On the d-Runaway Rectangle Escape Problem

Aniket Basu Roy[*], Sathish Govindarajan[*], Neeldhara Misra[*], Shreyas Shetty[*]

Abstract

In the Rectangle Escape problem, we are given a set of rectangles S in a rectangular region R, and we would like to extend these rectangles to one of the four sides of R while ensuring that the maximum number of overlaps is minimized. More formally, define the density of a point p in R as the number of extended rectangles that contain p. The question is then to find an extension with the smallest maximum density.

We consider the problem of maximizing the number of rectangles that can be extended when the maximum density allowed is at most d. It is known that this problem is polynomially solvable for $d = 1$, and NP-hard for any $d \geq 2$. We consider approximation and exact algorithms for fixed values of d. We also show that a very special case of this problem, when all the rectangles are unit squares from a grid, continues to be NP-hard for $d = 2$.

1 Introduction

The Rectangle Escape problem was introduced in [9], and was further explored in [2]. In its original formulation, the problem is the following. Let S be a set of rectangles in a rectangular region R. The goal is to extend these rectangles to one of the four sides of R while ensuring that the maximum number of overlaps is minimized. In particular, define the density of a point p in R as the number of extended rectangles that contain p. The question is then to find an extension with the smallest maximum density.

The problem finds its motivation in a closely related escape routing problem on the bus levels in PCBs. A detailed exposition of how the formulation above captures the essence of the bus-routing problem is provided in [2].

It turns out, by the combined results in [9, 2], that this question is intractable — indeed, it is NP-hard to determine if all rectangles can be extended with density at most d for any *fixed* $d \geq 2$ (the result was known for $d \geq 3$ in [9] and was established for $d = 2$ in [2]), even when the given set of rectangles are disjoint to begin with. However, the case when $d = 1$ is solvable in polynomial time — the first proposed algorithm from [8] had a running time of $O(n^6)$. Subsequently, Assadi et al. demonstrate a dynamic programming approach with an improved running time of $O(n^4)$ in [2]. The recursive formulations in the DP involve finding the maximum number of rectangles that can be routed in a given subset of directions while being completely disjoint in their extended state. For the problem of optimizing the density, a factor-4 approximation is known in general (by standard rounding techniques), and a PTAS can be obtained when the optimal density is high. We refer the readers to [2] for a more precise formulation.

In certain scenarios, the density of any point p in R cannot exceed a threshold value d, which is fixed by practical considerations. Here, the natural question is to maximize the number of rectangles that can be extended, subject to this fixed density d. This problem is clearly NP-hard for $d \geq 2$, since it is NP-hard to determine if the OPT in this setting is equal to n. We explore this problem from the point of view of approximation and fixed-parameter tractability. On the approximation front, we show that if the rectangles are disjoint, then we have a $4(1 + 1/(d - 1))$-approximation for the problem, and in general, we have a $(4d)$-approximation.

We also analyze the problems from a parameterized perspective. In this setting, each problem instance comes with a parameter k, and the central notion is *fixed parameter tractability* (FPT) which means, for a given instance (x, k), solvability in time $f(k) \cdot p(|x|)$, where f is a computable function of k and p is a polynomial in the input size $|x|$.

The decision version of the Rectangle Escape problem may be informally stated as follows: are there at least k rectangles that can be extended with density at most d? There are two natural parameters for this problem; namely k, the number of rectangles that we wish to extend, and d, the maximum density that is allowed. Since the problem is NP-complete even for constant values of d, we do not expect this problem to be fixed-parameter tractable parameterized by d alone. On the other hand, we show that when parameterized by k, for fixed d, the problem is indeed fixed-parameter tractable, as long as the input rectangles have density at most $(d - 1)$.

We also consider the following closely related question: can we extend at least p non-boundary rectangles horizontally (i.e, towards the right or left), and at least q non-boundary rectangles vertically (i.e, towards the top or bottom)? A non-boundary rectangle is one

[*]Department of Computer Science and Automation, Indian Institute of Science, aniket.basu|gsat|neeldhara|shreyas.shetty@csa.iisc.ernet.in

that doesn't share an edge with the boundary of R. It is natural to consider only non-boundary rectangles in our demand for extension, since the ones on the boundary, without loss of generality, can be "extended" to the boundary that they are on. We show that this problem is W[1]-hard, which implies that a fixed-parameter tractable algorithm does not exist unless the Exponential Time Hypothesis fails.

Finally, we consider the version of Rectangle Escape when all the rectangles are unit squares aligned to an underlying grid. For this problem when $d = 2$, we show a non-trivial reduction from a variant of Not-All-Equals SAT, establishing NP-hardness, and demonstrate that the problem enjoys a 2-factor approximation algorithm.

2 Preliminaries

Let S be a set of rectangles in a rectangular region R. For $T \subseteq S$, let $\Gamma(S,T)$ be obtained from S by extending all the rectangles in T to one of the four borders of R. We call $\Gamma(S,T)$ the *extended configuration of S with respect to T*. Further, we say that $\Gamma(S,T)$ has *density at most d* if every point in R is contained in at most d rectangles in the extended configuration.

For a fixed d, the size of the largest subset T for which the density of $\Gamma(S,T) \leq d$ is called the *runaway number of S with respect to d*, which we denote by $\rho(S,d)$. We study the following optimization version of the Rectangle Escape problem:

d-Runaway Rectangle Escape
Input: A set of n rectangles S in a rectangular region R, and an integer k.
Question: Is $\rho(S,d) \geq k$?

The exact algorithms are considered in the framework of parameterized complexity. We only introduce the terminology that we use in this work, the reader is referred to the books [10, 7, 4] for a comprehensive exposition. A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$, where Γ is a finite alphabet. An instance of a parameterized problem is a tuple (x,k), where k is called the parameter. A central notion in parameterized complexity is *fixed-parameter tractability (FPT)* which means, for a given instance (x,k), decidability is in time $f(k) \cdot p(|x|)$, where f is a computable function of k and p is a polynomial in the input size. We now define the notion of parameterized reduction.

