Rectangle Covers Revisited Computationally

LAURA HEINRICH-LITAN
Robert Bosch GmbH
and
MARCO E. LÜBBECKE
Technische Universität Berlin

We consider the problem of covering an orthogonal polygon with a minimum number of axis-parallel rectangles from a computational point of view. We propose an integer program which is the first general approach to obtain provably optimal solutions to this well-studied $\mathcal{NP}$-hard problem. It applies to common variants like covering only the corners or the boundary of the polygon and also to the weighted case. In experiments, it turns out that the linear programming relaxation is extremely tight and rounding a fractional solution is an immediate high-quality heuristic. We obtain excellent experimental results for polygons originating from VLSI design, fax data sheets, black and white images, and for random instances. Making use of the dual linear program, we propose a stronger lower bound on the optimum, namely, the cardinality of a fractional stable set. Furthermore, we outline ideas how to make use of this bound in primal–dual-based algorithms. We give partial results, which make us believe that our proposals have a strong potential to settle the main open problem in the area: To find a constant factor approximation algorithm for the rectangle cover problem.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Linear programming, integer programming

1. INTRODUCTION

A polygon with all edges either horizontal or vertical is called *orthogonal*. Given an orthogonal polygon P, the *rectangle cover problem* is to find a minimum number of possibly overlapping axis-parallel rectangles whose union is exactly

An extended abstract of this paper appeared in the *Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA05)* [Heinrich-Litan and Lübbecke 2005].
Authors' addresses: Laura Heinrich-Litan, Robert Bosch GmbH, Automotive Equipment, Division CM Car Multimedia, Department CM-DI/EAP, Daimlerstrasse 6, D-71229 Leonberg; email: lheinrich-litan@de.adit-jv.com, Laura.Heinrich-Litan@de.bosch.com; Marco E. Lübbecke, Technische Universität Berlin, Institut für Mathematik, Sekr. MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, Germany; email: m.luebbecke@math.tu-berlin.de.

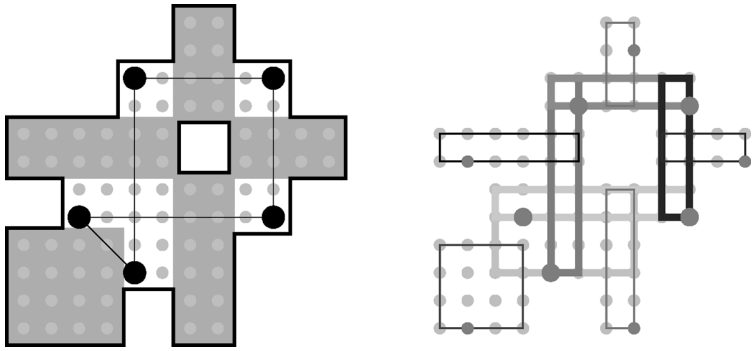

Fig. 1. The original counterexample to $\alpha = \theta$ by Szemerédi and indicated on the right-hand side is the odd whole, an optimal fractional cover. Thicker lines (points) indicate rectangles (pixels), which are picked to the extent of 0.5.

P. In computational geometry, this problem received considerable attention in the past 25 years, in particular, with respect to its complexity and approximability in a number of variants. Still, the intriguing main open question [Bern and Eppstein 1997] is: Is there a constant factor approximation algorithm for the rectangle cover problem? We do not answer this question now, but we offer a different and new kind of reply, which is "computationally, yes" for all our tests. In fact, we provide a fresh experimental view, the first of its kind, on the problem which has applications in the fabrication of masks in the design of DNA chip arrays [Hannenhalli et al. 2002], in VLSI design, and in data compression, and, in particular, in image compression.

1.1 Previous Work

Customarily, one thinks of the polygon P as a union of finitely many (combinatorial) pixels, sometimes also called a polyomino. The polygon P can be associated with a visibility graph G [Maire 1994; Motwani et al. 1989a; Motwani et al. 1989b; Schrijver 2003]: The vertex set of G is the set of pixels of P and two vertices are adjacent in G, if, and only if, their associated pixels can be covered by a common rectangle. Rectangles correspond to cliques in G. A clique is a set of vertices, any two of which are adjacent. We will denote by θ the number of rectangles in an optimal cover. An obvious lower bound on θ is the size α of a maximum stable set in G, also called maximum independent set. This is a set of pixels, no two of which are contained in a common rectangle. In the literature one also finds the notion of an antirectangle set.

Chvátal originally conjectured that $\alpha = \theta$; this is true for convex polygons [Chaiken et al. 1981] and a number of special cases. Szemerédi gave an example with $\theta \neq \alpha$, (see Figure 1). Intimately related to the initially stated open question, Erdős then asked whether θ/α was bounded by a constant. In Chaiken et al. [1981], an example is mentioned with $\theta/\alpha \geq 21/17 - \varepsilon$. However, this example cannot be reconstructed from Chaiken et al. [1981] and, thus, we cannot verify it. The best proven bound is $\theta/\alpha \geq 8/7$.

For polygons with holes and even for those without holes (also called simple polygons), the rectangle cover problem is $\mathcal{NP}$-hard [Masek 1979; Culberson

and Reckhow 1994] and Max$\mathcal{SNP}$-hard [Berman and DasGupta 1997]. In particular, this implies that there is no polynomial time approximation scheme. Since the rectangle cover problem is a special set cover problem, the standard greedy approach immediately gives an $O(\log n)$ approximation. For general polygons, Anil Kumar and Ramesh [2003] are the first (and currently best) to improve upon this obvious factor, giving an approximation guarantee of $O(\sqrt{\log n})$, where n is the number of edges of P. The proof of this result is rather involved. For simple polygons, Franzblau [1989] gives a sweepline algorithm that achieves an approximation factor of 2. In the general case, the proven factor of this algorithm is $O(\log n)$, but conjectured by Franzblau [1989] to be 3. The lower bounds used to prove approximation guarantees are the stable set size α and (an upper bound on) the size of a largest clique in G. A constant-factor approximation in the general case can be considered as *the* remaining white spot in the problem's complexity landscape.

Quite some research efforts have gone into finding polynomially solvable special cases; we mention only covering with squares [Aupperle et al. 1988; Levcopoulos and Gudmundsson 1997] and polygons, in general position [Bern and Eppstein 1997]. Interestingly, there is a polynomial time algorithm for partitioning a polygon into nonoverlapping rectangles [Ohtsuki 1982]. However, a polygon, similar to Figure 4 (see later), shows that an optimal partition size may exceed an optimal cover size by a linear factor, so this does not lead to a constant-factor approximation.

1.2 Our Contributions

Despite its theoretical hardness, we demonstrate the rectangle cover problem to be computationally very tractable. In particular, we study an integer programming formulation of the problem. Doing this, we are the first to offer an exact (of course nonpolynomial time) algorithm to obtain provably optimal solutions, and we are the first to introduce linear/integer programming techniques in this problem area. It has been pointed out, e.g., by Anil Kumar and Ramesh [2003] that the main difficulty in coming up with good approximation algorithms is to find good lower bounds on the optimum cover size. Based on a fractional solution to the (dual of the) linear programming relaxation, we propose a stronger lower bound, which we call the *fractional* stable set size. In fact, this new lower bound motivates us to pursue previously unexplored research directions to find a constant-factor approximation algorithm. These are the celebrated primal-dual scheme [Goemans and Williamson 1996], rounding a fractional solution, iterated rounding [Jain 2001], and a dual-fitting algorithm [Vazirani 2001].

Currently with "only" encouraging computational results in our hands, we are optimistic that our research will actually contribute to a positive answer to the initially stated long-standing open question. We sketch partial results and promising ideas. A fruitful contribution of our work is a number of open questions it spawns. Last, but not least, we hope to increase the awareness in the computational geometry community toward tools from mathematical programming.

