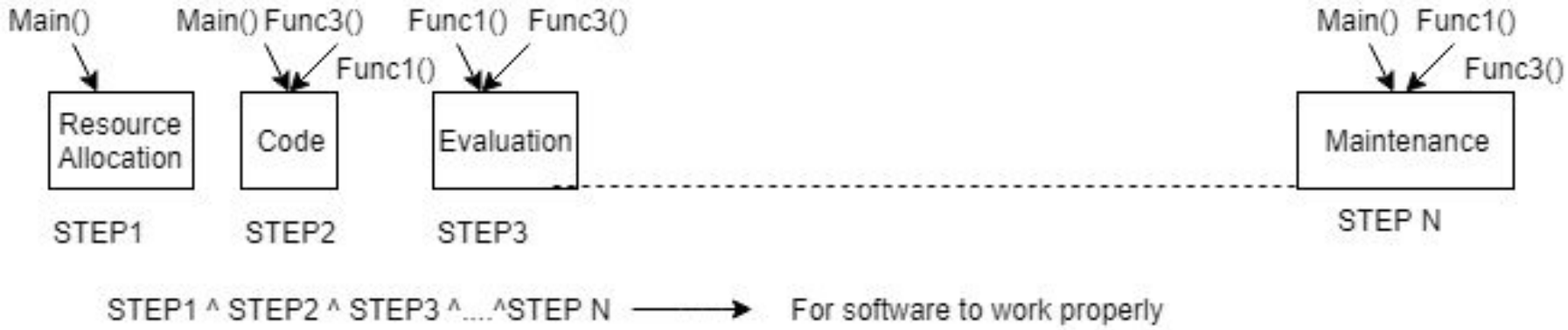
Hardware Acceleration of Minisat

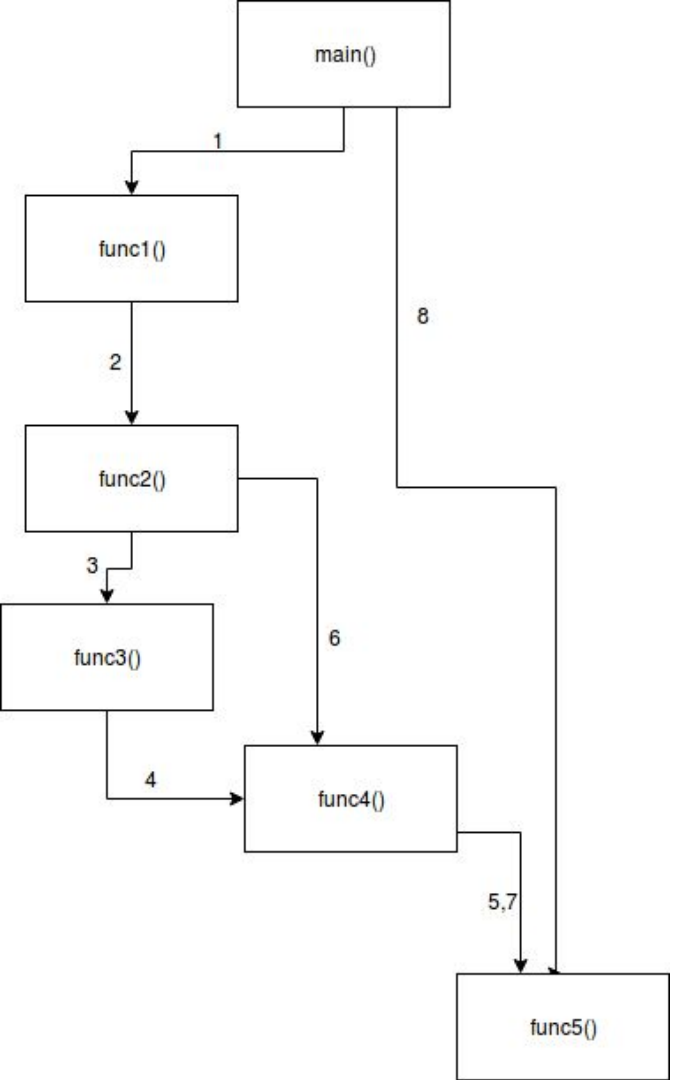
Shubhendra Pal Singhal

Applications of SAT Solver

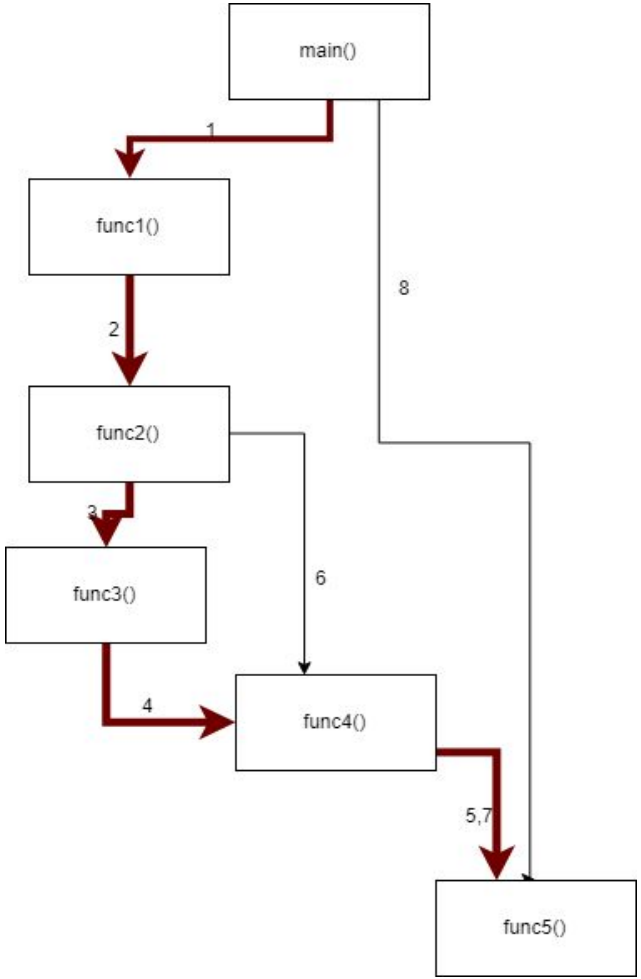


Insight of solution

There is a possibility of acceleration of SAT Solvers by a (hardware) chip. Instead of designing a hardware chip and testing it whether it really works, what we did was to emulate the results of hardware and feed these inputs to SAT solver and analyse the reduction in total execution time. This was achieved by using open source software **GPROF.** It analyses the flow of program and execution time at every call or instruction of a program.


main()
1
func1()
2
func2()
3
func3()
4
func4()
5,7
func5()
8
6

ARC


main()
1
func1()
2
func2()
func3()
4
func4()
5,7
func5()
8
6

Solution

Address, time etc. are all machine dependent, but the call graph is not. So we assign the index to every call and then change the time for those calls(corresponding to the block we need to change). Index is assigned to every call and the respective id can be printed so that the user can identify the call by noticing its parent and respective child.

Algorithm for changing execution time of a certain part of code

Assume min to be lower limit and max as upper limit of block that needs to be replaced. c be the changed value. In case of different values for every id, it needs to be specified in the form of an array of values.

