Expressive-WICED
C. E. Catalano and A. D. Parakkat (Editors)

EXPRESSIVE 2025

Modeling crochet patterns with a force-directed graph layout

Émile Greer and David Mould

Carleton University, Canada

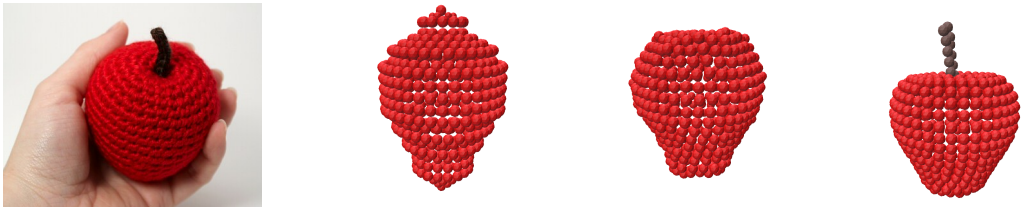


Figure 1: *Modeling a crocheted apple [Pla] using a force-directed graph layout. Left to right: a crocheted apple; an initial graph layout; model after a few iterations of optimizing; final layout.*

Abstract

Designing crochet patterns is a difficult, time-consuming task. Typically, an initial pattern is created and crocheted; after seeing how the object comes out, the pattern is modified and some amount of stitches are undone and remade, through some number of iterations. This process involves a lot of guesswork and the manual labor of physically crocheting. In this paper, we present a way of creating a 3D representation of a crochet pattern using a written pattern as input: we translate the written pattern into a graph and obtain a force-directed graph layout. The result is a 3D model that looks like the hand-crocheted pattern in shape and size, with the advantage that the designer does not need to physically crochet the pattern and can make adjustments based on the digital model. Our intended audience includes both professional designers as well as beginners, helping designers visualize their crochet pattern before investing the time and effort to physically make it. While our application is oriented towards amigurumi, it could be extended to work with clothing or other similar styles of crochet.

CCS Concepts

• ***Human-centered computing*** → ***Graph drawings;***

1. Introduction

Crochet is a type of yarn craft similar to knitting. A crocheter uses a single hook to pull yarn through loops to create new loops. We are mainly interested in a style of crochet called *amigurumi*, which is made with small, compact stitches and little to no negative space between stitches. Amigurumi is used to create plushies or other 3D pieces, as opposed to other styles of crochet used for primarily 2D pieces such as clothing, blankets, or doilies.

A pattern is a set of instructions to create a crocheted item; in amigurumi it is typically represented in written text as a combination of prose and row-by-stitch notation. Prose is often needed to specify how the different pieces in a pattern should be sewn together or where to add embroidered details, whereas row-by-stitch notation, sometimes referred to as textual notation [SRLH22], usually consists of a list of rows, and in each row a sequence of stitches. Stitch notation is not entirely standardized, and so a legend is sometimes provided by the pattern designer. The row-by-stitch portion of the pattern contains most or all of the instructions required by a crocheter to make that pattern; prose is optional.

Designing crochet patterns is usually a slow process. It is often not possible to create the desired structure as a single component, so the design process includes identifying the different parts of the project, such as head, body, legs, or ears, which is often done through sketching. Each separate part will have its own sub-pattern. Given a candidate pattern for a particular piece, the designer must crochet that piece to determine whether the result is as expected.

Individual designers have different approaches to creating and prototyping patterns. Some designers will crochet a piece and then write out the sequence of stitches they used in each row afterwards, while others will write out the stitches for a row before crocheting that row. In our experience and from discussions with other amigurumi pattern designers, it is most common to approach pattern design one row at a time, working on a row until the desired shape is achieved before moving on to the next row.

DOI: 10.2312/exw.20251057

The designer must check if the overall shape and size of crocheted pieces are correct, and if particular features are created accurately and in the proper location. In some cases, an issue may become apparent partway through the crocheting process; in other cases an issue may go unnoticed until all pieces have been crocheted and sewn together. When a problem is found, the crocheter must unravel their work, and start designing again. When instructions have been written down, they can typically be reused with minor modifications in certain rows, thereby saving the designer from remembering how specific shaping effects were made.

Crocheting and unraveling repeatedly can lead to damaged yarn and hours of extra work spent crocheting. In some cases, a designer may opt to abandon a particular piece and restart from scratch, leading to wasted materials. The ability to visualize crochet patterns could help designers prototype patterns more quickly and efficiently, with fewer crochet/unravel iterations. This would make the design process faster and more accessible, allowing designers to create patterns more quickly while also making it easier for less experienced crocheters to design their own patterns.

