A Greedy Algorithm for a Special Class of Geometric Set Covering Problems

Stolpe, Mathias; Bechmann, Andreas

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Stolpe, M., & Bechmann, A. (2012). A Greedy Algorithm for a Special Class of Geometric Set Covering Problems. DTU Wind Energy. (DTU Wind Energy E; No. 0013(EN)).

DTU Wind Energy

DTU

A Greedy Algorithm for a Special Class of Geometric Set Covering Problems

Department of Wind Energy Report 2012

Mathias Stolpe and Andreas Bechmann

DTU Wind Energy Report E-0013(EN)

October 2012

DTU Vindenergi
Institut for Vindenergi

Authors: Mathias Stolpe and Andreas Bechmann
Title: A Greedy Algorithm for a Special Class of Geometric Set Covering Problems
Department: Department of Wind Energy

Summary:

We consider the problem of covering a set of given points in the plane by the smallest number of axis aligned squares of a given fixed size. This problem is of importance for computational fluid dynamics simulations of both onshore and offshore wind turbine parks. For this special case of a geometric set covering problem we propose a greedy type algorithm. We also propose a linear mixed 0 – 1 formulation of the problem. For each problem instance this formulation is solved by a commercial branch-and-cut solver and the results are used to validate the quality of the solution from the greedy algorithm. The greedy algorithm finds the minimum number of squares for all but one problem instances from a set of 26 representative real-world examples.

DTU Wind Energy Report E-0013(EN)
October 2012

ISBN 978-87-92896-19-3

Contract no.:
11/02917

Project no.:
43087

Sponsor:
Business Innovation Fund
(www.fornyelsesfonden.dk)

Pages: 7
Tables: 1
References: 6

Technical University of Denmark
Department of Wind Energy
Frederiksborgvej 399
Building 118
4000 Roskilde
Denmark
Telefon 21 51 82 40
matst@dtu.dk
www.vindenergi.dtu.dk

1 Introduction

We consider a special class of a geometric set covering problem. The problem can be stated as follows: Given a set of n points in the plane $\mathbb{R}^2$, find the minimum number and positions of axis-aligned and potentially over-lapping squares of fixed width r which cover all points.

There is clearly a trivial feasible solution to any instance of this optimization problem. Let each point be the center of a square. This gives a solution with n squares which is unlikely to be optimal except for pathological situations.

The application we have in mind for this geometric set covering problem stems from computational fluid dynamic (CFD) simulations of wind turbine parks. The points refer to locations of wind turbines and meteorology masts while the squares represent computational domains. In order to adapt the CFD simulations to the wind resource assessment program WAsP [6] the size and orientation of the squares have been chosen to be fixed. Since the CFD simulations are expensive parallel computations we are interested minimizing the number of squares.

For a statement of the classical set covering problem we refer to e.g. [5]. A greedy heuristic for the set covering problem is presented in [2]. Studies of various types of geometric set covering problems are presented in recent articles such as [3] and [1].

2 Problem Reformulation

The stated geometric set covering problem can, perhaps not surprisingly, be almost equivalently reformulated as a linear mixed 0–1 problem with the use of certain "big-M" coefficients. We introduce the continuous variables $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ representing the center positions of the squares, and let the indicator variables $z_{ik} \in \{0,1\}$ be interpreted as

$$z_{ik} = \begin{cases} 1 & \text{if point } i \text{ is covered by square } k\text{, and} \\ 0 & \text{otherwise.} \end{cases}$$

We also introduce the binary variables $v_k \in \{0,1\}$ with the interpretation

$$v_k = \begin{cases} 1 & \text{if square } k \text{ is used to cover some points, and} \\ 0 & \text{otherwise.} \end{cases}$$

The variables z_{ik} are only allowed to be one if the corresponding v_k variable is one and are forced to be zero if v_k is zero. These requirements are modelled by the linear inequalities

$$z_{ik} \leq v_k \quad \forall\, (i,k).$$

Each point should be assigned to exactly one square. This is modelled by the linear equalities

$$\sum_{k=1}^{n} z_{ik} = 1 \quad \forall\, i = 1,\dots,n.$$

Notice that these constraints do not avoid squares which are overlapping, nor do they forbid a point to be covered (in the geometrical sense) by more than one square. We minimize the total number of used squares, i.e., we minimize the linear function

$$\sum_{k=1}^{n} v_k$$

We also ensure that the given points are inside the correct square. This can be modelled through the linear inequalities