id = 0, total_bin = 0, count = 0

If { id>=min-1 and id <= max-1 }{

total_bin += time

Time = c

}

count = count+1

```
int count = 0;
int count_arg = 0;
int tot_bin = 0;
static void
hist_assign_samples_1 (histogram *r)
{
  bfd_vma bin_low_pc, bin_high_pc;
  bfd_vma sym_low_pc, sym_high_pc;
  bfd_vma overlap, addr;
  unsigned int bin_count;
  unsigned int i, j, k;
  double count_time, credit;

  bfd_vma lowpc = r->lowpc / sizeof (UNIT);

  /* Iterate over all sample bins.  */
  for (i = 0, k = 1; i < r->num_bins; ++i)
    {
      bin_count = r->sample[i];
      if (! bin_count)
        continue;
      count_arg++;
      DBG(SAMPLEDEBUG,printf ("----------------------------  "));
      DBG(SAMPLEDEBUG,printf ("\n\nId is : %d\n",count_arg));
      bin_low_pc = lowpc + (bfd_vma) (hist_scale * i);
      bin_high_pc = lowpc + (bfd_vma) (hist_scale * (i + 1));
      if(count>=132 && count <=250) {
        tot_bin+=bin_count;
        DBG(SAMPLEDEBUG,printf ("Inside bin_count change"));
        bin_count = 1;
      }
      count++;
      count_time = bin_count;
      DBG (SAMPLEDEBUG,
     printf ("\n--->%s--->\n",symtab.base[k].name));

      DBG (SAMPLEDEBUG,
           printf (
      "[assign_samples] bin_low_pc=0x%lx, bin_high_pc=0x%lx, bin_count=%u\n",
                    (unsigned long) (sizeof (UNIT) * bin_low_pc),
                    (unsigned long) (sizeof (UNIT) * bin_high_pc),
                    bin_count));
      total_time += count_time;
```

```
int count = 0;
int count_arg = 0;
int tot_bin = 0;
static void
hist_assign_samples_1 (histogram *r)
{
  bfd_vma bin_low_pc, bin_high_pc;
  bfd_vma sym_low_pc, sym_high_pc;
  bfd_vma overlap, addr;
  unsigned int bin_count;
  unsigned int i, j, k;
  double count_time, credit;

  bfd_vma lowpc = r->lowpc / sizeof (UNIT);

  /* Iterate over all sample bins.  */
  for (i = 0, k = 1; i < r->num_bins; ++i)
    {
      bin_count = r->sample[i];
      if (! bin_count)
        continue;
      count_arg++;
      DBG(SAMPLEDEBUG,printf ("----------------------------  "));
      DBG(SAMPLEDEBUG,printf ("\n\nId is : %d\n",count_arg));
      bin_low_pc = lowpc + (bfd_vma) (hist_scale * i);
      bin_high_pc = lowpc + (bfd_vma) (hist_scale * (i + 1));
      if(count>=132 && count <=250) {
        tot_bin+=bin_count;
        DBG(SAMPLEDEBUG,printf ("Inside bin_count change"));
        bin_count = 1;
      }
      count++;
      count_time = bin_count;
      DBG (SAMPLEDEBUG,
     printf ("\n--->%s--->\n",symtab.base[k].name));

      DBG (SAMPLEDEBUG,
           printf (
      "[assign_samples] bin_low_pc=0x%lx, bin_high_pc=0x%lx, bin_count=%u\n",
                    (unsigned long) (sizeof (UNIT) * bin_low_pc),
                    (unsigned long) (sizeof (UNIT) * bin_high_pc),
                    bin_count));
      total_time += count_time;
```