Definition 1 *Let A, B be parameterized problems. We say that A is (uniformly many:1)* ***fpt-reducible*** *to B if there exist functions $f, g : \mathbb{N} \to \mathbb{N}$, a constant $\alpha \in \mathbb{N}$ and an algorithm Φ which transforms an instance (x,k) of A into an instance $(x', g(k))$ of B in time $f(k)|x|^\alpha$ so that $(x,k) \in A$ if and only if $(x', g(k)) \in B$.*

A parameterized problem is considered unlikely to be fixed-parameter tractable if it is $W[i]$-hard for some $i \geq 1$. To show that a problem is $W[1]$-hard, it is enough to give a parameterized reduction from a known $W[1]$-hard problem. It is well known that the parameterized version of the Maximum Clique problem is $W[1]$-hard. In particular, we use the following popular variant of the problem [6]:

Multi-Colored Clique
Input: A graph G whose vertex set is partitioned into k parts, $V_1 \uplus \cdots \uplus V_k$.
Question: Is there a subset S of vertices such that $G[S]$ is a clique and $|S \cap V_i| = 1$ for all $1 \leq i \leq k$?
Parameter: k.

3 An Approximation Algorithm

Let (R,S,d) be an instance of d-Runaway Rectangle Escape. In this section, we consider the optimization version of the problem, where the goal is to extend the maximum number of rectangles with density at most d. Let T denote an optimal solution. Note that at least half the rectangles in T are pushed either horizontally or vertically. So we consider the following problem: given the rectangles S, what is the largest number of rectangles that can be extended vertically with density at most d? We show that this can be approximated to within a factor of $2d$, and repeating the argument along the horizontal direction, and reporting the best of both solutions leads us to a $(4d)$-approximation overall.

We remark that in [2], the problem of determining if at least k rectangles can be extended vertically with density one (that is, with no overlapping rectangles) is shown to be polynomially solvable. They use a natural greedy strategy: consider the rectangles in the order of decreasing y-coordinate of the bottom edge, and let this order be $R_1, \ldots, R_n$. For $1 \leq i \leq n$, we try to extend R_i upwards if this causes no conflicts, else we attempt to extend it downwards. If R_i is blocked in both directions, we choose not to extend it and move to the next rectangle on the list. However, this strategy does not work as-is, for instance, when $d = 2$.

As a preprocessing step, we will first forbid some rectangles from consideration. For a rectangle Y whose x-projection is given by the interval (a,b), let us denote by $\mathcal{B}(Y)$ the intersection of R with the region between the lines $x = a$ and $x = b$. Call a rectangle Y *stuck* if there are points ℓ_1, ℓ_2 of density d (in the input configuration) contained in $\mathcal{B}(Y)$, with ℓ_1 above Y and ℓ_2 below Y. Note that the set of stuck rectangles do not participate in any solution. We now turn our attention to the remaining rectangles, which we refer to as "good" rectangles.

We begin by considering the projections of the good rectangles on the x-axis and arrange them according to their left endpoints. Choose a maximum independent set among these intervals by greedily choosing the intervals that end the earliest (and eliminating intervals that overlap with the chosen one). Let $\{X_1, \ldots, X_p\}$ be such an independent set, arranged according to their right endpoints, and let $\chi := \{T_1, \ldots, T_p\}$ denote the rectangles from S corresponding to this independent set. Let a_i and b_i denote, respectively, the left and right endpoints of X_i.

For a good rectangle Y, we say that Y is a child of X_i if the right endpoint of Y is at least b_i and (if $i+1 \leq n$) strictly less than b_{i+1}. Along these lines, we say that the children of X_i are the set of rectangles whose parent is X_i. Notice that every rectangle not in χ is the child of exactly one rectangle from χ. Thus, we may partition the set of rectangles into $\mathcal{R}_1 \uplus \cdots \uplus \mathcal{R}_p$, where $\mathcal{R}_i$ denotes the set of children of the set X_i.

Note that an optimal solution can extend at most $2d$ rectangles from $\mathcal{R}_i$. Indeed, if not, then at least $(d+1)$ rectangles in $\mathcal{R}_i$ are extended either upwards or downwards. The extended solution has density $(d+1)$ either on the top or the bottom, which is a contradiction. Therefore, if k denotes the size of the optimal solution, we have $k \leq 2dp$ (recall that the rectangles that are stuck do not contribute to any optimal solution).

On the other hand, the rectangles in χ can be extended either upward or downward — for each $T_i \in \chi$, there is at least one direction in which T_i can be extended with density at most d. Also, since the x-projections of the T_i's were non-overlapping, their extensions in the vertical directions also do not overlap with each other. Therefore, we have a solution with at least p rectangles, which is a $(2d)$-approximation to the optimal solution when the directions are vertically constrained. A similar argument holds for the problem of extending rectangles in the horizontal direction, and the better of the two solutions is a $(4d)$-approximation overall.

Improved Approximation With Disjoint Rectangles. We consider the d-Runaway Rectangle Escape when the rectangles are disjoint i.e., the input points have density at most unity to begin with. We recall that this problem remains NP-complete due to the reduction in [2] for all $d \geq 2$. For this case, we obtain an $4(1+1/(d-1))$-approximation algorithm. Let $\mathcal{S}$ be the optimal solution size for the given instance of the Disjoint d-Runaway Rectangle Escape problem. We restrict the extensions to one of the four directions at a time and choose the maximum among them. Let $\mathcal{S}_\lambda$ denote the maximum number of rectangles that can be extended in direction λ, where $\lambda \in \{\text{left, right, up, down}\}$, and let $\mathcal{S}^\dagger$ denote the maximum value of $\mathcal{S}_\lambda$. Clearly, $\mathcal{S}^\dagger \geq \mathcal{S}/4$. We now approximate $\mathcal{S}^\dagger$.

A *d-fold packing* is a collection of sets from a set system such that no element from the ground set is contained in more than d sets [3]. It is well known that an optimal d-fold packing for a system of intervals on the real line can be obtained in polynomial time [5].