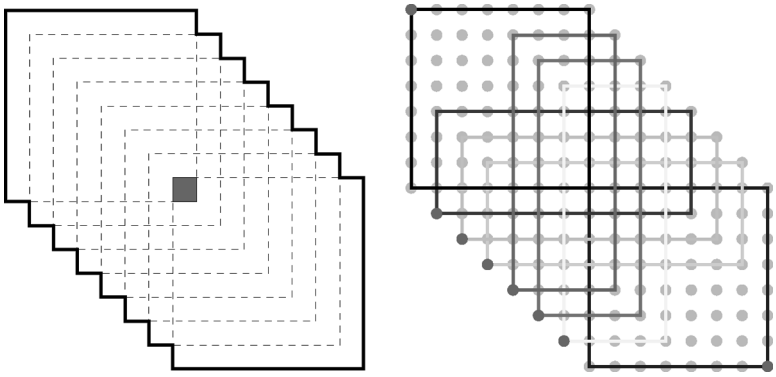

Fig. 2. (Left) The shaded center pixel is covered by *any* maximal rectangle; almost all pixels have nonconstant coverage. In an optimal cover, the coverage of the center pixel is linear in the cover size. The right figure schematically shows a minimal cover and a maximum stable set. Note that *there are* pixels of small, in fact, unique coverage in the cover; we call these *private*.

1.3 Preliminaries

Since we are dealing with a combinatorial problem, we identify P with its set of combinatorial pixels. This way we write $p \in P$ to state that pixel p is contained in polygon P. Let R denote the set of all rectangles in P. It is important that we only count rectangles and do not consider their areas. Thus, it is no loss of generality to restrict attention to inclusionwise maximal rectangles. We will do so in the following without further reference. The number of these rectangles can still be quadratic in the number n of edges of P [Franzblau 1989] (see also Figure 2).

2. AN INTEGER PROGRAM

Interpreting rectangles as cliques in G, we can make use of the standard integer programming formulation for the minimum clique cover problem in graphs [Schrijver 2003]. A binary variable x_r indicates whether rectangle $r \in R$ is chosen in the cover or not. For every pixel $p \in P$, at least one rectangle, which covers p, has to be picked and the number of picked rectangles has to be minimized:

$$\theta = \min \sum_{r \in R} x_r \tag{1}$$

$$\text{s. t.} \sum_{r \in R : r \ni p} x_r \geq 1 \quad p \in P \tag{2}$$

$$x_r \in \{0, 1\} \quad r \in R \tag{3}$$

This integer program (which we call the *primal* program) allows us to optimally solve any given instance of our problem; we will do so in our experiments. When we replace Eq. 3 by $x_r \geq 0, r \in R$ (3′), we obtain the associated linear programming (LP) relaxation. There is no need to explicitly require $x_r \leq 1$, $r \in R$, since we are minimizing. We call the optimal objective function value of the LP relaxation the *fractional cover size* of P and denote it by $\bar{\theta}$. Clearly, it holds that $\bar{\theta} \leq \theta$. In general, no polynomial time algorithm is known to

compute the fractional clique cover number of a graph, that is, for solving this linear program [Schrijver 2003]. In our case, however, the number of variables and constraints is polynomial in n, in fact, quadratic, because of the fact that we work with maximal rectangles only. Therefore, the fractional cover size $\bar{\theta}$ can be computed in polynomial time.

The integer program 1–3 immediately generalizes to the weighted rectangle cover problem, where rectangles need not have unit cost. It is straightforward and it does not increase the complexity, to restrict the coverage requirement to particular features of the polygon like the corners or the boundary—two well-studied variants [Berman and DasGupta 1997] for which no exact algorithm has been presented. It is also no coincidence that a formal dualization of our program leads to a formulation for the dual problem of finding a maximum stable set. A binary variable y_p, $p \in P$, reflects whether a pixel is chosen in the stable set or not. We have to require that no rectangle contains more than one of the chosen pixels and we maximize the number of chosen pixels. We call this the dual-integer program:

$$\alpha = \max \sum_{p \in P} y_p \tag{4}$$

$$\text{s. t.} \sum_{p \in P : p \in r} y_p \leq \quad 1 r \in R \tag{5}$$

$$y_p \in \{0, 1\} \quad p \in P \tag{6}$$

Again, when replacing Eq. 6 by $y_p \geq 0$, $p \in P$ (6′), we obtain the associated LP relaxation. We call its optimal objective function value $\bar{\alpha}$ the *fractional stable set size* of P. We refer to a feasible solution to the dual as a *fractional stable set*. It holds that $\bar{\alpha} \geq \alpha$. By strong linear programming duality, we have $\bar{\alpha} = \bar{\theta}$. We again stress the fact that we distinguish between the (primal and dual) *integer* programs, which solve the problems exactly, and their respective *continuous* linear programming relaxations, which give bounds. In general, optimal solutions to both linear programs (1–3′) and (4–6′) are fractional. However, using an interesting link to graph theory, in the case that G is perfect [Golumbic 1980], optimal solutions are automatically integer because of a strong duality between the *integer* programs [Schrijver 2003]. This link was already established early, (see e.g., Berge et al. [1981] and Motwani et al. [1989a, 1989b]), and our linear programs give optimal integer covers in polynomial time for this important class of polygons with $\alpha = \theta$.

2.1 About Fractional Solutions

Our computational experiments fuel our intuition. Anticipating our numerical results, we already give some qualitative observations here. In linear programming-based approximation algorithms the objective function value of a primal or dual fractional solution is used as a lower bound on the integer optimum. The more we learn about such fractional solutions, the more tools we may have to analyze the problem's approximability.

2.1.1 *General Observations.* The linear relaxations (1–3′) and (4–6′) appear to be easily solvable to optimality in a few seconds on a standard PC. The

vast majority of variables already assumes an integer value. A mere rounding of the remaining fractional variables typically gives an optimal or near-optimal integer solution (e.g., instance `night` is a bad example with "only" 95% integer values, but the rounded solution is optimal). For smaller random polygons, the LP optimal solution is very often already integer; this is an excellent quality practical heuristic, although memory expensive for very large instances.

2.1.2 *Odd Holes.* Figure 1 (left) shows Szemerédi's counterexample to the $\alpha = \theta$ conjecture. The five rectangles indicated by the shaded parts have to be in any cover. In the remaining parts of the polygon, there are five pixels that induce an odd-length cycle C ("odd hole") in the visibility graph G. To cover these pixels, at least three rectangles are needed, implying $\theta \geq 8$. On the other hand, at most, two of these pixels can be independent, that is, $\alpha \leq 7$. The odd hole C is precisely the reason why G is not perfect in this example. Figure 1 (right) shows that C is encoded in the optimal fractional solution as well: Exactly the variables corresponding to edges of C assume a value of 0.5. The same figure shows an optimal fractional stable set. Pixels corresponding to vertices of C assume a value of 0.5 (drawn wider in the figure). That is, $\bar{\alpha} = \bar{\theta} = 7.5$. This immediately suggests to strengthen the LP relaxation.

LEMMA 2.1. *For any induced odd cycle C with $|C| \geq 5$, the inequality $\sum_{r \in C} x_r \geq \lceil |C|/2 \rceil$ is valid for* (1–3)*, where $r \in C$ denotes the rectangles corresponding to the edges of C.*

The graph theoretic complements of odd holes are called odd *antiholes*. A graph is not perfect either if it contains an induced odd antihole. We can prove that there is no way of representing even the simplest nontrivial antihole with seven vertices in a rectangle visibility graph. Odd holes are, therefore, the only reason for imperfection. However, they are not the only reason for fractional solutions. From our experiments, arbitrary fractions are possible, not only halves. More explicitly, we know of no way of lower bounding the smallest occurring fraction and simply rounding a fractional solution does not give a constant factor approximation. Next, we pursue this a little further.