In this work, we present a modeling tool which takes as input a pattern written in the row-by-stitch notation, which is the most commonly used notation for amigurumi patterns. Our approach involves translating a written crochet pattern into a graph, then computing a graph layout to present a 3D model that is accurate in shape and size to real-world crocheted structures. In our model we represent each node in the graph as a sphere, which makes it easy for the user to see individual stitches.

2. Previous Work

Most yarn-related research is done in the context of knitted or woven textiles. Unlike crochet, knitted and woven textiles can be mass produced in factories, leading to more investment into modeling those textiles. Comparatively little research investigates the area of crochet. A notable exception is Taimina's exploration of hyperbolic crochet [Tai18], albeit not from an algorithmic standpoint.

Crane [Cra23] describes a parametric approach to modeling knitted yarn, presenting parametric equations for the trajectory and the ply-twist geometry of knitted yarn. Guo et al. [GLNM20] model the geometry of crocheted yarn by deconstructing crochet stitches into smaller building blocks, allowing them to combine those building blocks to represent some common crochet stitch types. In both papers, yarn texture (fuzziness) is not considered.

There is considerable interest in software to help with designing patterns across various fibre arts. One class of design software allows the user to sketch shapes and modify those shapes via sketching actions; e.g., Mori and Igarashi [MI07] created an application which allows a user to create sewing patterns for plushies using sketching, with numerous features such as cutting and reshaping. Crochet sketching software designed by Nakjan et al. [NRP18] has the user trace out 2D shapes, which the software converts to 3D shapes and then into a crochet pattern; however, it is limited to relatively symmetrical shapes such as spheres and cylinders.

Leake et al. [LBDA21] built a foundation paper-piecing quilt design tool that allows users to create geometric designs and get instant feedback about whether those designs are paper-pieceable, and if not it identifies which parts of the design are problematic. Their design tool allows users to add, remove, or modify seams interactively, and it provides all optional sewing orders, a task that is normally time-consuming and error-prone for quilt designers.

Several apps and websites help users create fibre art patterns. Plushify [Sta25] is a feature-rich plushie designer, in which users can transform a 3D model into a sewing pattern. The app allows them to specify size, colours, seam lines, the location of darts as well as any custom markings. *Amigurum.io* [HW] is a simple web app which provides some template shapes, and allows the user to mix and match different shapes to create a pattern. Currently, only a limited number of shapes are available, with more complex shapes requiring the user to pay. More recently, the web app *Crochet Parade* makes it possible for users to input crochet pattern instructions to generate a 3D representation, with the generated models being highly accurate. Although this is quite similar to our project, the implementation is not documented.

Gray et al. [GBK24] describe a graph layout approach for modeling 2D knitting patterns which takes into consideration the way yarn is interwoven to create stitches as well as the sizes of different stitch types, which they represent as edge length requirements in their layout algorithm. Ben-Chen et al. [EPIBC22] also use graphs in constructing crochet patterns from 3D models, although the resulting patterns are not ones that crocheters would use in practice for amigurumi, primarily due to the unintuitive segmentation.

3. Background

There are various styles of crochet, each with its own physical characteristics and terminology. We are primarily interested in amigurumi, which is characterized by dense uniform stitches with little or no negative space and which is used to make 3-dimensional patterns such as animals, plants, or characters. In particular, most amigurumi involves crocheting a shell and then filling the shell with stuffing to create crocheted plushies. Figure 2 shows some examples of amigurumi.

In amigurumi, patterns are usually presented in row-by-stitch notation, which is a list of rows and a list of stitches per row. A pattern consists of a number of rows, and each row is composed of a number of stitches. Stitches are made sequentially, one at a time, from right-to-left for right-handed crocheters and from left-to-right for left-handed crocheters. Figure 3 shows a portion of the instructions for a strawberry. Readers familiar with crochet will recognize that the patterns we tested are made in continuous rounds; we use the term rows generically to refer both to rounds and to rows.

4. Method

Creating a 3D representation of a crochet pattern from a set of written instructions involves several steps. First, we parse the pattern provided by the user and construct a graph. Next, we determine an initial graph layout. Finally, we apply the force-directed graph layout algorithm of Isenburg et al. [IGG01] to obtain the output model. Source code for our implementation is available at:

`https://github.com/EnbyMonkey/modeling-amigurumi.git`.

