many other shapes can be obtained by varying m. When $m < 1$, the super-quadric is concave, which is not considered here.

3.2.1 | Without consideration of IB

When IB is not considered, the first three polygons, denoted as P_1, P_2 and P_3, are placed tangent to each other in the centre of the 2D domain, as shown in Figure 7A. The position of the third polygon P_3 can be determined using the method

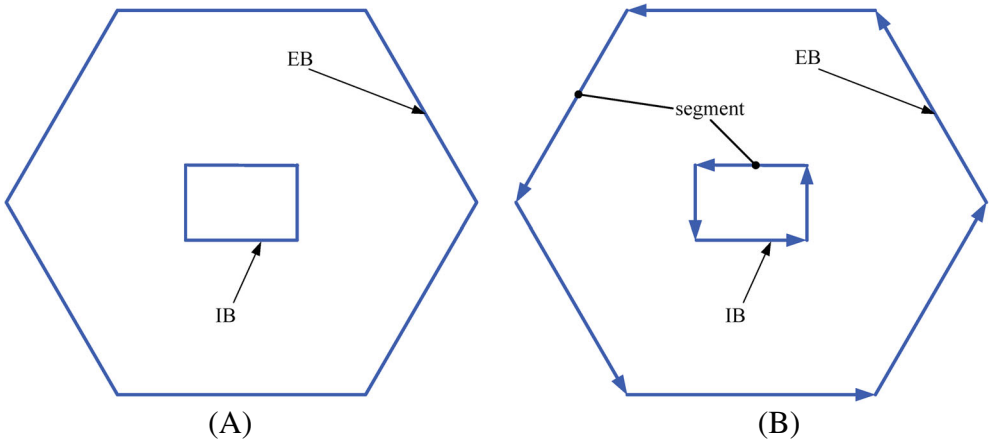

FIGURE 6 The boundaries of a planar domain with an interior cavity: (A) before discretised; (B) after discretised.

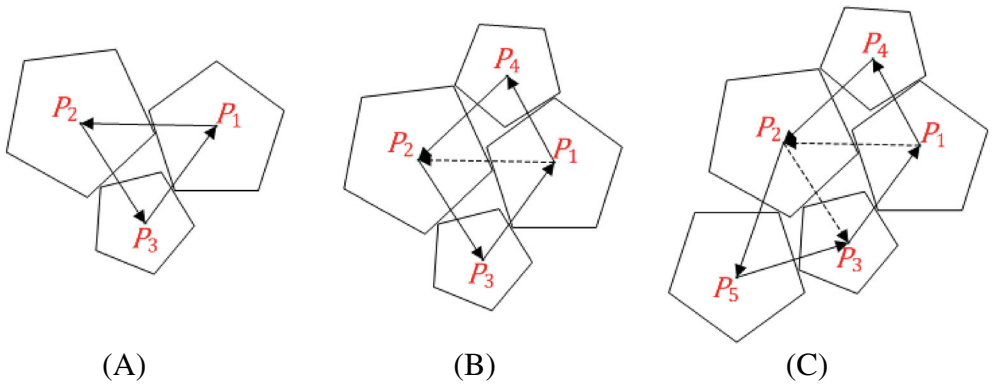

FIGURE 7 Initial front without IB: (A) first three polygons and initial front; (B) generation of polygon 4 and updated front; (C) generation of polygon 5 and updated front.

shown in Figure 4A as explained in Section 2.2. However, polygons P_1, P_2 and P_3 should form an anti-clockwise cycle, as in the case with discs in Feng et al.[16] Then $P_1 \to P_2 \to P_3 \to P_1$ form the initial front (Figure 7A).

3.2.2 | With consideration of IB

When IB is considered, the initial front is more complex. In this situation, the IB can be considered as a convex polygon. If the IB is formed from a circular solid region as shown in Figure 8A, the region is first approximated as a convex polygon.

To initiate the AFA process with IB, we designate the IB and the first polygon as P_0 and P_1, respectively. The position of the second polygon P_2 is determined using the method outlined in Section 2.2 to establish an initial front $P_0 \to P_1 \to P_2 \to P_0$. Subsequently, the last segment $P_2 \to P_0$ of the front is selected as the first active segment, as depicted in Figure 8A, to determine the third polygon P_3. It is worth noting that the third polygon P_3 must be positioned on the right-hand side of the segment, as shown in Figure 8B. As more polygons are generated, the front may be extended to $P_0 \to P_1 \to P_2 \to \cdots \to P_0$, as shown in Figure 8C. The initial front is considered complete when a polygon, such as P_{22}, touches the first polygon P_1. At this point, the front becomes $P_1 \to P_2 \to P_3 \to \cdots \to P_{22} \to P_1$, and the IB P_0 is removed from the front, as shown in Figure 8C.

Because the generated polygons forming the initial front conform to the IB, the problem of tangency between the generated convex non-circular particles and the IB is naturally resolved. This eliminates both the non-smoothness of particle packings and the relatively large gaps that would have been present near the IB as shown in Figure 8C.

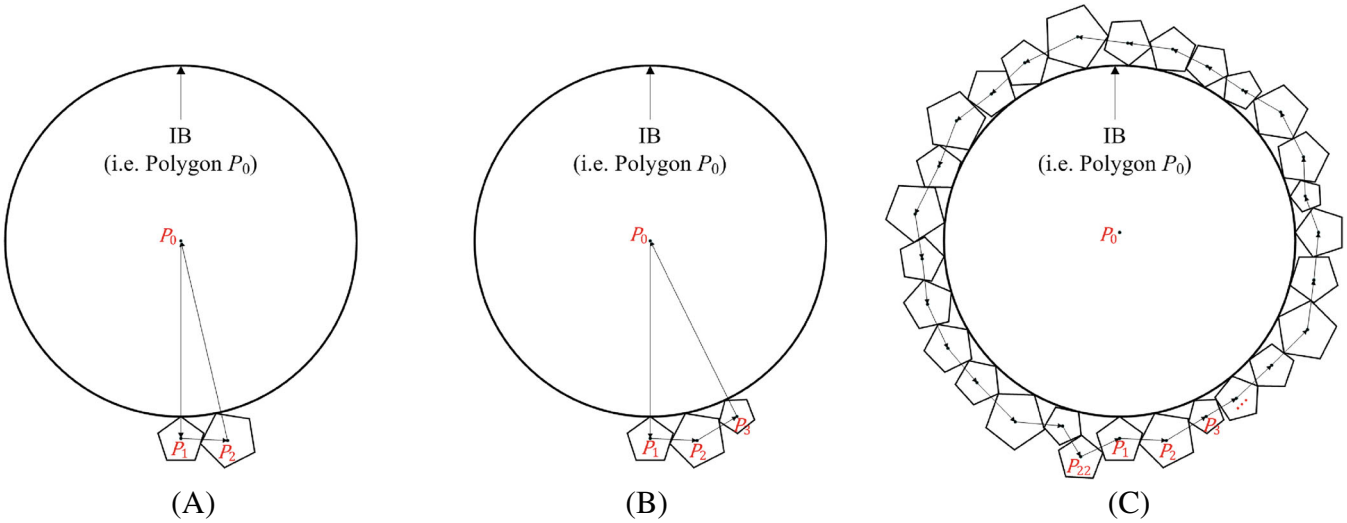

FIGURE 8 Initial front with IB: (A) The first two polygons; (B) generation of polygon 3; (C) initial front.

3.3 | New polygon generation and front update

Once the initial front has been formed, the domain can be filled by generating new polygons and incrementally advancing the front outward. The directions of the frontal segments are defined to ensure that any newly generated polygon is placed on the right-hand side when moving along the positive directions of the segments. This procedure of front update for convex noncircular particles is similar to the method used for discs in Feng et al.[16] For completeness, a brief summary of the front update procedure is provided below.

The initial front is updated by incrementally advancing it in the outward direction to generate new polygons and fill the domain. The first segment $P_1 \rightarrow P_2$ in the initial front without IB is chosen as the current active front. Then, a new polygon P_4 is generated and placed in contact with both P_1 and P_2, while lying on the right-hand side of the segment $P_1 \rightarrow P_2$, as described in Section 2.2. The initial front is updated by deleting the segment $P_1 \rightarrow P_2$ and insert two new segments $P_1 \rightarrow P_4$ and $P_4 \rightarrow P_2$ to form a new front $P_1 \rightarrow P_4 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$. The process is then repeated by selecting the segment $P_2 \rightarrow P_3$ as the next active front, and generating a new polygon P_5, as shown in Figure 7B,C.

However, it is possible that a newly generated polygon from an active segment may overlap with other existing polygons, resulting in four different cases as illustrated in Figure 9. To address this issue, a set of front updating rules for discs was proposed by Feng et al.[16] A similar methodology is adopted here for packing convex particles. More details can be found in Feng et al.[16]

3.4 | Treatment of polygons near EB

During the outwards-direction packing, some generated polygons may overlap with the EB. Discarding such polygons can result in large gaps near the EB, leading to non-smooth packing. To address this, the position and size of an overlapping polygon should be adjusted so that it becomes tangent to the EB.