2.1.3 *High Coverage.* We define the coverage of a pixel p as the number of rectangles that contain p. For the classical set-cover problem, rounding up an optimal fractional solution gives an f-approximate cover, where f is the maximum coverage of any element. In general, a pixel can have more than constant coverage; even worse, almost *no* pixel may have constant coverage. Even in an optimal cover of a simple polygon in general position pixels may have high coverage (see Figure 2). Unlike in the general set-cover case, high coverage is no prediction about the fractions in an optimal LP solution. In Figure 2 (right) there are no fractional variables; the solution is integer. The fractional (indeed, integer) optimal solution to this simple example has a remarkable property. Every rectangle in the optimal cover contains pixels of low coverage. More precisely, the following holds.

LEMMA 2.2. *In an optimal cover $\mathcal{C}$, every rectangle $r \in \mathcal{C}$ contains a pixel that is uniquely covered by r.*

This can be easily seen, since, otherwise, $\mathcal{C} \setminus \{r\}$ would be a cover, contradicting the optimality of $\mathcal{C}$. We call these uniquely covered pixels *private*. Since the number of private pixels is the number of rectangles in a cover, we deem this aspect of the problem worthy of more investigation. Even more so, since by complementary slackness (see below), private pixels (these fulfill the constraints (5) with equality) are the only candidates for being picked (at least fractionally) into a stable set.

Note that a set of private pixels need not be a stable set. It is, therefore, natural to relax the concept and ask: What are the characteristics of polygons where every pixel has only constant coverage? What kind of polygons have "many" pixels with "low" coverage? How can we exploit Lemma 2.2? Answers to these questions would turn LP rounding into a constant-factor approximation algorithm. These questions also keep appearing in different guises in the next section.

3. LP-BASED APPROXIMATION

There are more elaborate linear programming-based approaches to constant-factor approximation algorithms. They can be used as analytical tools to theoretically sustain our excellent computational results.

3.1 Iterated Rounding

In standard LP rounding (of all the fractional variables), the approximation factor hinges on the magnitude of the smallest fraction. Alternatively, Jain [2001] proposed to round up only one fractional variable at a time, adapt the linear program to reflect the rounding, solve the modified linear program again, and iterate. Adapting the linear program in our case means that rounding up a rectangle r at fractional value x is the same as reducing the coverage requirement for all the pixels in r from 1 to $1 - x$.

Proving an approximation factor via this *iterated rounding* scheme relies on two parts: The first is that one can always (in every iteration) find a rectangle/variable with a *large* value and the second is that a solution to the final (modified) linear program is still feasible to the initial linear program in order to guarantee that the initial linear program actually gives a lower bound on the cover size of the rounded solution. The latter is, indeed, immediate for our linear program. For the former, in our experiments, we always found (fractional) variables of value at least 0.5 and a geometric argument why this is always the case would be most interesting.

3.2 Primal-Dual Scheme

The primal-dual scheme [Goemans and Williamson 1996] builds on relaxing the complementary slackness optimality conditions [Schrijver 2003] in linear programming. The general scheme iteratively improves an initially infeasible integer primal solution, that is, a set of rectangles, to finally obtain a feasible cover. The improvement step is guided by a feasible fractional dual solution, that is a *fractional stable set*, which is improved in alternation with the primal solution. The relaxed complementary slackness conditions contain the key

information. In our case, they read

$$x_r > 0 \;\Rightarrow\; \frac{1}{d} \leq \sum_{p \in P : p \in r} y_p \qquad r \in R \tag{7}$$

for some constant d, and

$$y_p > 0 \;\Rightarrow\; \sum_{r \in R : r \ni p} x_r \leq c \qquad p \in P \tag{8}$$

for some constant c. First note that if a possibly infeasible primal integer solution is maintained, $x_r > 0$ means $x_r = 1$. An interpretation of condition 7 is that every rectangle in the constructed cover must cover at least $1/d$ pixels from the fractional stable set. Condition 8 states that a pixel in the fractional stable set must not be contained in more than c rectangles (regardless of whether in the cover or not).

We found two cases where we can compute a cover and a fractional stable set simultaneously, such that the two conditions hold. *Thin* polygons, as unions of width 1 or height 1 rectangles, are a class of polygons amenable to LP rounding and the primal-dual scheme. Since no pixel is covered by more than two rectangles, this gives a 2-approximation. More generally, polygons of bounded width (every pixel contains a boundary pixel in its "neighborhood") are a new nontrivial class, which allows a constant-factor approximation. Of course, the primal-dual scheme trivially applies when all rectangles have bounded (combinatorial) area.

3.3 Dual Fitting

Since $\alpha \leq \theta$ the former natural approach to approximation algorithms was to construct a large stable set usable as a good lower bound [Franzblau 1989]. Since $\alpha \leq \bar{\alpha}$, we propose to use the stronger bound provided by a *fractional* stable set. Our *dual-fitting* approach is to simultaneously construct a cover $\mathcal{C} \subseteq R$ and an *pseudostable set* $\mathcal{S} \subseteq P$ of pixels, with $|\mathcal{C}| \leq |\mathcal{S}|$ (we say that $\mathcal{S}$ *pays* for $\mathcal{C}$). "Pseudo" refers to allowing a constant number c of pixels in a rectangle, that is, we relax Eq. (5) to $\sum_{p \in P : p \in r} y_p \leq c$. From this constraint, we see that picking each pixel in $\mathcal{S}$ to the extent of $1/c$ (which is a division of all y_p variables' values by c) gives a feasible fractional solution to our dual linear program. A cover with these properties has a cost of

$$|\mathcal{C}| \leq |\mathcal{S}| \leq c\bar{\alpha} = c\bar{\theta} \leq c\theta \tag{9}$$

that is, it would yield a c-approximation. Actually, one does not have to require that $\mathcal{S}$ pays for the full cover, but $\frac{1}{d}|\mathcal{C}| \leq |\mathcal{S}|$ for a constant d suffices, which would imply a (cd)-approximation. This paying for a constant fraction of the primal solution only is a new proposal in the context of dual fitting. Here again, the question is how to *guarantee* our conditions, in general. From a computational point of view, we obtain encouraging results, which suggest that our proposal can be developed into a proven constant-factor approximation. In the next section, we sketch some ideas how this can be done.

4. TOWARD A CONSTANT-FACTOR APPROXIMATION

4.1 Obligatory Rectangles and Greedy

For set cover, the greedy algorithm yields the best possible approximation factor of $O(\log n)$. The strategy is to iteratively pick a rectangle that covers the most, yet uncovered, pixels. One expects that for our particular problem, the performance guarantee can be improved. Computationally, we answer strictly in the affirmative. Again, our contribution is the dual point of view. It is our aim to design an algorithm, which is based on the dual-fitting idea of Section 3.3 and we mainly have to say how to construct a feasible dual fractional solution.

