Robust Self-assembly of Graphs

Stanislav Angelov[1], Sanjeev Khanna[2], and Mirkó Visontai[3]

[1] Google, Inc., New York, NY 10011, USA
angelov@google.com
[2] Department of Computer and Information Science, University of Pennsylvania
Philadelphia, PA 19104, USA
sanjeev@cis.upenn.edu
[3] Department of Mathematics, University of Pennsylvania
Philadelphia, PA 19104, USA
mirko@math.upenn.edu

Abstract. Self-assembly is a process in which small building blocks interact autonomously to form larger structures. A recently studied model of self-assembly is the Accretive Graph Assembly Model whereby an edge-weighted graph is assembled one vertex at a time starting from a designated seed vertex. The weight of an edge specifies the magnitude of attraction (positive weight) or repulsion (negative weight) between adjacent vertices. It is *feasible to add* a vertex to the assembly if the total attraction minus repulsion of the already built neighbors exceeds a certain threshold, called the assembly temperature. This model naturally generalizes the extensively studied Tile Assembly Model.

A natural question in graph self-assembly is to determine whether or not there exists a sequence of feasible vertex additions to realize the entire graph. However, even when it is feasible to realize the assembly, not much can be inferred about its likelihood of realization in practice due to the uncontrolled nature of the self-assembly process. Motivated by this, we introduce the *robust* self-assembly problem where the goal is to determine if every possible sequence of feasible vertex additions leads to the completion of the assembly. We show that the robust self-assembly problem is co-NP–complete even on planar graphs with two distinct edge weights. We then examine the tractability of the robust self-assembly problem on a natural subclass of planar graphs, namely grid graphs. We identify structural conditions that determine whether or not a grid graph can be robustly self-assembled, and give poly-time algorithms to determine this for several interesting cases of the problem. Finally, we also show that the problem of counting the number of feasible orderings that lead to the completion of an assembly is #P-complete.

1 Introduction

Self-assembly is a process in which small building blocks interact autonomously to form larger structures. The self-assembly approach is especially suitable for building molecular scale objects with nano-scale features. Several representative applications and practical models of self-assembly are discussed in [1,2,3,4,5,6,7,8].

A. Goel, F.C. Simmel, and P. Sosík (Eds.): DNA 14, LNCS 5347, pp. 127–143, 2009.

Rothemund and Winfree [9] proposed the *Tile Assembly Model* to formalize and facilitate the theoretical study of the self-assembly process. This model extends the tiling models based on Wang tiles [10]. In their work, the building blocks, namely the DNA tiles, are abstracted as oriented unit squares. Each side of a tile has a glue type and a (non-negative) strength associated to it. An assembly starts from a designated *seed* tile and can be augmented by a tile if the sides of the tile match the glue types of its already assembled neighbors, and the total glue strength is no less than a threshold parameter τ, referred to as the *temperature* of the assembly.

Reif, Sahu, and Yin [11] introduced a generalization of the Tile Assembly Model, to one on general graphs, called the *Accretive Graph Assembly Model*. The accretive graph assembly is a *sequential* process where a given weighted graph is assembled one vertex at a time starting from a designated seed vertex. The weight of each positive (resp. negative) edge specifies the magnitude of attraction (resp. repulsion) between the adjacent vertices. It is *feasible to add* a vertex to the assembly if the total attraction minus the total repulsion of the already built neighbors is at least the temperature τ. Here, *accretive* suggests the monotone property of the process, i.e., once a vertex is added it cannot be removed later (cf. the *Self-Destructive Graph Assembly Model* [11] and the *Kinetic Tile Assembly Model* where tiles can fall off [12,13]).

The Accretive Graph Assembly Model addresses some of the deficiencies of the Tile Assembly Model. For example, it models repulsion and allows the assembly of general graph structures. A central problem in this model is the *Accretive Graph Assembly Problem* (`AGAP`): Given a weighted graph, a seed vertex, and an assembly temperature τ determine if there is a sequence of feasible vertex additions that builds the graph. Among other results, Reif et al. [11] showed that `AGAP` is NP-complete for graphs with maximum degree 4 and for planar graphs (`Planar AGAP`) with maximum degree 5. Subsequently, Angelov, Khanna, and Visontai [14] improved these results by giving a dichotomy theorem which completely characterized the complexity of `Planar AGAP` on graphs with maximum degree 3 and only 2 possible edge weights. Specifically, it was shown that whenever the allowed edge weights and τ satisfied a simple set of inequalities the problem is NP-complete, and poly-time solvable otherwise.

A drawback of the Accretive Graph Assembly Model is that even when there exists a feasible order of vertex additions to build the graph, its realization in practice may require a careful control over the order of assembly. Such control is arguably hard to implement at the molecular level, and perhaps, even in conflict with the notion of *self*-assembly. To alleviate this drawback, Reif et al. [11] considered a probabilistic variant of the model where at any point of time, the vertex to be build is chosen uniformly at random from the set of all vertices that can be added at that time to the partial assembly. Note that assembly still proceeds by adding one vertex at a time (cf. *insufficient attachment* in [12,13]). One of the main problems in this, so-called Stochastic Accretive Graph Assembly Model is to determine the probability of a graph system being assembled. One approach to estimating this probability is to consider the ratio of the number of

orderings that assemble the input graph to the total number of feasible maximal orderings. Reif et al. [11] showed that the problem of counting the number of ways a given *subgraph* can be assembled is #P-complete, and inferred that determining the probability of assembly of the subgraph is also #P-complete.

However, the following example shows that the number of orderings that assemble a graph against all possible ways to assemble a maximal subgraph can be arbitrary far from the actual probability of assembly. Consider the following graph with seed vertex s, a special vertex t, and two sets of vertices U and V, each of size n. The vertices in U are connected to s with edges with weight $\tau+1$, and to t with edges with weight -1. The vertices in V are connected only to t with edges with weight τ and there is an edge (s,t) with weight τ. Here τ is the assembly temperature. It is easy to see that starting from s, if the first vertex that is built is t, we can complete the remaining vertices in $(2n)!$ possible ways. On the other hand, if we build first any vertex from U, we make t infeasible. Furthermore, there are $n!$ orderings that cannot be extended with additional vertices and do not build the whole graph. Thus, the probability of assembly is exactly $\frac{1}{n+1}$. On the other hand, the ratio of feasible orderings that complete the graph to all possible ways to assemble a maximal subgraph is essentially 1.