Without loss of generality, let us assume that the restriction towards the top gives the maximum among the four directions. As before, let $\mathcal{S}^\dagger$ denote the maximum number of rectangles that can be extended upwards with density d. Let $\mathcal{I}$ denote the projections of the input rectangles on the x-axis. Consider an optimal $(d-1)$-fold packing of $\mathcal{I}$. Since the input rectangles are disjoint, the upward extensions of the rectangles corresponding to this packing constitute a feasible solution. Let OPT_{d-1} denote the size of this solution. The approximation algorithm outputs the rectangles corresponding to this solution.

We note that a d-fold packing may not be a feasible solution when the corresponding rectangles are extended upwards. This is because we may have an input rectangle positioned such that it intercepts a density d region from the extension, causing the overall density to "spill over" to $(d+1)$. On the other hand, since a d-fold packing on the interval projection can be obtained from an upward extension of density at most d, we have:

$$OPT_{d-1} \leq \mathcal{S}^\dagger \leq OPT_d$$

Now, we use the fact that any d-fold packing of intervals is a disjoint union of d independent sets of intervals (see, for example, [5]). Let $C_1, C_2, ..., C_d$ be the independent sets in an optimal d-fold packing of $\mathcal{I}$, where we index them in non-increasing order of their sizes. Let $t_i = |C_i|$ and also $t_i \geq t_{i+1}$ for $1 \leq i < d$. Thus, $OPT_d = \sum_{1 \leq i \leq d} t_i$ and $OPT_{d-1} \geq OPT_d - t_d$, since removing an independent set from a d-fold packing yields a $(d-1)$-fold packing. Now, by an averaging argument, we have $t_d \leq OPT_d/d$. Thus, $OPT_{d-1} \geq OPT_d(1-1/d)$. Hence, the following holds.

$$OPT_{d-1} \geq \mathcal{S}^\dagger(1 - 1/d)$$

Relating $\mathcal{S}^\dagger$ to $\mathcal{S}$ we have:

$$OPT_{d-1} \geq \frac{\mathcal{S}}{4(1+\frac{1}{d-1})},$$

giving us the desired approximation ratio.

Theorem 1 *There exists a polynomial time $4(1+\frac{1}{d-1})$-approximation algorithm for the* Disjoint d-Runaway Rectangle Escape *problem.*

4 Parameterized Algorithms and Hardness

In this section, we consider the d-Runaway Rectangle Escape problem parameterized by k. Note that if the input contains a point of density greater than d, then we have a trivial No-instance. On the other hand, we show that if all points in R have density at most $(d-1)$ in the input configuration, then the problem is FPT. Unfortunately, this algorithm does not immediately extend to accommodate the situation when the input may have points of density d.

We also consider a natural variation of this problem, for which we obtain a parameterized hardness result. Let us call a rectangle *internal* if none of its sides coincide with the boundaries of R. We introduce the following question:

d-Constrained Runaway Rectangle Escape

Input: A set of n rectangles S in a rectangular region R, and integers p, q.
Question: Is it possible to extend at least p internal rectangles along the horizontal axis, and at least q internal rectangles along the vertical axis, such that the extended configuration has density at most d?
Parameter: $p + q$

We show that this particular variant is in fact W[1]-hard by a reduction from Multi-Colored Clique, even when $d = 2$.

Fixed-Parameter Tractability. Let (R, S, k, d) be an instance of d-Runaway Rectangle Escape, where the density of every point in R is at most $(d-1)$ in the input configuration. We recall that this problem is NP-complete since the problem was shown, in [2] to be NP-complete for $d = 2, k = n$ even when all the rectangles are disjoint.

As with the approximation algorithm in the previous section, we consider the projections of the input rectangles on the x-axis and arrange them according to their left endpoints. Choose a maximum independent set among these intervals by greedily choosing the intervals that end the earliest (and eliminating intervals that overlap with the chosen one). Let $\{X_1, \ldots, X_p\}$ be such an independent set, arranged according to their right endpoints, and let $\mathcal{T} := \{T_1, \ldots, T_p\}$ denote the rectangles from S corresponding to this independent set. Let a_i and b_i denote, respectively, the left and right endpoints of X_i. Note that if $p \geq k$, then we may return Yes at this point, since the rectangles in χ can be extended upwards without any mutual conflicts, and this extension will have density at most d because all input points had density at most $(d-1)$ to begin with. Therefore, $p < k$.

We repeat this process on the y-projections of the rectangles, and let $\mathcal{T}' := \{T_1', \ldots, T_q'\}$ denote the rectangles from S corresponding to the independent set obtained in this case. We let (a_i', b_i') denote the top and bottom endpoints of the y-projection of T_i'. Again, we may assume that $q < k$, otherwise we are done. Now consider the lines given by $x = b_i$ for $1 \leq i \leq p$ and $y = b_j'$ for $1 \leq j \leq q$. Let $g(i,j)$ denote the intersection of the lines $x = b_i$ and $y = b_j'$. For every rectangle H in S, observe that there exists $1 \leq i \leq p$ and $1 \leq j \leq q$ such that H contains $g(i,j)$. Indeed, suppose not. Then this would imply, for instance, that H is not stabbed by any of the vertical lines $x = b_i$, which implies that there exists $i \in [p]$ for which the left endpoint of the x-projection of H is after b_i, and the right endpoint is before b_{i+1}. However, this contradicts the greedy construction of χ. A similar argument can be made for the horizontal lines.

Now, we have a collection of less than k^2 points that pierce all the rectangles in S. Since the input density was at most $(d-1)$, each $g(i,j)$ can be contained in at most $(d-1)$ of the input rectangles. Therefore, the total number of rectangles is at most (dk^2). We may now guess the subset of k rectangles that we would like to extend in time:

$$\binom{dk^2}{k} \leq \left(\frac{dk^2 e}{k}\right)^k = (dke)^k.$$

For each guess, we can further guess the direction of the extension of the chosen rectangles — noting that there are at most four possibilities for each rectangle, this is an additional overhead of 4^k. Note that we spend polynomial time in identifying the stabbing lines (and resolving the instance at that stage if it is called for). So overall, the running time of our algorithm is $O((dke)^k 4^k) n^{O(1)}$.

