# Using SAT Solvers

to investigate Erdős–Szekeres Conjecture for e(7)=33

Dumitru Bogdan

Faculty of Computer Science
University of Bucharest

December 7, 2023

## Table of contents

Section 1

## Introduction

## Problem Statement

In their classic 1935 paper [3], Erdos and Szekeres proved that, for every integer $k \geq 3$, there is a minimal integer $e(k)$, such that any set of $e(k)$ points in the plane in general position contains $k$ points in convex position, that is, they are the vertices of a convex *k-gon*

The *happy ending problem* (as it is sometimes called) was one of the original results that led to the development of *Ramsey theory*

## Problem Statement

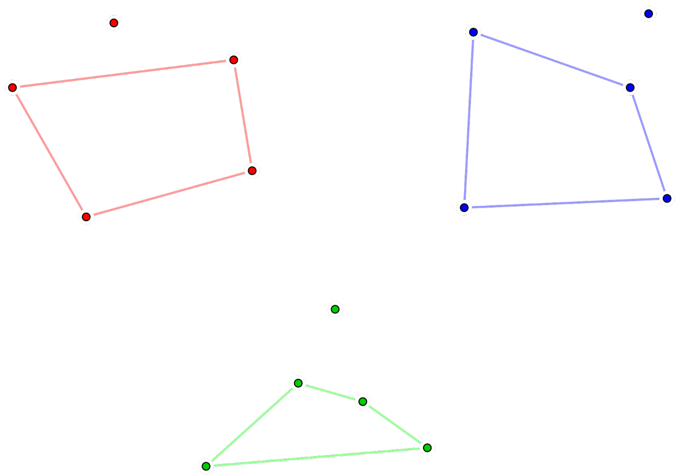

Figure: All possible configurations of five points

## Theoretical Results

- It has been proven [4] that $e(k) \geq 2^{k-2} + 1$ and conjectured this to be sharp
- In 2016, Suk [8] reduced the upper bound to $2^{k+o(k)}$
- In 2020, Holmsen et al. [5] claims an improvement on Suk, reducing the upper bound to $2^{k+O(\sqrt{k \log k})}$

## Computational Results

- In 2006 Szekeres and Peters [9] proposed a computer solution for $k = 6$ (1500 hours)
- In 2017 Balko and Valtr [1] proposed a ”A SAT attack” on the conjecture
- In 2017 Maric [7] proposed a fast formal proof for $k \leq 6$ (1700 sec)

- We investigate few methods to validate the conjecture for $k = 7$
- Using same methods for $k = 6$ we get a significantly lower time (80 sec)

Section 2

## Preliminaries

## Preliminaries

**Definition**

A set $S$ of $n$ points in cartesian space, is in general position if no 3 points are collinear

**Definition**

An ordered set of three points $\{p_1, p_2, p_3\} \in S$ where $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$ forms a *3cup* if the clockwise angle $\alpha$ formed by the base line passing thru $p_1$ and $p_2$ and the segment $p_2, p_3$ satisfies: $0 < \alpha < 180^\circ$

**Definition**

An ordered set of three points $\{p_1, p_2, p_3\} \in S$ where $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$ forms a *3cap* if the angle $\alpha$ formed by the base line passing thru $p_1, p_2$ and the segment $p_2, p_3$ satisfies: $180^\circ < \alpha < 360^\circ$

## Preliminaries

Intuitively if $p_3$ is on the right of the directed line passing thru $p_1, p_2$ we have a *3cup*. If $p_3$ is in the left of the line it is a *3cap*

From now on we use the term of *3c* to refer to an ordered set of three points which is either a *3cup* or a *3cap*