$$-(1-z_{ik})M + x_k - r/2 \leq \bar{x}_i \leq x_k + r/2 + (1-z_{ik})M \quad \forall\, (i,k)$$

and

$$-(1-z_{ik})M + y_k - r/2 \leq \overline{y}_i \leq y_k + r/2 + (1-z_{ik})M \quad \forall\,(i,k),$$

where $M > 0$ is a sufficiently large constant and $\overline{p}_i$ denotes the given position of the i-th point, i.e., $\overline{p}_i = \begin{pmatrix}\overline{x}_i & \overline{y}_i\end{pmatrix}^T$. The coefficients M can be estimated from the problem data. The problem formulation finally becomes

$$\begin{array}{lll}
\underset{v,z,x,y}{\text{minimize}} & \sum_{k=1}^{n} v_k & \\
\text{subject to} & \sum_{k=1}^{n} z_{ik} = 1 & \forall\, i \\
& z_{ik} \leq v_k & \forall\,(i,k) \\
& -(1-z_{ik})M + x_k - r/2 \leq \overline{x}_i & \forall\,(i,k) \\
& -(1-z_{ik})M - x_k - r/2 \leq -\overline{x}_i & \forall\,(i,k) \\
& -(1-z_{ik})M + y_k - r/2 \leq \overline{y}_i & \forall\,(i,k) \\
& -(1-z_{ik})M - y_k - r/2 \leq -\overline{y}_i & \forall\,(i,k) \\
& z_{ik} \in \{0,1\} & \forall\,(i,k) \\
& v_k \in \{0,1\} & \forall\, k.
\end{array} \tag{1}$$

The size of problem (1) can be reduced if, instead of allowing n squares to be present, only a smaller number of squares is allowed.

3 The Algorithms

In this section we propose a greedy type algorithm (see e.g. [5]) for the stated geometric set covering problem. Before the greedy algorithm is called some inexpensive pre-processing is performed.

3.1 Detecting Trivial Solutions

If all points are sufficiently close then only one square suffices to cover all points. This situation is detected by first computing the distance between all given points and then checking if

$$\max_{k \neq i}\{\max\{|\overline{x}_i - \overline{x}_k|, |\overline{y}_i - \overline{y}_k|\}\} \leq r. \tag{2}$$

If the inequality in (2) is satisfied then one square suffices. The square can for example be placed with center position at

$$\frac{1}{2}\left(\min_i\{\overline{x}_i\} + \max_i\{\overline{x}_i\}, \min_i\{\overline{y}_i\} + \max_i\{\overline{y}_i\}\right).$$

3.2 Problem Size Reduction

The problem can potentially be reduced in size by simple pre-processing. Points which are sufficiently far away from all other points require a separate square and can thus be disregarded from further consideration. A point i is isolated, and requires its own square, if the Euclidean distance to all other points is larger than $\sqrt{2}r$, i.e., if

$$\|\overline{p}_i - \overline{p}_k\|_2 > \sqrt{2}r \quad \forall\, i \neq k$$

or if the distances in x- and y-direction to all other points are too large, i.e., if

$$\min_{k \neq i}\{\max\{|\overline{x}_i - \overline{x}_k|, |\overline{y}_i - \overline{y}_k|\}\} > r.$$

These points are detected and assigned a square before the greedy algorithm is called.

3.3 The Greedy Algorithm

In the description of the greedy algorithm $\mathcal{C}$ and $\mathcal{U}$ denote the sets of currently covered and uncovered points, respectively.

Algorithm 1: The greedy algorithm for the special case of geometric set covering problem.

```
Let $\mathcal{C} = \emptyset$ and $\mathcal{U} = \{1, \dots, n\}$ and choose parameters $0 < \alpha \leq 1$ and $0 < \beta < 1$.
while $|\mathcal{U}| > 0$ do
    foreach $i \in \mathcal{U}$ do
        Construct the set of candidate points
            $\mathcal{S}_i = \{k \in \mathcal{U} \mid \|\overline{p}_i - \overline{p}_k\|_\infty \leq \alpha r\}.$
        If the points in $\mathcal{S}_i$ cannot be covered by a square of size $r$ then set $\mathcal{S}_i = \emptyset$.
    end
    if $S_i = \emptyset$ for all $i \in \mathcal{U}$ then
        Set $\alpha \leftarrow \alpha\beta$.
    else
        Compute $\hat{i} = \arg\max_{i \in \mathcal{U}} \{|\mathcal{S}_i|\}$.
        Set $\mathcal{C} \leftarrow \mathcal{C} \cup S_{\hat{i}}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{\hat{i}}$.
    end
end
```

Algorithm 1 will terminate in a finite number of steps. In the worst case it will find the trivial feasible solution that each point is covered by exactly one square. The algorithm described in Algorithm 1 is not truly greedy since there are no guarantees that the square which covers the most points is found in the top of the for-loop.