Figure 10A illustrates the distance d_n between the centroid of the new polygon P_n ($\mathbf{c}_n$, λ_n) and the boundary line B_t. If $d_n < \lambda_u r_{ref}$, then the new polygon may overlap with the boundary. To ensure tangency, a new polygon $P_n(\mathbf{c}, \lambda)$ is generated while maintaining the rotation angle α_n, but with adjusted size λ and position $\mathbf{c}$. The adjustment is achieved by solving the following problem:

$$\begin{cases} \mathbf{o} \in \partial\left(P_e \ominus P_n(\mathbf{c}, \lambda)\right) \equiv P_n \text{ touch } P_e \\ \mathbf{o} \in \partial\left(P_e \ominus P_n(\mathbf{c}, \lambda)\right) \equiv P_n \text{ touch } P_d, \quad \lambda \in [\lambda_l, \lambda_u] \\ d\left(P_n(\mathbf{c}, \lambda), B_t\right) = 0 \quad \equiv P_n \text{ touch } B_t \end{cases} \tag{15}$$

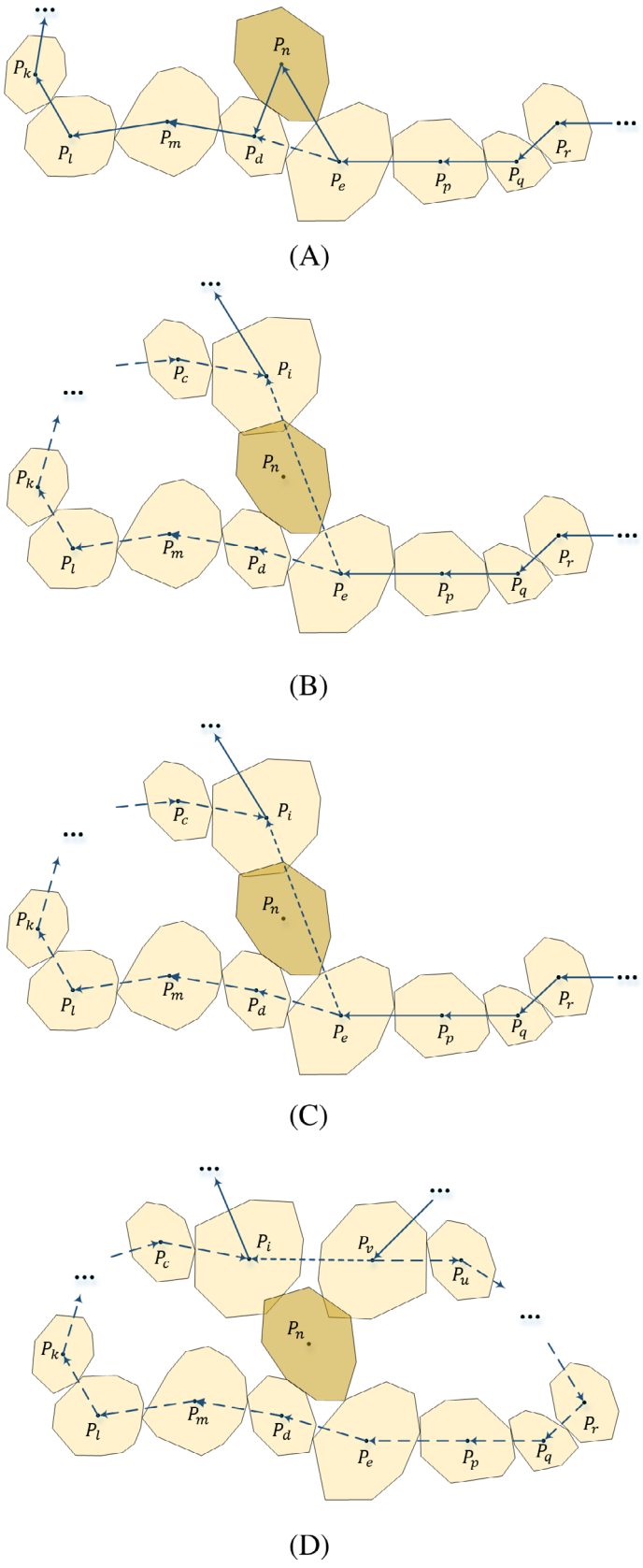

FIGURE 9 Four possible cases after a new polygon P_n is generated from the current front segment: (A) No overlapping; (B) Overlapping only with a polygon on the subsequent front; (C) Overlapping only with polygon on the preceding front; (D) Overlapping with polygons both on the preceding and subsequent fronts.

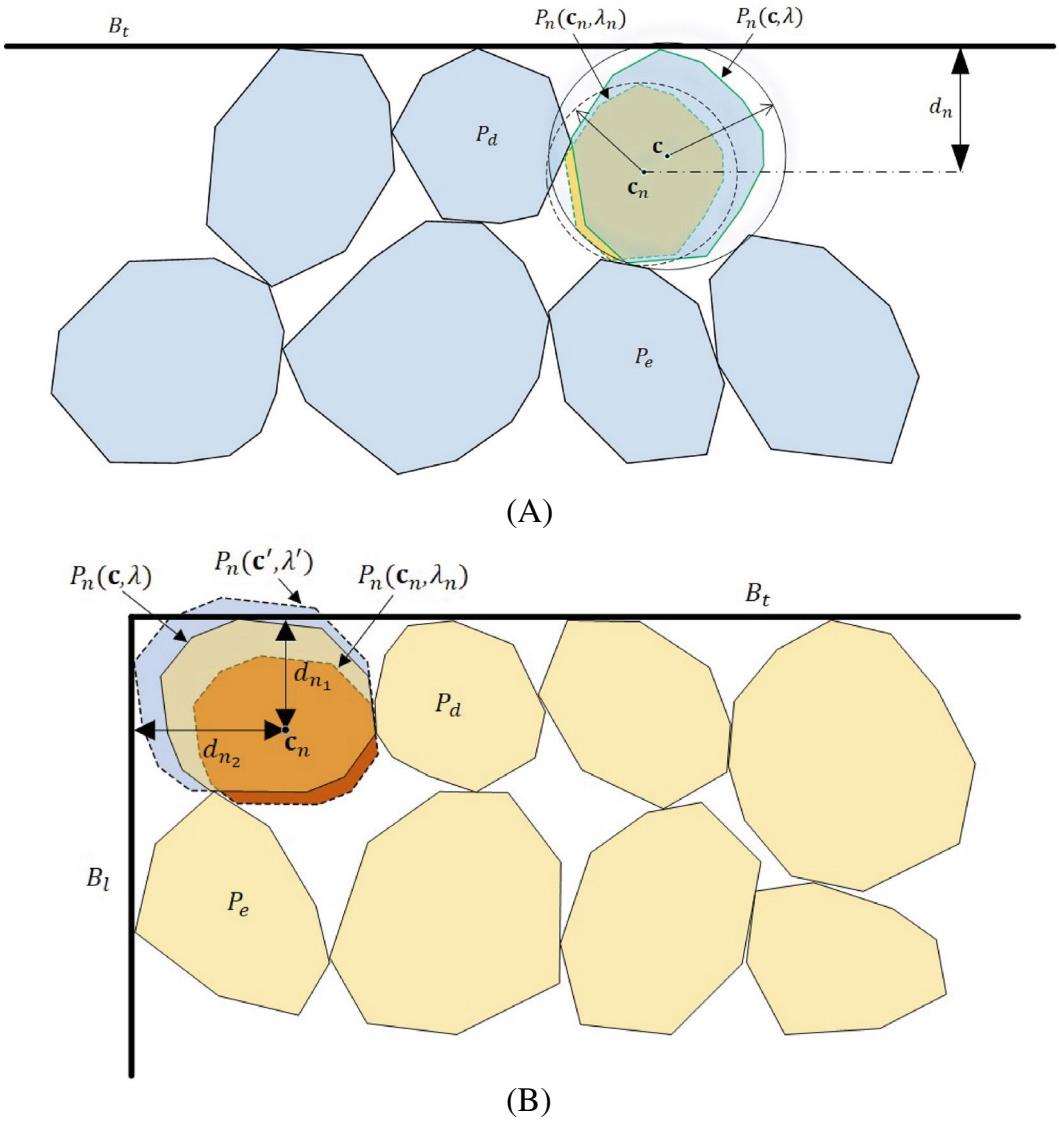

FIGURE 10 Treatment of polygons near EB: (A) single boundary; (B) corner.

However, there are two extremely cases that need to be considered: (1) If the largest polygon (its scaling coefficient is λ_u) is in separation with the boundary B_t, that is, $d\left(P_n\left(\mathbf{c},\lambda_u\right),B_t\right)>0$, another same sized polygon $P_n\left(\mathbf{c},\lambda_u\right)$ is generated and placed near the boundary B_t; and (2) If the smallest polygon (its scaling coefficient is λ_l) overlaps with the boundary B_t, that is, $d\left(P_n\left(\mathbf{c},\lambda_l\right),B_t\right)<0$, the polygon is rejected and no polygon is placed at this place.