Our Results and Techniques. We introduce a new accretive graph self-assembly problem that captures the uncontrolled nature of the self-assembly process: Given a graph G, does G assemble *robustly*, i.e., with probability 1? We refer to this problem as `Robust AGAP` and characterize its complexity as follows.

Theorem 1. `Robust AGAP` *with* 2 *weights is* co-NP*–complete on planar graphs. Moreover, when the number of weights is* 3*,* `Robust AGAP` *is* co-NP*–complete even on graphs with maximum degree* 3*.*

We use ideas developed in [11,14] along with several new combinatorial gadgets. The use of gadgets allows us to follow the same general framework while optimizing various parameters of the problem by finding equivalent gadgets for each case. We note that NP-completeness of `AGAP` on a family of instances does not imply that the corresponding `Robust AGAP` is co-NP–complete. It is easy to construct NP-hard instances of `AGAP` that admit a poly-time decision algorithm for `Robust AGAP`. Also, note when the number of allowed weights is one or the maximum degree is at most two, `Robust AGAP` is trivially solvable in poly-time.

In light of Theorem 1, it is natural to consider `Robust AGAP` with two weights on some subclasses of planar graphs. Towards this end, we study the tractability of `Robust AGAP` with two weights on grid graphs. The setting, with a positive weight w_p and a negative weight w_n modeling attraction and repulsion, respectively, is a natural analog of the Tile Assembly Model. We systematically analyze the complexity of `Robust AGAP` for all possible relationships between w_p, w_n, and the assembly temperature τ. We obtain the following partial characterization.

Theorem 2. `Robust AGAP` on grid graphs *is poly-time solvable when either* $\tau \leq w_p + 2w_n$ *or* $\tau > 2w_p + w_n$*.*

Finally, we strengthen a result in [11] by showing #P-hardness results for counting problems in the context of self-assembly. We omit the details from this version of the paper.

Theorem 3. *The problem of counting the number of ways an instance of* `AGAP` *can be assembled, namely* #`AGAP`*, is* #P*-complete.*

Organization. We begin by defining `AGAP` and `Robust AGAP`. In Section 3, we show hardness of `Robust AGAP` using reduction from `AGAP` by introducing modular gadgets. We also show hardness of `Robust AGAP` on planar graphs via a new reduction from DNF tautology. In Section 4, we study a related problem to tile assembly in the presence of repulsion, namely `Robust AGAP` on grid graphs.

2 Preliminaries

We adopt the *Accretive Graph Assembly Model* introduced in [11]. An *accretive graph assembly system* is a quadruple $\langle G, v_s, w, \tau \rangle$, where $G = (V, E)$ is undirected weighted simple connected graph, $v_s \in V$ is the seed vertex, $w : E \to \mathbb{Z}$ is a weight function on the edges, and $\tau \in \mathbb{N}$ is the temperature of the assembly. The assembly process is the following. Initially, the assembly consists of v_s only. The process is a *sequential* attachment of vertices to the assembly, i.e., vertices are built one by one. Given a partially assembled graph and $v \in V$, let $\Gamma(v)$ be the set of already built neighbors of v in G. Now, v can be built iff $\sum_{u \in \Gamma(v)} w(u, v) \geq \tau$. The model is *accretive* because once a vertex is built it cannot be detached from the assembly. For $u, v \in V$ we will use $u \prec v$ to denote that u has already been built when vertex v is built. Note that $\prec$ is an irreflexive, antisymmetric, and transitive relation. We consider the following problems:

Definition 1 (Accretive Graph Assembly Problem (`AGAP`)). *Given an accretive graph assembly system* $\langle G = (V, E), v_s, w, \tau \rangle$*, determine if* G *can be* assembled sequentially *(in short,* assembled*) starting from the seed vertex* v_s*, and provide a feasible order of assembly,* $v_s = v_{\pi(1)} \prec v_{\pi(2)} \prec \ldots \prec v_{\pi(n)}$*, if one exists. Here,* π *is a permutation of* $\{1, \ldots, n\}$ *and* $n = |V|$*. The* `AGAP` *problem restricted to planar graphs is referred to as* `Planar AGAP`*. When there are at most* k *different edge weights in* G*, we denote the problem as* k-`Wt. AGAP`*. Similarly, when the maximum degree of* G *is* d*, we use* d-`Deg. AGAP`*.*

`AGAP` and `Planar AGAP` are NP-complete [11]. Furthermore, `Planar AGAP` (hence `AGAP`) with maximum degree 3 and two distinct weights is NP-complete [14]. Note, even when an instance G of `(Planar)AGAP` can be assembled, a careful control over the order in which vertices are built may be required to assemble G. To deal with such situations, we introduce the notion of *robust* self-assembly.

Definition 2 (`Robust AGAP`). *Given an accretive graph assembly system with underlying graph* G*, determine if every* partial *feasible order of assembly of* G *can be extended to a* full *feasible order of assembly of* G*.*

`Robust AGAP` is in co-NP since given any ordering π, we can check in polynomial-time that π is a partial feasible assembly of a strict subset of V that cannot be extended to include additional vertices.

We also consider `Robust AGAP` on *grid graphs* due to its close connection with tile assembly with repulsion.