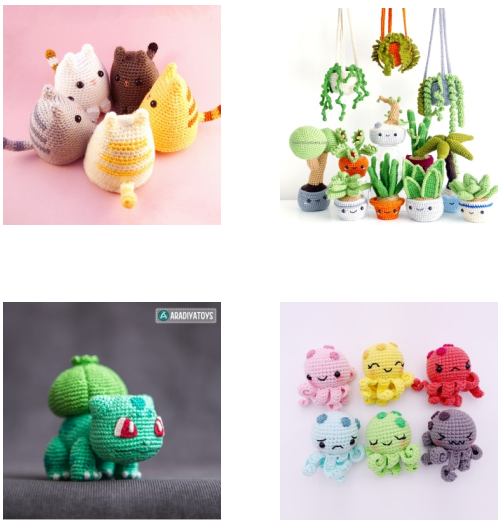


Figure 2: *Examples of amigurumi. Top left: Dumpling kitty [Cri]. Top right: Potted plants [Kno]. Bottom left: Bulbasaur [Ara]. Bottom right: Color octopus [Les].*

Strawberry (4.5mm hook, starting with green)

R1: Ch 2, 8 sc in second ch from hook (8) – **change colour to red in last st**

R2: [BLO inc] x8 (16)

R3: [Sc, inc] x8 (24)

R4: Sc, inc, [2 sc, inc] x7, sc (32)

R5-7: [Sc] x32 (32) *3 rounds*

Figure 3: *A portion of a written pattern of a strawberry [Stib].*

4.1. Constructing the graph

Each stitch is represented as a node in the graph, and the physical connections between stitches are represented as edges. The diagrams in Figure 5 show some common stitch types and their node-edge representation. In Figure 5, horizontal edges represent a sequential connection between stitches, where two nodes have a horizontal edge between them if the corresponding stitches were made one immediately after the other. Vertical edges represent working connections, where the top node/stitch was crocheted (worked) into the bottom node/stitch. Certain shaping effects are done without using crochet stitches, for example by passing yarn through different areas and pulling tight to create a dimpling effect, as seen in the apple pattern (Figure 7). These are represented with additional constraint edges.

For a given stitch, we can infer only the previous stitch and the working stitches in the previous row. Therefore, the connectivity graph is constructed sequentially, following the pattern in order as though we were crocheting it. For patterns in the round, the graph begins with a magic circle (mc) node. Then we read the instructions for the first row, starting with the first stitch.

Suppose the first row is "R1: 6sc (6)", indicating that in the first row, we make six single crochet stitches (into an anchor ring). A single crochet (sc) node is added to the graph, as well as an edge between the sc node and the anchor node. This edge represents a working (vertical) connection, since the stitch is worked into the anchor ring. For the next stitch, we add another 'sc' node to the graph, with a sequential edge to the first 'sc' node and a working edge to the anchor node. The rest of the first row is constructed in this way, with all 'sc' nodes connected to their predecessor and to the anchor node. Starting from the second row, stitches are worked into stitches from the previous row.

Most of the information required to construct the connectivity graph can be inferred when the pattern is in row/stitch notation, as in Figure 3. However, certain aspects of a pattern cannot easily be represented this way, and the user may be required to specify some things explicitly: e.g., information about how separate pieces should be joined together, or shaping effects which force the piece to take on a shape that would not arise naturally just from crocheting. Some of these can be represented using constraint edges as mentioned previously, while others merely indicate relative position and orientation of separate pieces and are not reflected in the graphs directly.

4.2. The graph layout

We implemented the force-directed graph layout described by Isenburg et al. [IGG01], who demonstrate that the underlying graph of a mesh contains some geometric information intrinsically via its connectivity. In essence, their algorithm takes a planar graph and inflates it by assigning a specific length to each edge and by minimizing overall curvature, balancing these two constraints so as to maximize volume. The graph we obtain from an amigurumi pattern is planar, edges have specific lengths, and stuffing the crocheted shell is equivalent to inflating the graph. For these reasons, we felt that this graph layout served as a good starting point for modeling crochet patterns.