We use some terminology from Hannenhalli et al. [2002]. Certain rectangles have to be in any cover. A *prime* rectangle contains a pixel, which is not contained in any other rectangle. Such a pixel is called a *leaf*. Every cover must contain all prime rectangles. For a given pixel p, we may extend horizontally and vertically until we hit the boundary; the rectangular area $R(p)$, defined by the corresponding edges at the boundary, is called the *extended* rectangle of p. $R(p)$ might *not* be entirely contained in the polygon, but, if so, it is a prime rectangle [Hannenhalli et al. 2002]. Moreover, let $\mathcal{C}' \subseteq \mathcal{C}$ be a subset of some optimal cover $\mathcal{C}$. If there is a rectangle r, which contains $(P \setminus \mathcal{C}') \cap R(p)$ for some extended rectangle $R(p)$, then there is an optimal cover, which contains $\mathcal{C}'$ and r [Hannenhalli et al. 2002]. In this context, let us call rectangle r *quasi-prime* and pixel p a *quasi-leaf*. The algorithm we use to compute a cover is a slight extension of Hannenhalli et al. [2002], but we will provide a new interpretation and, more importantly, a dual counterpart:

QUASI-GREEDY

1. pick all prime rectangles
2. pick a maximal set of quasi-prime rectangles
3. cover the remaining pixels with the greedy algorithm
4. remove redundant rectangles ("pruning")

It has not been observed before that a set of leafs and quasi-leafs forms a stable set. This leads to the idea to compute a pseudostable set containing a maximal set of leafs and quasi-leafs. Thus, in order to build a pseudostable set, we check for every rectangle in the greedy cover whether it contains

1. a leaf
2. a quasi-leaf
3. a corner pixel

(in this order). The first positive test gives a pixel, which we add to the pseudostable set. A *corner pixel* is a corner of a rectangle, which is private and a corner of the polygon. We already observed that pixels from steps 1 and 2 are independent. Furthermore, any rectangle obviously contains, at most, four corner pixels, and, since corner pixels are private, actually, at most, two of them. By our previous considerations, this would imply a 2-approximation if the constructed pseudostable set would pay for the whole cover. In general, we found

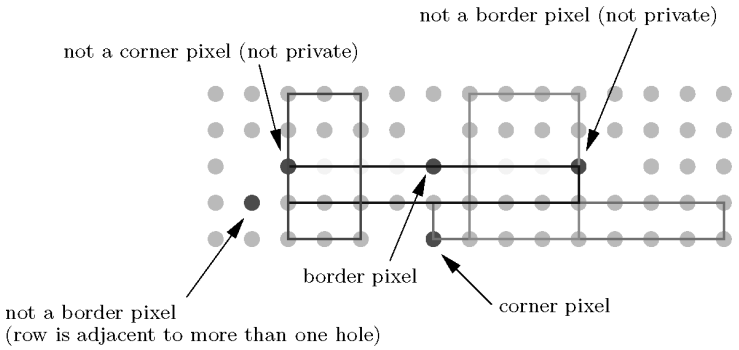

Fig. 3. Illustrating corner and border pixels; yellow pixels are the *row* of the border pixel (which is adjacent to only one hole).

this not to be true. We have constructed examples which suggest that one cannot guarantee that a constant fraction of the cover has been paid for. To achieve this latter goal, one has to add more pixels to the pseudostable set. To this end, we extend the above test and also check for every rectangle in the cover whether it contains

4. a border pixel

A *border pixel* p is private and adjacent to a nonpolygon pixel $\bar{p}$ (the outer face or a hole). The row (or column) of pixels, which contains p, which is adjacent to $\bar{p}$, and which extends to the left and the right (to the top and the bottom) until some nonpolygon pixel is hit, *must not be adjacent* to a different hole (or the outer face) *other* than the hole (or the outer face) the pixel $\bar{p}$ corresponds to (see Figure 3). These pixels also have a natural motivation: They are a kind of analog to corner pixels, in the sense that a border pixel *represents* an edge of a rectangle. It is, however, conceivable that some rectangle contains several border pixels picked into the pseudostable set, even though we conjecture than one cannot come up with a geometry that enforces such a situation.

Let us furthermore remark that after the pruning step in QUASI-GREEDY, every rectangle in the cover contains a private pixel (Lemma 2.2). This pixel is an intuitive candidate to become a pixel in a pseudostable set. This set would actually pay for the whole cover. However, it is not clear whether one can control how many pixels of this set can be contained in the same rectangle.

4.2 Using Boundary Covers

There is a simple 4-approximation algorithm for covering the boundary of an orthogonal polygon [Berman and DasGupta 1997]. In this context, a natural question arises: Can we always find an interior cover whose size is bounded from above by a constant multiple of the size θ_{boundary} of an optimal boundary cover? The answer is "no." Our counterexample in Figure 4 shows that there is an $O\left(\sqrt{n}\right)$-cover of the boundary of the polygon in the left figure with maximal horizontal and vertical strips. But the optimal interior cover needs $\Theta(n)$ rectangles, since the white uncovered pixels in the right figure are independent. Nevertheless, the latter observation is actually very encouraging. We conjecture

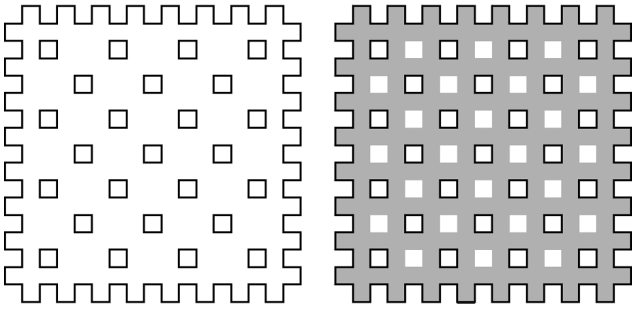

Fig. 4. A boundary cover may leave a nonconstant fraction of pixels uncovered. Note that in this example, all uncovered pixels are *leafs*, i.e., uniquely contained in some rectangle and, therefore, are all independent. A (4-approximation of a) boundary cover and a cover for these pixels obtained e.g., by our QUASI-GREEDY algorithm is thus a proven 5-approximation (in fact, actually optimal).

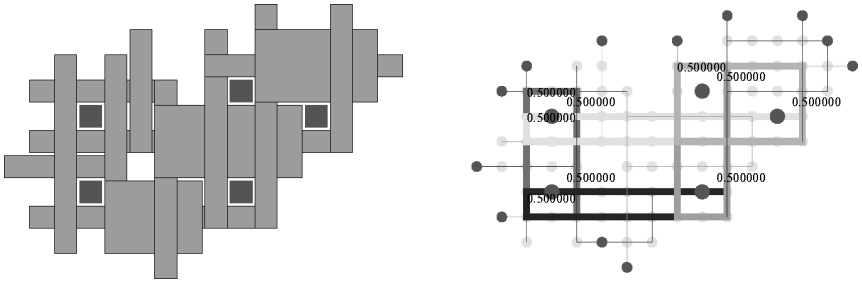

Fig. 5. An optimal boundary cover of this polygon is depicted with green rectangles; the red pixels form an *odd hole* which need to be covered by three additional rectangles. A fractional solution to the linear program is given on the right.

that one can find an interior cover of size less than $c_1\,\theta_{\text{boundary}} + c_2\,\alpha$, where c_1 are c_2 are appropriate constants. This would imply a constant-factor approximation for the rectangle cover problem.

We constructed an example (Figure 5) that shows that an optimal boundary cover may leave an *odd hole* of interior pixels uncovered. This implies that a solution to a linear program for covering the remaining pixels may be fractional.

4.3 Quasi-Prime Rectangles and Breaking Holes

There is a large class of polygons (e.g., polygons resulting from typical oligonucleotide masks [Hannenhalli et al. 2002]), where the optimal cover is found after the first two steps of the QUASI-GREEDY algorithm in Section 4.1. The cover then consists of only prime and quasi-prime rectangles. This is, of course, in general, not the case (see Figure 1). Now, consider the set $\mathcal{U}$ of pixels remained uncovered after step 2. We can prove that there is an induced cycle (a hole) in G whose vertices correspond to a subset of $\mathcal{U}$. Covering each second edge of this hole extends the previous partial cover. We call this covering step to "break a hole." A straightforward algorithm is the following: while the polygon is uncovered, iteratively pick a maximal set of quasi-prime rectangles, then find a hole and break it. We can iteratively extend also the partial pseudostable set. The quasi-prime rectangles are paid for by quasi-leafs, which form a stable set. The rectangles,

which break an even (odd) hole, can all (but one) be paid for by a stable set, as well.

