# On the Power of Restarts for CSP

Luís Baptista (student) and Francisco A. Azevedo (supervisor)

Department of Informatics,
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa, Portugal
lmtbaptista@gmail.com, fa@di.fct.unl.pt

**Abstract.** The use of restart techniques in solving Constraint Satisfaction Problems (CSPs) is considered of small importance for backtrack search algorithms. In this paper we propose to conduct a preliminary study on the impact of restarts in randomized backtrack search algorithms for solving CSPs. We show that the well-know n-queens problem has a heavy-tail distribution. We present empirical evidences that restarts can effectively improve the time to solve the n-queens problem. We implement a conflict-driven variable heuristic, and present empirical evidences that this heuristic effectively improve the time to solve the n-queens problem.

**Keywords:** search, constraint, restart, randomization, heuristic

## 1 Introduction

Constraint Satisfaction Problems (CSPs) are a well-known case of NP-complete problems [1]. They have extensive application in areas such as scheduling, configuration, timetabling, resources allocation, combinatorial mathematics, games and puzzles, and many other fields of computer science and engineering.

In this paper we propose to conduct a preliminary study on the impact of restarts in randomized backtrack search algorithms for solving CSPs. We also conduct a preliminary study on the impact of using knowledge from the past runs of the algorithm. We show that the runtime for computing the solution to the n-queens problem, using a randomized backtrack search algorithm, has a heavy-tail distribution. And we show that the use of restarts (restarting the algorithm) on a randomized backtrack search algorithm, with state-of-the-art techniques, improves the overall performance of backtrack search algorithm. And finally, adding a conflict-driven heuristic is shown to be better than the traditional, widely used, fail-first heuristic.

## 2 Constraint Satisfaction Problem

A Constraint Satisfaction Problem (CSP) consists of a set of variables, each with a domain of values, and a set of constraints on a subset of these variables. In this paper we will use a CSP with finite domains. A propositional satisfiability problem (SAT) is a particular case of a CSP where the variables are Boolean, and the constraints are defined by propositional logic expressed in conjunctive normal form.

Backtrack search algorithms are widely used for solving CSPs. It is commonly accepted that those algorithms should incorporate advanced search pruning techniques for space reduction, e.g., domain consistency techniques. Also, the use of heuristics based on the fail-first principle [2] is of great importance for efficiently solving CSPs. On the contrary, the use of restart techniques is considered of small importance for backtrack search algorithms. As a consequence they are not standard on state-of-the-art solvers.

The area of constraint satisfaction problems and the area of propositional satisfiability (SAT) share many techniques [3]. In SAT the use of restarts is a standard technique in state-of-the-art solvers. Restarts were essential in solving real-world instances of SAT [4, 5].

## 3 Restarts

A complete backtrack search algorithm is randomized by introducing a fixed amount of randomness in the branching heuristic [12]. The utilization of randomization results in different sub-trees being searched each time the search algorithm is restarted.

For many combinatorial problems different executions of a randomized backtrack search algorithms, on the same instance, can result in extremely different runtimes. This unpredictability in the runtime of the complete search procedures can be explained by the phenomena of heavy-tail distribution [12, 14, 13]. The heavy-tail distribution is characterized by long tails, as we can see in figure 1. The curve gives the cumulative fraction of successful runs as a function of the number of backtracks [13].

A randomized complete search algorithm is repeatedly run, each time limiting the maximum number of backtracks to a cutoff value. In practice a good cutoff values eliminates the heavy-tail phenomena, but unfortunately such a value has to be found empirically [12]. The resulting algorithm is not complete. A solution to this problem is to implement a policy for increasing the cutoff value [15]. A simple policy is to increment by a constant the cutoff value after each restart. The resulting algorithm is complete, and thus able to prove unsatisfiability [4].

As noted in [19] the impressive progress in SAT, unlike CSP, has been achieved using restarts and nogood recording [17, 18] (plus efficient lazy data structures). And this is starting stimulate the interest of the CSP community in restarts and nogood recording.

# 4 Experimental Results

In our empirical study we use the Comet System (http://www.comet-online.org), using the constraint programming solver over finite domains.

## 4.1 The N-Queens Problem

For our study of the impact of restarts we focus on instances of the well-known n-queen problem. This is a well studied problem [6], and has been used to illustrate various techniques used in solving CSPs [7]. When solving this problem, the use of domain consistence and fail-first heuristic are crucial.