The force-directed graph layout is a non-linear least-squares optimization problem requiring the minimization of two energy functions. The first energy function controls the lengths of edges in the graph, and the second energy function measures local curvature at each node in the graph. For simplicity, we start with the assumption that each edge should have length one and that each node should have minimal curvature. We can describe the edge length requirement as an energy function,

$$\mathrm{EL}(\mathcal{G}) = \sum_{(i,j)\in\mathcal{E}} \left(\|v_i - v_j\| - 1\right)^2$$

where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph, and v_i is the position in space of vertex i. We can also describe the curvature as an energy function,

$$\mathrm{C}(\mathcal{G}) = \sum_{i\in\mathcal{V}} L(v_i)^2$$

where $L(v_i)$ is the curvature of v_i. Isenburg et al. use the discrete uniform Laplacian to compute curvature,

$$L(v_i) = \frac{1}{d_i} \sum_{(i,j)\in\mathcal{E}} v_j - v_i,$$

where d_i is the number of neighbours of the vertex v_i. Although

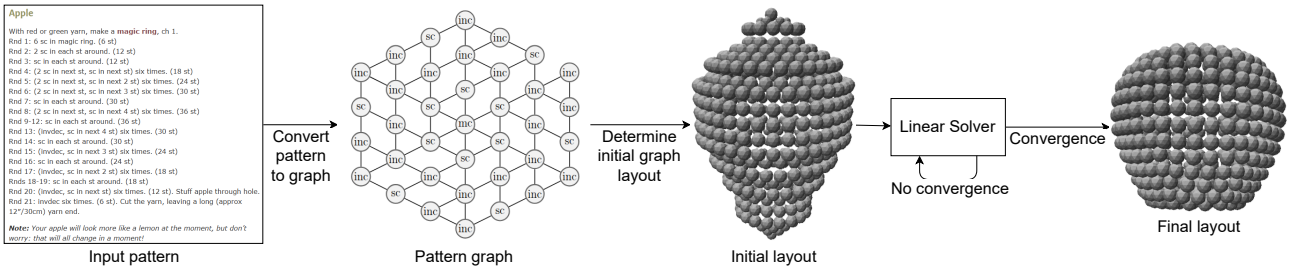
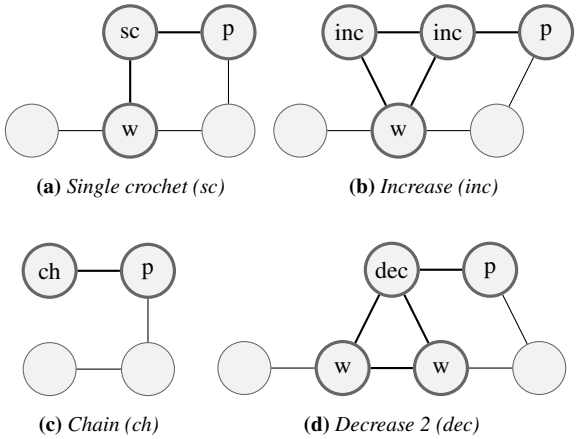


Figure 4: *Our method starts with a written crochet pattern. We convert that pattern to a graph, determine an initial graph layout and apply an edge-based force-directed graph layout algorithm. The final result is a 3D model representing the crochet pattern.*



(a) *Single crochet (sc)* **(b)** *Increase (inc)*

(c) *Chain (ch)* **(d)** *Decrease 2 (dec)*

Figure 5: *The connectivity of different stitch types. The nodes labeled p and w are the previous node and working node(s). Only increase stitches are represented as multiple nodes.*

the choice of curvature model used will affect the final result, the difference may not be significant enough to justify the higher computational costs of more precise methods. The paper then suggests minimizing the quantity

$$(1-\lambda)EL+\lambda C \tag{1}$$

where λ is chosen to maximize the volume of the graph layout. A linear solver, such as conjugate gradient descent, is used to compute the graph layout.

Our implementation used the Ceres Solver library [AMT23]. We used a static value of $\lambda = 0.65$ for all pieces, seeing no noticeable difference when testing other values, with the exception of the head and body of the bunny, where we used $\lambda = 0.9$. More exploration remains to determine when a static value of e.g. $\lambda = 0.65$ is sufficient, and when λ should be given a different value. The linear solver is by far the most time-expensive part of our application, and since we hope for it to be usable by a general audience it may not always be reasonable to optimize over possible values of λ.

5. Results and discussion

In this section, we will discuss some example results. First, we describe the result of applying our method to complete patterns, where we input the finished pattern in one step and compare the digital result with a previously-existing physical object. Subsequently, we will discuss how we envision our method being used in an iterative design process.

5.1. Modeling complete patterns

We tested our approach on several patterns that are available for free online. We chose relatively simple patterns: none of them would be considered beyond beginner-level in the crochet community. If the method could not successfully reproduce simple shapes, then we would know that it was likely not a viable approach for modeling crochet patterns. We did not model pieces which were flattened instead of stuffed, since the energy function seeks to maximize volume. We also limited the types of stitches tested to those which behave similarly to a single crochet stitch, which is the prototypical stitch in amigurumi. While these are reasonable initial restrictions for testing our method, they are too narrow for modeling intermediate or advanced patterns.