We have experimented with related and extended ideas based on the observations sketched in Sections 4.2 and 4.3 and obtained encouraging results. We hope that our findings will constitute an important step to enable researchers to finally come up with a constant-factor approximation algorithm.

5. COMPUTATIONAL EXPERIENCE

A technical investigation of our LP-based proposals for possible constant-factor approximations for the rectangle cover problem and their computational evaluation went hand in hand. Directly or indirectly, all primal-dual related questions boil down to "is it geometrically possible to select *many* pixels which are *almost* independent?" In this respect, we think that the QUASI-GREEDY algorithm is a representative method to provide partial answers. It is also our aim to further sustain the somewhat unexpected *well behavior* of the problem or our formulation.

5.1 "Real-World" Polygons

Our test set comprises polygons of various sizes, which are derived from VLSI mask design (instances `VLSI*`), a set of standard fax images[1] (instances `ccitt*`), and several black and white images (instances `marbles`, `mickey`, ...). The ratio between the number of available rectangles and the optimal cover size can be seen as one indicator about the *difficulty* of an instance (this has to be used carefully as Figure 2 shows an *easy* instance with θ^2 rectangles). Tables I and II summarize our results.

5.2 Random Polygons

We used two different generators for random polygons. The first, *flip-coin*, simply decides for each pixel (of a square of given size) with a given probability (the same for every pixel) whether it is in the polygon or not. The second, *union*, generates a given number of rectangles and outputs their union: Each generated rectangle has random upper left and lower right corners (up to a given maximum side length, common for width and height), i.e., it is placed at a random position. *Random* always refers to uniformly at random. Figure 6 shows sample polygons and Figures 7 and 8 try to capture the *difficulty* of our random polygons. Tables III and IV summarize our results.

5.3 Results

Tables I and III give overwhelming evidence for the excellent quality of the lower bound obtained from the LP relaxation of our integer program. The integrality gap (relative gap between linear and integer program in percentage) is always close to zero (for 1564 out of 1800 random polygons the gap actually is zero).

From Tables II and IV, we first observe that covers computed with our QUASI-GREEDY algorithm are generally very close to optimum. To computationally support our conjecture that the dual-fitting analysis of QUASI-GREEDY, indeed, leads

[1]Available at `ftp://nic.funet.fi/pub/graphics/misc/test-images/`

Table I. Results for the Primal and Dual Linear/Intege Programs[a]

Instance Characteristics				Dual (stable set size)			Primal (cover size)		
Instance	Size	Density (%)	Rectangles	Opt. LP	Opt. IP	LP gap (%)	Opt. LP	Opt. IP	LP gap (%)
VLSI1	68×35	50.25	45	43.000	43	0.000	43.000	43	0.000
VLSI2	3841×298	95.34	16694	4222.667	4221	0.039	4222.667	4224	0.032
VLSI3	148×135	45.09	78	71.000	71	0.000	71.000	71	0.000
VLSI5	6836×1104	55.17	192358	77231.167	77227	0.005	77231.167	77234	0.004
ccitt1	2376×1728	3.79	27389	14377.000	14377	0.000	14377.000	14377	0.000
ccitt2	2376×1728	4.49	30427	7422.000	7422	0.000	9127.000	7422	0.000
ccitt3	2376×1728	8.21	40625	21085.000	21085	0.000	21085.000	21085	0.000
ccitt4	2376×1728	12.41	101930	56901.000	56901	0.000	56901.000	56901	0.000
ccitt5	2376×1728	7.74	46773	24738.500	24738	0.002	24738.500	24739	0.002
ccitt6	2376×1728	5.04	30639	12013.000	12013	0.000	12013.000	12014	0.008
ccitt7	2376×1728	8.69	85569	52502.500	52502	0.001	52502.500	52508	0.010
ccitt8	2376×1728	43.02	41492	14024.500	14022	0.018	14024.500	14025	0.004
marbles	1152×813	63.49	56354	44235.000	44235	0.000	44235.000	44235	0.000
mickey	334×280	75.13	17530	9129.345	9127	0.026	9129.345	9132	0.029
day	480×640	64.63	45553	32191.000	32190	0.000	32191.000	32192	0.003
night	480×640	96.02	17648	7940.985	7938	0.038	7940.985	7943	0.025

[a]For each instance, we list its size in pixels, its number of (maximal) rectangles. For the dual and the primal programs (in that order), we give the optimal linear and integer program objective function values. The "LP gap" is the relative gap between linear and integer program. Notice that instances `mickey` and `night` do not have a fractional optimal solution with "nice" fractions. The 21086 reported as optimal for `ccitt3` in the extended abstract [Heinrich-Litan and Lübbecke 2005] is improved here, since the optimality tolerance of the solver was erroneously set to a nonzero value by default.

Table II. Details for the QUASI-GREEDY Algorithm of Section 4.1[a]

				QUASI-GREEDY Cover			Pseudostable set Characteristics			
Instance	Optimum	QUASI-GREEDY		Prime	Quasi-prime	Greedy	Corner	Border	Max pixels	Pays for (%)
VLSI1	43	43	1.000	41	2	0	0	0	1	100.00
VLSI2	4224	4701	1.113	1587	203	2911	1105	1279	4	88.79
VLSI3	71	71	1.000	71	0	0	0	0	1	100.00
ccitt1	14377	14457	1.006	10685	2099	1673	1632	28	2	99.91
ccitt2	7422	7617	1.026	3587	409	3621	3574	29	3	99.76
ccitt3	21086	21259	1.008	15691	2020	3548	3427	86	3	99.84
ccitt4	56901	57262	1.006	42358	8605	6299	6110	59	2	99.77
ccitt5	24739	24911	1.007	18529	2985	3397	3259	98	2	99.84
ccitt6	12014	12132	1.010	8256	1049	2827	2764	35	2	99.77
ccitt7	52508	52599	1.002	39230	10842	2525	2448	56	2	99.96
ccitt8	14025	14303	1.020	7840	1353	5110	5023	54	3	99.77
marbles	44235	44235	1.000	43548	687	0	0	0	1	100.00
mickey	9132	9523	1.043	5582	690	3251	528	1593	3	88.13
day	32192	32431	1.007	26308	3777	2346	749	900	4	97.85
night	7943	8384	1.056	4014	501	3869	762	1810	4	84.53

[a]We compare the optimal cover size against ours: We list the absolute numbers and the relative quality of QUASI-GREEDY covers as a factor from optimum; this is typically around 1.01. The following columns list the number of prime and quasi-prime rectangles and those picked by the greedy step. The number of corner and border pixels in the constructed quasi stable set $\mathcal{S}$ is then given (the number of (quasi-)leafs equals the number of (quasi-)primes). Finally, we state the maximal number of pixels of $\mathcal{S}$ in some rectangle and the fraction of the cover size for which $\mathcal{S}$ pays.

