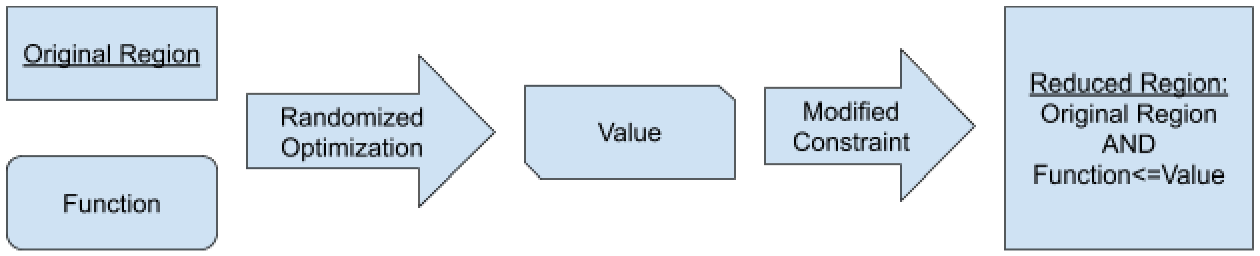
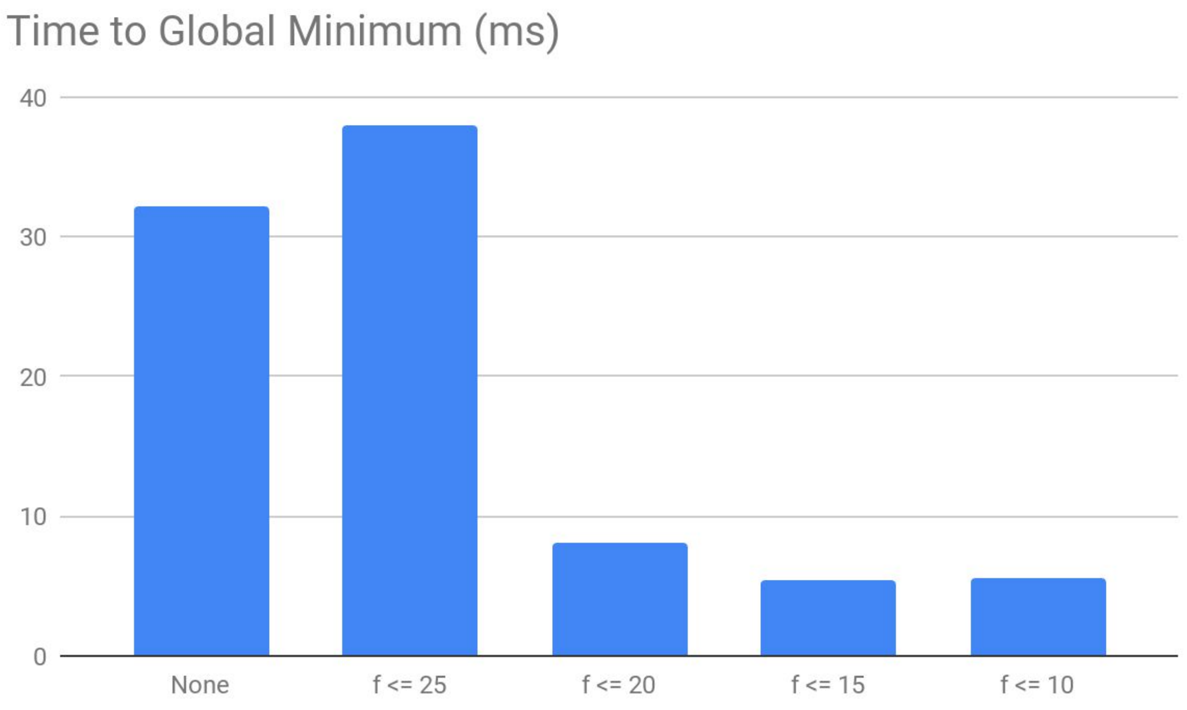
Project Overview

This project will test randomized optimization techniques as a way to reduce the initial region size for verified global optimization problems. By running the below step on the original region, we are able to further constrain the region before minimizing. We will compare time to optimal solution between the case when the optimization is run starting with the original box (direct minimization) and with the reduced box. In addition, we compare the time to verified global minimum between direct minimization and random optimization techniques followed up by proof of global minimum.



By simply appending the constraint $\{x \mid f(x) \leq Value\}$ to the region constraint, we propagate the previously learned values onto the region, essentially achieving the same result as running the randomized optimization before the counterexample guided pruning and minimizing from there (Estimate-Assisted Minimize). Below are some preliminary results from using the *Levi N. 13 Function* (Max around 375, Min of 0). **The values for f< are chosen arbitrarily.**



Here, we note that we must be careful to achieve good random optimizations. Looking at $f \leq 25$, we see that it took longer to compute than the baseline case of having no constraint at all. We suspect that this is because the extra clause takes time to compute. If the optimized value is not good enough, then the cost of computing the extra constraint outweighs the benefit of having the initial value optimized.

However, in some cases, getting too close to optimum might be a problem as well. For the *Holder Table 2D* function, simulated annealing actually achieves the global minimum. We see that getting closer than $\delta = 0.001$ to the global minimum results in a significant drop in performance, taking as much as 20 times the original, or 111 seconds. However, we see good performance (about 1.5x original speed) when we add δ to the estimate, loosening the bound. We suspect that this is because making the bound very tight makes the remaining region very hard to compute. Thus, we add δ to the estimate bound regardless; no significant decrease in speed is observed for the other functions after adding δ.

For *Trefethen* and *Mishra Bird*, we were unable to see positive results using the estimated bound. Instead, we see a decline in speed of about 10 to 40%. We suspect that these functions are harder to constrain by value when using the numerical libraries. On the other hand, we note that simply running optimization techniques (in this case, Simulated Annealing) until we reach within delta of global minimum, then proving it is the delta-global minimum achieves sufficient speedup (Random Search & Prove).

Direct Minimize vs Estimate-Assisted Minimize

Function name	Direct Minimize (ms)	Assisted Minimize (ms)	Speedup
Levi N.13	31.0563	6.38843	4.86134
Booth	55.0859	4.01405	13.7233
Holder Table 2D	6164.08	4224.89	1.45899
Trefethen	601.386	634.232	0.948213 (worse)
Mishra Bird	116.483	182.512	0.638334 (worse)

Direct Minimize vs Random Search + Proof

Function name	Direct Minimize (ms)	Random + Proof (ms)	Speedup
Trefethen	666.883	139.889	4.7669
Mishra Bird	157.117	26.1637	6.00515
Holder Table 2D	6558.41	369.725	17.386

Conclusion:

1. In some functions, global minimum estimates from random optimization techniques significantly speed up formal techniques to finding the minimum.
2. For others, *Trefethen* and *Mishra Bird*, the task of computing the region is heavier than the gain from the improved bound.
3. For low-dimensional function optimization, finding the optimum point and proving it to be the optimum ($\neg\exists$) is significantly faster than relying on an $\exists\forall$-solver to “synthesize” the optimum point.