Rectangle Covers Revisited Computationally

L. Heinrich-Litan[1] and M.E. Lübbecke[2]

[1] Technische Universität Braunschweig,
Institut für Mathematische Optimierung,
Pockelsstraße 14, D-38106 Braunschweig, Germany
litan@tu-bs.de
[2] Technische Universität Berlin,
Institut für Mathematik, Sekr. MA 6-1, Straße des
17. Juni 136, D-10623 Berlin, Germany
m.luebbecke@math.tu-berlin.de

Abstract. 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. 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.

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 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 [5] 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". In fact, we provide a fresh experimental

S.E. Nikoletseas (Ed.): WEA 2005, LNCS 3503, pp. 55–66, 2005.

view, the first of its kind, on the problem which has applications in the fabrication of masks in the design of DNA chip arrays [11], in VLSI design, and in data compression, in particular in image compression.

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 [15, 17, 18, 20]: 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. That is a set of vertices, any two of which are adjacent. Let θ denote 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 $\alpha = \theta$, and this is true for convex polygons [6] 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 [6] an example is mentioned with $\theta/\alpha \geq 21/17 - \varepsilon$, however, this example cannot be reconstructed from [6], and thus cannot be verified. The best proven bound is $\theta/\alpha \geq 8/7$.

For polygons with holes and even for those without holes (also called simple) the rectangle cover problem is $\mathcal{NP}$-hard [16, 7] and Max$\mathcal{SNP}$-hard [4], that is, there is no polynomial time approximation scheme. The best approximation algorithms known achieve a factor of $O(\sqrt{\log n})$ for general polygons [1] and a factor of 2 for simple polygons [8], where n is the number of edges of the polygon. Because of the problem's hardness quite some research efforts have gone into finding polynomially solvable special cases; we mention only covering with squares [2, 14] and polygons in general position [5]. Interestingly, there is a polynomial time algorithm for partitioning a polygon into non-overlapping rectangles [19]. However, a polygon similar to Fig. 3 shows that an optimal partition size may exceed an optimal cover size by more than constant factor, so this does not lead to an approximation.

Our Contributions. Despite its theoretical hardness, we demonstrate the rectangle cover problem to be computationally very tractable, in particular by studying an integer programming formulation of the problem. Doing this, we are the first to offer an exact (of course non-polynomial time) algorithm to obtain provably optimal solutions, and we are the first to introduce linear/integer programming techniques in this problem area. Based on a fractional solution to the (dual of the) linear programming relaxation we propose a stronger lower bound on the optimum cover size 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 [9], rounding a fractional solution, and a dual fitting algorithm [21]. We are optimistic that our research will actually contribute to a positive answer to the initially stated long standing open question, and due to space limitations

we only sketch some partial results and promising ideas. A fruitful contribution of our work is a number of open questions it spawns.

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 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 [8], see also Fig. 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 [20]. 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:

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

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

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

This integer program (which we call the *primal* program) allows us to optimally solve any given instance of our problem, and we will do so in our experiments. When we replace (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 [20]. In our case, however, the number of variables and constraints is polynomial in n, in fact quadratic, due to the fact that we work with maximal rectangles only. Therefore, the fractional cover size $\bar{\theta}$ can be computed in polynomial time.

This integer program 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 [4] for which no exact algorithm was known. 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:

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

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

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

Again, when replacing (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 stress again 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 [10], optimal solutions are automatically integer because of a strong duality between the *integer* programs [20]. This link was established already early, see e.g., [3,17,18], 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; therefore we discuss some observations first. 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.

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; and this is an excellent quality practical heuristic, though memory expensive for very large instances.