Except for the above two extremely cases, the contact state between a newly generated polygon P_n (with scaling coefficient λ_n) and the EB line B_t can be used to further narrow the range for the lower and upper bounds as follows:

$$\lambda\in\begin{cases}\left[\lambda_n,\lambda_u\right], & d\left(P_n\left(\mathbf{c}_n,\lambda_n\right),B_t\right)>0\\ \lambda_n, & d\left(P_n\left(\mathbf{c}_n,\lambda_n\right),B_t\right)=0\\ \left[\lambda_l,\lambda_n\right], & d\left(P_n\left(\mathbf{c}_n,\lambda_n\right),B_t\right)<0\end{cases}\tag{16}$$

To solve Equations (15) and (16), a bisection method for determining the polygon size and position near the EB are implemented in this work, and the algorithm is presented in Algorithm 4.

Algorithm 4. Bisection method for determining polygon size and position near EB

Input: Given two polygons P_e and P_d and the nearest EB line W; the newly generated polygon $P_n(\mathbf{c}_n, \lambda_n)$ that touches P_e and P_d; tolerances τ_1 and τ_2; Initial lower bound and upper bound for λ: λ_l and λ_u

Output: Size and position of a new polygon $P_n(\mathbf{c}, \lambda)$

1: Set $a \leftarrow \lambda_l$, $b \leftarrow \lambda_u$, k=1

2: Narrow the lower and upper bounds for λ

a. Compute the distance d_1 between $P_n(\mathbf{c}_n, \lambda_n)$ and W using Eq. (6)

b. **if** (abs(d_1)< τ_1): **output** $P_n(\mathbf{c}_n, \lambda_n)$, **stop**

else if (d_1< - τ_1) **then**

i. Set $b=\lambda_n$

ii. Determine the centroid of polygon $P_n(\mathbf{c}, \lambda_l)$ using **Algorithm 3**

iii. Compute the distance d_2 between $P_n(\mathbf{c}, \lambda_l)$ and W using Eq. (6)

iv. **if** (d_2<- τ_1): **reject** $P_n(\mathbf{c}, \lambda_l)$, **stop**

else if (abs(d_2)< τ_1): **output** $P_n(\mathbf{c}, \lambda_l)$, **stop**

else if (d_2> τ_1): $a=\lambda_l$, and GOTO 3

end

else if (d_1> τ_1) **then**

i. Set $a=\lambda_n$

ii. Determine the centroid of polygon $P_n(\mathbf{c}, \lambda_u)$ using **Algorithm 3**

iii. Compute the distance d_2 between $P_n(\mathbf{c}, \lambda_u)$ and W using Eq. (6)

iv. **if** (d_2<- τ_1): $b=\lambda_u$, and GOTO 3

else: **output** $P_n(\mathbf{c}, \lambda_u)$, **stop**

end

end

3: **loop**

a. Calculate $\lambda=(a+b)/2$

b. Determine the centroid of polygon $P_n(\mathbf{c}, \lambda)$ using **Algorithm 3**

c. Computed the distance d_3 between $P_n(\mathbf{c}, \lambda)$ and W using Eq. (6)

d. **if** (abs(d_3) < τ_1): **output** $P_n(\mathbf{c}, \lambda)$, **stop**

else if (d_3< - τ_1): $b=\lambda$

else: $a=\lambda$

end

e. $k \leftarrow k+1$

until abs(b-a)< τ_2

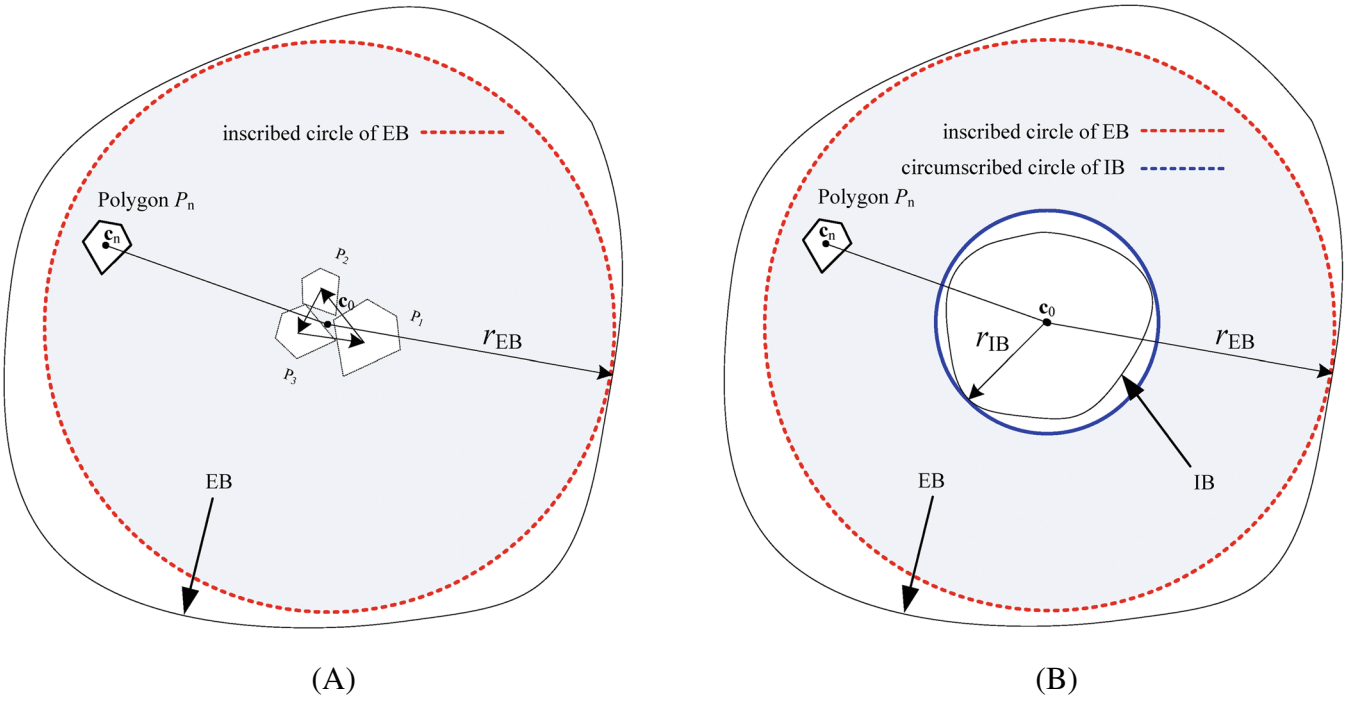

FIGURE 11 Inscribed and circumscribed IB/EB: (A) Without IB; (B) With IB.

When generating polygons near the boundary EB, it is important to check if the new polygon overlaps with other boundary lines, especially at the corners. As shown in Figure 10B, the new polygon $P_n\left(\mathbf{c}', \lambda'\right)$ may be closest to one boundary (e.g., B_l) before treatment. After determining the size and position of the polygon, it may be tangent to that boundary but overlap with other boundaries (e.g., B_t), as shown in Figure 10B. In this case, a new polygon $P_n\left(\mathbf{c}_n, \lambda_n\right)$ should be re-generated based on the overlapping boundary. This check for overlapping boundaries should be repeated until the new polygon does not overlap with any other boundaries.

After generating each new polygon, it is necessary to check whether it is near the EB or not. However, since most of the polygons (maybe 90% or more) are not near the EB, a significant amount of computational time can be wasted in this process. To address this issue, a simple yet effective scheme is proposed as shown in Figure 11. The scheme involves introducing two circles: the inscribed circle of EB and the circumscribed circle of IB. If there is no IB, the point $\mathbf{c}_0$ is selected as the geometric centre of the initial triangle $\Delta P_1P_2P_3$ (front). Otherwise, the point is chosen as the centre of the circumscribed circle of the IB. The radius of the inscribed circle of EB (i.e., r_{EB}) is defined as the minimum distance from the point to the exterior boundary W:

$$r_{EB} = \min\left\{\left(\mathbf{c}_0 - \mathbf{v}^s_{W_i}\right)\cdot \mathbf{n}_{W_i}\right\} \quad (i = 1, \ldots, k) \tag{17}$$

where $\mathbf{v}^s_{W_i}$ is the start point of the boundary line W_i (it also can be any point in the line W_i); $\mathbf{n}_{W_i}$ is the unit vector normal to the boundary line W_i pointing to inner domain; k is the total number of the boundary lines associated with EB.

The treatment for a newly generated polygon near the EB is active if the following criterion is satisfied:

$$|\mathbf{c}_0\mathbf{c}_n| + \lambda_u r_{ref} \geq r_{EB} \tag{18}$$

where $|\mathbf{c}_0\mathbf{c}_n|$ is the distance between the centroid of the new polygon P_n (i.e., $\mathbf{c}_n$) and the point $\mathbf{c}_0$.

Equation (18) reveals that a new polygon fully contained within the inscribed circle of the EB requires no further checks for proximity to the EB. With this approach, the algorithm can rapidly determine if a polygon is near the EB or not, without having to perform individual polygon checks. Consequently, this strategy significantly enhances the computational efficiency of the algorithm.