Definition 3 (Grid Graph). *An $m \times l$ grid graph $G_{m,l} = (V_{m,l}, E)$ is a graph such that its vertices can be arranged in an $m \times l$ rectangular (integer) grid with edges between vertices with ℓ_1 distance 1.*

3 Hardness of Robust Self-assembly

Planar 3SAT. In our hardness results, we will mostly use a reduction from `Planar 3SAT` similar to [11,14]. Lichtenstein proved that `Planar 3SAT`, i.e., `3SAT` with the restriction that the *identifying graph* is planar, remains NP-complete [15]. The identifying graph of a `3SAT` formula ϕ is the following graph G. Vertices of G correspond to literals and clauses of ϕ. There is an edge between a literal vertex and a clause vertex if the literal participates in the clause in ϕ, and there is an edge between every literal and its complement. Middleton showed that deciding the satisfiability of a `Planar 3SAT` formula with a modified identifying graph (see Fig. 1) obeying the following restrictions is still NP-complete [16]:

(1) There is a cyclic path, called the *loop* (the dashed circle denoted by L in Fig. 1), that can be drawn in the plane such that it passes between all pairs of complementary literals, but does not intersect any other edges of G.
(2) The formula ϕ contains only clauses in which the literals are either all positive or all negative.

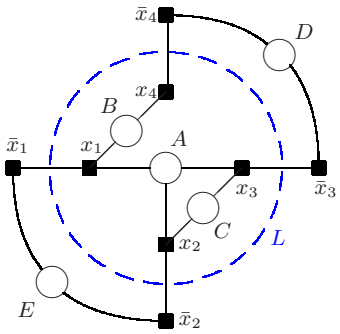


Fig. 1. The identifying graph for the formula $A \wedge B \wedge C \wedge D \wedge E = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2)$

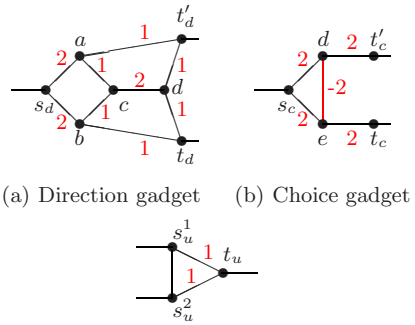


Fig. 2. Gadgets for $w_p = 2$, $w_n = -2$, and $w_o = 1$, and temperature $\tau = 2$. Edges without annotation have weight $w_p = 2$.

(3) G can be arranged so that interior (resp. exterior) clauses have positive (resp. negative) literals.
(4) Let $C(\ell)$ denote the set of clauses in which a literal ℓ participates, then $|C(\ell)| \leq 2$ for all ℓ in ϕ.

We assume the loop to be *directed*. This provides a natural (cyclic) ordering of the variables. For x and y we use the notation $xy \in L$ to denote that y succeeds x in L, e.g., $x_1x_2 \in L$, but $x_1x_3 \notin L$ in Fig. 1.

Gadgets. In our hardness constructions, we use modular composition of basic graph gadgets as outlined below. In parentheses, we give the identifying vertices of the gadgets (omitting any additional vertices for clarity). We first describe the gadgets when there are 3 distinct edge weights ($w_p \geq \tau$, $w_n < 0$, and $0 < w_o < \tau$) and then show the required modifications for 2 distinct edge weights ($w_p \geq \tau$ and $w_n < 0$ only). Note that we have maximum degree 3 in the first case, and maximum degree 5 in the latter. Also, the gadgets are planar and $\tau > 1$.

Direction gadget (s_d, t_d, t'_d) [14]**:** The gadget (see Fig. 2(a)) properties are as follows. Note, to realize the gadget, we require $2w_o \geq \tau$.

- If s_d is built, we can complete the gadget: i.e., $s_d \prec \{a, b\} \prec c \prec d \prec \{t_d, t'_d\}$.
- If t_d and t'_d are built, we can complete the gadget: i.e., $\{t_d, t'_d\} \prec d \prec c \prec \{a, b\} \prec s_d$.
- If only t_d or t'_d are built, but not both, we cannot build s_d via the gadget.

Choice gadget (s_c, t_c, [14]**:** The gadget (see Fig. 2(b)) properties are as follows. Note, to realize the gadget, we require $w_p + w_n < \tau$ and $2w_p + w_n \geq \tau$.

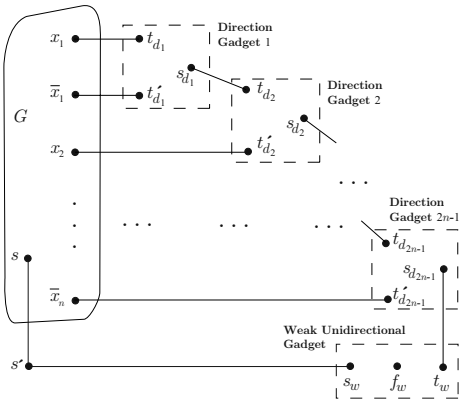
- If s_c is built, we can build either t_c or t'_c but not both via the gadget. For example, building d before e makes the net contribution to e from (s_c, e) and (d, e) equal to 0 which is less than $\tau = 2$.
- If only t_c (resp. t'_c) is built, we cannot build t'_c (resp. t_c) via the gadget. This property follows from a similar argument to the one above.

Unidirectional gadget ($s^1_u, \ldots, s^k_u, t_u$)**:** The gadget (see Fig. 2(c)) properties are as follows. Note, to realize the gadget, we require $kw_o \geq \tau$ ($k > 1$).

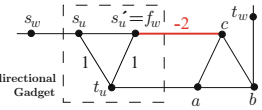
- If s^1_u or s^2_u are built we can build t_u.
- If only t_u is built, we cannot build s^1_u nor s^2_u via the gadget.

Weak unidirectional gadget (s_w, t_w, f_w)**:** The gadget (see Fig. 3(b)) properties are as follows. Note, to realize the gadget, we require $w_p + w_o + w_n < \tau$, $2w_o \geq \tau$, and $2w_p + w_n \geq \tau$.

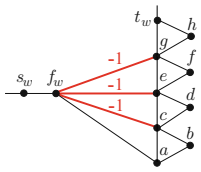
- If s_w is built but not t_w then in any feasible order of assembly for the gadget, we can build t_w, i.e., $s_w \prec s_u \prec s_{u'} = f_w \prec t_u \prec a \prec b \prec \{c, t_w\}$.
- If t_w is built before f_w, then there is an order of assembly in which f_w is made infeasible, i.e., cannot be built. For example, consider the order of assembly: $t_w \prec a \prec b \prec c$. The contribution to f_w from (c, f_w) is -2 which cannot be offset by the weights of (s_u, f_w) and (t_u, f_w).



