Research in Ramsey Theory and Automatic Theorem Proving

Yuan Chang
Joint work with William J. Wesley and (faculty mentor) Prof. De Loera

October 12, 2021

Introduction - Overview

- Ramsey Theory is the mathematical study of combinatorial objects in which a certain degree of order must occur as the scale of the object becomes large.

Introduction - Overview

- Ramsey Theory is the mathematical study of combinatorial objects in which a certain degree of order must occur as the scale of the object becomes large.
- In particular, Rado's theorem.

Introduction - Overview

- Ramsey Theory is the mathematical study of combinatorial objects in which a certain degree of order must occur as the scale of the object becomes large.
- In particular, Rado's theorem.
- Solving the problem in a different way, and through computers.

Introduction - Overview

- Ramsey Theory is the mathematical study of combinatorial objects in which a certain degree of order must occur as the scale of the object becomes large.
- In particular, Rado's theorem.
- Solving the problem in a different way, and through computers.
- In this project, we are concerned with linear homogeneous equations with 3 variables. For example, $ax + by = cz$, where $a, b, c \in \mathbb{Z}$.

Introduction - Definitions with example

Let me introduce the idea of **coloring** and **monochromatic solution** to an linear equation D with an example:

Introduction - Definitions with example

Let me introduce the idea of **coloring** and **monochromatic solution** to an linear equation D with an example:

Example

We can define a $2-$colorings of the integer from 1 to 6 as by splitting them into 2 sets. For example, $1, 3, 5, 2, 4, 6$.

Introduction - Definitions with example

Let me introduce the idea of **coloring** and **monochromatic solution** to an linear equation D with an example:

Example

We can define a $2-$colorings of the integer from 1 to 6 as by splitting them into 2 sets. For example, $1, 3, 5, 2, 4, 6$.
If we are concerned about equation $x + y = z$ where solutions are within the bounds $[1, 6]$, then we can see that $2 + 4 = 6$ is a monochromatic solution.

Introduction - Definitions cont'd

Introduction - Definitions cont'd

Definition (Rado Number)

For any equation D, the Rado Number $R_r(D)$ is the smallest N such that any $r-$coloring $\chi : \{1, 2, \ldots, N\} \rightarrow \{1, 2, \ldots, r\}$ must induce a monochromatic solution to D.

Introduction - Definitions cont'd

Definition (Rado Number)

For any equation D, the Rado Number $R_r(D)$ is the smallest N such that any $r-$coloring $\chi : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, r\}$ must induce a monochromatic solution to D.

Example

The 2-color Rado Number for equation $x + y = z$ is 5. Since we can color the first 4 integers $1, 4, 2, 3$, but 5 would be an issue.

Introduction - Definitions cont'd

Definition (Rado Number)

For any equation D, the Rado Number $R_r(D)$ is the smallest N such that any $r-$coloring $\chi : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, r\}$ must induce a monochromatic solution to D.

Example

The 2-color Rado Number for equation $x + y = z$ is 5. Since we can color the first 4 integers $1, 4, 2, 3$, but 5 would be an issue.

Example

We can avoid monochromatic solutions to equation $x + y = z$ if we $3-$color $1 - 13$ in the following way:

$$\{2, 3, 7, 12\}\{5, 6, 8, 9\}\{1, 4, 10, 11, 13\}$$

Introduction - Definitions cont'd

Definition (Rado Number)

For any equation D, the Rado Number $R_r(D)$ is the smallest N such that any $r-$coloring $\chi : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, r\}$ must induce a monochromatic solution to D.

Example

The 2-color Rado Number for equation $x + y = z$ is 5. Since we can color the first 4 integers $1, 4, 2, 3$, but 5 would be an issue.

Example

We can avoid monochromatic solutions to equation $x + y = z$ if we $3-$color $1 - 13$ in the following way:

$$\{2, 3, 7, 12\}\{5, 6, 8, 9\}\{1, 4, 10, 11, 13\}$$

In fact, $3-$coloring Rado Number for $x + y = z$ is 14.

Our Project

- There has been work on Rado numbers using computational methods before.

Our Project

- There has been work on Rado numbers using computational methods before.
- We are extending the computations, with optimizations, and computing a lot of new Rado numbers that had not been known before.

Our Project

- There has been work on Rado numbers using computational methods before.
- We are extending the computations, with optimizations, and computing a lot of new Rado numbers that had not been known before.
- Let computers do the heavy lifting for us.