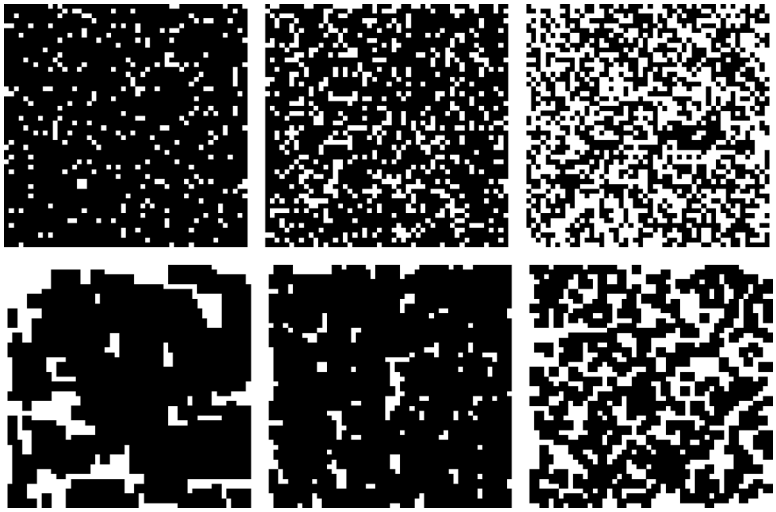

Fig. 6. Samples of random 50×50 polygons. The top row is generated with the *flip-coin* generator with pixel probabilities of 90, 75, and 50%. The bottom row is generated with the *union* generator: the characteristics as (maximal side length, number of rectangles) pairs are (10, 1000), (5, 2000), and (3, 2000), in that order. Even though *union*-type polygons in the bottom row look "more natural", our algorithms are more challenged by dense *flip-coin* polygons.

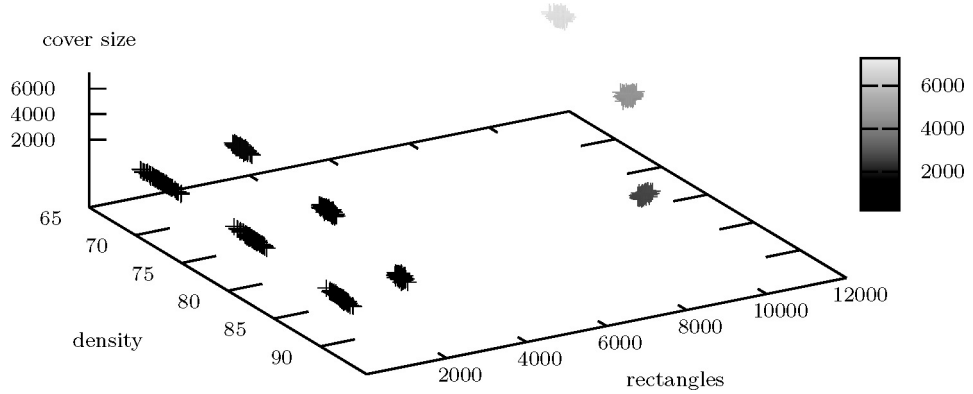

Fig. 7. Every cross in this and the following illustrations represents a single random instance. The online companion of this paper contains the `gnuplot` sources so that a better visual impression can be obtained. This illustration visualizes the very low *combinatorial complexity* of random *flip-coin* type polygons: Knowing the number of rectangles in an such a polygon almost allows *guessing* the size of a greedy cover.

to a constant approximation factor, we evaluate several more characteristics of the pseudostable sets we compute. Our findings are summarized in Figures 9–14.

We have never seen more than four pixels of a pseudostable set in a rectangle (which appears to be a sensible number to aim at in a geometry context). We remark that our observed worst case is four pixels, but the average fraction of rectangles containing three or four pseudostable pixels is almost negligibly

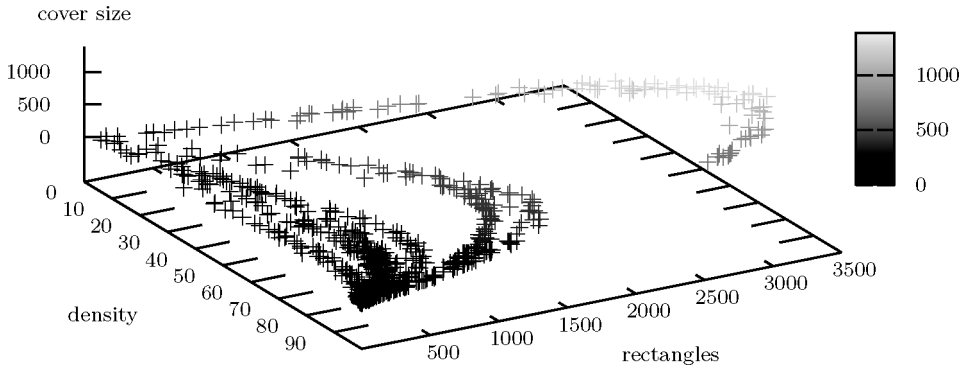

Fig. 8. *Combinatorial complexity* of random *union* type polygons. At a *medium* density around 50%, a cover uses the largest fraction of available rectangles.

Table III. Cumulated Results for Our Experiments with Random Polygons[a]

Instance generator	Density	Rectangles	Primal LP gap	Dual LP gap	Quasi-Greedy	
flip-coin(50,90%)	90.059	498.890	0.012	0.014	227.490	1.031
flip-coin(50,80%)	79.966	626.190	0.000	0.000	364.550	1.005
flip-coin(50,70%)	69.999	663.700	0.002	0.002	457.830	1.000
flip-coin(100,90%)	90.013	2013.160	0.029	0.029	887.190	1.048
flip-coin(100,80%)	80.008	2506.970	0.003	0.003	1430.310	1.008
flip-coin(100,70%)	69.960	2661.090	0.001	0.001	1806.960	1.000
flip-coin(200,90%)	90.031	8075.250	0.037	0.037	3506.940	1.058
flip-coin(200,80%)	79.971	10045.080	0.004	0.004	5677.470	1.009
flip-coin(200,70%)	69.974	10612.130	0.000	0.000	7166.130	1.001
union(50,10,1000)	91.630	65.360	0.000	0.000	30.780	1.005
union(50,5,2000)	79.967	225.710	0.014	0.014	98.260	1.021
union(50,3,2000)	72.162	364.020	0.009	0.009	172.130	1.022
union(100,20,1000)	91.173	89.910	0.000	0.000	42.060	1.002
union(100,10,3000)	86.760	352.440	0.018	0.018	137.100	1.027
union(100,6,4000)	76.995	976.780	0.015	0.015	403.070	1.039
union(200,30,1500)	89.397	244.130	0.006	0.006	94.250	1.016
union(200,15,5000)	85.595	852.010	0.009	0.012	299.810	1.044
union(200,9,6000)	61.749	2272.620	0.005	0.005	1038.140	1.016

[a]We give the characteristics for *flip-coin* polygons as (size, density) pairs and those for *union* polygons as triples (size, maximal side length, number of rectangles). We generated 100 polygons for each line in this table, 1800 in total. Headings' meanings are identical to those in Table I.

small. Our pseudostable sets consistently pay for more than 75% of the cover size. According to our line of argumentation in Section 4.1, this empirically supports that Quasi-Greedy be an $(4 \times \frac{4}{3})$-approximation algorithm for the rectangle cover problem. From Table IV, it appears that large and dense polygons obtained from the *flip-coin* generator come closest to refute this conjecture; we generated density 90% instances of size up to 1000×1000, however, to no negative effect to our conjecture.

