Restart Strategies for Constraint Satisfaction Problems

Alan M. Frisch   Yuan Zhan
Artificial Intelligence Group
Department of Computer Science
University of York
frisch@cs.york.ac.uk yuan@cs.york.ac.uk

1 Introduction

In solving constraint satisfaction problems, deterministic heuristic backtracking algorithms have been observed to solve some instances much faster than others. Though some of this variation results from the inherent difficulty of the instance, a considerable amount of the variation is an artefact of the algorithm. Small changes to the algorithm's heuristics can result in large changes in solution time for a particular problem instance. This is because the heuristics are not perfect and mistakes made at the top of the search space can lead to an enormous increase in solution time.

Gomes, Selman and Kautz [GSK98] have proposed a scheme, called randomisation and restart, for reducing this variation of solution times. The first step is to introduce an element of randomisation into the deterministic algorithm. When the algorithm evaluates a set of alternatives (alternative variables or alternative values) all those whose evaluation is within a fixed ratio of the best are considered to be viable alternatives, of which one is chosen at random with uniform probability. The distribution of run times of this randomised algorithm on a particular problem instance has sometimes been observed to have a "heavy tail" in which a significant number of runs take much longer than the median of the runtimes. The second step of the randomisation and restart scheme is aimed at reducing this heavy tail. This is achieved by imposing a runtime cutoff on the algorithm and restarting the algorithm each time the cutoff is reached without finding a solution.

Gomes, Selman and Kautz [GSK98] have applied their randomisation and restart scheme to two deterministic solvers and have tested the resulting randomised solvers on several problem instances. They have observed that with appropriate cutoff values, the mean time to solution for the randomised solver can be significantly less than the solution time for the original deterministic algorithm.

A restart strategy is an infinite sequence of natural numbers ($\boldsymbol{t}_1$, $\boldsymbol{t}_2$, $\boldsymbol{t}_3$, ...) which indicates that if the algorithm does not halt by $\boldsymbol{t}_1$ steps it is to be restarted. Then it continues to run with a cutoff of $\boldsymbol{t}_2$ steps, then $\boldsymbol{t}_3$ and so forth until a solution is found. Luby, Sinclair and Zuckerman [LSZ93] have shown that for every problem instance the restart strategy that results in the shortest mean time to solution is a fixed restart strategy, ($\boldsymbol{t}$, $\boldsymbol{t}$, $\boldsymbol{t}$, ...) for some value $\boldsymbol{t}$. The problem is that the value of $\boldsymbol{t}$ in the optimal strategy varies greatly from instance to instance. Furthermore, the performance of the algorithm can vary greatly with the setting of $\boldsymbol{t}$; if it is set low when it should be high or high when it should be low, then the mean time to solution can be increased by orders of magnitude. Though the optimal value of $\boldsymbol{t}$ can be computed directly from the run time distribution of the randomised algorithm on the problem instance, this distribution is not usually known.

To address situations in which one has no prior information about the runtime distribution for a problem instance, Luby, Sinclair and Zuckerman[LSZ93] introduced the idea of a universal strategy, which is designed to be generally used on all kinds of runtime distributions. Meanwhile they also suggested a universal strategy for practice. Walsh put forward another original universal strategy in [Walsh99]. But both of these strategies have their mutually exclusive advantages and disadvantages. Our interest is to evaluate the practical performance of these strategies upon different problems and to improve the practicability of these strategies.

2 Two universal restart strategies

Walsh introduced a novel universal restart strategy in which the cutoff values increase geometrically [Walsh99]. That is, each new trial has a cutoff value which is a constant factor ρ larger than the previous. He called it the RGR strategy (Randomisation and Geometric Restarts). The most important characteristic of it is that the cutoff value increases a bit too fast, but it is not sensitive to the problems. As we won't know which kind of problem we will meet, if we meet with a small problem which is easy to be solved or a problem with big left-hand heavy-tail, the cost of using a bigger value for ρ will be many times what we spend when a smaller factor is applied. On the contrary, if we meet with a hard problem or a problem which has no left-hand heavy-tail and we use a small factor, the cutoff value will still increase fast so that it won't cause too much waste. So it can be advantageous to use a small value for ρ like 1.1 or 1.2 for when we have no idea about the problem to be solved.