4 Implementation and Numerical Results

In this section we describe the implementation of the greedy algorithm and present numerical results obtained from solving a set of real-world problem instances. In connection with the EWEA conference in Copenhagen 2012, WAsP users were given the opportunity to have their wind turbine sites analyzed with CFD for free. The only requirements to the users were that the sites had recently been analyzed by WAsP and that the sites were located in complex terrain. In total 41 sites were analyzed but only 26 actually included wind turbine positions. These 26 sites have been used in the present work and should be representative of modern wind turbine parks in complex terrain.

4.1 Implementation and Parameters

The pre-processing algorithms and the greedy algorithm described in Section 3 are all implemented in MATLAB. In our implementation the parameters $\alpha = 1$ and $\beta = 0.9$.

In the implementation the coefficient M in the reformulated problem (1) is set to $M = 100r$, where $r = 2$ km is the fixed width of the squares in all examples. The problem formulation (1) is generally reduced in size by allowing fewer than n squares. In the implementation the maximum number of squares is set to $\min\{n, 15\}$. These choices of parameters are sufficient to allow feasible points to the linear mixed 0–1 reformulation (1) for all problem instances in the test set.

The reformulated problem (1) is solved by the efficient and robust branch-and-cut solvers in CPLEX version 21.1.1 [4]. No feasible points or valid inequalities are provided to the branch-and-cut method. All parameters and tolerances in CPLEX are set to default values, except for the number of threads which is set to one.

The numerical experiments are run on an AMD Opteron 6168 running at 1.9GHz.

4.2 Numerical Results

The greedy algorithm and the mixed integer 0–1 reformulation are used to solve all instances from the set of 26 real-world problems presented in Table 1. In the table we present the number of points (Points), the number of points assigned squares in the pre-processing phase (Pre), the number of squares suggested by the greedy algorithm (Greedy), the number of squares suggested by CPLEX, the number of nodes used by CPLEX (Nodes), and finally, the time in seconds used by CPLEX to solve the linear mixed 0–1 problem (1) (Time (s)).

Table 1. Problem statistics and computational results with the greedy algorithm and CPLEX *applied to the linear mixed 0–1 reformulation (1).*

Problem	Points	Pre	Greedy	CPLEX	Nodes	Time (s)
1	14	8	11	11	490	0.92
2	55	0	5	5	505	24.34
3	23	0	4	4	17	0.46
4	52	0	11	11	482	35.92
5	34	0	5	5	459	7.37
6	27	0	4	4	7	0.36
7	16	0	2	2	3	0.09
8	17	0	1	1	0	0.01
9	16	0	2	2	5	0.11
10	35	0	3	3	178	1.17
11	40	0	6	6	490	8.49
12	9	0	2	2	8	0.03
13	5	0	1	1	0	0.01
14	18	0	3	3	7	0.20
15	65	0	9	8	205	17.57
16	14	0	2	2	9	0.08
17	44	0	7	7	914	16.31
18	45	0	6	6	485	16.20
19	12	0	4	4	83	0.14
20	46	0	6	6	62	2.95
21	34	0	3	3	128	1.28
22	26	0	3	3	7	0.42
23	29	0	3	3	152	1.20
24	15	0	3	3	15	0.13
25	17	0	1	1	0	0.03
26	72	0	8	8	507	60.03

For three of the problems only one square is needed and this is recognized in the pre-processing phase. Similarly, CPLEX solves these three problems in the pre-processing phase without resorting to branch-and-cut. For two problems 11 squares are needed. On average 4.4 squares are needed to cover the given points. The pre-processing technique suggested in Section 3.2 is only able to reduce the size of Problem 1. One of the problems is shown in Figure 1. The distances are in kilometers (km.) and the size of the squares is 2×2 km. In the solution of this example there are several overlapping squares, but if necessary the solution can be manually post-processed.