(a) Composition of graph G induced by a formula ϕ (as in Theorem 4) with $2n-1$ direction gadgets and a weak unidirectional gadget.



(b) Weak unidirectional gadget for graphs with maximum degree 3 and three possible edge weights, e.g., $\{2, 1, -2\}$ for $\tau = 2$.



(c) Weak unidirectional gadget for graphs with maximum degree 5 and two possible edge weights, e.g., $\{2, -1\}$ for $\tau = 2$.

Fig. 3. Template for co-NP–hardness reductions for 3-Deg. 3-Wt. Robust AGAP and 5-Deg. 2-Wt. Robust AGAP and the corresponding weak unidirectional gadgets. Edges without annotation have weight equal to the temperature τ.

The gadgets above can also be constructed using only two edge weights (e.g., $w_p = 2$ and $w_n = -1$ at $\tau = 2$) by increasing the maximum degree to 5. For the Direction and Unidirectional gadgets, we model an edge (u, v) of weight 1 by creating a triangle adding vertex w and setting $(u, v) = -1$, $(u, w) = 2$, and $(w, v) = 2$. For the Weak unidirectional gadget, we can use the construction given in Fig. 3(b). In general, for $\tau > 1$, we require the following edge weight constraints to realize each gadget:

- Direction gadget: $2w_p + 2w_n \geq \tau$.
- Choice gadget: $w_p + w_n < \tau$ and $2w_p + w_n \geq \tau$.
- Unidirectional gadget: $2w_p + 2w_n \geq \tau$.
- Weak unidirectional gadget: $2w_p + 3w_n < \tau$ and $2w_p + w_n \geq \tau$.

In Section 3.2, we will also use the *Asymmetric gadget*.

Asymmetric gadget $(s_a, t_a; w_s \geq 0, w_t \geq 0)$**:** The gadget (see Fig. 5) property is that starting from s_a (resp. t_a) and building all vertices of the gadget except t_a (resp. s_a) the net (weight) contribution to t_a (resp. s_a) is w_t (resp. w_s).

3.1 Robust AGAP Is Co-NP–Complete

To show our hardness results, we reduce AGAP to Robust AGAP. Given an assembly system on graph G, we construct an instance H of Robust AGAP such that there is a maximal ordering that does not assemble all of H iff G can be

assembled. For the purpose, we will compose G with direction gadgets and one weak unidirectional gadget (identified by s_w, t_w, and f_w) such that if all of G can be assembled then the vertex f_w can be made infeasible (via t_w). But, if G cannot be assembled, H robustly assembles (via s_w).

For the basis of our reductions we will use the following result shown in [14].

Theorem 4 ([14]). *Given a* `Planar 3SAT` *formula ϕ, there is an instance of* `3-Deg. 3-Wt. Planar AGAP` *(also an instance of* `5-Deg. 2-Wt. Planar AGAP`*) where the underlying graph $G = (V, E)$ has a subset of vertices $V' \subset V$ satisfying:*

(i) V' consists of the seed vertex and the literals of ϕ: V'=$\{s, x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$,
(ii) each vertex in V' has degree 2,
(iii) G can be assembled iff all vertices in V' can be built, and
(iv) all vertices in V' can be built iff ϕ is satisfiable.

Furthermore, G consists of carefully composed choice and direction gadgets only.

We now show that `Robust AGAP` is co-NP–complete.

Theorem 5. `3-Deg. 3-Wt. Robust AGAP` *is* co-NP*–complete.*

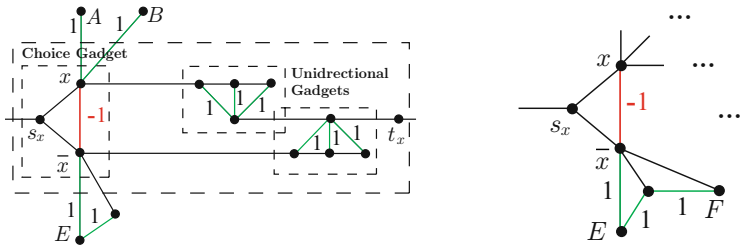
Proof. W.l.o.g., we show the proof for $\tau = 2$ and weights $\{2, 1, -2\}$. By using the same gadgets with different weights, the argument extends to any $\tau > 1$.

We use reduction from an `AGAP` instance with graph G, seed vertex s, edge weights $\{2, 1, -2\}$ and formula ϕ containing literals $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$ (as in Theorem 4). From G, we obtain graph H in the following way (see Fig. 3(a)). We use $2n-1$ direction gadgets where the ith gadget is identified by t_{d_i}, t'_{d_i}, and s_{d_i}. The first copy is connected to x_1 and $\bar{x}_1$ by edges (x_1, t_{d_1}) and $(\bar{x}_1, t'_{d_1})$. For $i > 1$, the ith gadget is connected to the $(i-1)$th gadget by an edge $(s_{d_{i-1}}, t_{d_i})$ and to G by an edge (y, t'_{d_i}), where $y = x_{\lfloor \frac{i}{2} \rfloor + 1}$ for even i, and $y = \bar{x}_{\lfloor \frac{i}{2} \rfloor + 1}$ otherwise. The last direction gadget is connected to a weak unidirectional gadget (identified by s_w, t_w, and f_w) by an edge $(s_{d_{2n-1}}, t_w)$. Finally, an additional vertex s' is set to be the seed vertex and is connected by edges (s', s) and (s', s_w). All connecting edges have weight equal to 2.

We now prove that there is a feasible maximal ordering H that does not assemble all of H iff G can be assembled. We will use the fact that $s_{d_{2n-1}}$ can be built without t_w being built iff G can be assembled (from Theorem 4 and properties of direction gadgets). Furthermore, if $s_{d_{2n-1}}$ is built, we can build all of G via the direction gadgets.