Address block

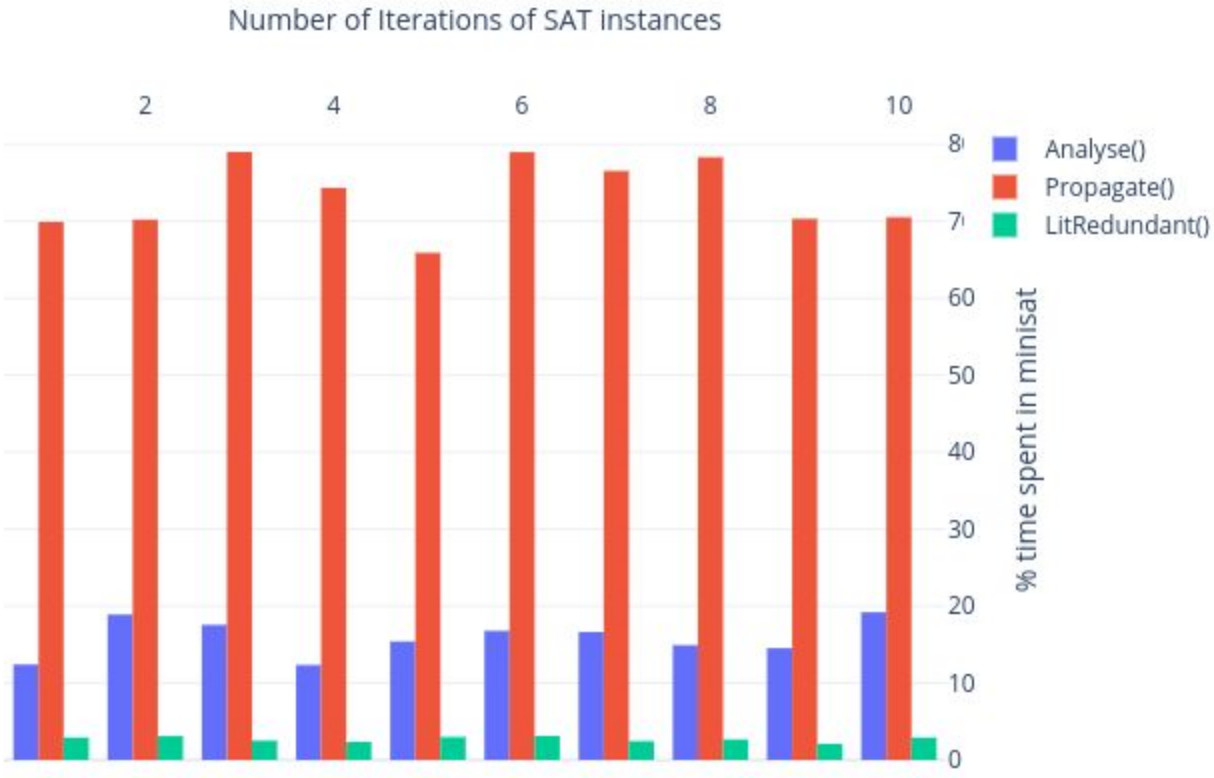
```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 76.57      8.82     8.82  2175936     0.00     0.00  Minisat::Solver::propagate(){44}  100%
 11.55     10.15     1.33  1017042     0.00     0.00  Minisat::Solver::analyze(unsigned int, Minisat::vec<Minisat::Lit, int>&, int&
  2.69     10.46     0.31  6832934     0.00     0.00  Minisat::Solver::litRedundant(Minisat::Lit){40}  100%
  2.52     10.75     0.29     1186     0.00     0.00  void Minisat::sort<unsigned int, reduceDB_lt>(unsigned int*, int, reduceDB_lt
  2.17     11.00     0.25  1019087     0.00     0.00  Minisat::Solver::~Solver(){31}  100%
  1.91     11.22     0.22     1192     0.00     0.00  Minisat::Solver::relocAll(Minisat::ClauseAllocator&){49}  100%
  1.13     11.35     0.13  1153853     0.00     0.00  Minisat::Solver::pickBranchLit(){39}  100%
  0.69     11.43     0.08     2046     0.00     0.01  Minisat::Solver::search(int){63}  100%
  0.35     11.47     0.04     1186     0.00     0.00  Minisat::Solver::reduceDB(){62}  100%
  0.17     11.49     0.02  1016029     0.00     0.00  Minisat::Solver::removeClause(unsigned int){36}  100%
  0.00     11.49     0.00  1017967     0.00     0.00  Minisat::Solver::attachClause(unsigned int){34}  100%
  0.00     11.49     0.00     2059     0.00     0.00  Minisat::Solver::simplify(){51}  100%
  0.00     11.49     0.00     1192     0.00     0.00  Minisat::SimpSolver::garbageCollect(){50}  100%
  0.00     11.49     0.00      936     0.00     0.00  Minisat::Solver::addClause_(Minisat::vec<Minisat::Lit, int>&){45}  100%
  0.00     11.49     0.00      240     0.00     0.00  Minisat::vec<Minisat::Option*, int>::push(Minisat::Option* const&){29}  100%
  0.00     11.49     0.00       60     0.00     0.00  Minisat::SimpSolver::eliminateVar(int){84}  100%
  0.00     11.49     0.00       60     0.00     0.00  Minisat::Solver::newVar(Minisat::lbool, bool){33}  100%
  0.00     11.49     0.00       11     0.00     0.00  Minisat::Solver::removeSatisfied(Minisat::vec<unsigned int, int>&){47}  100%