We use the *alldifferent* global constraint to post all the constraint of the problem, where the variables $x_1, \ldots, x_n$, are the columns of the chessboard, and the domains are the possible rows. All queens are placed in:

- different rows: *alldifferent*$(x_1, x_2, \ldots, x_n)$
- different downward diagonals: *alldifferent*$(x_1+1, x_2+2, \ldots, x_n+n)$
- different upward diagonals: *alldifferent*$(x_1-1, x_2-2, \ldots, x_n-n)$

## 4.2 Heavy tail distribution

The heavy-tail distribution in figure 1 was created with 872649 runs of a randomized backtrack search algorithm to solve the 8-queens problem. This algorithm does not use constraint propagation neither variable ordering heuristic. It simple selects the next variable and value randomly.

Figure 1 show that 50% of the runs solve the instance in 5400 backtracks (approximately) or less (the left part of the distribution). However, 1.2% of the runs do not result in a solution after 100000 backtracks (the right part of the distribution, the long tail). As we can observe the 8-queen problem has a heavy-tail distribution.

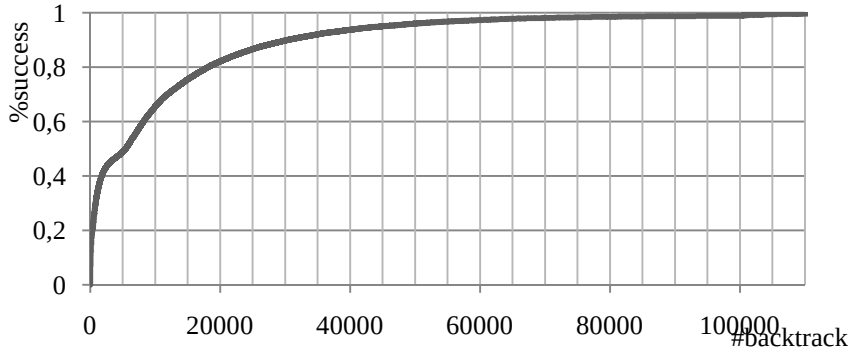


**Fig. 1.** 8-queens heavy-tail distribution

### 4.3 Restarts

In this session we use as a base configuration for all algorithms a backtrack search with constraint propagation and the fail-first heuristic. For the experimental results we use the following algorithms:

- **BK**, we are using the base configuration.
- **BK+Rand**, base configuration and randomly choose among the possible values for the variable.
- **BK+Rst**, base configuration, randomly choose among the variables with the three best values (according to the fail-first heuristic), and restart the search (the initial cutoff value is 1; the increment to the cutoff value after each restart is 10).

The results of running the algorithms are the number of backtracks needed to solve the instance. The backtrack limit for each instance was set to 500000. Since randomization was used in the last 2 algorithms, the number of runs was set to 10. Hence, the results shown correspond to the average values for all the runs.

**Table 1.** Results for different instances (with different number *n* of queens)

| ***n*** | **BK** | **BK+Rand** | **BK+Rst** |
|---|---|---|---|
| 100 | 29 | 58,8 | 84,2 |
| 200 | 200217 | 35525,9 | 63,1 |
| 500 | (1) | 5791,8 | 352,7 |
| 1000 | 2 | 103460,4 (2) | 846,2 |
| 1500 | 4265 | 100310,2 (2) | 2069,0 |
| 2000 | (1) | 50003,6 (1) | 857,2 |

In table 1, the values in parenthesis specify the number of times the backtrack limit was reached (without solving the instance).

As we can observe, the results in table 1 uncover the power of using restarts:

- Restarts allow the algorithm to solve all the instances in all the runs.
- Restarts allow the algorithm to solve the harder (bigger) instances more efficiently (need fewer backtracks).
- Without restarts the algorithms can exhibit long run times, because of the heavy-tail distribution. But with restart the algorithm avoid the long tail of the distribution.

### 4.4 Conflict-driven heuristic

A very important decision heuristic in SAT is based on clause recording (nogood recording) [5]. The general idea is to increment the value of literal involved in conflicts. The heuristic then select the variables involved in more conflicts.

We implement one counter for each variable. When the algorithm does not have more value to assign to a variable (a conflict) we increment that variable counter by 1.

The variable heuristic select the variable involved in more conflicts, and break ties with the fail-first heuristic.

**Table 2.** Result for the conflict-driven heuristic

| *n* | BK+Rst | BK+Rst+Conf |
|---|---|---|
| 100 | 84,2 | 30,9 |
| 200 | 63,1 | 142,0 |
| 500 | 352,7 | 126,2 |
| 1000 | 846,2 | 197,8 |
| 1500 | 2069,0 | 268,5 |
| 2000 | 857,2 | 213,8 |

The label **BK+Rst+Conf** represent the backtrack search algorithm with restarts and with the conflict-driven heuristic. As we can see this heuristic improves, except in one case, the number of backtrack to solve the n-queens instances. So, in those instances this heuristic is better than the fail-first heuristic.