(only-if) Suppose G cannot be assembled. Then in any feasible order of assembly of H, $s' \prec s_w \prec f_w \prec t_w \prec s_{d_{2n-1}}$. Now, once $s_{d_{2n-1}}$ is built, we can assemble all vertices of G corresponding to literals via the direction gadgets. Therefore, we can assemble all of G and thus all of H.

(if) On the other hand, if G can be assembled then we can build $s_{d_{2n-1}}$ and therefore t_w before s_w and f_w. Using the properties of the weak unidirectional gadget, we conclude f_w can be made infeasible.



(a) Gadget replacing a variable x, participating in clauses A, B, E along the loop L.

(b) Fragment: Literal participating in two clauses with two literals each.

Fig. 4. Construction for 6-`Deg.` 3-`Wt.` `Robust AGAP` on planar graphs for edge weights $\{-1, 1, 3\}$ and temperature $\tau = 3$. Edges without annotation have weight 3.

Using the equivalent gadgets for the case when there are only 2 possible edges weights, we obtain the next corollary.

Corollary 1. 5-`Deg.` 2-`Wt.` `Robust AGAP` *is* co-NP*–complete.*

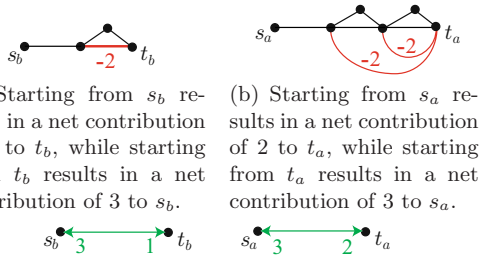
3.2 `Robust AGAP` on Planar Graphs Is Co-NP–Complete

Note that in the previous section the constructed graph H is *not* planar regardless of G being planar. We show that `Robust AGAP` on planar graphs is co-NP–complete by using reduction from DNF tautology. We construct a planar graph that robustly self-assembles iff the underlying formula is a tautology.

Let formula ϕ be a `Planar 3SAT` formula. Then $\bar{\phi}$ is a DNF formula that has the same identifying graph as ϕ up to a permutation of the variables. Let the loop L induce the ordering of the variables $x_1, \ldots, x_n$ and recall that each clause has either 2 or 3 literals. Given $\bar{\phi}$ and its identifying graph, we modify the graph similarly to the constructions given in [11,14] but using different gadgets. We then connect all clauses (preserving planarity) such that if any one clause is built we can build the remaining clauses and all other vertices. In the end, we show that the graph robustly self-assembles iff $\bar{\phi}$ is a tautology.

We now describe the details of the construction below for temperature $\tau = 3$ and draw the edge weights from the set $\{3, 1, -1\}$. For every variable x and its negation $\bar{x}$, we replace the edge $(x, \bar{x})$ in the graph (Fig. 1) with the gadget depicted in Fig. 4(a). For x and y, $xy \in L \backslash \{x_n x_1\}$, we connect the corresponding gadgets with edge (t_x, s_y) with weight $w(t_x, s_y) = 3$. The gadget ensures that unless all literal vertices adjacent to a clause are built (i.e., the clause is satisfied) then for each variable x at most one of x or $\bar{x}$ can be built following the ordering induced by L. Formally, let i be the largest index such that x_i or $\bar{x}_i$ is built. Then, exactly one of x_j or $\bar{x}_j$, for all $j \leq i$, are built if there is no clause with all literals already built.

We connect each literal ℓ to the clauses it participates as follows. If a clause $A \in C(\ell)$ has two literals, and ℓ is induced by the variable with smaller index,



(a) Starting from s_b results in a net contribution of 1 to t_b, while starting from t_b results in a net contribution of 3 to s_b.

(b) Starting from s_a results in a net contribution of 2 to t_a, while starting from t_a results in a net contribution of 3 to s_a.

(c) Simplified graphical notation of the gadgets.

Fig. 5. Asymmetric gadgets for $\tau = 3$ and weights $\{-2, 3\}$. Graphically, we will represent the gadgets with bidirectional edge with corresponding weights at end-points.

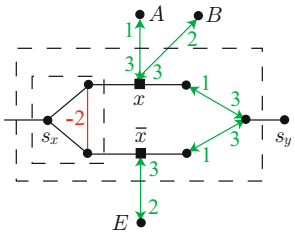


Fig. 6. Composition of gadgets for 9-Deg. 2-Wt. Robust AGAP on planar graphs for edge weights $\{-2, 3\}$ and $\tau = 3$.

then ℓ and A are connected with an edge of weight 2. Otherwise, ℓ and A are connected with an edge of weight 1. We simulate edge weight 2 by a triangle with two edges of weight 1 and one with weight 3 (see Fig. 4(a)). If a literal is connected to two clauses in this manner, then the induced two triangles share the edge with weight 3 adjacent to the literal (see Fig. 4(b)).

Finally, we want to connect all clauses with paths of edges with weight 3 such that if one clause is built then we can build the remaining clauses, planarity is preserved, and the maximum degree of the resulting construction is low. Consider the clauses with only positive literals (similarly negative). A clause $A = x_a \wedge x_b \wedge x_c$ $(a < b < c)$ is contained in clause $B = x_i \wedge x_j \wedge x_k$ $(i < j < k)$ iff $i < a < c < j$ or $j < a < c < k$. Note that the relation is transitive. Let $p(A)$ denote the parent of A, i.e., the clause B that contains A such that there is no other clause C such that C contains A and B contains C. Note that clauses are properly nested and thus preserve planarity.

Connect all clauses with a common parent in a binary tree where edges have weight 3 and clauses are the leaves. *W.l.o.g.*, this tree preserves planarity. Furthermore, it ensures that if a clause can be built then all clauses with the same parent can be built. We then connect the root of the tree with the parent clause of the leaves. We introduce vertices, r_p and r_n, corresponding to the null parents of the clauses with positive and negative literals, respectively, and an edge (r_p, r_n) of weight 3. Note that since we did not connect x_n and x_1 above, this edge also preserves planarity.

The maximum degree of the above construction is 6. Each clause has three edges due to connections with literals (if the clause has two literals, one literal contributes two edges). One edge connects the clause to its parent and to the root of at most two trees of contained clauses. The literals have degree at most 6 and all other vertices have degree at most 5.