Computations

- We are solving the Rado Number problem with Boolean algebra.

Computations

- We are solving the Rado Number problem with Boolean algebra.
- In particularly, we are translating the Rado Number problems into propositional logic satisfiability problem(SAT). Then, use SAT-solvers to solve them.

Computations

- We are solving the Rado Number problem with Boolean algebra.
- In particularly, we are translating the Rado Number problems into propositional logic satisfiability problem(SAT). Then, use SAT-solvers to solve them.
- The high-level idea of encoding the problem is very similar of M.Heule's Schur Number Five Paper.

Computations

- We are solving the Rado Number problem with Boolean algebra.
- In particularly, we are translating the Rado Number problems into propositional logic satisfiability problem(SAT). Then, use SAT-solvers to solve them.
- The high-level idea of encoding the problem is very similar of M.Heule's Schur Number Five Paper.
- Let me introduce some basic Boolean algebra terminology.

Encoding notations

- Let $R_n^r(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.

Encoding notations

- Let $R_n^r(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.
- If $R_n^r(D)$ is satisfiable, then we can avoid monochromatic solutions with this coloring and $R_r(D) > n$. If it's unsatisfiable, then $R_r(D) \leq n$.

Encoding notations

- Let $R^r_n(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.
- If $R^r_n(D)$ is satisfiable, then we can avoid monochromatic solutions with this coloring and $R_r(D) > n$. If it's unsatisfiable, then $R_r(D) \leq n$.
- For encoding of $R^r_n(D)$, we use Boolean variable v^i_j, which indicates that number j has color i.

Encoding notations

- Let $R_n^r(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.
- If $R_n^r(D)$ is satisfiable, then we can avoid monochromatic solutions with this coloring and $R_r(D) > n$. If it's unsatisfiable, then $R_r(D) \leq n$.
- For encoding of $R_n^r(D)$, we use Boolean variable v_j^i, which indicates that number j has color i.
- The encoding is split into three sections, positive, negative and optional clauses.

Encoding notations

- Let $R_n^r(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.
- If $R_n^r(D)$ is satisfiable, then we can avoid monochromatic solutions with this coloring and $R_r(D) > n$. If it's unsatisfiable, then $R_r(D) \leq n$.
- For encoding of $R_n^r(D)$, we use Boolean variable v_j^i, which indicates that number j has color i.
- The encoding is split into three sections, positive, negative and optional clauses.
- The positive clauses encode that every number j must have at least one color.

Encoding notations

- Let $R_n^r(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.
- If $R_n^r(D)$ is satisfiable, then we can avoid monochromatic solutions with this coloring and $R_r(D) > n$. If it's unsatisfiable, then $R_r(D) \leq n$.
- For encoding of $R_n^r(D)$, we use Boolean variable v_j^i, which indicates that number j has color i.
- The encoding is split into three sections, positive, negative and optional clauses.
- The positive clauses encode that every number j must have at least one color.
- The negative clauses encode that every solution to the equation D, the variables in D cannot have the same color.

Encoding notations

- Let $R_n^r(D)$ denote the encoding of a $r-$coloring of the integers from 1 to n which avoids monochromatic solutions to the equation D.
- If $R_n^r(D)$ is satisfiable, then we can avoid monochromatic solutions with this coloring and $R_r(D) > n$. If it's unsatisfiable, then $R_r(D) \leq n$.
- For encoding of $R_n^r(D)$, we use Boolean variable v_j^i, which indicates that number j has color i.
- The encoding is split into three sections, positive, negative and optional clauses.
- The positive clauses encode that every number j must have at least one color.
- The negative clauses encode that every solution to the equation D, the variables in D cannot have the same color.
- The optional clauses encode that every number has at most one color.

Encoding Example

- Let D be $x + y = z$, we want to encode $R_4^3(D)$.