Odd Holes. Figure 1 (left) shows Szemerédi's counterexample to the $\alpha = \theta$ conjecture. The 5 rectangles indicated by the shaded parts have to be in any cover. In the remaining parts of the polygon, there are 5 pixels which induce an odd-length cycle C ("odd hole") in the visibility graph G. To cover these pixels, at least 3 rectangles are needed, implying $\theta \geq 8$. On the other hand, at most 2 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

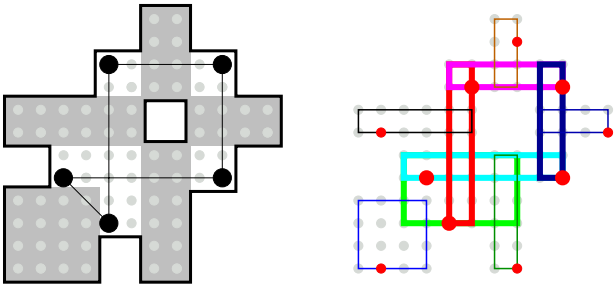

Fig. 1. The original counterexample to $\alpha = \theta$ by Szemerédi and (to the right) an optimal fractional cover. Thicker lines (points) indicate rectangles (pixels) which are picked to the extent of 0.5

optimal fractional stable set. Pixels corresponding to vertices of C assume a value of 0.5 (drawn fatter in the figure). That is, $\bar{\alpha} = \bar{\theta} = 7.5$. This immediately suggests to strengthen the LP relaxation.

Lemma 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. However, we can prove that there is no way of representing even the simplest non-trivial antihole with 7 vertices in a rectangle visibility graph. Odd holes are therefore the only reason for imperfection. Unfortunately still, from our experiments, arbitrary fractions are possible, not only halves, and simply rounding a fractional solution does not give a constant factor approximation, as discussed next.

High Coverage. We define the coverage of a pixel p as the number of rectangles which 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 Fig. 2). Unlike in the general set cover case, high coverage is no prediction about the fractions in an optimal LP solution: In Fig. 2 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. *In an optimal cover $\mathcal{C}$, every rectangle $r \in \mathcal{C}$ contains a pixel which 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*. It is no coincidence that the pixels in a maximal stable set are private. It is natural to ask

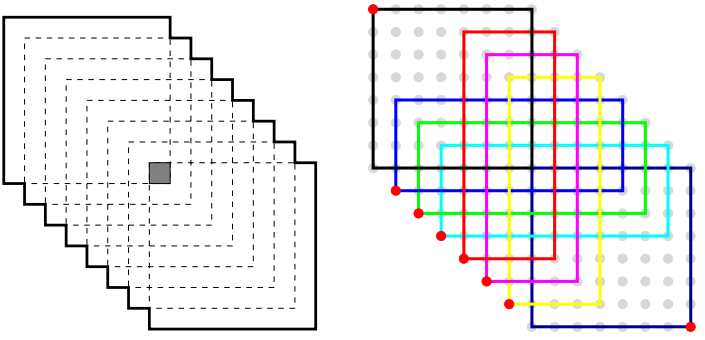

Fig. 2. Left: The shaded center pixel is covered by *any* maximal rectangle; almost all pixels have non-constant coverage. In an optimal cover, the coverage of the center pixel is linear in the cover size. The right figure schematically shows a minimal cover and a maximal stable set

(since an answer immediately turns LP rounding into a constant factor approximation algorithm): 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? These questions are intimately related to 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 Primal-Dual Scheme

The primal-dual scheme [9] builds on relaxing the well-known complementary slackness optimality conditions [20] 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 non-trivial class which allows a constant factor approximation.

3.2 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 [8]. Since $\alpha \leq \bar{\alpha}$ we propose to use the stronger bound provided by a *fractional* stable set. Our *dual fitting* approach [21] is to simultaneously construct a cover $\mathcal{C} \subseteq R$ and an *pseudo stable 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 (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

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

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 $(c \cdot d)$-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 Towards 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 which 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 3.2, and we mainly have to say how to construct a feasible dual fractional solution.

We use some terminology from [11]. 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 [11]. 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 [11]. 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 [11], 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 pseudo stable set containing a maximal set of leafs and quasi-leafs. Thus, in order to build a pseudo stable set we check for every rectangle in the greedy cover whether it contains

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

The first positive test gives a pixel which we add to the pseudo stable 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 4 corner pixels, and since corner pixels are private, actually at most 2 of them. By our previous considerations, this would imply a 2-approximation if the constructed pseudo stable set would pay for the whole cover. In general, we found 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 pseudo stable 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 non-polygon 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 non-polygon 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. Also these pixels have a natural motivation.