Stitches typically have an inherent orientation. The bottom of a stitch is the part that is worked into another stitch, the front of a stitch is the (typically) visible part of the stitch, the back of a stitch is the (typically) hidden part of a stitch, and the top of the stitch is usually where new stitches are worked into. However, some stitch types do not follow this pattern; e.g., the slip stitch is worked into the front of another stitch, partially covering the face of the other stitch.

Moreover, to achieve more interesting and unique shapes, it is quite common to change where and how one stitch is worked into another, which results in their relative orientations being subtly or dramatically different. This can be used, for example, to create sharp edges or corners, which are nearly impossible to achieve otherwise. We hope to expand our method to allow more versatility in the future.

Finally, we simulated attaching different pieces together with ad-hoc specification of position and orientation. We would like to devise a more systematic approach. These criticisms notwithstanding, we found that our models were accurate in shape and size to their

real-world crochet counterparts. We discuss some specific examples next.

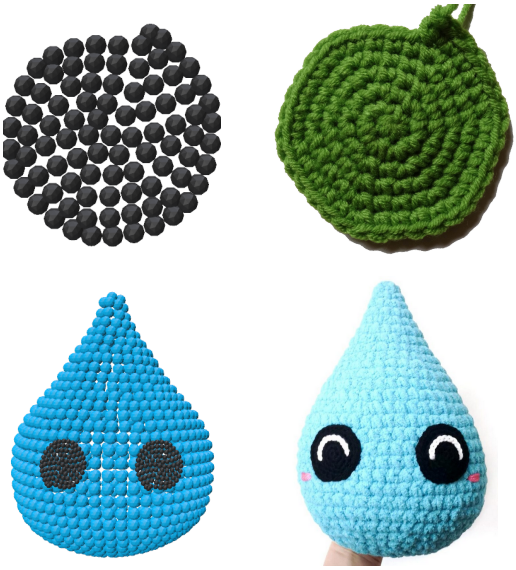


Figure 6: *Top left: the model of the raindrop eye; top right: the crocheted eye (in green for better visibility); bottom left: the complete raindrop model; bottom right: the crocheted raindrop.*

Figure 6 shows results from a raindrop pattern [Stia]. The lower portion of the pattern is rounded similar to the reference image. However, our model is not quite as tall as the crocheted version. We assumed constant edge length of 1, and this assumption is not accurate in general; differences in individual yarn tension, hook size, yarn weight and stitch type all lead to different stitch sizes. In the raindrop reference image, we can see that the stitches appear to be slightly taller than they are wide, which would lead to a slightly taller shape overall. Determining stitch dimensions is a necessary element in producing highly accurate models; the user might provide swath measurements (measurements of, e.g., a rectangle made with 10 rows and 10 stitches per row) with the yarn and hook they intend to use for their project.

In the eyes, we can observe a 6-point spiral in our model as well as in the reference image, although the effect is quite subtle. This effect is commonly seen in amigurumi; our model successfully recreates it using only the graph and graph layout. Note that while the eyes are flat in their natural resting state, they take on the curvature of the body when they are sewn into place; the graph layout does not have enough information to reproduce this effect. Also note that we omitted the white highlighting of the eyes.

Figure 7 shows results from an apple pattern [Pla]. The main body of the apple, when crocheted, initially has a shape similar to a lemon. The crocheter then uses a sewing needle to pass yarn through the top and bottom of the apple a few times, and pulls on the yarn until the desired shape is achieved, at which point the two yarn ends are tied together and hidden inside the body of the apple. Additional details can be found on the pattern website.

We recreated this pulling-in effect by adding a constraint edge

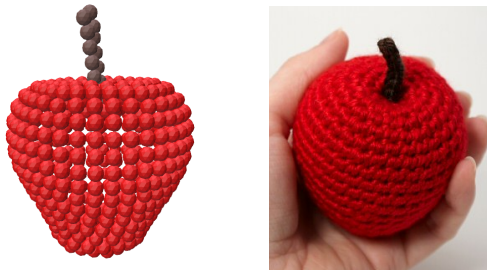


Figure 7: *Left: a model of an apple; right: the crocheted pattern.*

linking the first node and last node in the graph. An edge length of 6 was chosen through trial and error; Figure 8 shows the result with different edge lengths. This experimentation parallels the crocheter's approach, where they pull on the yarn until they are satisfied with the resulting shape. Note that for the graph nodes involved in this edge constraint, we omitted curvature computations completely, since we aimed for sharp points at those nodes.