## Preliminaries

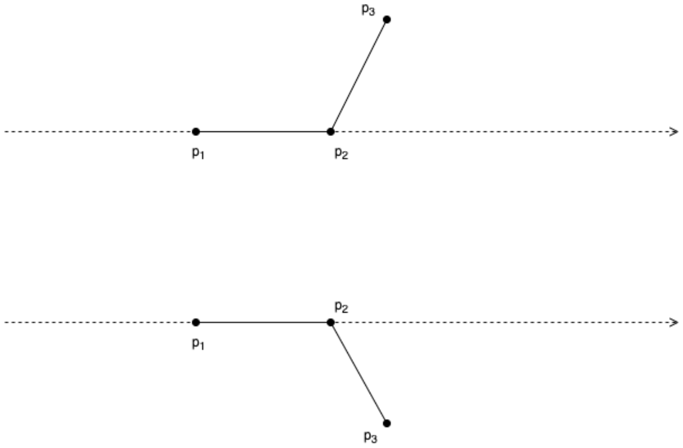

Figure: Visual representation of a 3cap (above) and a 3cup (below)

## Preliminaries

**Definition**

A chain of size $k$ or a k-chain is ordered set of $k$ *3cs* such that every two adjacent *3c* have the form $\{p_{i_1}, p_{i_2}, p_{i_3}\}$ and $\{p_{i_2}, p_{i_3}, p_{i_4}\}$, where $p_{i_k} \in S$

**Definition**

A cycle of size $k$ or a *k-cycle* is a *(k+2)-chain* with the property that $p_1 = p_{k+1}$ and $p_2 = p_{k+2}$ where $\{p_1, p_2, p_3\}$ is the first *3c* of the chain and $\{p_k, p_{k+1}, p_{k+2}\}$ is the last *3c* element of the chain

**Definition**

A cycle is *convex* if all the elements are *3cap* (*3cup*)

## Preliminaries

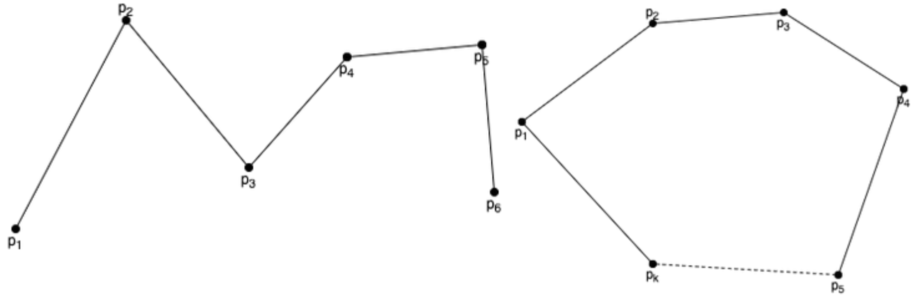

Figure: Visualisation of a chain and a cycle

## Preliminaries

**Definition**

A set $S$ of $k$ points is in *convex position* if exists an ordering of the set $S$ such that it forms a convex *k-cycle*

**Definition**

Given a set $S$ of $n$ points in general position, we define the set $\mathcal{S}$ of size $\binom{n}{3}$, as the set containing all possible ordered (by a point associated index) sub-sets of any three points from $S$ along with the type: *3cup* or *3cap*. The set $\mathcal{S}$ is called *3c extension* of $S$

## Preliminaries

### Example

Let $S = \{(0,0),(2,2),(0,2),(2,0)\}$. We assign index 1 to $(0,0)$, 2 to $(2,2)$, 3 to $(0,2)$ and index 4 to $(2,0)$. Arranging the set of 3c elements in lexicographic order we get:

$$\mathcal{S} = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}\}$$

We observe that $\{1,2,3\}$ and $\{2,3,4\}$ are *3cap* and $\{1,2,4\}$ and $\{1,3,4\}$ are *3cup*. We assign 0 to *3cup* and 1 to *3cap* and we get a simplified representation of $\mathcal{S}$ as:

$$\mathcal{S} = \{1,0,0,1\}$$

## Preliminaries

**Example**

For any $S$ with $|S| = 4$ we have 14 possible representations (or configurations): 6 convex configurations $\{(1,1,1,1),(0,0,0,0),(1,1,0,0),(0,0,1,1),(1,0,0,1),(0,1,1,0)\}$ and 8 non-convex configurations: $\{(1,0,0,0),(0,1,1,1),\dots\}$

