Factorization with SAT –
classical propositional calculus
as a programming environment

April 20, 2004

Mateusz Srebrny

<mati@students.mimuw.edu.pl>
Factorization with SAT

Plan

- SAT
- RSA
- goal – translation
- details of the translation
- experimental results
- conclusions, questions
Factorization with SAT

SAT

- satisfiability of classical propositional formulas
- Is a given formula satisfiable?
  - Find a satisfying valuation for a given formula.
  - Find all satisfying valuations.
- NP-complete (Cook 1971)
- exponential time required in general
Factorization with SAT

**SAT solvers**

- many SAT instances can be tested efficiently
- many competing algorithms/implementations
- Davies-Putnam Procedure, 1960
- CNF: conjunctive normal form, conjunction of disjunctions of literals
Factorization with SAT

Well-known SAT solvers

- zChaff (http://www.ee.princeton.edu/ chaff/zchaff.php)
- BerkMin (http://eigold.tripod.com/BerMin.html)
- Stalmarck’s PROVER
- GSAT, SATO, MAXSAT, WalkSat, GRASP
RSA (1977) – Factorization

- two factors: \( n = pq \), \( p, q \) prime
- historical anecdote
- \( p, q \) are cryptographically strong (detailed requirements)
  - \( p, q \) are odd
  - \( n \) large, \( p, q \) are of similar length
Complexity of factorization

- noone knows any polynomial time algorithm in general (asymptotically)
- all non-polynomial time algorithms are not feasible
- starting point for complexity theory – non-polynomial implies infeasible
- today – about $2^{60}$ instructions is an upper bound
RSA challenge

http://www.rsasecurity.com/rsalabs/challenge/factoring/numbers.html

- **512** – Factored, 1999
  
  7.5 month, about 300 computers (including one Cray)
  
  general number field sieve

- **576** – Factored, December 3, 2003
  
  \[18819881292060796383869723946165043980716356337941738270076335642298885971523466548531906060650474304531738801\]
  
  \[1303396716199692321205734031879550656996221305168759307650257059\]

- **640** – Not factored, reward $10K
  
  \[31074182404900437213507500358885679300373460228427545720161948823206440518081504556346829671723286782437916272\]
  
  \[838033415471073108501919548529007337724822783525742386454014691736602477652346609\]

- **1000-bit** is safe for now
Programming paradigm

• to solve a problem:
  1. translate the problem to SAT
  2. let SAT checker solve its satisfiability
• for a given \( n \),
generate such a propositional formula
  that its satisfying valuation encodes two integer factors \( p \) and \( q \) of \( n \)
Representation of integers

- $l$-bit integer $p \rightarrow l$ propositional variables $P_0, \ldots, P_{l-1}$

- e.g. $13 = (1101)_2$ is represented by formula $P = 13$:

$$\neg P_4 \land P_3 \land P_2 \land \neg P_1 \land P_0$$
$R = P$ represents $r = p$

$$\bigwedge_{i=0}^{l-1} (R_i \land P_i) \lor (\neg R_i \land \neg P_i)$$

(CNF):

$$\bigwedge_{i=0}^{l-1} (R_i \lor \neg P_i) \land (P_i \lor \neg R_i)$$
$R = 2P$ represents $r = 2p$

$$\neg R_0 \land \bigwedge_{i=1}^{l-1}(R_i \land P_{i-1}) \lor (\neg R_i \land \neg P_{i-1})$$

(CNF):

$$\neg R_0 \land \bigwedge_{i=1}^{l-1}(R_i \lor \neg P_{i-1}) \land (\neg R_i \lor P_{i-1})$$
$R = BP$ represents $r = bp$

$$
\bigwedge_{i=0}^{l-1} ( (R_i \land (B \land P_i)) \lor (\neg R_i \land \neg (B \land P_i) ) )
$$

(CNF):

$$
\bigwedge_{i=0}^{l-1} ( (B \lor \neg R_i) \land (P_i \lor \neg R_i) \land (R_i \lor \neg B \lor \neg P_i) )
$$
\[ R = P + Q \text{ represents } r = p + q \text{ (carry)} \]

\[
\begin{array}{c}
110100_c \\
01101_p \\
+ 00101_q
\end{array}
\]

\((C_{-1}, C_0, \ldots, C_{i-1})\) – carry bits

for \(i > -1\):

\[
(C_i \land ((C_{i-1} \land P_i) \lor (C_{i-1} \land Q_i) \lor (P_i \land Q_i))) \\
\lor
\]

\[
(\neg C_i \land (((\neg C_{i-1} \land \neg P_i) \lor (\neg C_{i-1} \land \neg Q_i) \lor (\neg P_i \land \neg Q_i)))
\]

(CNF):

\[
(\neg C_i \lor P_i \lor C_{i-1}) \land (\neg C_i \lor P_i \lor Q_i) \land \\
(\neg C_i \lor Q_i \lor C_{i-1}) \land (C_i \lor \neg P_i \lor \neg C_{i-1}) \land \\
(C_i \lor \neg P_i \lor \neg Q_i) \land (C_i \lor \neg Q_i \lor \neg C_{i-1})
\]
\[ R = P + Q \] (result)

\[
\begin{align*}
R_i &\land ((C_{i-1} \land \neg P_i \land \neg Q_i) \lor (\neg C_{i-1} \land P_i \land \neg Q_i)) \\
&\lor
\neg C_{i-1} \land \neg P_i \land Q_i) \lor (C_{i-1} \land P_i \land Q_i))) \\
&\lor
\neg R_i \land ((C_{i-1} \land P_i \land \neg Q_i) \lor (\neg C_{i-1} \land P_i \land Q_i) \lor \\
&\lor
C_{i-1} \land \neg P_i \land Q_i) \lor (\neg C_{i-1} \land \neg P_i \land \neg Q_i))
\end{align*}
\]