  0.00     11.49     0.00       10     0.00     0.00  Minisat::IntOption::parse(char const*){23}  100%
  0.00     11.49     0.00        8     0.00     0.00  Minisat::BoolOption::parse(char const*){24}  100%
  0.00     11.49     0.00        7     0.00     0.00  Minisat::DoubleOption::parse(char const*){67}  100%
  0.00     11.49     0.00        7     0.00     0.00  Minisat::Solver::rebuildOrderHeap(){48}  100%
  0.00     11.49     0.00        2     0.00     0.00  Minisat::SimpSolver::strengthenClause(unsigned int, Minisat::Lit){75}  100%
  0.00     11.49     0.00        2     0.00     0.00  Minisat::Solver::detachClause(unsigned int, bool){35}  100%
  0.00     11.49     0.00        1     0.00     0.00  _GLOBAL__sub_I__ZN7Minisat10SimpSolverC2Ev{5}  100%
  0.00     11.49     0.00        1     0.00     0.00  _GLOBAL__sub_I__ZN7Minisat6SolverC2Ev{4}  100%
  0.00     11.49     0.00        1     0.00     0.00  _GLOBAL__sub_I_elapsed_t0{2}  100%
  0.00     11.49     0.00        1     0.00     0.00  memReadPeak(){57}  100%
  0.00     11.49     0.00        1     0.00     0.00  mkElimClause(Minisat::vec<unsigned int, int>&, int, Minisat::Clause&){71}  10
  0.00     11.49     0.00        1     0.00     0.00  Minisat::SimpSolver::removeClause(unsigned int){74}  100%
  0.00     11.49     0.00        1     0.00     0.00  Minisat::SimpSolver::backwardSubsumptionCheck(bool){81}  100%
  0.00     11.49     0.00        1     0.00     0.00  Minisat::SimpSolver::~SimpSolver(){70}  100%
  0.00     11.49     0.00        1     0.00     0.00  Minisat::memUsedPeak(bool){58}  100%
  0.00     11.49     0.00        1     0.00     0.00  Minisat::StringOption::parse(char const*){15}  100%
  0.00     11.49     0.00        1     0.00     0.00  Minisat::RegionAllocator<unsigned int>::alloc(int){89}  100%
  0.00     11.49     0.00        1     0.00    11.49  Minisat::Solver::solve_(){64}  100%
  0.00     11.49     0.00        1     0.00     0.00  Minisat::Solver::Solver(){32}  100%
Tot is: 11.491194
Total time in bin is : 0
```