5.4 Corner and Boundary Covers

Finding a minimum cover only for the corners or the boundary of a polygon is $\mathcal{NP}$-hard as well [Berman and DasGupta 1997]. Just out of interest, and to

Table IV. Cumulated Results for Our Random Experiments with Polygons of *Flip-coin* and *Union* Type[a]

	QUASI-GREEY Cover Characteristics				Pseudostable Set Characteristics			
Instance generator	QUASI-GREEDY	Prime	Quasi-prime	Greedy	Corner	Border	Max pixels	Pays for
flip-coin(50,90%)	227.490	106.190	45.470	77.310	20.170	33.700	2.420	90.333
flip-coin(50,80%)	364.550	240.820	90.740	33.490	15.380	10.910	2.060	98.157
flip-coin(50,70%)	457.830	363.100	88.310	6.460	4.010	1.530	1.560	99.807
flip-coin(100,90%)	887.190	372.720	105.690	417.820	109.110	173.280	2.980	85.747
flip-coin(100,80%)	1430.310	912.340	331.680	188.580	84.190	61.130	2.320	97.133
flip-coin(100,70%)	1806.960	1407.230	367.510	32.430	19.810	7.830	2.010	99.746
flip-coin(200,90%)	3506.940	1381.700	292.770	1875.160	489.380	767.020	3.060	83.573
flip-coin(200,80%)	5677.470	3556.520	1270.610	859.720	390.540	278.280	2.890	96.802
flip-coin(200,70%)	7166.130	5551.210	1473.220	142.640	86.490	34.520	2.070	99.711
union(50,10,1000)	30.780	21.660	6.160	2.970	1.260	1.430	1.290	99.371
union(50,5,2000)	98.260	59.720	17.050	21.900	8.670	10.280	1.840	97.166
union(50,3,2000)	172.130	103.870	32.660	36.200	14.420	14.670	1.920	95.386
union(100,20,1000)	42.060	30.550	6.720	4.870	2.520	2.000	1.380	99.534
union(100,10,3000)	137.100	80.030	20.470	37.520	14.850	17.790	2.050	96.819
union(100,6,4000)	403.070	219.120	62.500	124.500	47.750	53.690	2.380	93.931
union(200,30,1500)	94.250	59.460	13.240	21.900	11.090	9.150	1.860	98.658
union(200,15,5000)	299.810	166.190	33.830	102.380	40.820	48.550	2.510	96.086
union(200,9,6000)	1038.140	691.800	154.850	195.540	92.980	75.340	2.080	97.825

[a]Headings' meanings are identical to those in Table II. Large and dense instances of this type appear to be most critical to refute our conjecture that our dual-fitting approach leads to a constant-factor approximation algorithm; see our comments in the text.

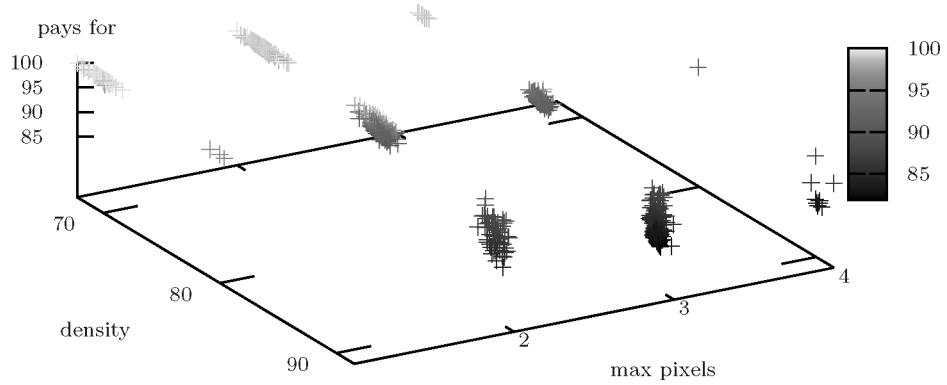

Fig. 9. QUASI-GREEDY's computation of pseudo stable sets on *flip-coin* instances. The axis labels correspond to the headings in Tables II and IV. Denser polygons make it harder for a pseudostable set to pay for the whole cover. The number of pseudostable pixels in some rectangle never exceeded four.

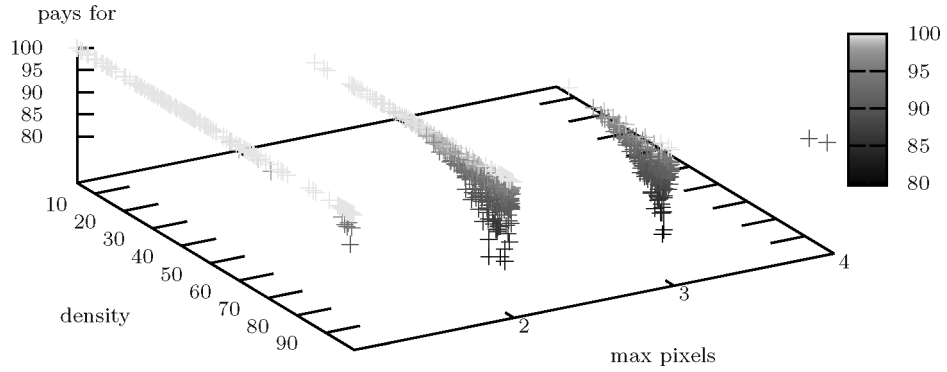

Fig. 10. Same as Figure 9 for *union* instances.

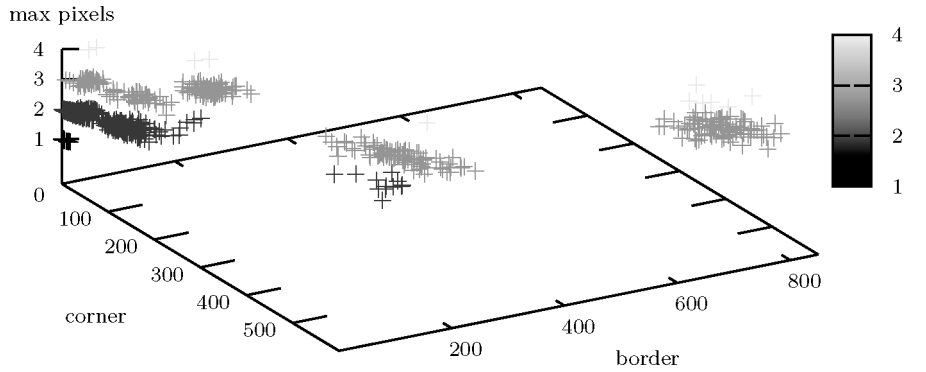

Fig. 11. QUASI-GREEDY's computation of pseudostable sets on *flip-coin* instances. The need to use border and corner pixels in a pseudostable set increases chances that more pseudostable pixels are contained in some rectangle, as was to be expected.

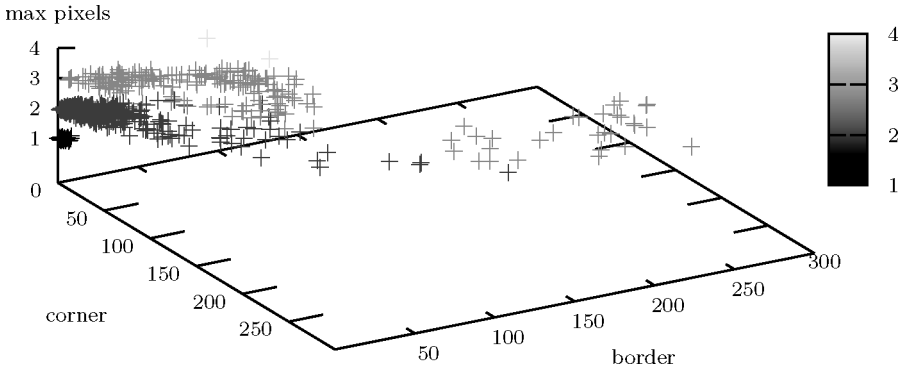

Fig. 12. Same as Figure 11 for *union* instances.