Let us furthermore remark that after the pruning step in Quasi-Greedy, every rectangle in the cover contains a private pixel (Lem. 2). This pixel is an intuitive candidate to become a pixel in a pseudo stable 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 [4]. 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 Fig. 3 shows that there is an $O(\sqrt{n})$-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 that one can find an interior cover of size less than $c_1 \cdot \theta_{\text{boundary}} + c_2 \cdot \alpha$ where c_1 are c_2 are appropriate constants. This would imply a constant factor approximation for the rectangle cover problem.

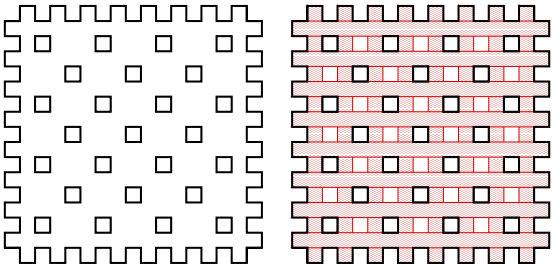

Fig. 3. A boundary cover may leave a non-constant fraction of pixels uncovered

4.3 Quasi-Prime Rectangles and Breaking Holes

There is a large class of polygons (e.g., polygons resulting from typical oligonucleotide masks [11]) where the optimal cover is found after the first two steps of the Quasi-Greedy algorithm in Section 4.1. Then the cover consists of only prime and quasi-prime rectangles. This is of course in general not the case (see Fig. 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 pseudo stable 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, too.

We have experimented with related and extended ideas based on the observations sketched in Sections 4.2 and 4.3 and obtained encouraging results. These methods and their approximation potential are currently under investigation.

5 Computational Experience

We experimented with small polygons occurring in VLSI mask design (instances `VLSI*`), a set of standard fax images[1] (instances `ccitt*`), and several black and white images (instances `marbles`, `mickey`, ...). Further, we have two strategies to construct random polygons. The first is to eliminate a varying fraction of single pixels uniformly from a square of size up to 750×750 pixels. The second is a union of uniformly placed rectangles of random sizes.

Table 1. Results for the primal and dual linear/integer programs. For each instance we list its size in pixels, its number of pixels (as a fraction), and its number of 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

	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%	7422.000	7422	0.000%
`ccitt3`	2376×1728	8.21%	40625	21085.000	21085	0.000%	21085.000	21086	0.005%
`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%

The extremely small integrality gaps listed in Tab. 1 and experienced for thousands of random polygons (not listed here) are a strong vote for our integer programming approach. On the downside of it, integer programs for industrial

[1] Available at `http://www.cs.waikato.ac.nz/~singlis/ccitt.html`

Table 2. Details for the Quasi-Greedy algorithm of Section 4.1. We compare the optimal cover size against ours (they differ by only 3–7%). The following columns list the number of prime and quasi-prime rectangles, and those picked by the greedy step. Then, the number of corner and border pixels in the constructed quasi stable set $\mathcal{S}$ is 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

Instance	optimum cover size	prime	quasi-prime	greedy	corner	border	max pixels	pays for
VLSI1	43	43	41	2	0	0	1	100.00%
VLSI2	4224	4701	1587	203	2911	1105	1279	4	88.79%
VLSI3	71	71	71	0	0	0	0	1	100.00%
ccitt1	14377	14457	10685	2099	1673	1632	28	2	99.91%
ccitt2	7422	7617	3587	409	3621	3574	29	3	99.76%
ccitt3	21086	21259	15691	2020	3548	3427	86	3	99.84%
ccitt4	56901	57262	42358	8605	6299	6110	59	2	99.77%
ccitt5	24739	24911	18529	2985	3397	3259	98	2	99.84%
ccitt6	12014	12132	8256	1049	2827	2764	35	2	99.77%
ccitt7	52508	52599	39230	10842	2525	2448	56	2	99.96%
ccitt8	14025	14303	7840	1353	5110	5023	54	3	99.77%
marbles	56354	44235	43548	687	0	0	0	1	100.00%
mickey	9132	9523	5582	690	3251	528	1593	3	88.13%
day	32192	32431	26308	3777	2346	749	900	4	97.85%
night	7943	8384	4014	501	3869	762	1810	4	84.53%

size polygons, e.g., from VLSI design are extremely large. The generation of the integer programs consumes much more time than solving them which takes typically only a few seconds using the standard solver CPLEX [13]. As a remedy we propose a column generation approach, that is, a dynamic generation of the variables of the linear program. This enables us to attack larger instances.

