|  |  |  |  |
| --- | --- | --- | --- |
| Date | Things done | Result | Interpretation of the result |
| 9/12/2020 | Implement the naïve version of the 2^n solver in python | The program can take a QBF as a string and solve it | The solver cannot adapt to the standard QDIMACS format, modification required |
| 18/12/2020 | Implement the brute force solver as well as the QDIMACS parser | The parser can parse the QDIMACS file correctly | The solver is too slow, PNS should be introduced |
| 21/12/2020 | Implement the first version of the PNS algorithm | The PNS works correctly, all formula are stored in memory | No standard SAT solving tricks like unit propagation or pure literal elimination was used, the solver works very bad on the 3x3 example |
| 24/12/2020 | Implement the unit propagation algorithm | The number of iterations was cut significantly | 3x3 example becomes solvable |
| 28/12/2020 | Memory optimized the PNS algorithm | Only formulas of the current searching path are stored in memory | After this optimization, the 3x3 example can be solved nearly 8 times faster |
| 1/01/2021 | Implement the pure literal elimination with a naïve data structure | The number of iterations was reduced to 1/5  [result\_PNS.csv](https://github.com/hharryyf/Taste-of-research-2020/blob/main/result_PNS.csv) | The run time was not improved on the 3x3 example, hence data structure optimization on the formula class is required |
| 3/01/2021 | Implement the mobility initialization, which is (1, branching factor) or (branching factor, 1) instead of (1, 1) | The number of iterations was halved | The run time was not improved on the 3x3 example, hence data structure optimization on the formula class is required |
| 7/01/2021 | Implement the data structure optimization of the formula class | The run time was improved significantly, the result is in a csv file called [resultshuffle\_dtopt.csv](https://github.com/hharryyf/Taste-of-research-2020/blob/main/resultshuffle_dtopt.csv) | It can be seen later that the vast majority of time spent by the algorithm is no longer on unit propagation or pure literal elimination, it is on duplicate the formula instead. This is a very annoying things, because under the current implementation, such thing cannot be optimized easily |
| 8/01/2021 | Implement the dynamic branching | The program can have a branching factor of 2,4,8,16 instead of only 2. Result is recorded in [resultshuffle\_bf.csv](https://github.com/hharryyf/Taste-of-research-2020/blob/main/resultshuffle_bf.csv) | In all such cases, PNS with mobility initialization would take around half the iterations than standard PNS initialization |
| 10/01/2021 | Reserve a place for using a SAT solver in the last quantifier block | Tested correctly, if I get the SAT solver library, the solver should be invoked on time |  |
| 23/01/2021 | Detected a bug in the brute force implementation, I forgot to simplify the formula | Completely unexpected behaviour, the brute force solver is even faster than PNS when solving gttt3x3 example |  |
| 25/01/2021 | Allow 2 parameters, branching factor for existential and branching factor for universal |  |  |
| 31/01/2021 | An update version of PNS, referenced article Reducing the Seesaw Effect with Deep Proof-Number Search | Run time improvement |  |
| 4/02/2021 | Implement the binary frequency method and frequency method | Improve the performance of the gttt3x3 example significantly | We can further try if this method can be used for MPN |