YaSat

Usage

- Simply enter **make [options]** and then **./yasat [input file]**
- There are two options
  - RESTART=1 (enable random restart)
  - NIVER=1 (enable NiVER preprocessing)
- Recommand option **make RESTART=1**
- Uncomment the last line in **sat.cpp** to attain the running time

Implementations

Random Restart

Add some **randomness**!
Randomly choose one variable from the top three variables with highest scores,
in order to do so, I implement my own heap instead of using **std::make_heap**

Restart the solver when $100 * 1.5^{restart\_count}$ conflicts occurred

Multi-threading

Since randomness is added, let all 8 threads do the searching independently,
once a thread find the answer, then output the answer, there's no need to wait for the other threads.
This make the solver be more consistent in the aspect of time.

NiVER

Implement the preprocessor NiVER
But, the sample benchmarks seem not be benefited by **NiVER** and thus **NiVER** even become a overhead.
so, probably don't activate it.

Summary

After i added **randomness**, the execution time for some sample benchmarks has increased significantly, such as "dubois100.cnf".
I think the reason is that when performing the very first **decision**, every variable has the same score,
my implementation in **Milestone 2** has luckly picked a good one among all the candidates, so it runs pretty fast.
Nevertheless, with randomness, there are slim chances to pick a good one among all candidates, but **random restart** and **multi-threading** could more or less improve it.

The execution time of "dubois100.cnf" and "par16-1(-c).cnf" are still inconsistent, it can be from 4 times faster to 10 times slower

On the other hand, the benchmarks "jnh{}.cnf" and some of "aim-{}" are at least 2~3 times faster

Problem

As mention in **summary**, the inconsistence in aspect of time is yet to be fixed,
maybe i messed up something or misunderstand some concepts so that causing this phenomenon.