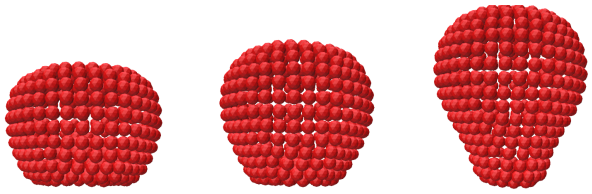


Figure 8: *The result of setting different edge length values for the constraint edge between the first and last stitch of the apple. From left to right, the values are 1, 5, and 10.*

The stem of the apple consists of only two rows of stitches. Our model is quite straight, whereas in the reference image the stem curves to one side, as is commonly the case with long, narrow pieces made with only a few rows. This is mainly because the first row is a foundation chain (a sequence of chain stitches), and chain stitches are slightly smaller than other stitch types types. We used uniform edge lengths which does not recreate the curvature.

Figure 9: *Left: the model of the elephant head; right: the crocheted elephant head.*

Figure 9 shows the head from an elephant pattern [Cha]. The

reference image shows asymmetry between the forehead and chin, with the forehead being more pronounced. Our model similarly emphasizes the forehead, although without the eyes as a point of reference, the comparison is somewhat difficult. The trunk on our model has the same shape as the crocheted piece, and the relative sizes of the head and trunk are nearly identical to the crocheted piece.

Figure 10: *Left: the model of the bunny; right: the crocheted bunny.*

Lastly, we tested a bunny pattern [Lov] consisting of a head with a crown of flowers, body, arms, legs, ears, and a tail. We modeled all pieces except the ears and crown; we omitted the crown because it consisted of stitches we were not yet testing, and we omitted the ears because they were meant to be flattened, and therefore were also beyond the scope of our implementation.

The arms and legs of the bunny rendered quickly and with the correct shape. The body and head of the bunny led to considerably longer run times for the layout algorithm. Moreover, with an initial value of $\lambda = 0.65$ (Equation 1), the bottoms of the body and head were unnaturally flat. To fix this, we experimented with some different values of λ, although we did not attempt to find values of λ which would maximize volume. Figure 11 shows the difference in the shape of the head when $\lambda = 0.65$ and $\lambda = 0.9$.

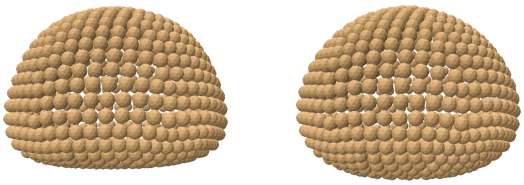


Figure 11: *The head of the bunny with* $\lambda = 0.65$ *(left) and* $\lambda = 0.9$ *(right).*

5.2. Iterative and incremental design workflow

The usual process of designing a crochet pattern is iterative: the designer makes a first guess about the pattern, crochets it, adjusts the pattern, crochets it again, and repeats until satisfied with the result. In general, a designer will only guess one row or a few rows at a time, crocheting as they go. As soon as they identify a problem they stop, unraveling their work only as needed, making the design process incremental as well as iterative.

We believe that this design workflow is well-suited to our graph layout method. Ceres Solver [AMT23], the linear solver library we used in our implementation, makes it easy to stop the linear solver after a desired number of iterations, make adjustments to the linear system in question, and restart the linear solver without the added overhead of setting up the solver each time. This means we can run the linear solver a small number of iterations each time the designer makes a change to their pattern. The designer can then have near-instant feedback to any changes they make, and provided the changes are not too dramatic, the model displayed should provide a useful reference.

We can also target the linear solver to only the parts of the pattern that the designer is making changes to. This would be useful, for example, with very large patterns, where changes in one area are not likely to affect the shape in other areas. This could be interspersed periodically with an application of the linear solver to the entire graph, to make sure the full model remains accurate. The exact balance of local vs. global applications of the graph layout would likely be determined through experimentation or by allowing the designer to set their own preferences.

5.3. Performance

Force-directed graph layouts are generally slow, and the particular approach we used is no exception. Larger patterns and patterns with higher curvature, such as the apple, took 30 seconds or more to run; in comparison, small patterns typically converged in a few seconds. For example, the relatively simple patterns for the legs (318 stitches) and arms (264 stitches) of the bunny reached convergence in under 5 seconds. On the other hand, neither the head (798 stitches) and body (708 stitches) of the bunny reached convergence even after running for well over two minutes.