For graph G, a maximal order of assembly has the following properties.

- For each variable x, the vertex s_x is built. Therefore, vertex x or $\bar{x}$ (or both) is also built.
- Assume there is a built clause and let C be the first such clause in the ordering. Then all literals participating in C are built before C.
- If a clause is built then all clauses are built: Recall all clauses are connected by weight 3 edges and adjacent to only positive edges.
- If all clauses are built then all vertices are built: Since all clauses and all s_x's are built, each literal receives contribution of at least $3+(-1)+1 \geq 3$. After the literals are built, all of the remaining vertices can be built.
- If no clause is built then exactly one of x and $\bar{x}$ is built, for each variable x. Such a partial assembly corresponds to a certificate that $\overline{\phi}$ is not a tautology.

Hence, we obtain the following theorem.

Theorem 6. 6-`Deg.` 3-`Wt.` `Robust AGAP` on planar graphs *is* co-NP*–complete.*

Using $\tau = 3$ and only two edge weights, we can modify the above construction but the resulting maximum degree will be 9 (see Figs. 5 and 6).

Corollary 2. 9-`Deg.` 2-`Wt.` `Robust AGAP` on planar graphs *is* co-NP*–complete.*

4 `Robust AGAP` on Grid Graphs

Grid graphs are of particular interest due to their correspondence to the Tile Assembly Model. For completeness, we mention that `AGAP` on grid graphs with 3 weights is NP-complete by embedding on a grid the hardness construction of [14] for planar graphs of maximum degree 3 and 2 weights (the third weight is introduced to pad the construction to be a grid graph).

A qualitative difference between `AGAP` and `Robust AGAP` is that in instances that assemble robustly, finding a feasible order of assembly is easy, i.e., a simple algorithm of building any vertex (which is feasible at the time) should be able to find such an order of assembly. On the other hand, it is enough to show one maximal ordering on vertices that only partially assembles the input graph to certify a graph is a NO instance for the `Robust AGAP`.

For our analysis, we introduce the recurring notions of *inextensibility* and *forbidden structures*.

Definition 4 (Inextensibility). *Given an accretive graph assembly system $\langle G, v_s, w, \tau\rangle$, a subgraph G' of G with $V(G') \subsetneq V(G)$ is called inextensible if G' can be assembled starting from the seed vertex without building any vertex in $V(G) \setminus V(G')$, and once G' is built no other vertex can be added to the assembly. Such an order of assembly of G' is referred to as* inextensible ordering.

Remark 1. We assume that each vertex is reachable through a path of positive edges. Otherwise it is clear that the instance cannot be assembled.

Definition 5 (Forbidden Structure). *Let G be a grid graph and H a connected subgraph of G. We call $v \in V(H)$ a* boundary *vertex if v is on the grid boundary or $\exists u \in V(G) \setminus V(H)$ such that $(v,u) \in E$. We say H is a forbidden structure if the seed vertex $v_s \notin V(H)$ and, for each boundary vertex v, $\sum_{u \in (V(G) \setminus V(H)) \cap \Gamma(v)} w(v,u) < \tau$, where $\Gamma(v)$ denotes the set of vertices adjacent to v. The size of H is $|V(H)|$.*

Intuitively, the boundary vertices of a forbidden structure can be made infeasible by assembling all the outside neighbors of these vertices. The following theorem gives a *sufficient* condition when these neighbors can be assembled, and hence gives a *partial* characterization of `Robust AGAP` in terms of forbidden structures.

Theorem 7. *If G cannot robustly self-assemble, then there exists a forbidden structure. Conversely, consider a forbidden structure H in G. Let B denote the set of boundary vertices of H, and $\Gamma(B)$ the set of vertices in $V(G) \setminus V(H)$ which are adjacent to some vertex in B or are on one diagonal from a vertex of B. Then if every edge (u,v) such that $u \in \Gamma(B)$ and $v \in V(G) \setminus V(H)$ has weight at least τ, G cannot robustly self-assemble.*

Proof. The first part follows from the definition of forbidden structure. For the second part, consider a forbidden structure H with a maximum number of vertices on the grid boundary. Note that in this case the subgraph induced by $\Gamma(B)$ is connected and have positive edges only. Furthermore, every path from the seed to a vertex in $V(H)$ crosses $\Gamma(B)$. Fix an order of assembly for G and consider the first time it reaches a vertex in $\Gamma(B)$. Since no vertex in $V(H)$ is built at this point, we can build all vertices in $\Gamma(B)$ without using any vertex in $V(H)$. Since $\Gamma(B)$ includes all outside neighbors of H, this ordering makes H infeasible. □

4.1 `Robust AGAP` on Grid Graphs with 2 Weights

We now focus on `Robust AGAP` on grid graphs with two possible edge weights. The case when there is only one possible weight is trivial, i.e., the graph robustly self-assembles iff this weight is $\geq \tau$. Table 1 summarizes all possible cases when there are two possible edge weights, $w_p \geq \tau$ and $w_n < 0$. Note that when

Table 1. Cases of `Robust AGAP` on grid graphs with 2 edge weights $w_p \geq \tau$ and $w_n < 0$

Case	Results
$\tau \leq w_p + 3w_n$	Poly-time solvable
$\tau \in (w_p + 3w_n, w_p + 2w_n]$	Poly-time solvable
$\tau \in (w_p + 2w_n, w_p + w_n]$	Open problem
$\tau \in (w_p + w_n, 2w_p + 2w_n]$	Open problem
$\tau \in (2w_p + 2w_n, 2w_p + w_n]$	Open problem
$\tau \in (2w_p + w_n, 3w_p + w_n]$	Poly-time solvable (Theorem 9)
$\tau > 3w_p + w_n$	Poly-time solvable

both weights are positive the instance is trivial [11]. When there is only one positive weight, it must be at least τ, otherwise the instance is not feasible. When $\tau \leq w_p + 3w_n$, the graph G robustly self-assembles iff there is a spanning tree of positive edges (see Remark 1) since even 3 negative neighbors cannot make a vertex infeasible. If $w_p + 3w_n < \tau \leq w_p + 2w_n$, then we can show G robustly self-assembles iff there does not exists a vertex (other than the seed vertex) with 3 negative edges incident on it. Furthermore, if $3w_p + w_n < \tau$ then the graph G robustly self-assembles iff there is no negative edge in G.