Encoding Example

- Let D be $x + y = z$, we want to encode $R_4^3(D)$.
- Positive clauses:
  $(v_1^1 \vee v_1^2 \vee v_1^3) \wedge (v_2^1 \vee v_2^2 \vee v_2^3) \wedge (v_3^1 \vee v_3^2 \vee v_3^3) \wedge (v_4^1 \vee v_4^2 \vee v_4^3)$

Encoding Example

- Let D be $x + y = z$, we want to encode $R_4^3(D)$.
- Positive clauses:
  $(v_1^1 \vee v_1^2 \vee v_1^3) \wedge (v_2^1 \vee v_2^2 \vee v_2^3) \wedge (v_3^1 \vee v_3^2 \vee v_3^3) \wedge (v_4^1 \vee v_4^2 \vee v_4^3)$
- Negative clauses:
  $(\overline{v}_1^1 \vee \overline{v}_1^1 \vee \overline{v}_2^1) \wedge (\overline{v}_2^1 \vee \overline{v}_1^1 \vee \overline{v}_3^1) \wedge (\overline{v}_3^1 \vee \overline{v}_1^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_2^1 \vee \overline{v}_3^1) \wedge (\overline{v}_2^1 \vee \overline{v}_2^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_3^1 \vee \overline{v}_4^1) \wedge$

Encoding Example

- Let D be $x + y = z$, we want to encode $R_4^3(D)$.
- Positive clauses:
  $(v_1^1 \vee v_1^2 \vee v_1^3) \wedge (v_2^1 \vee v_2^2 \vee v_2^3) \wedge (v_3^1 \vee v_3^2 \vee v_3^3) \wedge (v_4^1 \vee v_4^2 \vee v_4^3)$
- Negative clauses:
  $(\overline{v}_1^1 \vee \overline{v}_1^1 \vee \overline{v}_2^1) \wedge (\overline{v}_2^1 \vee \overline{v}_1^1 \vee \overline{v}_3^1) \wedge (\overline{v}_3^1 \vee \overline{v}_1^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_2^1 \vee \overline{v}_3^1) \wedge (\overline{v}_2^1 \vee \overline{v}_2^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_3^1 \vee \overline{v}_4^1) \wedge$
  $(\overline{v}_1^2 \vee \overline{v}_1^2 \vee \overline{v}_2^2) \wedge (\overline{v}_2^2 \vee \overline{v}_1^2 \vee \overline{v}_3^2) \wedge (\overline{v}_3^2 \vee \overline{v}_1^2 \vee \overline{v}_4^2) \wedge (\overline{v}_1^2 \vee \overline{v}_2^2 \vee \overline{v}_3^2) \wedge (\overline{v}_2^2 \vee \overline{v}_2^2 \vee \overline{v}_4^2) \wedge (\overline{v}_1^2 \vee \overline{v}_3^2 \vee \overline{v}_4^2) \wedge$

Encoding Example

- Let D be $x + y = z$, we want to encode $R_4^3(D)$.
- Positive clauses:
$(v_1^1 \vee v_1^2 \vee v_1^3) \wedge (v_2^1 \vee v_2^2 \vee v_2^3) \wedge (v_3^1 \vee v_3^2 \vee v_3^3) \wedge (v_4^1 \vee v_4^2 \vee v_4^3)$
- Negative clauses:
$(\overline{v}_1^1 \vee \overline{v}_1^1 \vee \overline{v}_2^1) \wedge (\overline{v}_2^1 \vee \overline{v}_1^1 \vee \overline{v}_3^1) \wedge (\overline{v}_3^1 \vee \overline{v}_1^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_2^1 \vee \overline{v}_3^1) \wedge (\overline{v}_2^1 \vee \overline{v}_2^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_3^1 \vee \overline{v}_4^1) \wedge$
$(\overline{v}_1^2 \vee \overline{v}_1^2 \vee \overline{v}_2^2) \wedge (\overline{v}_2^2 \vee \overline{v}_1^2 \vee \overline{v}_3^2) \wedge (\overline{v}_3^2 \vee \overline{v}_1^2 \vee \overline{v}_4^2) \wedge (\overline{v}_1^2 \vee \overline{v}_2^2 \vee \overline{v}_3^2) \wedge (\overline{v}_2^2 \vee \overline{v}_2^2 \vee \overline{v}_4^2) \wedge (\overline{v}_1^2 \vee \overline{v}_3^2 \vee \overline{v}_4^2) \wedge$
$(\overline{v}_1^3 \vee \overline{v}_1^3 \vee \overline{v}_2^3) \wedge (\overline{v}_2^3 \vee \overline{v}_1^3 \vee \overline{v}_3^3) \wedge (\overline{v}_3^3 \vee \overline{v}_1^3 \vee \overline{v}_4^3) \wedge (\overline{v}_1^3 \vee \overline{v}_2^3 \vee \overline{v}_3^3) \wedge (\overline{v}_2^3 \vee \overline{v}_2^3 \vee \overline{v}_4^3) \wedge (\overline{v}_1^3 \vee \overline{v}_3^3 \vee \overline{v}_4^3)$