## Preliminaries

- *3c* will be the primary tool in our encoding process. They will be the variables in our CNF formula
- To have a proper geometric representation of a set $S$ of points using *3c* we need have an axiomatic approach
- We will use *CC axioms* from Knuth [6]

## Preliminaries

**Axiom**

1. *If* $p_1p_2p_3$ *is a 3cup then* $p_2p_3p_1$ *is a 3cup (cyclic symmetry)*
2. *If* $p_1p_2p_3$ *is a 3cup then* $p_1p_3p_2$ *is a 3cap (antisymmetry)*
3. $p_1p_2p_3$ *is a 3c (nondegeneracy)*
4. *If* $p_1p_2p_4$, $p_2p_3p_4$ *and* $p_3p_1p_4$ *are 3cups then* $p_1p_2p_3$ *is a 3cup (interiority)*
5. *If* $p_1p_2p_3$, $p_1p_2p_4$, $p_1p_2p_5$, $p_1p_3p_4$ *and* $p_1p_4p_5$ *are 3cups then* $p_1p_3p_5$ *is a 3cup (dual transitivity)*

Section 3

## SAT encoding

## Boolean Satisfiability Problem

The Boolean satisfiability problem asks whether there is at least one combination of binary input variables $x_i$ for which a Boolean logic formula returns true. When this is the case, we say the formula is satisfiable.

A SAT solver is an algorithm for establishing satisfiability. It takes the boolean logic formula as input and returns **SAT** if it finds a combination of variables that can satisfy it or **UNSAT** if it can demonstrate that no such combination exists. In addition, it may sometimes return **UNKNOWN** if it cannot determine whether the problem is SAT or UNSAT

## SAT Encoding

We aim to transform our problem into a boolean logic formula and let SAT solver demonstrate that no combination of variables exists to satisfy it. SAT Solver should return **UNSAT**

Our goal is to find an efficient encoding both in terms of space and time. While we do not control how the SAT solver works, we will assume that if the sum of variables and clauses is lower, so is the running time

## Bad news/Good news

- *Bad news*: SAT problem is proven to be NP-complete and it follows that there is no known polynomial algorithm for establishing satisfiability in the general case (except 2SAT)
- *Good news*: modern SAT solvers are very efficient and can often solve problems involving hundreds of millions of variables and clauses in practice
- We will push current SAT solvers to their limits with our encoding

## First Encoding

- For each 3$c$ we create a (boolean) variable. In total we have $\binom{n}{3}$ because we use only 1 out of 6 possible combination for any 3 points. The correct relation between the 6 combinations is enforced by the axiom encodings
- We encode the set of 5 axioms. The biggest effort is with axiom 5 because we need to handle all $\binom{n}{5} \times 5!$ cases which of course lead to the same number of clauses for our SAT solver to handle
- We encode the non-convexity constrains. Given a $k$ points we need to search if there is a *k-cycle* on each possible configuration, which is $k!$. We need $\binom{n}{k} \times k!$ additional clauses
- We end up with a CNF formula with $\binom{n}{3}$ variables and $\binom{n}{5} \times 5! + \binom{n}{k} \times k!$ clauses
- For $n = 33$ and $k = 7$ we have 5454 variables and $2.15 \times 10^{10}$ clauses. Out of reach for any SAT Solver (most of them are unable even to load the CNF file)

## Second Encoding

- We keep the same encoding principles but we perform rudimentary improvements
- In case of axiom 5 encoding we can reduce the number of clauses three times by removing redundancy (heuristics, not yet a formal proof)
- For non-convexity constrains we can reduce the number of of clauses for a set of $k$ points by $k$ because we eliminate all rotations of a permutation, hence we need $\binom{n}{k} \times (k-1)!$
- For $n = 33$ and $k = 7$ we have 5454 variables and $2.15 \times 10^9$ clauses. Not enough!