For random instances the relation between the different objective function values is very similar to Tab. 1 and is not reported separately in this abstract. The excellent performance of the Quasi-Greedy algorithm can be seen in Tab. 2. We remark that we never observed more than 4 pixels of a pseudo stable set in a rectangle, and the pseudo stable set pays for significantly more than 50% of the cover size. This supports that Quasi-Greedy could be an 8-approximation algorithm for the rectangle cover problem (see Section 4.1).

6 Conclusions

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:

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 Fig. 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

1. V.S. Anil Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. *SIAM J. Comput.*, 32(6):1509–1541, 2003.
2. L.J. Aupperle, H.E. Conn, J.M. Keil, and J. O'Rourke. Covering orthogonal polygons with squares. In *Proc. 26th Allerton Conf. Commun. Control Comput.*, pages 97–106, 1988.
3. C. Berge, C.C. Chen, V. Chvátal, and C.S. Seow. Combinatorial properties of polyominoes. *Combinatorica*, 1:217–224, 1981.
4. P. Berman and B. DasGupta. Complexities of efficient solutions of rectilinear polygon cover problems. *Algorithmica*, 17(4):331–356, 1997.
5. M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In Hochbaum [12], chapter 8, pages 296–345.
6. S. Chaiken, D.J. Kleitman, M. Saks, and J. Shearer. Covering regions by rectangles. *SIAM J. Algebraic Discrete Methods*, 2:394–410, 1981.
7. J.C. Culberson and R.A. Reckhow. Covering polygons is hard. *J. Algorithms*, 17:2–44, 1994.
8. D.S. Franzblau. Performance guarantees on a sweep-line heuristic for covering rectilinear polygons with rectangles. *SIAM J. Discrete Math.*, 2(3):307–321, 1989.
9. M.X. Goemans and D.P. Williamson. The primal-dual method for approximation algorithms and it application to network design problems. In Hochbaum [12], chapter 4.
10. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.
11. S. Hannenhalli, E. Hubell, R. Lipshutz, and P.A. Pevzner. Combinatorial algorithms for design of DNA arrays. *Adv. Biochem. Eng. Biotechnol.*, 77:1–19, 2002.
12. D.S. Hochbaum, editor. *Approximation Algorithms for $\mathcal{NP}$-Hard Problems*. PWS Publishing Co., Boston, MA, 1996.
13. ILOG Inc., CPLEX Division. *CPLEX 9.0 User's Manual*, 2004.
14. C. Levcopoulos and J. Gudmundsson. Approximation algorithms for covering polygons with squares and similar problems. In *Proceedings of RANDOM'97*, volume 1269 of *Lect. Notes Comput. Sci.*, pages 27–41, Berlin, 1997. Springer.
15. F. Maire. Polyominos and perfect graphs. *Inform. Process. Lett.*, 50(2):57–61, 1994.
16. W.J. Masek. Some NP-complete set covering problems. Unpublished manuscript, MIT, 1979.
17. R. Motwani, A. Raghunathan, and H. Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. System Sci.*, 40:19–48, 1989.
18. R. Motwani, A. Raghunathan, and H. Saran. Perfect graphs and orthogonally convex covers. *SIAM J. Discrete Math.*, 2:371–392, 1989.
19. T. Ohtsuki. Minimum dissection of rectilinear regions. In *Proc. 1982 IEEE Symp. on Circuits and Systems, Rome*, pages 1210–1213, 1982.
20. A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003.
21. V.V. Vazirani. *Approximation Algorithms*. Springer, Berlin, 2001.