3.5 | Further issues

3.5.1 | Overlap checking of two polygons

It is worth noting that checking for overlaps between polygons is a computationally intensive process, as it must be conducted for each new polygon with respect to all the existing polygons on the front. Therefore, the overlap checking algorithm must be as efficient as possible. In this study, we have utilised four different algorithms for this purpose:

1. The simplest algorithm is the direct search algorithm (DSA), which sequentially checks the intersection of each edge of one polygon with each edge of the other polygon.
2. The linear algorithm (LA), initially proposed by O'Rourke et al.[35] and further enhanced by Han et al.,[34] uses two 'bugs' to chase each other for determining the intersection points of two convex polygons. The detailed introduction of the linear algorithm can be found in Han et al.[34]
3. The Minkowski difference algorithm (MDA) constructs the Minkowski difference of two convex polygons explicitly first, and then checks whether the origin is in the corresponding Minkowski difference or not. It is straightforward to use MDA for conducting overlap detection between convex particles.[31]
4. The Gilbert-Johnson-Keerthi (GJK) proposed by Gilbert et al.[36] does not need to construct Minkowski difference of two convex polygons explicitly, but check if the origin is enclosed in the Minkowski difference through iterations using support points of the polygons. The detail of the GJK algorithm can be found in Feng and Tan[32,33] and is also outlined in Algorithm 5 for the completeness.

Algorithm 5. The GJK algorithm

Input: Two convex polygons A and B
Output: Contact state between A and B

1: Initialization

a. Set Simplex **S** to be empty: $\mathbf{S}=\emptyset$

b. Choose an (arbitrary) initial search direction $\mathbf{v}\neq 0$ (It is suggested to be the joint vector of the centroids of objects A and B)

2: Iteration

a. Compute the support point $\mathbf{a}=\mathbf{p}_s(A\ominus B, \mathbf{v})$

b. If $\mathbf{a}\cdot\mathbf{v}<0$, $\mathbf{S}=\emptyset$, **output** (**NO CONTACT**), **stop**

c. Update Simplex **S**

i. Add **a** to **S** as a new vertex

ii. If **S** contains the origin **o**, **output** (**CONTACT**), **stop**

iii. Reduce **S** to the lowest dimension possible that still contains the closest feature to the origin by discarding vertices

iv. Compute a new search direction **v** that is toward the origin normal to the new simplex; and GOTO 2.a

In order to compare the performance of these four overlap-checking algorithms, a large number of packing cases with up to 5 million convex polygons of different shapes are employed to examine their computational costs and the results are depicted in Figure 12. The efficiency of these four algorithms is in this order: GJK > DSA > LA > MDA when the number of vertices is lower than 5, and GJK > LA > DSA > MDA when the number is greater than 5. Obviously, GJK is the most efficient, while MDA, which is independent of polygonal shapes, is the worst. Most importantly, the packing speed using GJK is about 2 times faster than that of MDA. Thus, GJK is used to speed up the overlapping checking for new generated polygons with the existing polygons on the front. Also note that the actual computational efficiency of each algorithm is sensitive to the number of vertices of polygons.

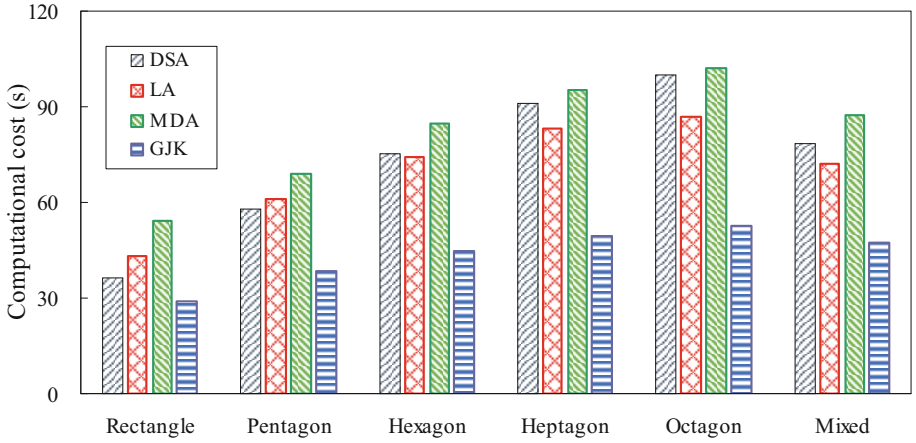

FIGURE 12 Performance comparison for packing 5 million polygons using four different overlap checking algorithms.

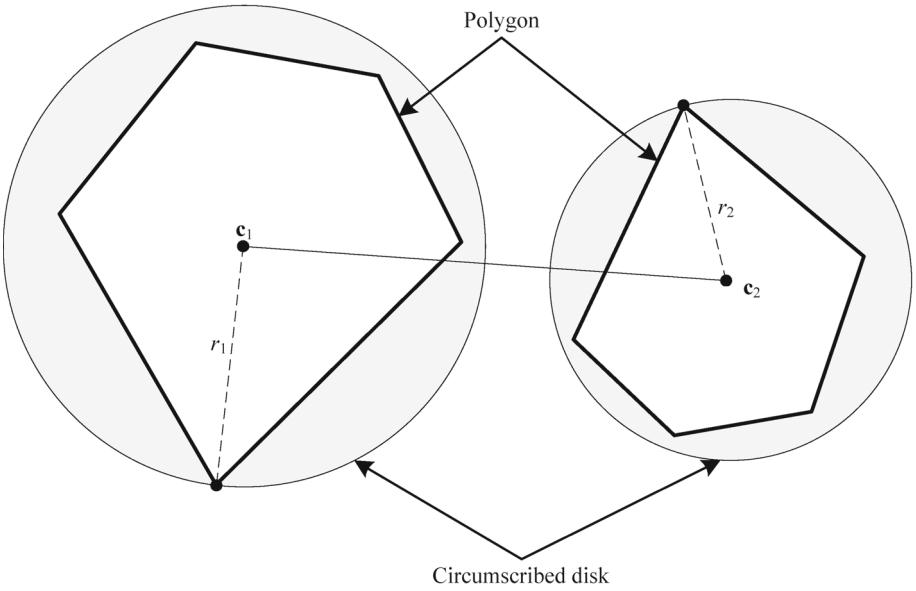

FIGURE 13 Rough contact detection between convex polygons using circumscribed disk.

Furthermore, prior to any overlap checking using GJK, a broader contact detection can be used to exclude those newly generated polygons that cannot be in contact with the existing polygons, thereby further improving the performance of the overlap check. As described in Section 3.1, the circumscribed disc is assigned to each polygon. So if the distance between the centroid of two polygons is greater than the sum of the two radii of their circumscribed discs (i.e., $|\mathbf{c}_1\mathbf{c}_2| > r_1 + r_2$), they can be excluded from further overlap checking, as shown in Figure 13.

3.5.2 | The value of λ_l and λ_u

λ_l and λ_u are the lower and upper bounds of that variable λ that controls the size of polygons near the EB. Thus, the value of λ_u should be moderate. On the other hand, to avoid unwanted small polygons generated near the EB, if the scaling coefficient of a new polygon is less than λ_l, the polygon should be abandoned. In the present study, $\lambda_l = \lambda_{\min}$, $\lambda_u = \lambda_{\max}$. In this situation, all generated polygons in the domain, including those generated near the EB, are within the range $[\lambda_{\min}, \lambda_{\max}]$. Other lower and upper bounds can also be chosen by users.

The complete procedure for generating convex noncircular particles in a complex domain with IB/EB using the closed form is given in Algorithm 6.

Algorithm 6. The complete procedure for closed form with IB/EB

Input: Reference polygon P_{ref}; IB/EB geometry; Lower bound and upper bound for λ: $\lambda_{\min}$ and $\lambda_{\max}$; PDF for λ; Lower bound and upper bound for λ near EB: λ_l and λ_u; The number of polygons to be generated in the domain $NP_{\max}$

Output: The total number of generated polygons NP, and their positions and sizes (i.e. $P_n(\mathbf{c}_n,\lambda_n)$) ($n$=1,…, NP)

1: Compute all polygons $P_n(\mathbf{o}, \lambda_n)$ (n=1,…, $NP_{\max}$) based on the reference polygon P_{ref} using Eqs. (11) and (12)

2: Discretise IB/EB into a set of connected segments as shown in Fig.6

3: Form the initial front

if (IB is not considered) **then**

Generate three contacting polygons in the centre of the domain as shown in Fig.7(a)

else

Generate a array of smoothly placed polygons which conforms to the IB shape using **Algorithm 3** as shown in Fig. 8(c)

end

4: **loop** the active segments of the present front

a. Select an active front and generate a new polygon P_n which is in contact with other two polygons/lines in the front using **Algorithm 3**

b. Check if the polygon at this position is near EB

i. Find the closest EB line to the new polygon P_n to be generated

ii. Compute the distance d_n between the centroid of the polygon and the closest EB line

iii. **if** (d_n<$\lambda_u r_{ref}$) **then**

① Re-determine this polygon size and position using **Algorithm 4**

② Check if the new polygon overlaps with other EB lines, if (overlap): GOTO ① using the present overlap boundary line

end

c. Check if the polygon at this position overlaps with any existing polygon on the front using **Algorithm 5**

if (no overlap): Accept the new polygon, and GOTO 4.a

else: Reject the new polygon, update front as shown in Fig. 9, and GOTO 4.a

end

until all segments in the front are inactive

4 | ADVANCING FRONT APPROACH: THE OPEN FORM

4.1 | Packing first layers of polygons

The open form is different from the closed form as it generates polygons from the bottom of the boundary layer by layer and from the left to right corner. A rectangular domain with four boundaries (left B_l, bottom B_b, right B_r and top B_t) is considered in this section as an example. The initial front is $B_l \rightarrow B_b \rightarrow B_r$, and $B_l \rightarrow B_b$ is selected as the first active front.