(CNF):
\[
\begin{align*}
R_i &\lor Q_i \lor P_i \lor \neg C_{i-1}) \land (R_i \lor Q_i \lor \neg P_i \lor C_{i-1}) \land \\
(R_i \lor \neg Q_i \lor P_i \lor C_{i-1}) \land (R_i \lor \neg Q_i \lor \neg P_i \lor \neg C_{i-1}) \land \\
(\neg R_i \lor Q_i \lor P_i \lor C_{i-1}) \land (\neg R_i \lor Q_i \lor \neg P_i \lor \neg C_{i-1}) \land \\
(\neg R_i \lor \neg Q_i \lor P_i \lor \neg C_{i-1}) \land (\neg R_i \lor \neg Q_i \lor \neg P_i \lor C_{i-1})
\end{align*}
\]
\[ R = P + Q \text{ (the whole)} \]

\[-C_{-1} \land -C_{l-1} \land \]
\[\land_{i=0}^{l-1} (\]
\[(-C_i \lor P_i \lor C_{i-1}) \land (-C_i \lor P_i \lor Q_i) \land \]
\[(-C_i \lor Q_i \lor C_{i-1}) \land (C_i \lor -P_i \lor -C_{i-1}) \land \]
\[(C_i \lor -P_i \lor -Q_i) \land (C_i \lor -Q_i \lor -C_{i-1}) \]
\[) \land \]
\[\land_{i=0}^{l-1} (\]
\[(R_i \lor Q_i \lor P_i \lor -C_{i-1}) \land (R_i \lor Q_i \lor -P_i \lor C_{i-1}) \land \]
\[(R_i \lor -Q_i \lor P_i \lor C_{i-1}) \land (R_i \lor -Q_i \lor -P_i \lor -C_{i-1}) \land \]
\[(-R_i \lor Q_i \lor P_i \lor C_{i-1}) \land (-R_i \lor Q_i \lor -P_i \lor -C_{i-1}) \land \]
\[(-R_i \lor -Q_i \lor P_i \lor -C_{i-1}) \land (-R_i \lor -Q_i \lor -P_i \lor C_{i-1}) \]
\[) \]
\[ N = PQ \text{ represents } n = pq \]

\[ pq = q_0 p + q_1 2p + q_2 2^2 p + \ldots + q_{l-1} 2^{l-1} p \]

\[ (S^0 = P) \land (\bigwedge_{i=1}^{l-1} S^i = 2S^{i-1}) \land \]

\[ (\bigwedge_{i=0}^{l-1} M^i = Q_i S^i) \land \]

\[ (R^0 = M^0) \land (\bigwedge_{i=1}^{l-1} R^i = R^{i-1} + M^i) \land \]

\[ (R^{l-1} = N) \]
Complexity of the formula $N = PQ$ – variables

In the formula there are this many $l$-bit numbers represented:

- $P, Q - 2$
- shifts of $P (S^i) - l$
- products $Q_i S^i (M^i) - l$
- sums $R^{i-1} + M^i (R^i) - l$
- implicit carry – $l$

$4l + 2$ numbers $\longrightarrow 4l^2 + 2l$ propositional variables
Complexity of the formula – clauses

- \( l - 1 \) shifts \((S^i = 2S^{i-1})\) – \( 2l - 1 \) clauses each
- \( l \) bit multiplications \((M^i = Q_i S^i)\) – \( 3l - 3 \) each
- \( l - 1 \) sums \((R^i = R^{i-1} + M^i)\) – \( 14l + 2 \) each
- \( 2 \) equalities \((S^0 = P, R^0 = M^0)\) – \( 2l \) each
- \( 1 \) equality \((R^{l-1} = n)\) – \( l \)

Total: \( 19l^2 - 13l - 1 \) clauses
Optimizations of the formula

- the formula can have \( l^2 + O(l) \) variables
- e.g. \( p, q \) – about \( \frac{l}{2} \)-bit
- e.g. \( S^i \) – \( i \) first bits are zeros
Experimental results

- RSA is out of reach of this exact method :(

- Task: to test what can be done on computers available today

- Experiments on 2GHz, 2GB RAM computer:
  - 32-bit – 15 seconds
  - 46-bit – 3 hours
  - 47-bit – 3 days wasn’t enough
  - 48-bit – 33 hours
  - 49-bit – 3 days wasn’t enough
  - 212-bit (random) – 10 seconds (one of the factors: 11)
The limits of what currently can be done

- What can be done with the fastest computers ($35 \cdot 10^{12}$ ops)?
- How long time is needed for 50-bit and 640-bit with the fastest computers?
- How many bits is the limit of the fastest computers?
- How to evaluate the limits of large nets (e.g. the Internet)?
Factorization with SAT

SAT solver dedicated for factorization?

- zChaff has neither specific structural nor number theoretic info about the formula
- probably brute force is used
- How to teach/optimize zChaff about/for the formula?

Ideas:
- grouping of clauses, sorting of clauses and variables
- parallelization
SAT solver dedicated for factorization?

- zChaff has neither specific structural nor number theoretic info about the formula
- probably brute force is used
- How to teach/optimize zChaff about/for the formula?

Ideas:
- grouping of clauses, sorting of clauses and variables
- parallelization
Thanks for your attention