## Sparsifying axiom 5

**Property**

*For any set $S$ of $n$ points, axiom 5 cases can be reduced by a factor of 3*

**Proof.**

**Proof sketch**: If $p_1p_2p_3$, $p_1p_2p_4$, $p_1p_2p_5$, $p_1p_3p_4$, $p_1p_4p_5$ and $p_1p_3p_5$ are $3cups$ then there is at least another configuration bounded by $p_3p_4$ and/or $p_3p_5$ to satisfy axiom 5 conditions $\square$

This property holds for $(n, k) = (9, 5)$ and $(n, k) = (17, 6)$. It seems to hold for $(n, k) = (33, 7)$. We can look at this relaxed axiom as a more general case of the problem and if it proves to be UNSAT then our more restricted case is also UNSAT

## Third Encoding

- We want to create an order relation on $S$. It will help reduce the number of clauses and the number of variables (by assign them a predefined value)
- One method (not particularly efficient) is to use a *sliding line* as follows:
  1. starting from the left we slide a vertical line until intersects a point from $S$
  2. we assign the index 0 to that point
  3. next we clockwise rotate the line using 0 point as pivot
  4. while rotating the line we intersect points from $S$ and we assign them an index in increasing consecutive order (e.g $1, 2, \dots |S| - 1$)
- After this procedure we end up with an ordered set $S$ and few interesting properties

## Third Encoding

**Property**

*Every 3c of the form $0p_ip_{i+k}$ where $i > 0$ and $k > i$ is a 3cup*

**Proof.**

Since there is no other point on the left of vertical line passing thru 0 then all points are on the right (under) the line passing thru 0 and 1, meaning that all *3c* of the form $01p$ are *3cup*. For all $0ij$, $j > i$ we reason in a similar manner $\square$

## Third Encoding

**Property**

*Given $k$ points, checking for k-cycle requires $2^{k-1}$ verification steps (or clauses)*

**Proof.**

Assume we have a *k-cycle*. Our sliding line with 0 as pivot will intersect at some moment a point form out *k-cycle*. Because we assign an index in increasing order to every point it means that this point has the lowest index in our *k-cycle*. Next points from our *k-cycle* will intersected by our line at some moment and they will receive a greater index. The next point intersected could be geometrically in two places: on the left (clockwise) of our first point or on the right. Hence $2^{k-1}$ possible configurations □

## Third Encoding

- Using this new encoding we now have $(n-2)(n-1)/2$ assigned variables
- Axiom 5 clauses remains the same as in previous encoding
- For non-convexity constrains we reduced the number of clauses to $\binom{n}{k} \times 2^{k-1}$
- For $n = 33$ and $k = 7$ we have 4960 variables and $1.46 \times 10^8$ clauses.
- Resulting CNF formula has apx. $6GB$, can be generated reasonably fast (apx. 3 min) and loaded by any performant SAT solver on an average system
- Now we have the tools to run more experiments, faster

## Third Encoding

We can implement and test different orderings on our set $S$. One particular variation provided good results (as time):

1. starting from the left we slide a vertical line until intersects a point from $S$
2. we assign the index 0 to that point
3. we rotate clockwise the line using 0 point as pivot until we intersect a point which will be assigned 1
4. next we rotate counter-clockwise using 1 as a pivot until we intersect a point
5. we repeat the last two steps until all points of $S$ are indexed

## Third Encoding

- While possible to run a SAT solver on a $n = 33$ encoding, after running for few months, the SAT solver seems to have been exhausted all tricks and optimisations (stuck in 0 eliminated clauses and 0 variable removed)
- This is because all this CNF encoding are highly symmetrical (all variables appear in the same number and type of clauses)
- This means(?) that SAT solver is doing brute-force search on the remained combinatorial space
- We do not know how large this space is so it is safe to assume it is large
- A stronger encoding is needed, with fewer clauses and more assigned variables

