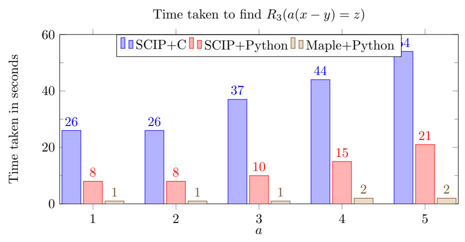
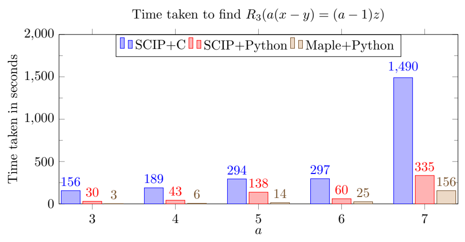
1 Experiment Data

In the following tables of data, the language combination in the format of `Solution generation` + `CNF generation`.
In the beginning of this project, `SCIP` was used to compute all the integer solutions for equation $\mathcal{E}$ given a bound of $[1, n]$. `C` was used to implement the logic for the positive and optional clauses, as well as generating the CNF files.
For example, `SCIP+C` suggests that the input CNF file is generated with code that is a combination of `SCIP` and C, where `SCIP` was responsible to generate all possible solutions for the negative clauses and `C` was responsible for writing clauses to file.
The SAT solver used in all experiment is `SATCH`. The main hope of the experimental data was to show off the speedup with the change of the choice of language used and the change of methodology in terms of solution generation.

1.1 Graphs





The combination where the entire algorithm is written in `Maple` is not part of the section because it is only viable for small instances of Rado Numbers. It would take too long to run the experiment and it will make the graph out of proportion. Thus, we simply leave the early implementation of the algorithm in `Maple` purely conceptual.
In order to make sure the experiments were fair, the upper and lower bounds for each Rado Number calculation were kept the same to ensure each version of the algorithm is searching the same number of times. Refer to the tables below for the bounds.

Bounds for $R_3(a(x-y)=bz)$

b \ a	1	2	3	4	5
1	$[10, 20]$	$[10, 20]$	$[20, 40]$	$[30, 70]$	$[50, 150]$

Bounds for $R_3(a(x+y)=(a-1)z)$

a	3	4	5	6	7
1	$[150, 350]$	$[200, 400]$	$[600, 1000]$	$[600, 1000]$	$[2000, 4000]$