Theorem 2 *The d-Runaway Rectangle Escape problem can be resolved in time $2^{O(k \log k)} n^{O(1)}$ when the input configuration has density at most $(d-1)$.*

Note that the difficulty with input configurations that have points of density d is that having an independent set of size k on the x-projections does not imply that we have a solution, because quite possibly many of the rectangles in the independent set are "blocked" by points of density d. Even if we forbid such rectangles upfront, as in the previous section, we may have an unbounded number of rectangles that are stuck horizontally or vertically. As it turns out, the rectangles that are stuck horizontally *and* vertically pose no problems, because they can be declared forbidden and eliminated from the search space. Similarly, the number of rectangles that are not stuck in either direction can be bounded by $(k^2 d)$ by an argument along the lines of what we had for Theorem 2. However, there may be an unbounded number of rectangles that are stuck only vertically (or only horizontally), and this is where the argument for Theorem 2 does not extend to the case when the input configuration has points of density d.

W-hardness We now turn to the 2-Constrained Runaway Rectangle Escape. Let (G, V, k) be an instance of Multi-Colored Clique, where the partitions of V are given by $V_1 \uplus \cdots \uplus V_k$. We assume, without loss of generality, that all the parts have the same number of vertices. We denote the vertices in V_i by $v_i[1], \ldots, v_i[t]$.

The reader is referred to Figure 1 for a schematic of the construction that we are going to describe. We first introduce the rectangles corresponding to vertices, which we call *selection gadgets*. Every vertex in V is associated with a rectangle of unit width and height $(t+1)$. We use $T_i[j]$ to refer to the rectangle corresponding to the vertex $v_i[j]$, where $1 \leq i \leq k$ and $1 \leq j \leq t$.

We place the bottom-left corner of $T_i[j]$ at $(2+2j, 3+j+(2t+5)(i-1))$. This is simply a collection of t rectangles cascading successively in the top-right direction, with the collection of rectangles for vertices in V_i appropriately offset from the collection corresponding to vertices in V_{i+1}. We use $\mathcal{T}_i$ to refer to $\{T_i[j] \mid j \in [t]\}$. Note that at most two of the rectangles from any $\mathcal{T}_i$ can be extended either to the right or the left (since $d = 2$ and all of these rectangles are stabbed by a single horizontal line). We will refine this observation further with the help of additional rectangles, to ensure that at most one of them can be extended to the right, and none of them in any of the other directions.

We now turn to the edges in G. For each $e_\ell \in G$, we introduce an unit square T_ℓ with its lower left corner at $(3t + 12\ell, (2t + 5)^2)$. Informally, all the squares corresponding to the edges are placed on one horizontal line, suitably spaced out. Further, the y-coordinates of their lower-left corners are large enough to ensure that the squares are placed above all the vertex gadgets.

Next, we add incidence gadgets. These rectangles ensure that if $T_i[j]$ is extended to the right and T_ℓ is extended downwards, then the edge e_ℓ is incident to $v_i[j]$. We first informally describe the setup. Let $B_a[r]$ denote the rectangle obtained by extending $T_a[r]$ towards the right. Let $e_\ell = (v_a[p], v_b[q])$. We will place two rectangles $W_a[p], Z_a[p]$ to the right of $\mathcal{T}_a$ and below T_ℓ. The rectangle $W_a[p]$ will intercept the bands $B_a[r]$ for $r > p$, but will not overlap $B_a[p]$, and the rectangle $Z_a[p]$ will intercept the bands $B_a[r]$ for $r < p$, but again will not overlap $B_a[p]$. This ensures that if $T_a[r]$ is extended to the right and e_ℓ is not incident to r, then a point of density two is created by the overlap of either $W_a[p]$ or $Z_a[p]$ with the extended rectangle $T_a[r]$, thus forbidding T_ℓ from being extended downwards. This process is repeated for the collection $\mathcal{T}_b$.

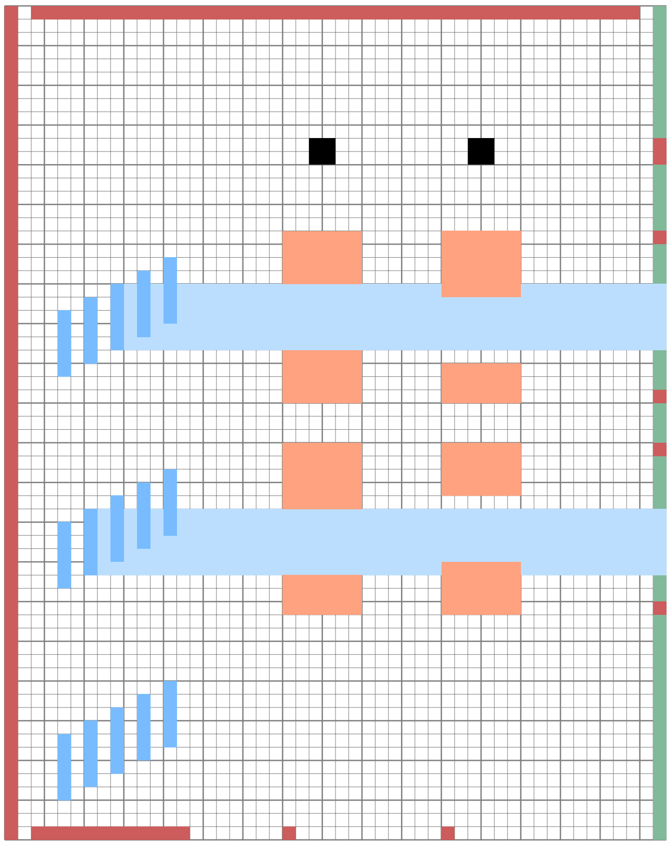
Figure 1: A cross-section schematic, *not* drawn to scale, of the reduction from Multi-Colored Clique. The groups of blue rectangles correspond to vertices from a particular partition in the instance of Multi-Colored Clique. The red rectangles indicate two overlapping rectangles placed along the borders of R, while the green rectangle is a single rectangle, again aligned to the right border of R. The orange rectangles are the incidence gadgets and the black rectangles correspond to the edges.