## Fourth Encoding

- All selected encodings had the property of being realistically implementable, testable for $n = 33$
- We have avoided costly constructions and conditions (e.g. AtMostK, AtLeastK). Only, *AtLeastOne* or *All* constraints were allowed
- Generating CNF files was done using *python+pysat* and *nim*
- Kissat SAT solver [2] was mostly used as it is the fastest and has the most added heuristics on top of classic CDCL algorithm
- Parallel SAT solvers were used without efficient output
- *Note*: it is time for a truly parallel SAT solver, but CDCL approach must be dropped (e.g. 1-bit transformers)

## Fourth Encoding

- To further reduce the number of clauses we focus on non-convexity constraints and we try to generalize the concept of 3$c$ by moving one dimension up to 4$c$, to *4-gons* more precisely
- To increase the number of assigned variables but more importantly to distribute the assignments more widely (in previous encodings assigned variables where all *3c* starting with 0) we devise another ordering method that will split our problem in sub-problems
- Next two properties facilitates this approach

## Fourth Encoding

**Property**

*All 4-gons of a convex n-gon* $(n \geq 4)$ *are convex*

**Proof.**

By assuming there is at least one non-convex *4-gon* it follows that the *n-gon* is non-convex which is a contradiction □

## Fourth Encoding

**Property**

*Every set $S$ of $n$ points can be written as an union of sub-set where every sub-set is a convex $l$-gon ($l \leq k$)*

**Proof.**

We construct our sub-sets as follows: Start with a vertical line on the left-most of the cartesian plane. We move it to the right until we intersect a point. Next, using that point as pivot we rotate the line clockwise until we intersect with another point. The last intersected point becomes the pivot. We continue this process until we intersect the initial point. At this moment we have a *$l$-gon*. We remove *$l$-gon* points from $S$ and we start the process again, until we eliminated all the points. $\square$

## Fourth Encoding

### Definition

The union of convex *l-gons* $L_1, L_2, \ldots, L_r$ is called a configuration and is denoted $|L_1|, |L_2|, \ldots, |L_r|$

### Example

If we have a set $S$ of 9 points and the first/outer convex polygon is a triangle, the second one is a triangle and the third one is also a triangle then the configuration of $S$ is $3, 3, 3$

## Fourth Encoding

- Again, we start our encoding with *3c* variables and axiom clauses
- We add a 14 new variables for each *4-gon* which are set in accordance with the configuration of the *4-gon* (see example with $\mathcal{S}$)
- If any of last 8 variables is *true* then the *4-gon* is convex
- Then we add a simple *if AllLeastOneTrue* clause for any *k-gon* and this will ensure non-convexity constrain
- In total we add $(6+1)\binom{n}{4}$ new variables and $((14 * 5 + 1) + 1)\binom{n}{k}$ clauses to our CNF

## Fourth Encoding

- Regarding the ordering of points of $S$, based on previous property we transform $S$ into a union of *l-gons*, $l < k$
- We assign an index in increasing order to every point of the outer *l-gon*, then we continue with the next ones based on inclusion order
- Now we can assign *true* value to any *3c*(variable) of the form $p_{i_1}p_{i_2}p_{i_3}$, where $p_{i_1}$ and $p_{i_2}$ are adjacent vertices of a *l-gon* and $p_{i_3}$ is any point inside the *l-gon*
- For $(n, k) = (33, 7)$ we assign 661 variables, compared to 496 when using the sliding method
- The main benefit here is that we spread the assignments into the whole structure of $S$

## Fourth Encoding

- For $3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3$ configuration of $S$, we end up with 578336 variables and 16671965 clauses
- For $6, 6, 6, 3, 6, 6$ configuration, we have have 16671435 clauses
- Compared with previous encodings ($10^9$, $10^8$ clauses), we now have $10^7$ clauses
- Regarding variables we now have quite a big number. However active variables are the same
- The worst part is that for $n = 33$ we can have $3.1 \times 10^4$ possible configurations
- *Axiom* 5 *clauses make up more than half of the total clauses*