As B_l and B_b are two lines, the first polygon P_1 can be generated using the approach shown in Figure 4C in Section 2.2. After P_1 is generated, the front becomes $B_\mathrm{l} \to P_1 \to B_\mathrm{b} \to B_\mathrm{r}$. Then $P_1 \to B_\mathrm{b}$ is chosen as the next active front. The second polygon P_2, which is tangent both to the first polygon P_l and the boundary line B_b, can be generated using the approach shown in Figure 4B in Section 2.2. Accordingly, the front is now updated as $B_\mathrm{l} \to P_1 \to P_2 \to B_\mathrm{b} \to B_\mathrm{r}$.

The last polygon in the first layer, for instance, polygon P_i as marked in brown in Figure 14A, should be in contact with a previous polygon P_n and the bottom wall B_b, but is in overlap with the right wall B_r. In this case, the polygon P_i should be placed to touch P_n and the left wall B_r, as the one marked in yellow in Figure 14A, to complete the first layer of polygons. The corresponding front becomes $B_\mathrm{l} \to P_1 \to P_2 \to \cdots \to P_i \to B_\mathrm{r}$ and the bottom wall B_b has been removed from the front.

However, the first layer generated in this way can cause the one-sided lifting problem as shown in Figures 14A and 15, with the following two issues: (1) the particles near the right boundary will continue to grow in the subsequent packing steps, as shown in Figure 9B, which further increases the computing cost and decreases the packing density; and (2) a potential discontinuity from the bottom corner of one side to the top corner of the other side may be encountered in the packing of mono-sized polygons without rotation. A similar phenomenon has been reported in disc packings.[26]

To overcome the problem, the following scheme is proposed. If a polygon P_n $(\mathbf{c}_n, \lambda_n)$ is near the right boundary B_r (see Figure 14B), its size λ and position $\mathbf{c}$ should be re-determined by satisfying the following conditions:

$$\begin{cases} \mathbf{o} \in \partial\left(P_e \ominus P_n(\mathbf{c}, \lambda)\right) & \equiv \ P_n \ \text{ touch } \ P_e \\ d\left(P_n(\mathbf{c}, \lambda), B_r\right) = 0 & \equiv \ P_n \ \text{ touch } \ B_r, \ \ \lambda \in [\lambda_\mathrm{l}, \lambda_u] \\ d\left(P_n(\mathbf{c}, \lambda), B_b\right) = 0 & \equiv \ P_n \ \text{ touch } \ B_b \end{cases} \tag{19}$$

Equation (19) is similar to Equation (15), except that the polygon P_d is replaced by the right wall B_r. Then Algorithm 4 can be used to determine the size and position of the polygon near the right boundary. After this treatment, the right-sided lifting problem can be resolved as shown in Figures 14B and 15.

4.2 | Packing subsequent layers of polygons

The second and third layers can be built upon the first layer following a similar procedure, as shown in Figure 15B,C. However, each newly generated polygon must be checked for possible overlap with all existing polygons on the front.

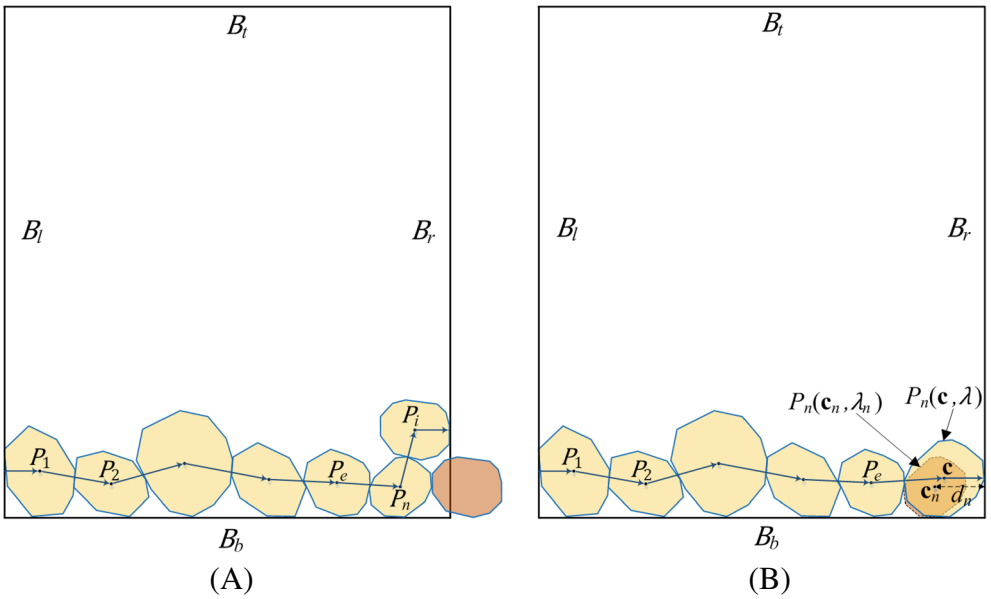

FIGURE 14 Treatment of polygons near right boundary: (A) before treatment; (B) after treatment.

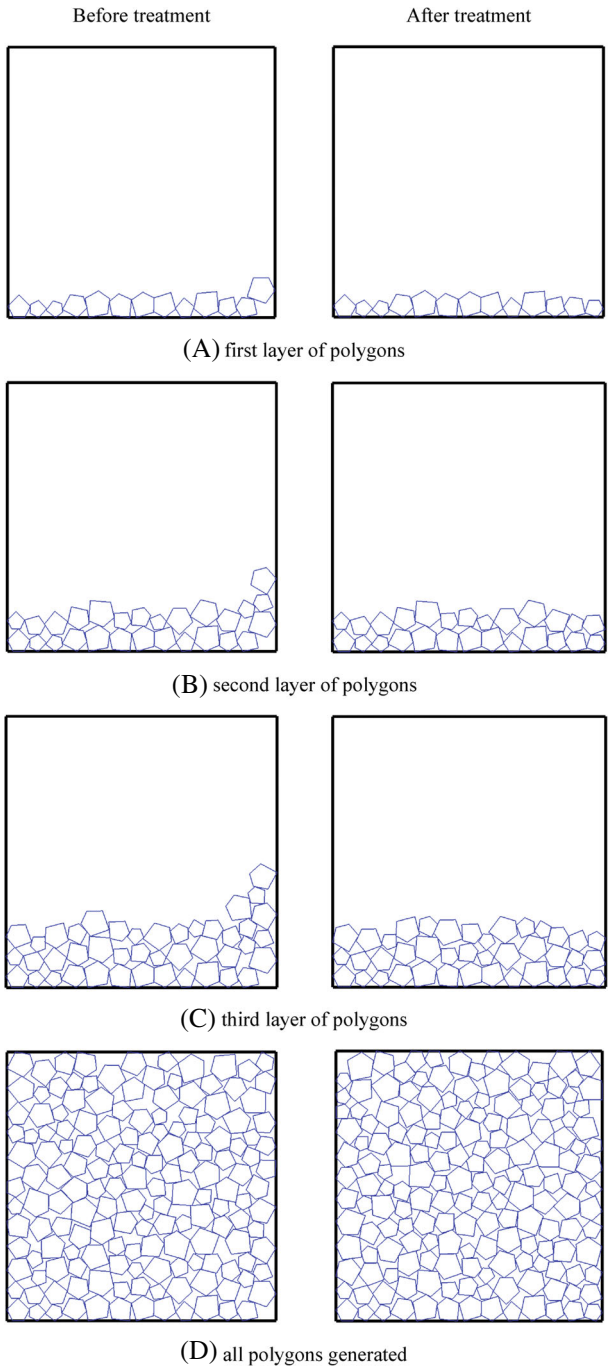

FIGURE 15 Advancing front-open form.

The same four cases as described in Section 3.3 can occur, and the same actions must be taken, as shown in Figure 9. Figure 15B,C also demonstrate that the number of polygons on the front is nearly the same as when using the proposed algorithm to solve the one-sided lifting issue.

Successive layers can be placed in a similar fashion. It should be noted that this study takes the top and right boundaries into account. When a polygon is near either boundary, its size and position need to be re-determined using Algorithm 4. Specifically, after treatment of polygons near the right boundary, the packing densities with the final assembly (Figure 15D) increases from 0.72 to 0.74. This indicates that the packing density can be further improved through treatment. The one-sided lifting issue can create a 'void' or 'hole' in the domain, which reduces the packing density.

