On estimating total time to solve SAT in distributed computing environments: Application to the SAT@home project

Oleg Zaikin and Alexander Semenov

Institute for System Dynamics and Control Theory SB RAS, Irkutsk, Russia

zaikin.icc@gmail.com, biclop.rambler@yandex.ru

Monte Carlo method to search for a partitioning with good properties

X – a set of Boolean variables of CNF.
$\tilde{X}, \tilde{X} \subset X$ – a set, that defines the partitioning.

The problem: how to know if this partitioning is good or bad?
The answer: estimate the time required to process the partitioning using a Monte Carlo method. $t_A(\tilde{X})$ – time required to process the partitioning, A – complete SAT solving algorithm (that eventually stops for any given input). $\xi_A(\tilde{X})$ – random variable with range $\{\xi_A(\tilde{X}, \alpha)\}_{\alpha \in \{0,1\}^{|\tilde{X}|}}$, $\xi_A(\tilde{X}, \alpha)$ – time of processing of CNF $C|_\alpha$, α is chosen from $\{0,1\}^{|\tilde{X}|}$ in accordance with a uniform distribution. Then the following equation holds

$$t_A(\tilde{X}) = 2^{|\tilde{X}|} \cdot E[\xi_A(\tilde{X})].$$

We estimate the value of $E[\xi_A(\tilde{X})]$ using the Monte Carlo method.
Main formula of the Monte Carlo method is:

$$Pr\left\{\left|\frac{1}{N} \cdot \sum_{j=1}^{N} \xi^j - E[\xi]\right| < \frac{\delta_\gamma \cdot \sigma}{\sqrt{N}}\right\} = \gamma, \qquad \gamma = \Phi(\delta_\gamma)$$

We calculate the value of predictive function, that gives an estimation of time required to process the partitioning.

$$F_{A,C}(\tilde{X}) = \frac{2^{|\tilde{X}|}}{N} \cdot \sum_{j=1}^{N} \xi^j, \qquad \xi^j = \xi_A\left(\alpha_1^j, \dots, \alpha_{|\tilde{X}|}^j\right).$$

The procedure of search for a partitioning with good time estimation

We represent an arbitrary $\tilde{X}$ by the following Boolean vector

$$\chi(\tilde{X}) = (\chi_1, \dots, \chi_{|X|}), \chi_i = \begin{cases} 1, x_i \in \tilde{X} \\ 0, x_i \notin \tilde{X} \end{cases}$$

For each vector $\chi(\tilde{X})$ we construct a random sample

$$\{(\alpha_1^1, \dots, \alpha_{|\tilde{X}|}^1), \dots, (\alpha_1^N, \dots, \alpha_{|\tilde{X}|}^N)\}.$$

Than we calculate the value of predictive function using this sample. After this we try to improve that value in the neighborhood of $\chi(\tilde{X})$.

It is important to note, that the calculation of predictive function in an arbitrary point of the search space is an expensive process, so it is desirable to perform it not more than once for any given point. To take this into account we developed a tabu search algorithm described below.

Computational implementation of TS-algorithm on a computing cluster

If in the process of calculation of the prediction function value for some point it is already greater than its best known value, then we can safely interrupt calculations for this point and proceed to the next point from the search space.