Formally, for every edge $e_\ell = (v_a[p], v_b[q])$, we place the following four rectangles, which we call *incidence gadgets*. All these rectangles are seven units wide, and the x-coordinate of their lower-left corner is three units less than the x-coordinate of the lower-left corner of T_ℓ. That is, if we consider the x-projections of these four rectangles along with the x-projection of T_ℓ, then we will find the x-projection of T_ℓ exactly at the center, and the remaining four intervals coinciding. We now describe how $W_a[p], Z_a[q]$ are placed along the y-axis, and note that the rectangles $W_b[q]$ and $Z_b[q]$ are placed similarly.

1. The rectangle $W_a[p]$. The y-coordinate of the bottom edge of $W_a[p]$ is the same as the y-coordinate of the upper edge of $T_a[p]$. The y-coordinate of the top edge of $W_a[p]$ is two more than the y-coordinate

of the upper edge of $T_a[t]$, that is, we make sure that this rectangle "juts out" over and above the last rectangle in the group $\mathcal{T}_a$. This will be useful later, when we would like to forbid this rectangle from extending to the right, *without* forbidding any of the rectangles in $\mathcal{T}_i$ from extending to the right.

2. The rectangle $Z_a[p]$. The y-coordinate of the top edge of $Z_a[p]$ is the same as the y-coordinate of the bottom edge of $T_a[p]$. The y-coordinate of the bottom edge of $Z_a[p]$ is two less than the y-coordinate of the bottom edge of $T_a[1]$.

We now incorporate some rectangles along the boundary, which we will refer to as *guards*. The purpose here is to "block" certain extensions. To begin with, we place two overlapping unit-width rectangles along the entire left boundary, and two overlapping unit-height rectangles along the top boundary of R. This ensures, for example, that none of the internal rectangles can be extended to either the left or the top. Further, we place a single rectangle, denoted by H, that covers the entire right boundary (stopping short of the guard rectangles on top to avoid a region of density three).

Next, we would like to ensure that the rectangles $T_i[j]$ can only be extended to the right. To this end, we place two overlapping rectangles of unit height along the bottom boundary of R, wide enough to block any $T_i[j]$ from extending downwards, for $1 \leq i \leq k$ and $1 \leq j \leq t$. Specifically:

- The x-coordinate of the bottom left corner of these rectangles is one less than the x-coordinate of the bottom left corner of $T_1[1]$.
- The x-coordinate of the bottom right corner of these rectangles is one more than the x-coordinate of the bottom right corner of $T_1[t]$.

We add a unit square on the right boundary (overlapping H), whose lower-right corner has the y-coordinate $(2t+5)^2$. This effectively blocks the rectangles corresponding to the edges from extending to the right.

Finally, we add rectangles along the bottom and right boundaries to ensure that the rectangles in the incidence gadgets are blocked from being extended to either the right or the downwards. In this context, we introduce unit squares $H_1, \ldots, H_k$ and $H_1^\dagger, \ldots, H_k^\dagger$, to be placed along the right boundary. The y-coordinate of the upper-right corner of the square H_i is two more than the y-coordinate of the upper edge of $T_i[t]$. This, together with H, ensures that the rectangles $W_i[j]$ are blocked from extending towards the right, for any $1 \leq i \leq k$ and $1 \leq j \leq t$. Similarly, the y-coordinate of the lower-right corner of the square $H_i^\dagger$ is two less than the y-coordinate of the bottom edge of $T_i[1]$. Again, together with H, this ensures that the rectangles $Z_i[j]$ are blocked from extending towards the right, for any $1 \leq i \leq k$ and $1 \leq j \leq t$. Note that these rectangles do not block any rectangles in $\mathcal{T}_i$ from extending to the right, because of their unit height.

For this instance, we let $p = k$ and $q = \binom{k}{2}$. This completes the description of the construction, and we now turn to a proof of correctness. It is useful to keep in mind that the guards are the only rectangles that are not internal.

In the forward direction, let $c_1, \ldots, c_t$, $c_i \in [t]$, be such that the vertices $v_i[c_i]$ form a multi-colored clique. We then extend the rectangles $T_i[c_i]$ to the right, and the unit squares T_ℓ corresponding to the edges of the clique downwards. It is easy to check that the guards do not interfere with any of these extensions, that is, there are no points of density three on the boundary after these rectangles are extended as described. Also, extending the rectangles from the selection gadgets alone creates no points of density greater than two. We now address the edge extensions. Let $e_\ell = (v_i[c_i], v_j[c_j])$ be an edge in the clique. Observe that the rectangles in the incidence gadget corresponding to the rectangle T_ℓ skirt the edges of the bands $B_i[c_i]$ and $B_j[c_j]$, without overlapping them. Therefore, it can be verified that we create no points of density greater than two when the square T_ℓ is extended downwards.

In the reverse direction, we observe that at most one rectangle can be extended to the right from $\mathcal{T}_i$, and none of them can be extended to the left. Further, none of the other internal rectangles can be extended along the horizontal axis while maintaining density at most two. Since we have to extend at least k rectangles along the horizontal axis, it follows that any solution extends exactly one rectangle from each $\mathcal{T}_i$, for $1 \leq i \leq k$. Let $1 \leq c_i \leq t$ be such that $T_i[c_i]$ was the rectangle that was extended to the right. We claim that the vertices $v_i[c_i]$ form a multi-colored clique in G. Indeed, observe that if T_ℓ is extended downwards, where $e_\ell = (v_i[p], v_j[q])$, then the rectangle extended from $\mathcal{T}_i$ must be $T_i[p]$ and the rectangle extended from $\mathcal{T}_j$ must be $T_j[q]$ — indeed, the extension of any other rectangle from either collection will lead to a point of density three (combined with the incidence gadgets for T_ℓ. Therefore, a rectangle corresponding to an edge can be extended downwards only if it is an edge from $G[\{v_1[c_1], \ldots, v_t[c_t]\}]$. Recall that the guard vertices are positioned so that none of the internal rectangles can be extended upwards, and only the squares corresponding to the edges can be extended downwards. Therefore, if the claimed subgraph does not induce a clique, we conclude that the solution falls short of the $\binom{k}{2}$ extensions that were required along the vertical axis. Thus, we have shown the following.