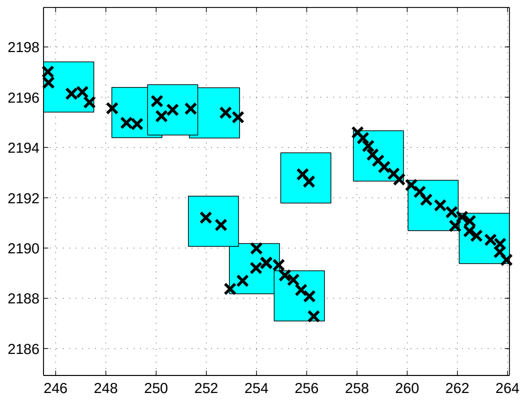

Figure 1. An example of a geometric set covering problem with 52 given points. The greedy algorithm suggests that 11 squares suffices to cover all points. The optimality of the solution is confirmed by solving the linear mixed 0–1 reformulation (1) of the set covering problem.

The branch-and-cut solver only requires, for most problems, less than a few seconds. Even the most difficult example in the test set requires less than a thousand nodes and a minute to be solved. The greedy algorithm needs less than 0.1 second in total for solving all problem instances. Surprisingly, the greedy algorithm finds the minimum number of squares for all but one of the problems. For Problem 15, CPLEX finds a solution with eight squares while the greedy algorithm suggests nine squares. Both solutions are shown in Figure 2. These observations suggest that the real-world problems in the set are moderately difficult to solve.

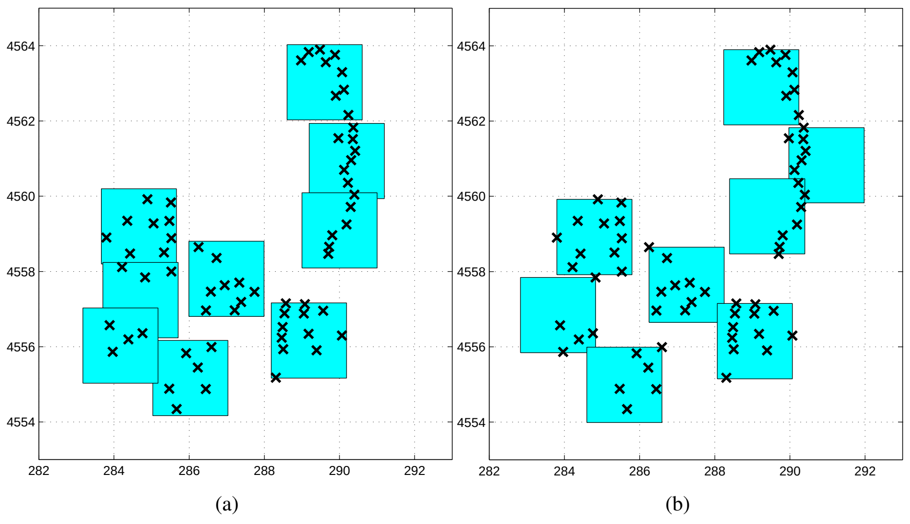

Figure 2. An example of a geometric set covering problem with 65 given points. The greedy algorithm suggests that nine squares suffices to cover all points (a) whereas the solution obtained by CPLEX *contains only eight squares (b).*

5 Conclusions

We consider the special geometric set covering problem of finding the smallest number of fixed size squares which cover a set of given points in the plane. For this problem we propose a simple greedy algorithm. Despite its simplicity, the algorithm finds the smallest number of squares for all but one problem instance in a set of 26 real-world benchmark problems. The results are confirmed by solving a linear mixed 0–1 reformulation of the problems to global optimality using a commercial branch-and-cut solver.

Acknowledgements

This work is part of the research project *Beregning af vindenergi – computermodellering af vindressourcer i komplekst terræn* which is co-funded by the Danish Business Innovation Fund. For additional information please see `www.fornyelsesfonden.dk`.

References

[1] C. Chekuri, K.L. Clarkson, and S. Har-Peled. On the set multi-cover problem in geometric settings. In *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, pages 341–350. ACM, 2009.

[2] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[3] K.L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

[4] IBM. *IBM ILOG CPLEX V12.1 User's Manual for CPLEX*, 2012. `www.ibm.com`.

[5] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999.

[6] I. Troen and E. Petersen. *European Wind Atlas*. Risø National Laboratory, Roskilde, 1989.

Technical University of Denmark
DTU Vindenergi
Frederiksborgvej 399
Building 118
4000 Roskilde
Denmark
Telefon 46 77 50 85
info@vindenergi.dtu.dk
www.vindenergi.dtu.dk