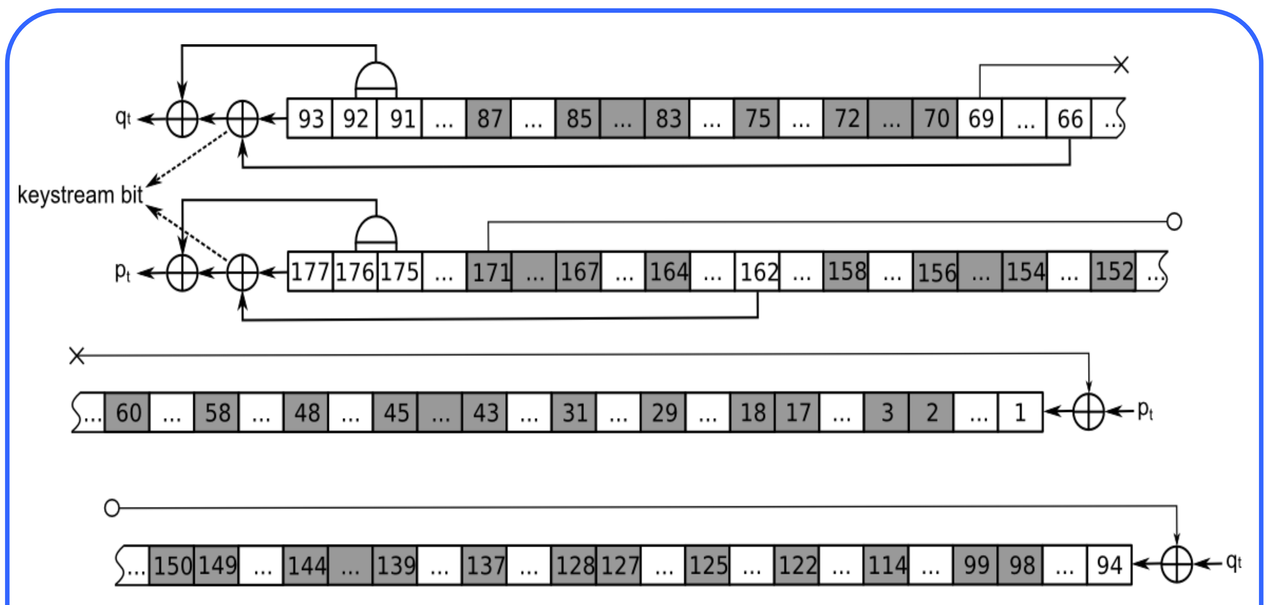
In practice this technique makes it possible to interrupt calculations in more than 90 % of cases.

$x_{current}$ — center of neighborhood that is being checked currently.
$N(\cdot)$ — neighborhood of point (according to Hamming distance).
L_1 — set of checked points with fully checked neighborhood.
L_2 — set of checked points with partly checked neighborhood.

Algorithm 1: Tabu search for the optimization of predictive function

Input: CNF C, start point x_{start}
Output: estimation F_{record} of time of solving SAT problem for C

$x_{record} \leftarrow x_{current} \leftarrow x_{start}$
$F_{record} \leftarrow F(x_{start})$
$L_1 \leftarrow \emptyset$
$L_2 \leftarrow \{x_{start}\}$
while *TRUE* **do**
  $UncheckedPoints \leftarrow$ unchecked points from $N(x_{current})$
  while $size(UncheckedPoints) > 0$ **do**
    $x \leftarrow$ randomly selected point from $UncheckedPoints$
    compute $F(x)$
    $L_2 \leftarrow L_2 \cup \{x\}$
    forall the $y \in L_2$ **do**
      if $x \in N(y)$ **then**
        mark x as checked in $N(y)$
    forall the $y \in L_2$ *with fully checked* $N(y)$ **do**
      $L_2 \leftarrow L_2 \setminus \{y\}$
      $L_1 \leftarrow L_1 \cup \{y\}$
    if $F(x) < F_{record}$ **then**
      $x_{record} \leftarrow x_{current} \leftarrow x$
      $F_{record} \leftarrow F(x)$
      break
    $UncheckedPoints \leftarrow UncheckedPoints \setminus \{x\}$
  if *timeExceeded() or* $L_2 = \emptyset$ **then**
    return F_{record}
  if $size(UncheckedPoints) = 0$ **then**
    $x_{current} \leftarrow$ point from L_2 with the largest heuristic value



Decomposition set of 47 variables for cryptanalysis of Bivium that was found by tabu search algorithm

Number of known bits (from 177) of initial state of Bivium	Time estimation for 120-core cluster	Avg. solving time (10 tests) on 120-core cluster	Time estimation for SAT@home
20 (weakened problem)	1 h. 5 min.	1 h.	Too simple problem
18 (weakened problem)	5 h. 34 min.	1 h. 50 min.	Too simple problem
10 (weakened problem)	244 days	-	41 days
0 (original problem)	35 years	-	6 years

SAT@home is a volunteer computing project aimed to solve hard problems (cryptanalysis, discrete optimization, etc.) that can be effectively reduced to SAT.
Join us at http://sat.isa.ru/pdsat/

State (July 5, 2013)

- **Application versions**: windows and linux, x86 and x64.
- **Active users**: 1044. **Active PCs**: 2234.
- **Performance**: 4.3 TFLOPs.

- Implemented using BOINC platform.
- Client application is based on slightly modified MiniSat.

Finished experiment
- 10 problems of cryptanalysis of the generator A5/1, that can't be solved using known Rainbow tables. Experiment took about 6 months.

Planned experiment
- Cryptanalysis of weakened Bivium (10 bits known).

POWERED BY BOINC

SAT@home

Proud participant of IDGF

International Desktop Grid Federation | Supported by EDGI & DEGISCO | http://desktopgridfederation.org