Theorem 3 *The* 2-Constrained Runaway Rectangle Escape *is* $W[1]$*-hard.*

5 The Square Escape Problem

In this section, we look in to a special case of the Rectangle Escape problem, where the rectangular region R is given as a grid of unit squares, and every rectangle is a unit square aligned to the grid. This is same as having grid points instead of squares and orthogonal line segments joining the grid points to the boundary of R instead of extensions of the squares. As it turns out this problem, in the latter guise, has been studied for unit density and an $O(n^2 \log n)$ time algorithm is devised in [11]. We show that despite being a rather severely specialized version of Rectangle Escape, even this formulation is NP-hard for density two. In particular, the problem of determining if all the squares can be extended while maintaining density two is NP-hard, while the "runaway" version enjoys an improved approximation algorithm and is fixed-parameter tractable irrespective of the density of the input squares.

d-Runaway Square Escape
Input: A set S of n squares from an $m \times m$ grid R, and an integer k.
Question: Is $\rho(S, d) \geq k$?

We first show that the d-Runaway Square Escape is NP-hard even when $k = n$ and $d = 2$. We reduce from the version of Not-All-Equals SAT where every clause has two or three variables, all variables appear in their positive form, and every variable occurs in at most three clauses. It is known that not-all-equals satisfiability continues to be NP-complete for this restricted formulation [1].

Before we describe the construction, we introduce some terminology. For a square s located on the i^{th} row and the j^{th} column of the given grid, we use $R(s)$ and $C(s)$ to refer to i and j, respectively. We say we place a square at (i, j) to indicate that a square is placed in the location determined by the intersection of the i^{th} row and j^{th} column.

Suppose we are working on an $m \times m$ grid. When we say that we *block* a square s, say, in the upward direction, then this means that we introduce two overlapping squares at $(m, C(s))$, if they are not already present. We are only allowed to block a square s if there are either no squares at $(m, C(s))$, or if there are two squares at $(m, C(s))$. The terminology is motivated by the fact that when we block a square s in the upward direction, no extension of density at most two can extend s in the upward direction.

When we say that we *partially block* a square s in the upward direction, then this means that we introduce one square at $(m, C(s))$, if not already present. In particular, a square on column j cannot be partially blocked if there are two squares placed already at (m, j). These squares are called *guards*.

We let ϕ denote an instance of Not-All-Equals SAT where every clause has two or three variables and every variable occurs in at most three clauses. Let $v_1, \ldots, v_n$ be the variables involved in ϕ, and let $C_1, \ldots, C_m$ denote the clauses of ϕ.

For every variable, we will introduce three squares corresponding to the variable, which we simply call the variable gadget. We then add more squares to ensure that these three squares are always extended in the same direction, and these collections of squares are called the copy gadgets. Finally, we add three squares for every clause, which we call the clause gadgets.

For a variable v_i, let $\mathcal{V}_i := \{s_i[1], s_i[2], s_i[3]\}$ denote the three squares involved in the corresponding variable gadget. We say we place $\mathcal{V}_i$ at (x, y) to mean that $s_i[1]$ is placed at $(x, y+4)$, $s_i[2]$ is placed at $(x+2, y+2)$ and $s_i[3]$ is placed at $(x+4, y)$. The envelope of a variable gadget that is placed at (x, y), denoted by $\mathcal{E}_i$, is defined as the rectangular region whose lower-left corner is at (x, y) and whose upper-right corner is at $(x+25, y+25)$. All the squares that participate in the copy gadget for $\mathcal{V}_i$ will be placed in $\mathcal{E}_i$. We now describe the individual components of the construction.

Variable Gadgets The variable gadget corresponding to v_1 is placed at $(0, 0)$. The variable gadget corresponding to v_i is placed at the top-right corner of $\mathcal{E}_{i-1}$, for $2 \leq i \leq n$. All the squares in the variable gadgets are blocked downwards and to their left, while they are partially blocked upwards and to their right.

Clause Gadgets Let $C_1, \ldots, C_m$ be an arbitrary but fixed ordering of the clauses. Let $C_j = \{v_{i_1}, v_{i_2}, v_{i_3}\}$ be a clause of length three. Within a clause, we order the variables according to increasing order of their indices. Let C_j be the $f_j[x]^{th}$ clause that v_{i_x} appears in. For example, for a clause $C_3 := \{v_2, v_3, v_7\}$, we may have $f_3[1] = 2$ to denote the fact that C_3 is the second clause that v_2 appears in. Note that $f_j[x] \in \{1, 2, 3\}$ for the particular instance of Not-All-Equals SAT that we have started with.

For this clause C_j, we introduce squares $t_j[1]^U, t_j[2]^U, t_j[3]^U$ and $t_j[1]^R, t_j[2]^R, t_j[3]^R$, placed in the following manner.

1. For $x \in \{1, 2, 3\}$, the square $t_j[x]^U$ is placed in the same column as $s_{i_x}[f_j[x]]$. The row it is placed in is $2j + 25n + 10$. In particular, it is $2j + 10$ units above $\mathcal{E}_n$.

2. For $x \in \{1, 2, 3\}$, the square $t_j[x]^R$ is placed in the same row as $s_{i_x}[f_j[x]]$. The column it is placed in is $2j + 25n + 10$. In particular, it is $2j + 10$ units to the right of $\mathcal{E}_n$.

If we have a clause of length two, then we place squares $t_j[1]^U, t_j[2]^U$ and $t_j[1]^R, t_j[2]^R$ exactly as described above. Further, we add two dummy squares P and Q, where P is placed on the same row as $t_j[1]^U, t_j[2]^U$, and is placed on an empty column c such that $C(t_j[1]^U) < c < C(t_j[2]^U)$. Similarly, Q is placed on the same column as $t_j[1]^R$, and is placed on an empty row r such that $R(t_j[1]^U) < r < R(t_j[2]^U)$. The square P is blocked up and down, while the square Q is blocked on the right and left. We note that if empty rows or columns are not available, then the spacing between the envelopes of the variables can be easily adjusted to free up space. We do not incorporate this detail in the interest of a simpler presentation.