We expect that this processing time can be mitigated by the iterative and incremental design process described in the previous step. When testing, we input the pattern in one step and ran the linear solver on the entire graph for each iteration, but we don't expect to do that during a typical design workflow. However, in cases where a designer has a pattern already written that they wish to make changes to, an upfront run-time cost may be unavoidable.

A final point to consider is whether it is even necessary for the linear solver to reach convergence. We often found that increasing the number of iterations beyond a certain point made no noticeable difference to the final shape of the model. We anticipate being able to raise the threshold for convergence significantly; allowing the user to set their preference would also let them decide for themselves what visual vs. run-time trade-offs they are willing to make.

5.4. Limitations

Other than the performance issues just mentioned, there are some limitations with the particular formulation of the force-directed graph layout itself. The graph layout aims to maximize volume by minimizing curvature, which we believe is a limitation with this approach. Least squares optimization problems are formulated as minimizing a set of cost functions, which is incompatible with our objective of maximizing volume. As a workaround, we instead minimize curvature which results in a sub-maximal volume.

In general, a pattern may not have minimal local curvature everywhere. In the case of the apple, we excluded certain nodes from the curvature minimization computation, and more broadly, there are crocheting techniques to create a specific amount of curvature. Requiring certain nodes to have set curvatures could possibly be incorporated into the graph optimization.

Finally, the accuracy of any representation is subjective. Differences in crocheting ability or technique, yarn tension and choice of yarn, and the amount and type of stuffing used all affect the shape of a crochet piece. Therefore, a representation that might seem correct to some may seem incorrect to others. The goal should be to present something that is accurate enough to be useful, with an understanding that small differences may be unavoidable.

6. Conclusion

We presented a method for previewing outcomes from amigurumi crochet patterns, used to make 3D figures such as plushies. We converted written instructions for an amigurumi crochet pattern into a graph, and applied the force-directed graph layout process described by Isenburg et al. [IGG01]. The resulting shapes were similar, though not identical, to the real-world crochet patterns. Some differences, such as the difference in height when modeling the raindrop, may have been caused by the choice of edge length (1 for nearly all edges). Other differences, such as the shape of the head of the bunny, may be caused by the parameters used in the graph layout, or may be intrinsic to the graph layout itself.

Testing a wider range of patterns, and more complex patterns, might demonstrate that the two features being measured in the graph layout (edge length and curvature) are not sufficient for obtaining models that are as accurate as possible. We might find that edge length and curvature are simply not appropriate metrics for modeling certain types of patterns, such as 2-dimensional patterns made in rows. The sheer versatility of crochet, and especially amigurumi, makes it unlikely that a one-size-fits-all graph layout will be up to the task.

Beyond expanding the graph layout to a wider range of patterns, we also need a way to deform the graph layout. In amigurumi, it is common to crochet a piece as 3-dimensional made in rounds and then flatten it into a 2-dimensional surface, as we saw with the ears of the bunny (Figure 10). This is a preferred method in many cases because the resulting piece is more stable and retains it shape better than a 2-dimensional piece made in rows. The texture of a piece crocheted in rounds vs. rows will also look different, and so for uniformity we generally prefer to have the same texture throughout an entire project. Deformation should be done in such a way that relative positions of neighboring stitches is preserved as much as possible.

To create a model that is as accurate as possible also requires augmenting the model with yarn geometry, ply, fuzziness, and other details. Because the target audience is crocheters who want to design patterns, we should not simply tile a model with a generic crochet stitch texture. Firstly, crochet stitch textures are as not simple as knitted yarn textures (the classic V shapes), so even coming up with such a generic texture is no simple task; in AI-generated images of crochet, it is impossible to tell what kind of stitches a piece is made of. Secondly, most crocheters rely on subtle cues to distinguish, for example, between a single crochet stitch and an increase stitch or between a single crochet stitch and double crochet stitch. In fact, the number of different crochet stitches is enormous, with each one looking distinct.

We have shown that simple crochet patterns can be modeled with a relatively high degree of accuracy. Even without generalizing to more complex patterns and stitch configurations, it is likely that design software based on our method would still be useful in the crochet community, by reducing the time spent designing simpler shapes and instead allowing designers to focus on the more complex aspects of their patterns. Extending this foundation to additional features, such as specialized arrangements of stitches, stitch variations, and more complex constraints, will further increase its value to the crocheting community.

Acknowledgements

Thanks to the reviewers for careful and thoughtful remarks; their suggestions improved the paper considerably. Thanks also to Oliver van Kaick and other members of the GIGL group for helpful suggestions along the way. This work received financial support from NSERC's Discovery Grant program.