Section 4

## Results

## $(n, k) = (17, 6)$

- Using sliding ordering and *4-gon* non-convexity constraints we find UNSAT in apx. 80 sec on an average system
- Using sliding ordering and regular non-convexity constraints we find UNSAT in apx. 200 sec on an average system
- Using convex union ordering and *4-gon* non-convexity constraints we find UNSAT in apx. 5 sec on an average system for $3, 3, 3, 3, 3$ configuration
- Using convex union ordering and *4-gon* non-convexity constraints we find UNSAT in apx. 1 sec on an average system for $5, 6, 4$ configuration
- We have 96 different configurations

## $(n, k) = (33, 7)$

- Using sliding ordering and 4-gon non-convexity constraints solver is still running after many months
- Using sliding ordering and regular non-convexity constraints is still running
- Using convex union ordering and 4-gon non-convexity constraints we find UNSAT in apx. 2 days on an average system for (example) $6, 6, 6, 3, 6, 6$ configuration
- Using convex union ordering and 4-gon non-convexity constraints we find UNSAT in apx. 7 days on an average system for $3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3$ configuration
- Over 40 configurations have been tested
- We have 31422 different configurations
- $e(7) = 33$ is now tangible!

## $(n, k) = (33, 7)$

- Testing one configuration requires a SAT solver instance that uses 1 thread and apx 5GB of memory
- Over 10 instances can be executed simultaneously on a regular system (starting time should be deferred)
- Using 1000 systems or cloud instances that run only 1 solver instance for 4 days, $e(7)$ can be closed in **4 month**

## Next steps

- Reducing axiom 5 clauses will reduce average time per instance
- No intention to continue further unless FMI is seeing a benefit
- After 5-10 years if no advance is done, we retest using up to date hardware
- A new approach, highly parallel SAT Solver?
- $\mathit{Ramsey}(5,5)$ :)

Section 5

## References

## References I

- Martin Balko and Pavel Valtr. "A SAT attack on the Erdős–Szekeres conjecture". In: *European Journal of Combinatorics* 66 (Dec. 2017), pp. 13–23. ISSN: 0195-6698. DOI: 10.1016/j.ejc.2017.06.010. URL: http://dx.doi.org/10.1016/j.ejc.2017.06.010.
- Armin Biere. *GitHub - arminbiere/kissat — github.com*. https://github.com/arminbiere/kissat. [Accessed 06-12-2023].
- P. Erdos and G. Szekeres. "A combinatorial problem in geometry". In: (1935), pp. 463–470.
- P. Erdos and G. Szekeres. "On some extremum problems in elementary geometry". In: (1961), pp. 53–63.
- Andreas F. Holmsen et al. "Two extensions of the Erdős-Szekeres problem". In: (2020). arXiv: 1710.11415 [math.CO].

## References II

Donald E Knuth. *Axioms and Hulls*. en. Ed. by Donald E Knuth. 1992nd ed. Lecture notes in computer science. Berlin, Germany: Springer, June 1992.

Filip Marić. "Fast Formal Proof of the Erdős–Szekeres Conjecture for Convex Polygons with at Most 6 Points". In: *Journal of Automated Reasoning* 62.3 (Sept. 2017), pp. 301–329. ISSN: 1573-0670. DOI: 10.1007/s10817-017-9423-7. URL: http://dx.doi.org/10.1007/s10817-017-9423-7.

Andrew Suk. "On the Erdos-Szekeres convex polygon problem.". In: *CoRR* abs/1604.08657 (2016). URL: http://dblp.uni-trier.de/db/journals/corr/corr1604.html#Suk16.

George Szekeres and Lindsay Peters. "Computer solution to the 17-point Erdős-Szekeres problem". In: *The ANZIAM Journal* 48.2 (Oct. 2006), pp. 151–164. DOI: 10.1017/s144618110000300x. URL: https://doi.org/10.1017/s144618110000300x.