The remaining cases for 2 edge weights appear nontrivial. In what follows, we make progress towards understanding the complexity of those cases by giving a poly-time algorithm that solves one of the cases. In our analysis, we will assume that the seed vertex is connected to its neighbors in G with positive weight edges.

4.2 `Robust AGAP` on Grid Graphs with $2w_p + w_n < \tau \leq 3w_p + w_n$

We now consider a nontrivial case of `Robust AGAP` on grid graphs when there are 2 edge weights, w_p and w_n, such that $2w_p + w_n < \tau \leq 3w_p + w_n$. We show that this case is poly-time solvable since in this case the existence of a forbidden structure is both sufficient and necessary condition of the fact that G cannot robustly assemble. Therefore, we first categorize forbidden structures into groups and then proceed with a theorem giving the desired characterization.

Definition 6 (Nearby Negative Edges). *A pair of disjoint edges e_1 and e_2 with* negative *weights are* nearby *iff they have adjacent nodes (see Fig. 7(a)).*

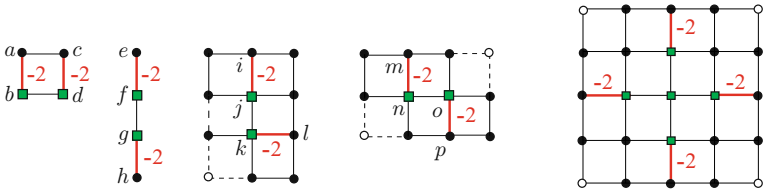
Theorem 8. *If there is a forbidden structure in G with $2w_p+w_n < \tau \leq 3w_p+w_n$ then at least one of the following conditions holds:*

(i) there is a negative path of length 2 or more, or
(ii) there is a negative edge with at least one end-point on the grid boundary, or
(iii) there is an elementary forbidden structure (shown in Fig. 7).

Furthermore, if (i), (ii), or (iii) holds, then G cannot robustly self-assemble.

Proof (of the first part of Theorem 8). If we have a negative path of length 2 or a negative edge with at least one of its end-point on the grid boundary, the statement is trivial. In fact, these are the only forbidden structures of size 1 (see Definition 5). Assume now that there are neither negative paths of length 2 nor negative edges with at least one end-point on the grid boundary.

Consider the boundary vertices of the forbidden structure. If there are two boundary vertices which are adjacent to each other, then we have nearby negative edges. Thus the only possibility is that the boundary vertices are from at distance two from each other (on the diagonal). Also note that if there are three boundary vertices on the same diagonal line, the middle vertex will be an end-point of a negative edge which has a nearby edge (one of the edges with an end-point on the diagonal). Hence, the only remaining possibility is if the boundary of the forbidden structure consists of vertices which are distance two from each other



(a) Nearby negative edges (b) Non-nearby edges

Fig. 7. Elementary forbidden structures for weights $\{-2, 1\}$ and $\tau = 1$. Solid edges without annotation have weight 1; dashed edges have weight 1 or -2. The vertices of the forbidden structures are sown with square nodes.

(only two vertices on each diagonal). The only such structure is formed by 4 vertices vertices arranged in a diamond shape (if one of the 4 vertices is missing we can always complete the structure). In this case, depending on the orientation of the incident edges, we either have nearby edges or the elementary forbidden structure in Fig. 7(b).

Now we prove the second part of the theorem in the following lemmas.

Lemma 1. *Let (u, v) be a negative edge and let p be a neighbor of u that has common neighbors with v. If π is a feasible order of assembly such that $p \prec u \prec v$ in π, then either there is an ordering (possibly inextensible) where v is built before u, or there is an ordering when u is built but v is made infeasible.*

Proof. Note that if (u, v) is a negative edge and u is built before v, then when v is built, it must have 3 neighbors connected with positive edges that have already been built. Now consider the time when p is built in π. If we cannot build the common neighbor, say q, of p and v, then by building u we make v infeasible. This is because q depends on v to be built and vice versa. Now if q can be built, we can either build v, or by building u before, we make v infeasible as it has at most two neighbors connected with positive edge.

Lemma 2. *If there is a negative edge with at least one end-point on the grid boundary or if there is a path of negative edges of length at least* 2 *then the grid cannot robustly self-assemble.*

Proof. Consider a negative edge (u, v) with at least one end-point on the grid boundary. If both u and v are on the boundary then there is no feasible order of assembly since by building one of the vertices, we make the other one infeasible. When only one end-points is on the boundary, say u, then it must be the case $u \prec v$ in any order of assembly since $2w_p + w_n < \tau$. However, since any neighbor of such u is as in Lemma 1, we can either built v before u or make v infeasible.

Now assume that all negative edges have end-points strictly inside the grid. Let $P = \{(v, u), (u, w)\}$ be a negative path. Consider a feasible order of assembly π. Clearly, $u \prec \{v, w\}$. Since each neighbor of u other than v and w is as in Lemma 1 with respect to either v or w, the claim follows.

Lemma 3. *If there are two nearby negative edges, then the grid cannot robustly self-assemble.*

Proof. W.l.o.g., the premise conditions of Lemma 2 do not hold. We proceed by case analysis on the types of nearby edges given in Fig. 7(a). When there are two parallel nearby negative edges (a, b) and (c, d), consider the first vertex from $\{a, b, c, d\}$ that is built, say a. Now, building c right after a completes the forbidden structure $\{b, d\}$. For what follows, we assume there are no parallel nearby negative edges.