Algorithm 7 outlines the complete procedure for generating convex noncircular particles in complex domains using the open form with EB.

Algorithm 7. The complete procedure of the open form with EB

Input: Reference polygon P_{ref}; EB geometry; Lower bound and upper bound for λ: $\lambda_{\min}$ and $\lambda_{\max}$; PDF for λ; Lower bound and upper bound for λ near EB: λ_l and λ_u; The number of polygons to be generated in the domain $NP_{\max}$

Output: The total number of generated polygons NP, and their positions and sizes (i.e. $P_n(\mathbf{c}_n,\lambda_n)$) ($n$=1,…, NP)

1: Compute all polygons $P_n(\mathbf{o},\lambda_n)$ (n=1,…, $NP_{\max}$) based on the reference polygon P_{ref} using Eqs. (11) and (12)

2: Discretise EB into a set of connected segments as shown in Fig.6

3: Packing the first layer of polygons

a. Form the initial front with three exterior boundaries (i.e. $B_l \rightarrow B_b \rightarrow B_r$) as shown in Fig.14

b. **loop** the present front

i. Select an active front and generate a new polygon P_n which is in contact with other two polygons/lines in the front using **Algorithm 3**

ii. Compute the distance d_n between the centroid of the polygon and the right boundary line

iii. **if** ($d_n<\lambda_u r_{ref}$) **then**

① Re-determine this polygon size and position using **Algorithm 4**

② Update the front and GOTO 4

end

iv. update the front

until the newly generated polygons touch the right wall B_r

4: **loop** the active segment of the present front

a. Select an active front and generate a new polygon P_n which is in contact with other two polygons/lines on the front using **Algorithm 3**

b. Check if the new polygon P_n is near the right boundary

i. Compute the distance d_n between the centroid of the polygon and the right boundary line

ii. **if** ($d_n<\lambda_u r_{ref}$) **then**

① Re-determine this polygon size and position using **Algorithm 4**

② Update the front and GOTO 4.c

end

c. Check if the new polygon P_n is near EB (except the left, bottom and right boundary)

i. Find the closest EB line to the new polygon P_n to be generated

ii. Compute the distance d_n between the centroid of the polygon and the closest EB line

iii. **if** ($d_n<\lambda_u r_{ref}$) **then**

① Re-determine this polygon size and position using **Algorithm 4**

② Check if the new polygon overlaps with other EB lines, if (overlap): GOTO 4.c.iii.① using the present overlap boundary line

end

d. Check if the polygon at this position overlaps with any existing polygon on the front using **Algorithm 5**

if (no overlap): Accept the new polygon, and GOTO 4.a

else: Reject the new polygon, update front as shown in Fig. 9, and GOTO 4.a

end

until all segments in the front are inactive

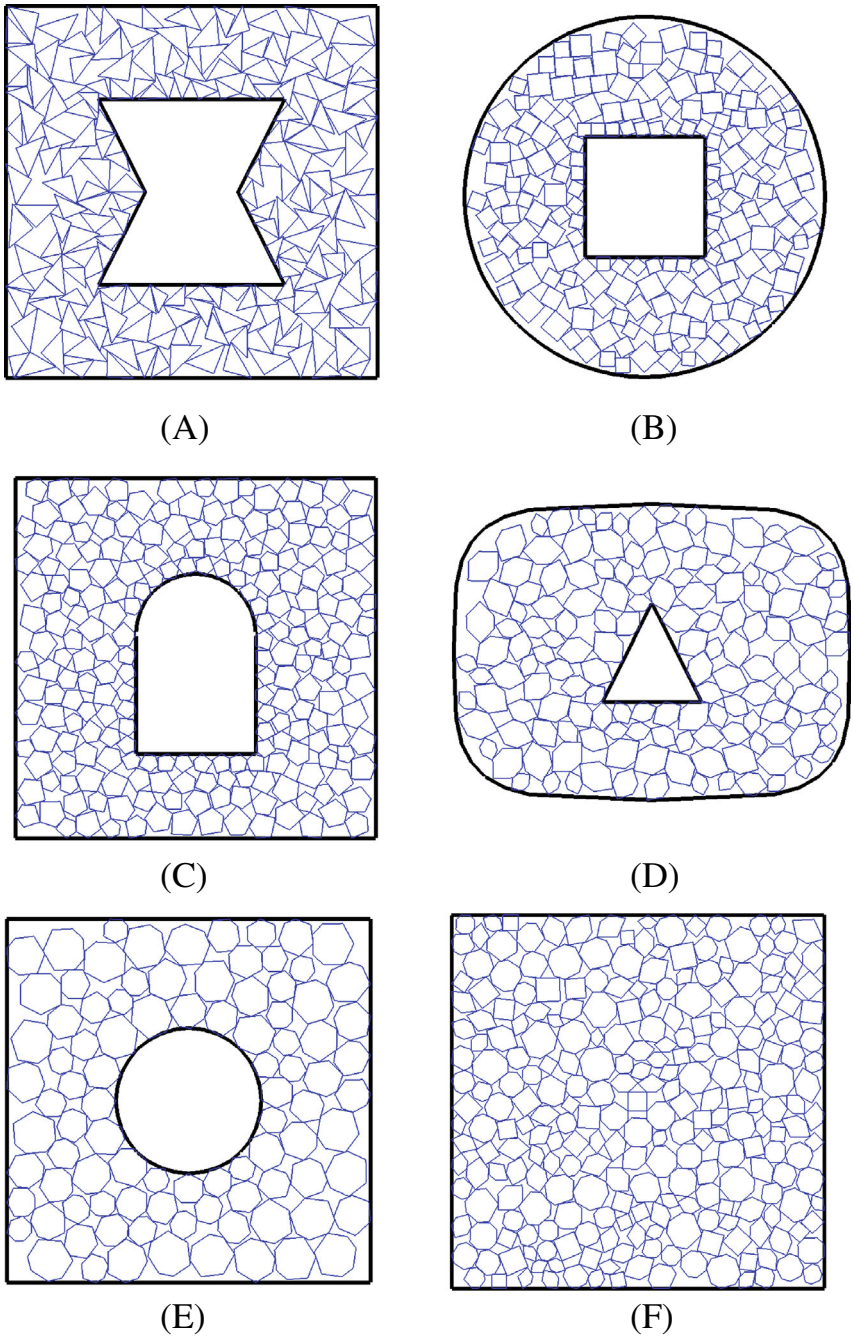

FIGURE 16 Closed form packing of convex polygons: (A) triangle; (B) rectangle; (C) pentagon; (D) hexagon; (E) heptagon; (F) mixed.

5 | EXAMPLES

To demonstrate the effectiveness and versatility of the proposed approach, various particle assemblies have been generated using the closed form and the open form on an Intel Core i5-8300H CPU 2.3 GHz laptop. The approach is programmed using Fortran95 in an in-house code.

Figures 16 and 17 illustrate the packing examples for different polygons generated using the closed form and open form respectively. The proposed approach is capable of generating various convex non-circular particle shapes including tri-, quad-, pen-, hex-, hept-agonal and mixed shapes. The packing assemblies show smoothness and minimal gaps near the EB, indicating that the proposed approach is robust and able to handle any type of convex non-circular particles. The IB includes triangular, rectangular, horseshoe and circular shapes, while the EB includes rectangular, circular and

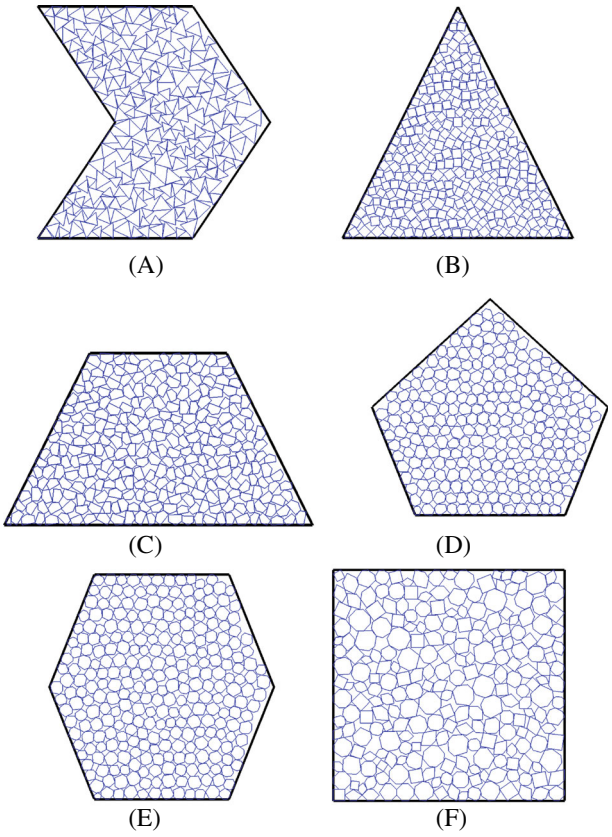

FIGURE 17 Open form packing of convex polygons: (A) triangle; (B) rectangle; (C) pentagon; (D) hexagon; (E) heptagon; (F) mixed.