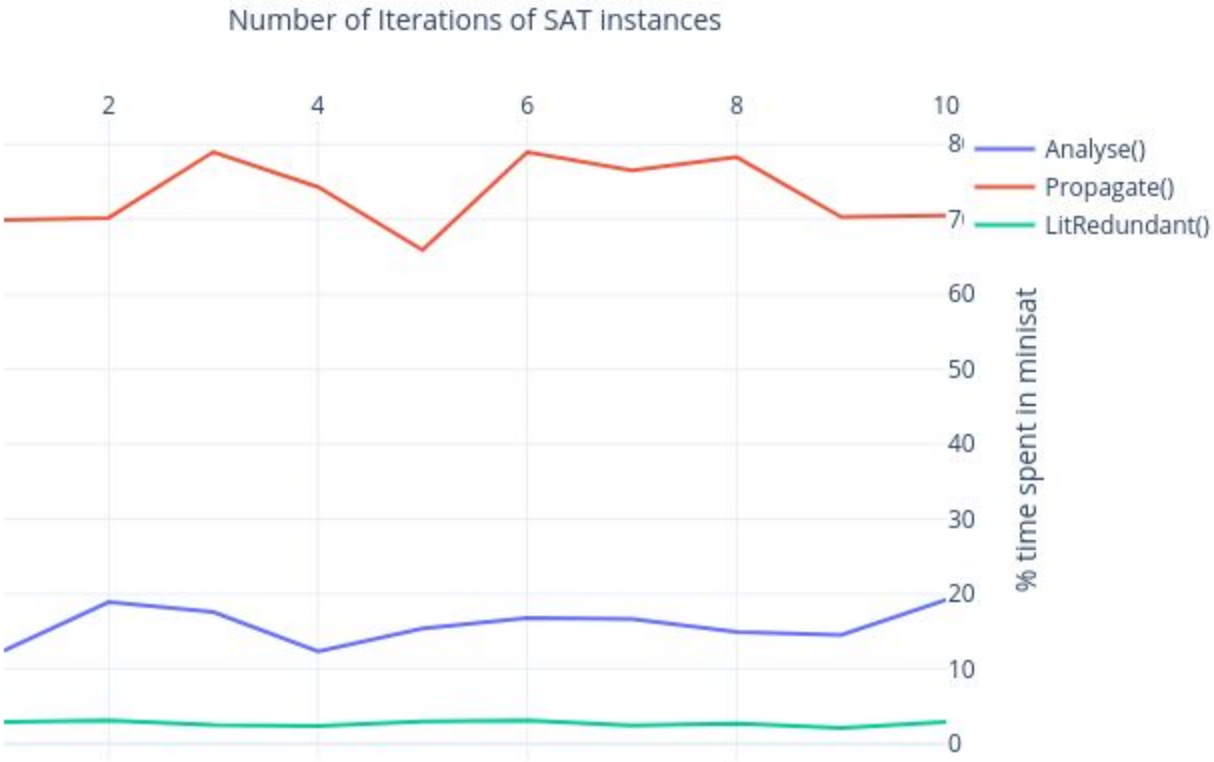
In-depth Analysis

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 76.57      8.82     8.82  2175936     0.00     0.00  Minisat::Solver::propagate(){44}  100%
 11.55     10.15     1.33  1017042     0.00     0.00  Minisat::Solver::analyze(unsigned int, Minisat::vec<Minisat::Lit, int>&, int
  2.69     10.46     0.31  6832934     0.00     0.00  Minisat::Solver::litRedundant(Minisat::Lit){40}  100%
  2.52     10.75     0.29     1186     0.00     0.00  void Minisat::sort<unsigned int, reduceDB_lt>(unsigned int*, int, reduceDB_
  2.17     11.00     0.25  1019087     0.00     0.00  Minisat::Solver::~Solver(){31}  100%
  1.91     11.22     0.22     1192     0.00     0.00  Minisat::Solver::relocAll(Minisat::ClauseAllocator&){49}  100%
  1.13     11.35     0.13  1153853     0.00     0.00  Minisat::Solver::pickBranchLit(){39}  100%
  0.69     11.43     0.08     2046     0.00     0.01  Minisat::Solver::search(int){63}  100%
  0.35     11.47     0.04     1186     0.00     0.00  Minisat::Solver::reduceDB(){62}  100%
  0.17     11.49     0.02  1016029     0.00     0.00  Minisat::Solver::removeClause(unsigned int){36}  100%
  0.00     11.49     0.00  1017967     0.00     0.00  Minisat::Solver::attachClause(unsigned int){34}  100%
```

After replacing propagate() by 1 in histogram from bin_count = 132-250

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 34.02      1.33     1.33  1017042     0.00     0.00  Minisat::Solver::analyze(unsigned int, Minisat::vec<Minisat::Lit, int>&, in
 31.59      2.57     1.24  2175936     0.00     0.00  Minisat::Solver::propagate(){44}  100%
  7.93      2.88     0.31  6832934     0.00     0.00  Minisat::Solver::litRedundant(Minisat::Lit){40}  100%
  7.42      3.17     0.29     1186     0.00     0.00  void Minisat::sort<unsigned int, reduceDB_lt>(unsigned int*, int, reduceDB_
  6.39      3.42     0.25  1019087     0.00     0.00  Minisat::Solver::~Solver(){31}  100%
  5.63      3.64     0.22     1192     0.00     0.00  Minisat::Solver::relocAll(Minisat::ClauseAllocator&){49}  100%
  3.33      3.77     0.13  1153853     0.00     0.00  Minisat::Solver::pickBranchLit(){39}  100%
  2.05      3.85     0.08     2046     0.00     0.00  Minisat::Solver::search(int){63}  100%
  1.02      3.89     0.04     1186     0.00     0.00  Minisat::Solver::reduceDB(){62}  100%
  0.51      3.91     0.02  1016029     0.00     0.00  Minisat::Solver::removeClause(unsigned int){36}  100%
  0.00      3.91     0.00  1017967     0.00     0.00  Minisat::Solver::attachClause(unsigned int){34}  100%
```


Number of Iterations of SAT instances
2
4
6
8
10
Analyse()
Propagate()
LitRedundant()
% time spent in minisat
60
50
40
30
20
10
0


Number of Iterations of SAT instances
2
4
6
8
10
80
70
60
50
40
30
20
10
0
% time spent in minisat
Analyse()
Propagate()
LitRedundant()