In [5] the counters are periodically divided by a constant, but in our implementation we do not divide the counters. This could be the reason why in some situations this heuristic behaves poorly. Nevertheless, this heuristic show promising results.

## 5 Conclusions

This paper gave a preliminary study on restarts applied in CSP. Restarts are not standard in CSP backtrack search algorithms, and are not considered useful in improving the algorithm. This paper contributes by contradicting the conventional wisdom. It shows that:

- Restarts can effectively improve the time to solve the n-queens problem.
- Conflict-driven variable heuristic can also effectively improve the time to solve the n-queens problem.

Restart do not substitute other techniques, it complements, empower other techniques, like constraint propagation and heuristic decisions. Restarts are an open area for CSP. Progress in SAT was due to restarts and nogoods [19].

In the near future we expect:

- Confirm the results with other CSP instances (real-world and randomly generated).
- Include in the study nogood recording from conflict (learning).
- Study the impact of different cutoff and cutoff increment values.
- Enhance the conflict-driven heuristic.
- Test other forms of making the algorithm with restarts complete.

# 6 References


1. Apt, K.R.: Principles of constraint programming, Cambridge University Press (2003).
2. Haralick, R.M., Elliott, G.L.: Increasing tree search efficiency for constraint satisfaction problems. Proceedings of the 6th international joint conference on Artificial intelligence - Volume 1. pp. 356-364 Morgan Kaufmann Publishers Inc., Tokyo, Japan (1979).
3. Bordeaux, L., Hamadi, Y., Zhang, L.: Propositional Satisfiability and Constraint Programming: A comparative survey, ACM Comput. Surv., vol. 38, 2006, p. 12.
4. Baptista, L., Silva, J.P.M.: Using Randomization and Learning to Solve Hard Real-World Instances of Satisfiability. Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming. pp. 489-494 Springer-Verlag (2000).
5. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. Proceedings of the 38th annual Design Automation Conference. pp. 530-535 ACM, Las Vegas, Nevada, United States (2001).
6. Bell, J., Stevens, B.: A survey of known results and research areas for n-queens, Discrete Mathematics, vol. 309, Jan. 2009, pp. 1-31.
7. Rossi, F., Beek, P.V., Walsh, T.: Handbook of constraint programming, Elsevier (2006).
8. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, Prentice Hall (2002).
9. Selman, B., Kautz, H.: Domain-independent extensions to GSAT: solving large structured satisfiability problems. Proceedings of the 13th international joint conference on Artifical intelligence - Volume 1. pp. 290-295 Morgan Kaufmann Publishers Inc., Chambery, France (1993).
10. Hoos, H.H., Stützle, T.: Stochastic local search: foundations and applications, Morgan Kaufmann (2005).
11. McAllester, D., Selman, B., Kautz, H.A.: Evidence for Invariants in Local Search, 1997.
12. Gomes, C.P., Selman, B., Kautz, H.: Boosting combinatorial search through randomization. Proceedings of the fifteenth national conference on Artificial intelligence. pp. 431-437 American Association for Artificial Intelligence, Madison, Wisconsin, United States (1998).
13. Gomes, C.P., Selman, B., Crato, N., Kautz, H.: Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems, JOURNAL OF AUTOMATED REASONING, vol. 24, 2000, pp. 67-100.
14. Gomes, C., Selman, B., Crato, N.: Heavy-Tailed Distributions in Combinatorial Search, Principles and Practices of Constraint Programming, 1997, pp. 121-135.
15. Walsh, T.: Search in a Small World. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. pp. 1172-1177 Morgan Kaufmann Publishers Inc. (1999).
16. Baptista, L., Lynce, I., Marques-Silva, J.: Complete Search Restart Strategies for Satisfiability, IN IJCAI WORKSHOP ON STOCHASTIC SEARCH ALGORITHMS (IJCAI-SSA, 2001.
17. Lecoutre, C., Sais, L., Tabary, S., Vidal, V.: Nogood recording from restarts. Proceedings of the 20th international joint conference on Artifical intelligence. pp. 131-136 Morgan Kaufmann Publishers Inc., Hyderabad, India (2007).
18. Lecoutre, C., Saïs, L., Tabary, S., Vidal, V.: Recording and Minimizing Nogoods from Restarts, Journal on Satisfiability, Boolean Modeling and Computation, vol. 1, 2007, pp. 147–167.
19. Lecoutre, C.: Constraint Networks: Techniques and Algorithms, Wiley-ISTE (2009).