Encoding Example

- Let D be $x + y = z$, we want to encode $R_4^3(D)$.
- Positive clauses:
  $(v_1^1 \vee v_1^2 \vee v_1^3) \wedge (v_2^1 \vee v_2^2 \vee v_2^3) \wedge (v_3^1 \vee v_3^2 \vee v_3^3) \wedge (v_4^1 \vee v_4^2 \vee v_4^3)$
- Negative clauses:
  $(\overline{v}_1^1 \vee \overline{v}_1^1 \vee \overline{v}_2^1) \wedge (\overline{v}_2^1 \vee \overline{v}_1^1 \vee \overline{v}_3^1) \wedge (\overline{v}_3^1 \vee \overline{v}_1^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_2^1 \vee \overline{v}_3^1) \wedge (\overline{v}_2^1 \vee \overline{v}_2^1 \vee \overline{v}_4^1) \wedge (\overline{v}_1^1 \vee \overline{v}_3^1 \vee \overline{v}_4^1) \wedge$
  $(\overline{v}_1^2 \vee \overline{v}_1^2 \vee \overline{v}_2^2) \wedge (\overline{v}_2^2 \vee \overline{v}_1^2 \vee \overline{v}_3^2) \wedge (\overline{v}_3^2 \vee \overline{v}_1^2 \vee \overline{v}_4^2) \wedge (\overline{v}_1^2 \vee \overline{v}_2^2 \vee \overline{v}_3^2) \wedge (\overline{v}_2^2 \vee \overline{v}_2^2 \vee \overline{v}_4^2) \wedge (\overline{v}_1^2 \vee \overline{v}_3^2 \vee \overline{v}_4^2) \wedge$
  $(\overline{v}_1^3 \vee \overline{v}_1^3 \vee \overline{v}_2^3) \wedge (\overline{v}_2^3 \vee \overline{v}_1^3 \vee \overline{v}_3^3) \wedge (\overline{v}_3^3 \vee \overline{v}_1^3 \vee \overline{v}_4^3) \wedge (\overline{v}_1^3 \vee \overline{v}_2^3 \vee \overline{v}_3^3) \wedge (\overline{v}_2^3 \vee \overline{v}_2^3 \vee \overline{v}_4^3) \wedge (\overline{v}_1^3 \vee \overline{v}_3^3 \vee \overline{v}_4^3)$
- Optional clauses:
  $(\overline{v}_1^1 \vee \overline{v}_1^2) \wedge (\overline{v}_1^1 \vee \overline{v}_1^3) \wedge (\overline{v}_1^2 \vee \overline{v}_1^3) \wedge (\overline{v}_2^1 \vee \overline{v}_2^2) \wedge (\overline{v}_2^1 \vee \overline{v}_2^3) \wedge (\overline{v}_2^2 \vee \overline{v}_2^3) \wedge$
  $(\overline{v}_3^1 \vee \overline{v}_3^2) \wedge (\overline{v}_3^1 \vee \overline{v}_3^3) \wedge (\overline{v}_3^2 \vee \overline{v}_3^3) \wedge (\overline{v}_4^1 \vee \overline{v}_4^2) \wedge (\overline{v}_4^1 \vee \overline{v}_4^3) \wedge (\overline{v}_4^2 \vee \overline{v}_4^3)$

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

There are methods which we can speed up each of the 3 parts.

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

There are methods which we can speed up each of the 3 parts.

- Faster software implementation (Maple $\rightarrow$ C $\rightarrow$ Python)

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

There are methods which we can speed up each of the 3 parts.

- Faster software implementation (Maple $\rightarrow$ C $\rightarrow$ Python)
- Smarter solution generation (Hermite Normal Form).

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

There are methods which we can speed up each of the 3 parts.

- Faster software implementation (Maple $\rightarrow$ C $\rightarrow$ Python)
- Smarter solution generation (Hermite Normal Form).
- Hardware optimization (loop unrolling).

Optimizations

When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

There are methods which we can speed up each of the 3 parts.

- Faster software implementation (Maple $\rightarrow$ C $\rightarrow$ Python)
- Smarter solution generation (Hermite Normal Form).
- Hardware optimization (loop unrolling).
- Symmetry breaking (Asymmetric branching).