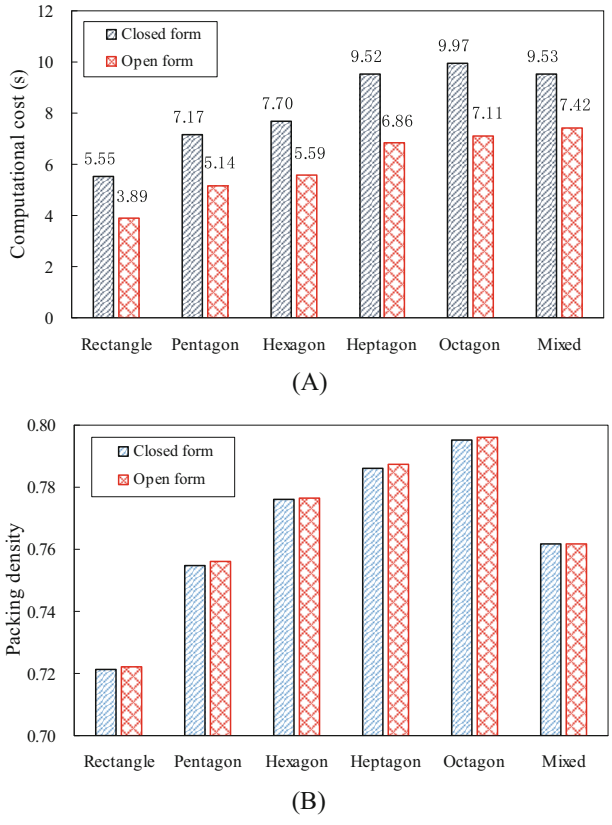

FIGURE 18 Performance comparison for packing 1 million polygons: (A) computational cost; (B) packing density.

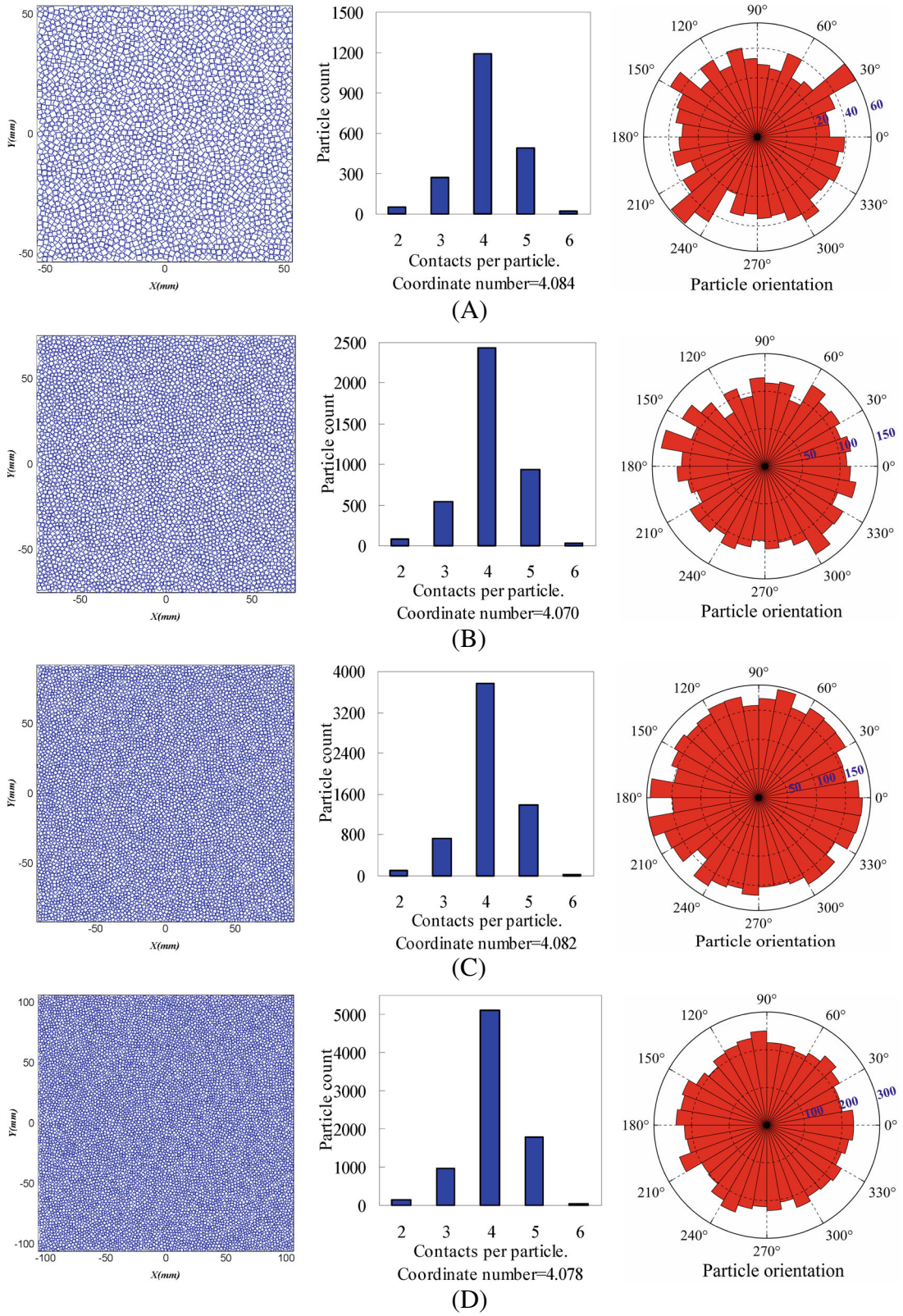

FIGURE 19 Rectangular particle packings and contact statistics in four square domains: (A) 2016 particles; (B) 4007 particles; (C) 6007 particles; (D) 8007 particles.

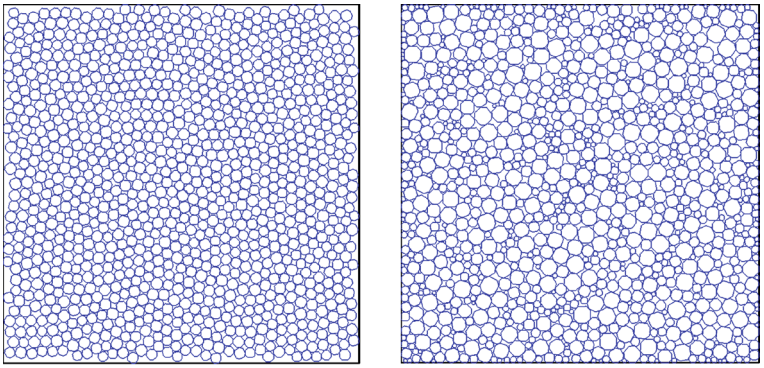

FIGURE 20 Closed form advancing front-generated packings in a unit square for two different scaling coefficients: (A) scaling coefficient = 1.2; (B) scaling coefficient = 5.0.

super-elliptical shapes, demonstrating the versatility of the proposed approach to handle different packing domains with either convex or concave interior cavity.

To assess the computational efficiency of the proposed approach, we measured the time required to pack 1 million convex non-circular particles of different shapes into a rectangular domain, taking into account EB. In these simulations, we limited the number of front segments checked during polygon generation to a maximum value of 10 for both forward and backward checks, and the scaling coefficient was uniformly distributed between 1.0 and 1.2, while the particle rotation ranged from 0 to 2π.

The results, presented in Figure 18A, show that although the open form algorithm is generally more complex to programme, it is more efficient than the closed form algorithm. For instance, the open form approach generated 1 million convex rectangles, pentagons, hexagons, heptagons, octagons, and mixed polygons in 3.89, 5.14, 5.59, 6.86, 7.11, and 7.42 s, respectively, using an Intel Core i5-8300H CPU 2.3 GHz laptop. In contrast, Du et al.[31] required 1 s to generate 1101 rectangles with an aspect ratio of 1 using MATLAB on an Intel Core i7-6700k CPU with 16 GB of RAM. Remarkably, our approach generated 1 million rectangles with an aspect ratio of 1 in only 3.89 s, highlighting its superior performance.

We also found that the computational cost is linearly proportional to the number of edges/vertices in a planar domain, provided the number of polygons to be generated is fixed. Notably, the open form algorithm only required 7.11 s to generate 1 million convex octagons, demonstrating its effectiveness. Furthermore, our approach produced a higher packing density between 0.72 and 0.80, as shown in Figure 18B. As the surface of a convex particle becomes more circular, a higher density is achieved.

Figure 19 shows the packings of rectangular particles and contact statistics in four square domains. The coordination number is the average number of contacts per particle. It can be seen that the generated packing assembles have almost the same average coordinate number around 4.07, indicating the homogeneity of the packing. There is also a similar particle orientation distribution.