References

[AMT23] AGARWAL S., MIERLE K., THE CERES SOLVER TEAM: Ceres Solver, 10 2023. URL: https://github.com/ceres-solver/ceres-solver. 4, 6

[Ara] ARADIYATOYS: Bulbasaur ("pokemon"). URL: https://www.amigurumi.com/Cartoons-and-Games/Bulbasaur/. 3

[Cha] CHASE N.: Elliot the elephant. URL: https://www.thenicolechase.com/patterns/elliot-the-elephant. 5

[Cra23] CRANE K.: A Simple Parametric Model of Plain-Knit Yarns. online, 3 2023. URL: https://github.com/keenancrane/plain-knit-yarn. 2

[Cri] CRITTERBEANS: Dumpling kitty. URL: https://www.amigurumi.com/Cats/Dumpling-Kitty/. 3

[EPIBC22] EDELSTEIN M., PELEG H., ITZHAKY S., BEN-CHEN M.: Amigo: Computational design of amigurumi crochet patterns. In *Symposium on Computational Fabrication (SCF '22)* (2022), ACM. doi:https://doi.org/10.1145/3559400.3562005. 2

[GBK24] GRAY K., BELL B., KOBOUROV S.: A graph model and a layout algorithm for knitting patterns, 2024. URL: https://arxiv.org/abs/2406.13800, arXiv:2406.13800. 2

[GLNM20] GUO R., LIN J., NARAYANAN V., MCCANN J.: Representing crochet with stitch meshes. In *Symposium on Computational Fabrication (SCF '20)* (2020), ACM. doi:https://doi.org/10.1145/3424630.3425409. 2

[HW] HA P., WIEHLER J.: Amigurum.io: Amigurumi pattern generator. URL: https://make.amigurum.io/. 2

[IGG01] ISENBURG M., GUMHOLD S., GOTSMAN C.: Connectivity shapes. In *Proceedings of Visualization 2001* (2001), pp. 135–142. URL: https://www.cs.unc.edu/~isenburg/connectivityshapes/. 2, 3, 7

[Kno] KNOTMONSTER: 12 potted plants bundle. URL: https://www.amigurumi.com/shop/Knotmonster/Knotmonsters-Potted-Plants-Bundle/. 3

[LBDA21] LEAKE M., BERNSTEIN G., DAVIS A., AGRAWALA M.: A mathematical foundation for foundation paper pieceable quilts. *ACM Trans. Graph. 40*, 4 (July 2021). URL: https://doi.org/10.1145/3450626.3459853, doi:10.1145/3450626.3459853. 2

[Les] LES PETITES MAINS DE KHUC CAY: The color octopus. URL: https://khuccay.com/the-color-octopus/. 3

[Lov] LOVELY CRAFT: Crochet bunny moo amigurumi pdf pattern. URL: https://www.lovelycraft.com/crochet-bunny-moo-amigurumi-pdf-pattern/. 6

[MI07] MORI Y., IGARASHI T.: Plushie: an interactive design system for plush toys. *ACM Trans. Graph. 26*, 3 (July 2007), 45–es. URL: https://doi.org/10.1145/1276377.1276433, doi:10.1145/1276377.1276433. 2

[NRP18] NAKJAN P., RATANOTAYANON S., PORWONGSAWANG N.: Automatic crochet pattern generation from 2d sketching. In *2018 10th International Conference on Knowledge and Smart Technology (KST)* (2018), pp. 170–175. doi:10.1109/KST.2018.8426123. 2

[Pla] PLANET JUNE: Amigurumi apples. URL: https://www.planetjune.com/blog/free-crochet-patterns/amigurumi-apples/. 1, 5

[SRLH22] SEITZ K., REIN P., LINCKE J., HIRSCHFELD R.: Digital crochet: Toward a visual language for pattern description. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (New York, NY, USA, 2022), Onward! 2022, Association for Computing Machinery, p. 48–62. URL: https://doi.org/10.1145/3563835.3567657, doi:10.1145/3563835.3567657. 1

[Sta25] STAMM S.: Plushify, 2025. URL: https://plushify.net/. 2

[Stia] STITCH BY FAY: Amigurumi raindrop crochet pattern. URL: https://www.stitchbyfay.com/amigurumi-raindrop-crochet-pattern/. 5

[Stib] STITCH BY FAY: Easy crochet strawberry pattern. URL: https://www.stitchbyfay.com/crochet-strawberry-pattern/. 3

[Tai18] TAIMINA D.: *Crocheting Adventures with Hyperbolic Planes*. CRC Press, Boca Raton, 2018. 2