Now consider the case of nearby edges (e, f) and (g, h) with forbidden structure $\{f, g\}$. Consider the first time in some feasible ordering of assembly, a vertex adjacent to say one of $\{e, f\}$ other than g is built. If it is a neighbor of e we can extend the ordering so far to build e. If it is a neighbor of f, we use Lemma 1 to argue that e can be built before f, otherwise e can be made infeasible. Similarly, we can argue that we can build h before g. Note that if after building e, we cannot reach a neighbor of $\{g, h\}$ then the resulting ordering is inextensible. For what follows, we assume there are no such nearby edges.

It remains to argue that if there is any of the remaining combinations of nearby edges (forbidden structures) then there is an inextensible ordering. Consider an order of assembly up to the point where we reach a vertex at distance 1 from a vertex of nearby edges for the first time. Call this vertex r.

First, consider the case of nearby edges (i, j) and (k, l). If r is the North neighbor of i, after r we can build i and follow clockwise the black nodes to build l. Similarly, if r is the East neighbor of l we build l first and proceed in counterclockwise direction to build i. For the remaining choices of r, we can follow the unique paths on the black nodes from r to i and from r to l that do not include the empty node (which might have a neighbor that is built and connected with a negative edge to it). The black nodes are such that if they cannot be built along this path (because of negative edge) it would contradict the choice of r since we reached such neighbor of nearby edges earlier. A special care is needed for the edges shown with dashed line which might be negative edges. However, the only possibility an end-point of such an edge is included in the above paths is if it coincides with r, contradicting the choice of r.

Now, let the reached nearby edges be in configuration as (m, n) and (o, p). Furthermore, *w.l.o.g.* there is no reached configuration as in the previous case (i.e., none of the horizontal dashed edges are negative). Applying the same argument as above does not work since a path from m to p always includes one of the depicted empty nodes and furthermore it can be blocked by the vertical dashed edges if negative. Suppose both vertical dashed edges are negative edges. Then, depending on where r is, we can build all vertices in the row of m (resp. p) making o (resp. n) infeasible. Now assume that at most one of the vertical dashed edges is a negative edge, say the leftmost one. We can show that even if the empty node on the row of m had a neighbor connected with negative weights that is already built we can construct m and p from r. We note that when we try to use such "disabled" vertices it is the case that they are not part of nearby edges. Our argument uses the fact that if we have a negative edge (u, v) that is

not nearby other negative edge, if we build v we can build the two horizontal (resp. vertical) neighbors of u by building the horizontal (resp. vertical) neighbors of v. For example, if the negative edge that disables one of the empty nodes in the Fig. 7(b) is horizontal and r is adjacent to m, then we can build the east vertex of o and therefore the east neighbor of p and p itself. The cases when the negative edge is vertical or r is some other vertex are slightly more involved (we need to argue for at most two non nearby edges) but use the same ideas.

Lemma 4. *If there is a forbidden structure shown in Fig. 7(b) the grid cannot robustly self-assemble (where the seed vertex is not one of the square nodes).*

Proof. W.l.o.g., we can assume that there are no nearby negative edges, otherwise the claim follows from Lemma 3. Consider the first time a round node is reached on this structure. Since there are no nearby negative edges, the round end-points of negative edges shown Fig. 7(b) can be built. Now, the square end-points of those edges cannot be built since the center vertex cannot be built.

Since the conditions of Theorem 8 are poly-time testable (and hence existence of forbidden structures is poly-time decidable), we obtain the following theorem.

Theorem 9. `Robust AGAP` on grid graphs *is poly-time solvable when* $2w_p+w_n < \tau \leq 3w_p + w_n$.

References

1. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544 (1998)
2. Rothemund, P.: Using lateral capillary forces to compute by self-assembly. Proc. Nat. Acad. Sci. U.S.A. 97, 984–989 (2000)
3. LaBean, T.H., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J.H., Seeman, N.C.: Construction, analysis, ligation, and self-assembly of DNA triple crossover complexes. J. Amer. Chem. Soc. 122, 1848–1860 (2000)
4. Yan, H., LaBean, T.H., Feng, L., Reif, J.H.: Directed nucleation assembly of DNA tile complexes for barcode-patterned lattices. Proc. Nat. Acad. Sci. U.S.A. 100, 8103–8108 (2003)
5. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology 2, 2041–2053 (2004)
6. Chelyapov, N., Brun, Y., Gopalkrishnan, M., Reishus, D., Shaw, B., Adleman, L.M.: DNA triangles and self-assembled hexagonal tilings. J. Amer. Chem. Soc. 126, 13924–13925 (2004)
7. He, Y., Chen, Y., Liu, H., Ribbe, A.E., Mao, C.: Self-assembly of hexagonal DNA two-dimensional (2D) arrays. J. Amer. Chem. Soc. 127, 12202–12203 (2005)
8. Malo, J., Mitchell, J.C., Vénien-Bryan, C., Harris, J.R., Wille, H., Sherratt, D.J., Turberfield, A.J.: Engineering a 2D protein-DNA crystal. Angewandte Chemie International Edition 44, 3057–3061 (2005)
9. Rothemund, P.W.K., Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In: STOC, pp. 459–468 (2000)
10. Wang, H.: Proving theorems by pattern recognition II. Bell Systems Technical Journal 40, 1–41 (1961)

11. Reif, J.H., Sahu, S., Yin, P.: Complexity of graph self-assembly in accretive systems and self-destructible systems. In: Carbone, A., Pierce, N.A. (eds.) DNA 2005. LNCS, vol. 3892, pp. 257–274. Springer, Heidelberg (2006)
12. Winfree, E., Bekbolatov, R.: Proofreading tile sets: Error correction for algorithmic self-assembly. In: Chen, J., Reif, J.H. (eds.) DNA 2003. LNCS, vol. 2943, pp. 126–144. Springer, Heidelberg (2004)
13. Chen, H.-L., Goel, A.: Error free self-assembly using error prone tiles. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 62–75. Springer, Heidelberg (2005)
14. Angelov, S., Khanna, S., Visontai, M.: On the complexity of graph self-assembly in accretive systems. Natural Computing 7, 183–201 (2008)
15. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. 11, 329–343 (1982)
16. Middleton, A.A.: Computational complexity of determining the barriers to interface motion in random systems. Phys. Rev. E 59, 2571–2577 (1999)