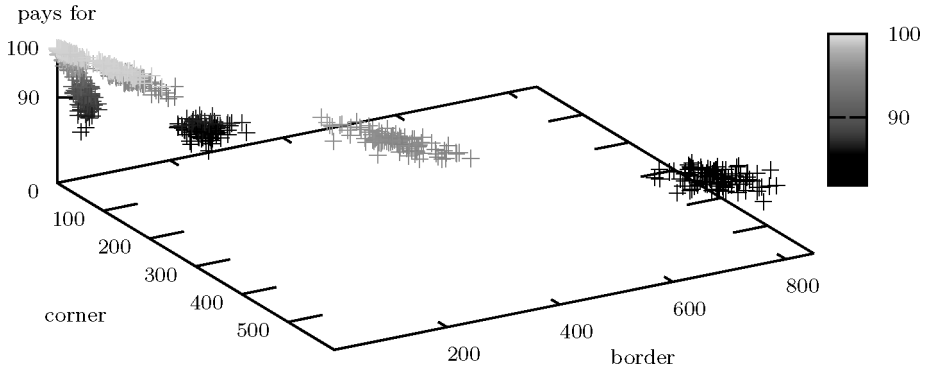

Fig. 13. QUASI-GREEDY's computation of pseudostable sets on *flip-coin* instances. This is the analog to Figure 11, evaluating the fraction of the cover the pseudostable set pays for. Here again, the need to use border and corner pixels decreases chances that the pseudostable set is able to pay for the whole cover or a very large fraction of it.

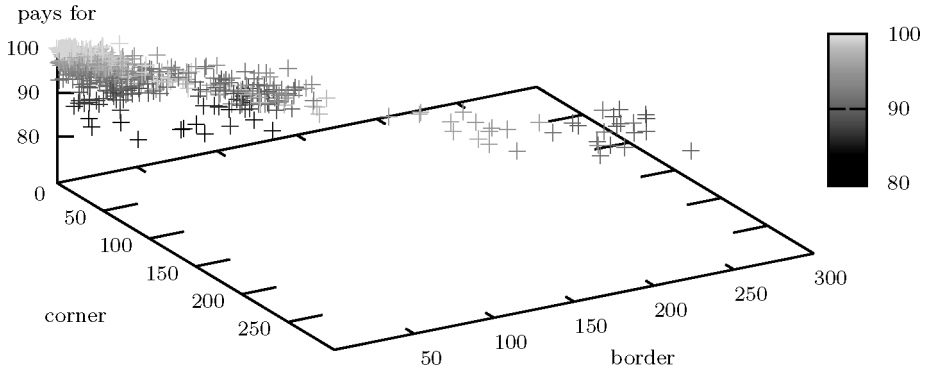

Fig. 14. Same as Figure 13 for *union* instances.

initiate a fresh thinking here as well, we computationally related the optimum cover sizes for these problems to minimum rectangle (interior) cover sizes.

The relation $\theta_{\text{corner}} \leq \theta_{\text{boundary}} \leq \theta$ is obvious. In computations, we found that θ_{boundary} is closer to θ than to θ_{corner} and it appears that $\theta \leq 2\theta_{\text{corner}}$, which implied that $\theta \leq 2\theta_{\text{boundary}}$. Recall that there is a 4-approximation for the boundary cover problem.

6. CONCLUSIONS

The rectangle cover problem turns out to be computationally very tractable. The extremely small integrality gaps are a strong vote for our integer-programming approach. Despite such successes and the ability to obtain probably optimal solutions to geometric problems, this methodology is not very common in computational geometry. We hope to contribute with our work to more curiosity towards mathematical programming techniques in this community. On the downside, integer programs for industrial size polygons, e.g., from VLSI design, are extremely large. The generation of the integer programs consumes much more time than solving them (solution time is typically only a few seconds using the standard solver CPLEX 9.1.0 [ILOG Inc., CPLEX Division 2004]). As a remedy, we also experimented with a column generation approach, that is, a dynamic generation of the variables of the linear program. This enables us to attack larger instances.

It is common that theory is complemented by computational experience. In this paper, we did the reverse: We found promising research directions by a careful study of computational experiments. Finally, we propose:

6.1 Restatement of Erdős' Question

Is it true that *both*, the integrality gap of our primal and that of our dual integer program are bounded by a constant? The example in Figure 1 places lower bounds on these gaps of $\theta/\bar{\theta} \geq 16/15$ and $\bar{\alpha}/\alpha \geq 15/14$, implying the already known bound $\theta/\alpha \geq 8/7$. We conjecture that these gaps are, in fact, tight. Originally, we set out to find an answer to Erdős' question. We conclude with an answer in the affirmative, at least computationally.

ACKNOWLEDGMENTS

We thank Sándor Fekete for fruitful discussions and Ulrich Brenner for providing us with polygon data from the mask fabrication process in VLSI design.

REFERENCES

ANIL KUMAR, V. AND RAMESH, H. 2003. Covering rectilinear polygons with axis-parallel rectangles. *SIAM J. Comput. 32*, 6, 1509–1541.

AUPPERLE, L., CONN, H., KEIL, J., AND O'ROURKE, J. 1988. Covering orthogonal polygons with squares. In *Proc. 26th Allerton Conf. Commun. Control Comput.* 97–106.

BERGE, C., CHEN, C., CHVÁTAL, V., AND SEOW, C. 1981. Combinatorial properties of polyominoes. *Combinatorica 1*, 217–224.

BERMAN, P. AND DASGUPTA, B. 1997. Complexities of efficient solutions of rectilinear polygon cover problems. *Algorithmica 17*, 4, 331–356.

BERN, M. AND EPPSTEIN, D. 1997. Approximation algorithms for geometric problems. See Hochbaum [1996], Chapter 8, 296–345.

CHAIKEN, S., KLEITMAN, D., SAKS, M., AND SHEARER, J. 1981. Covering regions by rectangles. *SIAM J. Algebraic Discrete Methods 2*, 394–410.

CULBERSON, J. AND RECKHOW, R. 1994. Covering polygons is hard. *J. Algorithms 17*, 2–44.

FRANZBLAU, D. 1989. Performance guarantees on a sweep-line heuristic for covering rectilinear polygons with rectangles. *SIAM J. Discrete Math. 2*, 3, 307–321.

GOEMANS, M. AND WILLIAMSON, D. 1996. The primal-dual method for approximation algorithms and it application to network design problems. See Hochbaum [1996], Chapter 4.

GOLUMBIC, M. 1980. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.

HANNENHALLI, S., HUBELL, E., LIPSHUTZ, R., AND PEVZNER, P. 2002. Combinatorial algorithms for design of DNA arrays. *Adv. Biochem. Eng. Biotechnol. 77*, 1–19.

HEINRICH-LITAN, L. AND LÜBBECKE, M. 2005. Rectangle covers revisited computationally. In *Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA05)*, S. Nikoletseas, Ed. Lect. Notes Comput. Sci., vol. 3503. Springer-Verlag, New York. 55–66.

HOCHBAUM, D., Ed. 1996. *Approximation Algorithms for $\mathcal{NP}$-Hard Problems*. PWS, Boston, MA.

ILOG INC., CPLEX DIVISION. 2004. *CPLEX 9.0 User's Manual*.

JAIN, K. 2001. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica 21*, 1, 39–60.

LEVCOPOULOS, C. AND GUDMUNDSSON, J. 1997. Approximation algorithms for covering polygons with squares and similar problems. In *Proceedings of RANDOM'97*. Lect. Notes Comput. Sci., vol. 1269. Springer, New York. 27–41.

MAIRE, F. 1994. Polyominos and perfect graphs. *Inform. Process. Lett. 50*, 2, 57–61.

MASEK, W. 1979. Some NP-complete set covering problems. Unpublished manuscript, MIT.

MOTWANI, R., RAGHUNATHAN, A., AND SARAN, H. 1989a. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. System Sci. 40*, 19–48.

MOTWANI, R., RAGHUNATHAN, A., AND SARAN, H. 1989b. Perfect graphs and orthogonally convex covers. *SIAM J. Discrete Math. 2*, 371–392.

OHTSUKI, T. 1982. Minimum dissection of rectilinear regions. In *Proc. 1982 IEEE Symp. on Circuits and Systems, Rome*. 1210–1213.

SCHRIJVER, A. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, New York.

VAZIRANI, V. 2001. *Approximation Algorithms*. Springer, New York.

Received October 2005; revised January 2006; accepted February 2006