Optimizations

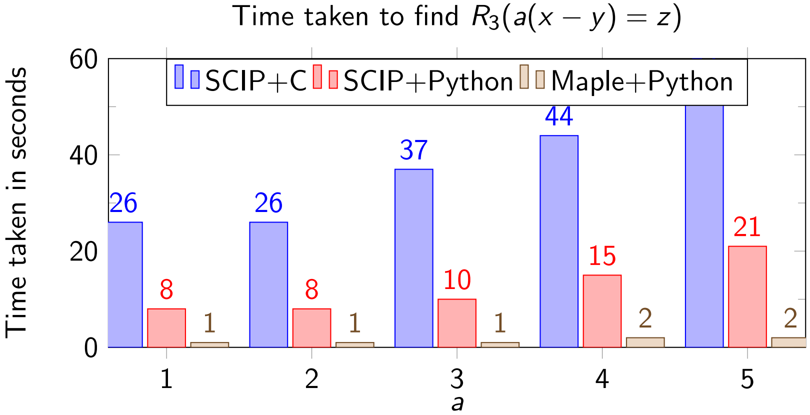
When solving each instance, there are 3 main parts within the algorithm that take up the majority of the time.

- SAT generation(Generate solutions)
- Writing clauses to file
- Solve the SAT problem.

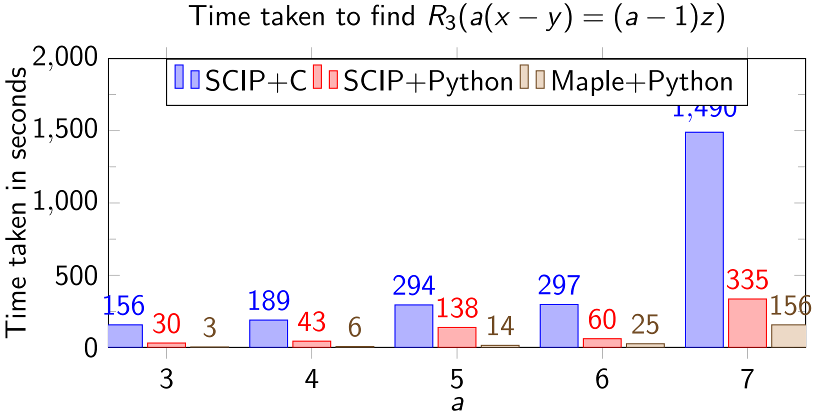
There are methods which we can speed up each of the 3 parts.

- Faster software implementation (Maple $\rightarrow$ C $\rightarrow$ Python)
- Smarter solution generation (Hermite Normal Form).
- Hardware optimization (loop unrolling).
- Symmetry breaking (Asymmetric branching).
- Multi-threaded SAT solvers (Glucose).

Speedup results



Speedup results



Future work

- Probabilistic solution generation.

Future work

- Probabilistic solution generation.
- Smarter and heavier symmetry breaking technique.

Future work

- Probabilistic solution generation.
- Smarter and heavier symmetry breaking technique.
- Pipeline style SAT solving, avoid secondary storage.

Future work

- Probabilistic solution generation.
- Smarter and heavier symmetry breaking technique.
- Pipeline style SAT solving, avoid secondary storage.
- Implementation in a logical programming language such as Prolog.

Future work

- Probabilistic solution generation.
- Smarter and heavier symmetry breaking technique.
- Pipeline style SAT solving, avoid secondary storage.
- Implementation in a logical programming language such as Prolog.
- $4-$colorings.

Acknowledgment

I would like to thank again:

- William Wesley, who guided me throughout the project, helped me with all the necessary background knowledge.

Acknowledgment

I would like to thank again:

- William Wesley, who guided me throughout the project, helped me with all the necessary background knowledge.
- Professor Jesús De Loera, whose expertise helped push the project forward, also the countless insightful comments throughout the summer.

Acknowledgment

I would like to thank again:

- William Wesley, who guided me throughout the project, helped me with all the necessary background knowledge.
- Professor Jesús De Loera, whose expertise helped push the project forward, also the countless insightful comments throughout the summer.
- I am grateful for the financial support provided NSF grant 1818969 to Prof. Jesus De Loera. The National Science Foundation's Summer Scholars Internship Program provided me with this opportunity and funding for this project. Thanks to Professor Jesús De Loera for nominating me to this summer program.

Thank you!

Thank you very much everyone for your time!