Copy Gadgets Let $\mathcal{V}_i$ be a variable gadget placed at (x, y). Then we introduce the following squares in the copy gadget corresponding to $\mathcal{V}_i$:

- We place squares at $(x, y + 8), (x + 2, y + 12), (x + 2, y+16), (x+4, y+20)$. Further, we place squares at $(x+8, y+2), (x+12, y+4), (x+16, y), (x+20, y+2)$.
- We place squares at $(x+8, y+8), (x+12, y+12), (x+16, y+16), (x+20, y+20)$. We call these the *blockers*.
- For each blocker at (p, q), we place two additional squares at $(p-2, q)$ and $(p, q-2)$. These we call the *anchors*. The anchors at $(p-2, q)$ are blocked upwards and downwards, while the rest of the anchors are blocked to their left and right.

The variable gadgets, their corresponding copy gadgets, and the clause gadgets, together comprise the reduced instance. We now argue the equivalence of the two instances.

In the forward direction, let $\tau : \{v_1, \ldots, v_n\} \to \{0, 1\}$ be a not-all-equals satisfying assignment. If $\tau(v_i) = 1$, then we extend all the squares in $\mathcal{V}_i$ to the right, and if $\tau(v_i) = 0$, then we extend all the squares in $\mathcal{V}_i$ upwards. It can be shown that all the squares in the copy gadgets continue to have a valid extension (see Figure 2 onwards for illustration). It is important to ensure here that for any fixed column (or row), we extend at most one square upwards (or rightwards) along that column (or row). This ensures that all the "crossings" encountered when we proceed to extend the squares corresponding to clause gadgets have density at most two. Also, extending two squares to the left or downwards along any row or column causes no problems, because any potential interference comes from clause gadgets being extended (respectively) downwards or to the left — but the placements of these gadgets are such that these extensions are guaranteed to be parallel, and consequently, noncrossing.

Among the squares $t_j[1]^U, t_j[2]^U, t_j[3]^U$, notice that at most two of them are in locations with density two because at most two of the corresponding squares in the variable gadgets were extended upwards (recall that we start with a not-all-equals satisfying assignment). Therefore, we extend the square that is free upwards, and the other two to the left and right, respectively. A similar argument works for the squares $t_j[1]^R, t_j[2]^R, t_j[3]^R$. Finally, all the guards can be trivially extended to the edge that they are the closest to.

In the reverse direction, we first note that in any valid extension, all the squares in a variable gadget must be extended in the same direction. For example, for any $1 \leq i \leq n$, if $s_i[1]$ and $s_i[2]$ are extended upwards and to the right respectively, then there are corresponding anchor squares that are forced to be extended to the right and upwards, which then create a point of density three at the corresponding blocker square. It can be argued, therefore, that $s_i[1]$ and $s_i[2]$ must be extended in the same direction, and similarly, that $s_i[2]$ and $s_i[3]$ must be extended in the same direction. It follows that all three of them must be extended in the same direction — and since they are blocked on the left and downwards, they must be extended either upwards or to the right.

We suggest an assignment to the variables of ϕ as follows. If the squares in $\mathcal{V}_i$ are extended to the right, then we set v_i to 1 and to 0 otherwise. If this is not a valid not-all-equals assignment, then consider the squares corresponding to a violated clause. Assume, without loss of generality, that all variables in this clause were set to one, therefore, the squares corresponding to the variables were extended to the right. If this was a clause of length three, then observe that the square in the clause gadget corresponding to the second variable is now blocked in all four directions (recall that the squares corresponding to the variables were partially blocked on the right), and cannot be extended. If this was a clause of length two, then the dummy square Q corresponding to the clause is similarly blocked in all four directions. In all four cases, we get the desired contradiction. Thus we have shown the following.

Theorem 4 *d-*Runaway Square Escape *is* NP-complete *even when $d = 2$ and $k = n$.*

On the other hand, we know that for the d-Runaway Square Escape problem, we may find the maximum number of rectangles that can be extended vertically in polynomial time. Indeed, for every column, we extend the d squares "closest to the top" upwards, and the d squares "closest to the bottom" downwards. This is evidently an optimal solution. A similar argument holds for finding the maximum number of rectangles that can be extended horizontally. Since any solution that extends the squares in any of the four directions extends at least half of the squares either vertically or horizontally, we have a simple two-approximation algorithm. It is also easy to check that the fixed-parameter tractable algorithm described for rectangles works for squares with no assumptions on the density of the input configuration.

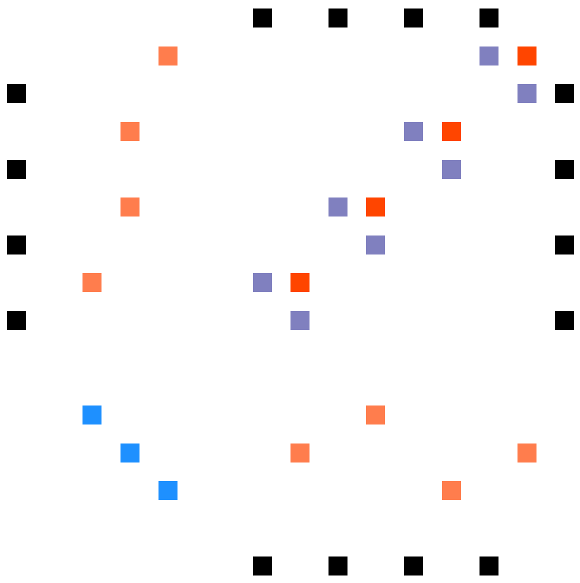
Figure 2: A general schematic of a copy gadget.