Luby et al. provided a universal strategy that gives "a sequence of runs whose lengths are powers of two, and each time a pair of runs of a given length has been completed, a run of twice of that length is immediately executed" [LSZ93]. The sequence is as follows:

(1,1,2,1,1,2,4,1,1,2,1,1,2,4,8,1,1,2,1,1,2,4,1,1,2,1,1,2 ,4,8,16,1,1,2……).

It has been proved that for every problem instance the runtime of this universal strategy is at most logarithmically slower than the optimal strategy for the instance.

Luby et al's strategy avoids the big search cost in the bigger cutoff runs where the optimal cutoff is smaller, but it may spend too much in the smaller cutoff runs. On the contrary, Walsh's strategy escapes the waste on the repetitions of small cutoff runs within which there's no solution or little possibility of solution, but may pass by the optimal cutoff fleetly and fall into the waste of big cutoff runs. Therefore, we are thinking of some tradeoffs.

3 Our two intuitions

Luby et al's above sequence is based on powers of 2. To get the sequence to grow faster, it could be based on a power greater than 2. For example, we could use a power of 4:

(1,1,1,1,4,1,1,1,1,4,1,1,1,1,4,1,1,1,1,4,16,1,1,1,1,4,1, 1,1,1,4,1,1,1,1,4,1,1,1,1,4,16,64,1,1,1,1,4,1,1……)

or a power of 8:

(1,1,1,1,1,1,1,1,8,1,1,1,1,1,1,1,1,8,1,1,1,1,1,1,1,1,8,1, 1,1,1,1,1,1,1,8,1,1,1,1,1,1,1,1,8,1,1,1,1,1,1,1,1,8,1,1, 1,1,1,1,1,1,8,1,1,1,1,1,1,1,1,8,64,1,1,1,1……)

Notice that the strategy of using a factor of 8 performs a total of only 128 backtracks before reaching a run of 64 backtracks, whereas for a factor of 2 we need 384 backtracks. So we hope it can be more practical and overcome the shortcoming of Luby et al's strategy, which converges to the solution too slowly.

This strategy can be deemed as a derivation of Luby et al's strategy, only changing the factor of it. However, this strategy still contains the feature of being sensitive to the problem as Luby et al's strategy. If the problem doesn't have a left-hand heavy-tail, it wastes much time in the small runs until it arrives at a run with the cutoff number around the median. So although we expect this change can improve Luby et al's strategy in some extent, we couldn't expect it to improve much.

Another intuition is to generate a hybrid strategy of both the strategies contributed by Luby et al and Walsh, that is adding some repetition into Walsh's strategy with the hope of overcoming the defect of increasing too fast. The cutoff sequence can be like this:

(1,1,5,1,1,5,25,1,1,5,1,1,5,25,125,1,1,5,1,1,5,25,1,1,5 ,1,1,5,125,625,1,1,5,……)

or use a bigger factor 10 like this:

(1,1,10,1,1,10,100,1,1,10,1,1,10,100,1000,1,1,10,1,1, 10,100,1,1,10,1,1,10,100,1000,10000,1,1,10……)

This strategy almost keeps the advantage of Walsh's RGR strategy, which is not sensitive to the problems themselves as well. So we hope it can be more effective in practice when we apply it to a new problem which we have no knowledge about it's runtime distribution.

Acknowledgements

We thank Toby Walsh for useful guidance throughout this project.

References

[GW99] Ian P. Gent and Toby Walsh. The Search for Satisfaction. 1999. http://dream.dai.ed.ac.uk/group/tw/sat/

[GSK98] C. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomisation. In Proceedings of 15th National Conference on Artificial Intelligence, pages 431-437. AAAI Press/The MIT press, 1998.

[GSCK00] C. Gomes, G. Selman, N.Crato and H. Kautz. Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems. J. Automated Reasoning, Special Issue "SAT'2000", 2000.

[GSC97] C. Gomes, G. Selman and N.Crato. Heavy-Tailed Distributions for Backtrack Search. Proc. Third Intl. Conf. On Principles and Practice of Constraint Programming (CP-97), pages 121-135, 1997.

[LSZ93] Michael Luby, Alistair Sinclair, David Zuckerman. Optimal Speedup of Las Vegas Algorithms. In *Proceedings of the 2nd Israel Symposium on Theory of Computing and Systems, Jerusalem*. June 1993

[Walsh99] Toby Walsh. Search in a Small World. Proceedings of IJCAI-99, 1999.