The proposed algorithms can also handle particles with a wide range of size ratios. Figure 20 shows the packings of polygons generated in a square by the closed form algorithm for two different scaling coefficients: 1.2 and 5.0. Tests also indicate that the computation cost increased for a greater size ratio is negligible.

6 | CONCLUSIONS

This work presents a Minkowski difference-based advancing front approach, with both a closed form and an open form, for generating convex noncircular particles with arbitrary shapes in complex domains. The approach places new polygons in contact with existing polygons/lines using the Minkowski difference, which considers both interior and exterior boundaries. To ensure that polygons generated near exterior boundaries are strictly tangent to those boundaries, the bisection method is introduced.

The efficiency of overlap checking between new and existing polygons using four different algorithms is compared, with the Gilbert-Johnson-Keerthi (GJK) algorithm being the most efficient and the explicit construct Minkowski difference being the slowest. The GJK is about two times faster than the explicit construct Minkowski difference. During the packing process using the open form, the one-sided lifting problem is identified and addressed after treating polygons near the right boundary, which further improves computational efficiency and packing density.

Several numerical examples with various interior and exterior boundaries demonstrate the effectiveness of the proposed approach, which can generate 1 million rectangles and octagons in just 3.89 and 7.11 s, respectively, on a laptop with a 2.3 GHz processor. The results show that the proposed approach is highly efficient and robust and can achieve a packing density between 0.72 and 0.80.

It should be noted that the proposed computational framework is limited to generating arbitrary convex particles. However, it can be extended to packings of concave particles with two modifications: (1) When concave particles can be decomposed into convex components, their Minkowski difference can be obtained as the union of individual differences of the convex components; (2) The GJK algorithm is no longer valid for detecting contact states for concave particles, so the DSA contact detection algorithm can be used instead. Additionally, this work is limited to two dimensions, and future work will explore the extension to 3D.

ACKNOWLEDGEMENTS

This work is financially supported by the National Natural Science Foundation of China (No. 12072217), the Natural Science Foundation of Hunan Province (No. 2022JJ30567), and the Scientific Research Foundation of Education Department of Hunan Province (No. 20B557).

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Ming Xia https://orcid.org/0000-0003-1596-8156
Min Wang https://orcid.org/0000-0002-4454-2480
Y. T. Feng https://orcid.org/0000-0002-6396-8698

REFERENCES

1. Cundall PA, Strack ODL. A discrete numerical model for granular assemblies. *Geotechnique*. 1979;29(1):47-65.
2. Feng YT. An energy-conserving contact theory for discrete element modelling of arbitrarily shaped particles: basic framework and general contact model. *Comput Methods Appl Mech Eng*. 2021;373:113454.
3. Feng YT. An energy-conserving contact theory for discrete element modelling of arbitrarily shaped particles: contact volume based model and computational issues. *Comput Methods Appl Mech Eng*. 2021;373:113493.
4. Feng YT. Thirty years of developments in contact modelling of non-spherical particles in DEM: a selective review. *Acta Mech Sinica*. 2023;39(1):722343.
5. Huang GH, Tong CX, Zhang S, Chen XF. A thermo-solid coupling model for disk discontinuous deformation analysis to simulate heating and stirring particles in rotary drums. *Powder Technol*. 2022;402:117326.
6. Huang GH, Lv GS, Zhang S, et al. Numerical analysis of debris flows along the Sichuan-Tibet railway based on an improved 3D sphere DDA model and UAV-based photogrammetry. *Eng Geol*. 2022;305:106722.
7. Shi GH. *Discontinuous Deformation Analysis – A New Model for the Statics and Dynamics of Block Systems*. Ph.D thesis. University of California; 1988.
8. Krabbenhoft K, Huang J, da Silva MV, Lyamin AV. Granular contact dynamics with particle elasticity. *Granular Matter*. 2012;14:607-619.
9. Krabbenhoft K, Lyamin AV, Vignes C. Computational plasticity algorithm for particle dynamics simulations. *Comput Part Mech*. 2018;5:103-111.
10. Moreau JJ. Some numerical methods in multibody dynamics: application to granular materials. *Eur J Mech A/Solid*. 1994;13:93-114.
11. Jiang MJ, Konrad JM, Leroueil S. An efficient technique for generating homogeneous specimens for DEM studies. *Comput Geotech*. 2003;30(7):579-597.
12. Seelen LJH, Padding JT, Kuipers JAM. A granular discrete element method for arbitrary convex particle shapes: method and packing generation. *Chem Eng Sci*. 2018;189:84-101.
13. Visscher WM, Bolsterli M. Random packing of equal and unequal spheres in two and three dimensions. *Nature*. 1972;239(5374):504-507.
14. Bagi K. An algorithm to generate random dense arrangements for discrete element simulations of granular assemblies. *Granul Matter*. 2005;7(1):31-43.
15. Cui L, O'Sullivan C. Analysis of a triangulation based approach for specimen generation of discrete element simulations. *Granul Matter*. 2003;5:135-145.

16. Feng YT, Han K, Owen DRJ. Filling domains with disks: an advancing front approach. *Int J Numer Methods Eng*. 2003;56(5):699-713.
17. Jerier J-F, Richefeu V, Imbault D, Donzé F-V. Packing spherical discrete elements for large scale simulations. *Comput Methods Appl Mech Eng*. 2010;199:1668-1676.
18. Löhner R, Oñate E. A general advancing front technique for filling space with arbitrary objects. *Int J Numer Methods Eng*. 2004;61:1977-1991.
19. Lozano E, Gattass M. A new constructive algorithm for random polydisperse dense disk pack generation. *Int J Numer Methods Eng*. 2021;122:5812-5843.
20. Mollon G, Zhao J. 3D generation of realistic granular samples based on random fields theory and Fourier Shape Descriptors. *Comput Methods Appl Mech Eng*. 2014;279:46-65.
21. Fang X, Liu Z, Tan J. Algorithm with hybrid method based for sphere packing in two-dimensional region. *J Zhejiang Univ Eng Sci*. 2011;45(4):650-655.
22. Han K, Feng YT, Owen DRJ. Sphere packing with a geometric based compression algorithm. *Powder Technol*. 2005;155(1):33-41.
23. Zhao TT, Feng YT, Zhang J, Wang ZH, Wang ZY. Discrete element modelling of dynamic behaviour of rockfills for resisting high speed projectile penetration. *Comput Model Eng Sci*. 2021;127(2):721-735.
24. Recarey C, Pérez I, Roselló R, et al. Advances in particle packing algorithms for generating the medium in the Discrete Element Method. *Comput Methods Appl Mech Eng*. 2019;345:336-362.
25. Dong Q, Wang Y, Feng D. An algorithm to generate dense and stable particle assemblies for 2D DEM simulation. *Eng Anal Bound Elem*. 2020;114:127-135.
26. Xu X, Xia M. An efficient algorithm for dense discs packing considering domain boundaries. *J Xiangtan Univ Nat Sci Ed*. 2023;45(2):18-28.
27. Morfa CAR, Pérez Morales IP, de Farias MM, Oñate E, Valera RR, Casañas H. General advancing front packing algorithm for the discrete element method. *Comput Part Mech*. 2018;5:13-33.
28. Altuhafi FN, Coop MR, Georgiannou VN. Effect of particle shape on the mechanical behavior of natural sands. *J Geotech Geoenviron Eng*. 2016;142(12):04016071.
29. Wang X, Yin ZY, Su D, Wu XX, Zhao JD. A novel approach of random packing generation of complex-shaped 3D particles with controllable sizes and shapes. *Acta Geotech*. 2022;17:355-376.
30. Feng YT, Han K, Owen DRJ. An advancing front packing of polygons, ellipses and spheres. In: Cook BK, Jensen RP, eds. *Discrete Element Methods*. ASCE; 2002:93-98.
31. Du LB, Liu XR, Han YF, Deng ZY. Generation of irregular particle packing with prescribed statistical distribution, spatial arrangement, and volume fraction. *J Rock Mech Geotech Eng*. 2023;15(2):375-394.
32. Feng YT, Tan Y. The Minkowski overlap and the energy-conserving contact model for discrete element modeling of convex nonspherical particles. *Int J Numer Methods Eng*. 2021;122(22):6476-6496.
33. Feng YT, Tan Y. On Minkowski difference-based contact detection in discrete/discontinuous modelling of convex polygons/polyhedra: algorithms and implementation. *Eng Comput*. 2020;37(1):54-72.
34. Han K, Feng YT, Owen DRJ. Polygon-based contact resolution for super-quadrics. *Int J Numer Methods Eng*. 2006;66(3):485-501.
35. O'Rourke J, Chien CB, Olson T, Naddor D. A new linear algorithm for intersecting convex polygons. *Comput Graph Image Process*. 1982;19(4):384-391.
36. Gilbert EG, Johnson DW, Keerthi SS. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Trans Robot Automat*. 1988;4(2):193-203.

How to cite this article: Xia M, Xu X, Gong F, Wang M, Feng YT. A Minkowski difference-based advancing front packing technique for generating convex noncircular particles in complex domains. *Int J Numer Methods Eng*. 2023;1-27. doi: 10.1002/nme.7318