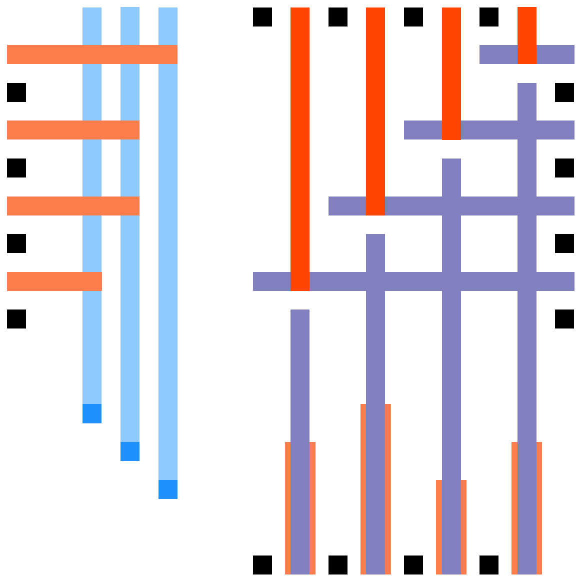
Figure 3: When all copies of a variable are extended upwards.

6 Future Directions

Studying this natural optimization version of Rectangle Escape leads us to several new questions. First, to

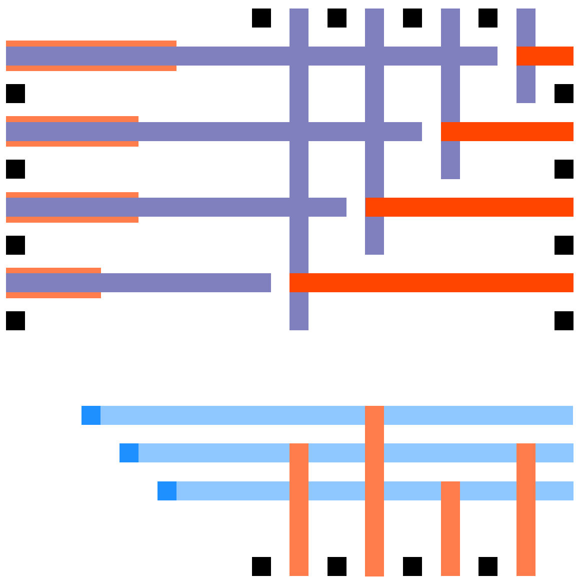
Figure 4: When all copies of a variable are extended to the right.

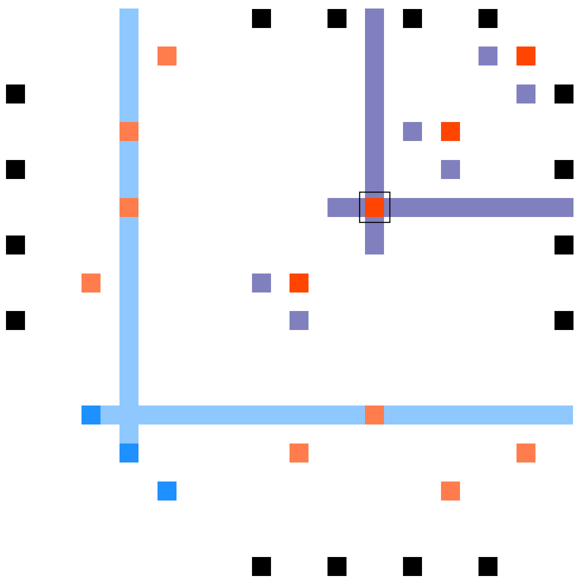
Figure 5: A violation when two copies are extended in different directions.

obtain a constant-factor approximation algorithm that is independent of d, we would like to be able to answer the question of whether at least k rectangles can be pushed along one direction in polynomial time, and further address the question of whether k rectangles can be pushed up or down with density at most d in polynomial time. From the reduction in [9] it can be seen that the question of whether all rectangles can be pushed with density at most three when the only available directions are top and right, is already NP-hard. It would be inter-

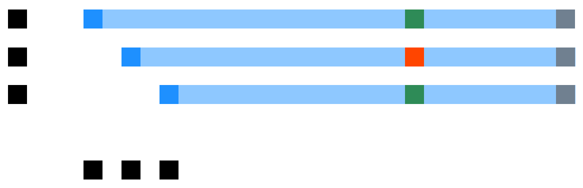

Figure 6: A schematic of the clause gadget.

esting to examine what happens when the combinations of directions that are available are parallel (like up and down, or right and left), and one of the motivations is that this directly impacts the approximation ratio.

For optimizing density, a randomized PTAS is known when the instance has a large value of OPT. We leave open the question of whether the question of extending the maximum number of rectangles for a fixed value of density admits a PTAS, at least for the case when the rectangles are squares from a grid.

There are unresolved questions in the parameterized context as well. For example, is the problem fixed-parameter tractable when the input configuration has points of density d? Further, for the cases when the input configuration has density at most $(d-1)$, the algorithm presented here has a running time of $2^{O(k \log k)} n^{O(1)}$. Can this be improved, for instance, $2^{O(k)} n^{O(1)}$?

A general direction of interest is to obtain substantially improved algorithms for the special case when the rectangles are squares aligned to a grid, for which we establish NP-hardness here.

References

[1] B. M. Anthony and R. Denman. k-Bounded Positive Not All Equal LE3SAT. In *Brown Working Papers*, 2009.

[2] S. Assadi, E. Emamjomeh-Zadeh, S. Yazdanbod, and H. Zarrabi-Zadeh. On the rectangle escape problem. In *Canadian Conference on Computational Geometry (CCCG)*, pages 235–240, 2013.

[3] P. Brass, W. Moser, and J. Pach. *Research Problems in Discrete Geometry.* Springer, 2006.

[4] R. G. Downey and M. R. Fellows. *Parameterized complexity*, volume 3. Springer Heidelberg, 1999.

[5] U. Faigle and W. M. Nawijn. Note on scheduling intervals on-line. *Discrete Applied Mathematics*, 58(1):13–17, 1995.

[6] M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.

[7] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[8] H. Kong, Q. Ma, T. Yan, and M. D. F. Wong. An optimal algorithm for finding disjoint rectangles and its application to pcb routing. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 212–217, New York, NY, USA, 2010. ACM.

[9] Q. Ma, H. Kong, M. D. F. Wong, and E. F. Y. Young. A provably good approximation algorithm for rectangle escape problem with application to pcb routing. In *ASP-DAC*, pages 843–848, 2011.

[10] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[11] L. Palios. Connecting the maximum number of nodes in the grid to the boundary with nonintersecting line segments. *J. Algorithms*, 22(1):57–92, Jan. 1997.