Incorporating Variance in Impact-Based Search

Serdar Kadioglu[1], Eoin O'Mahony[2], Philippe Refalo[3], and Meinolf Sellmann[4]

[1] Brown University, Dept. of Computer Science, Providence, RI 02912, USA
serdark@cs.brown.edu
[2] Cornell University, Dept. of Computer Science, Ithaca, NY 14850, USA
eoin@cs.cornell.edu
[3] IBM, 1681 route des Dolines, 06560 Sophia-Antipolis, France
philippe.refalo@fr.ibm.com
[4] IBM Watson Research Center, Yorktown Heights, NY 10598, USA
meinolf@us.ibm.com

Abstract. We present a simple modification to the idea of impact-based search which has proven highly effective for several applications. Impacts measure the average reduction in search space due to propagation after a variable assignment has been committed. Rather than considering the mean reduction only, we consider the idea of incorporating the variance in reduction. Experimental results show that using variance can result in improved search performance.

Keywords: Search Strategies, Impact-based Search, Robust Search.

1 Introduction

Impact-based search strategies give efficient variable and value ordering heuristics to solve decision problems in constraint programming [12]. This method learns information about the importance of variables and values choices by averaging the observed search space reduction due to constraint propagation after an assignment. It's a simple way to exploit parts of the search tree that are apparently not useful because they do not lead to a solution.

Other impact measures have been designed and subjected to experimental validation. They refine or take into account more information in order to obtain better strategies. In [14] the solution density of constraints and occurrences of values in constraints' feasible assignments are used to guide search. In [1] the measure of the impact of an assignment is based on explanations provided by the constraint programming solver. These approaches can be more effective than regular impacts on some problems.

We propose in this paper a new way to refine the classical averaging of impact observations by taking into account the *variance* of the observations. In practice, when one needs to choose between two variables that have the same average impact, one can break this tie by taking into account the distribution of the observed impacts. Assuming that the two distributions have different variances, a risk-free choice will choose the variable with the smallest variance, while an optimistic choice will choose the variable with the largest variance.

Incorporating variance in impact based search is rather natural since impacts are based on taking the mean of observed domain reductions. Moreover, in practice, impact

J. Lee (Ed.): CP 2011, LNCS 6876, pp. 470–477, 2011.

values are normally distributed [13]. Experimental validation was performed to determine the best way to use variance in practice. We present our results on quasi-group completion problems, magic squares and on the Costas array problem. Our results show that including variance can be rewarding in several cases and that it is an enhancement to be considered for impact-based search implementations.

2 Impact-Based Search

In constraint programming (CP) we strive to find feasible solutions, and our main inference mechanism is constraint propagation. Namely, by considering the problem constraints, one at a time, we eliminate potential values for the variables involved in the constraint. We iterate this process until no one constraint alone can eliminate values from the domain of variables anymore.

If we want to avoid an explicit enumeration of all potential solutions, we must obviously rely on constraint propagation to discard most of these solution candidates *implicitly*. Therefore, the way how we partition the space needs to enable constraint propagation to function well.

There are several ways how search methods try to provide the underlying inference mechanism with the necessary "grip." One traditional method is based on the fail-first principle which states: "To succed, try first where you are most likely to fail." [7]. To list only a few others, solution-density guided search [14] finds a constraint where one variable clearly favors one value in the sense that in most assignments that obey this constraint the variable is overwhelmingly assigned to the respective value. The method branches over that variable in the hope that the constraint will fail quickly when one of the other values is assigned to it. [1]

In mathematical programming a well-known and successful technique is pseudo-cost branching. While solution-density guided search looks ahead, pseudo-cost branching keeps a running average of the change in relaxation objective value due to the branching on a variable. That is, pseudo-cost branching extrapolates the past search experience to make predictions which search partition is likely to affect the inference mechanism the most.

Impact-based search in constraint programming is following the same motivation. Lacking an objective function, [12] proposes to keep a running average of the reduction in search space that is observed after committing a variable assignment $X = v$. Assume the Cartesian product of the variables' domains *before* committing the assignment has size $B \in \mathbb{N}$ and the product of all domain sizes *after* committing and propagating the assignment is $A \in \mathbb{N}$. Then, the impact of the assignment is defined as

$$I(X = v) = 1 - \frac{A}{B}.$$

The running average of these values is denoted with $\bar{I}(X = v)$.

From these values we can derive the expected search space reduction factor (ERF) for a variable. Namely, the sum of the Cartesian products of all domain sizes after

[1] Note that this is our summary which does not quite align with the motivation given in [14].

committing in turn $X = v$ for all values v in the domain $D(X)$ is expected to be multiplied by

$$\mathrm{ERF}(X) = 1 - \sum_{v \in D(X)} \bar{I}(X = v).$$

In [12] it was proposed to branch over the variable with the lowest corresponding ERF: As we assume that we are searching an infeasible part of the search space, we expect that all alternative values for X must be explored. The lower the ERF, the smaller we expect the union of the remaining search spaces to be after committing assignments $X = v$ for all $v \in D(X)$. This method has since proven to work very well in various domains such as latin square completion, magic square, and multi-knapsack problem.

3 Impact Variance

Obviously, when estimating the ERF by computing a running average of reductions in search space that we observe, our estimate will come with some uncertainty.

3.1 Variance

there is no reason to believe that the past predicts the present

To assess the confidence that we have in our estimate, variance is the statistical quantity that comes to mind first. Between two variables that have the same low ERF, being risk averse clearly we would favor the one that has exhibited less variance in search space reduction. On the other hand, if we are optimistic we might want to choose a variable that offers the potential of significantly reducing the search space.

If we incorporate variance, we now have two quantities that we want to optimize. The natural question is what should be the right trade-off between both objectives. If we assumed that the ERFs of a variable are normally distributed, then

- with about **68%** probability the real reduction factor will be lower than the mean **plus the standard deviation** (i.e., the square root of the variance),
- with about **95%** probability the real reduction factor will be lower than the mean plus **two times** the standard deviation, and
- with about **99.7%** probability the real reduction factor will be lower than the mean plus **three times** the standard deviation.

Alternatively, if we take the optimistic viewpoint and value variables with larger potential more, for a normal distribution we can argue that

- with about **32%** probability the real reduction factor will be lower than the mean **minus the standard deviation**, and
- with about **5%** probability the real reduction factor will be even lower than the mean minus **two times** the standard deviation.

Even though we cannot assume that the real reduction factors will be exactly normally distributed, the trend will be the same for all distributions. We therefore propose to

choose an $\alpha \in \mathbb{Q}$ (where $\alpha > 0$ means we are risk averse, and $\alpha < 0$ means "we are feeling lucky") and to compute the *adjusted reduction factor*

$$\mathrm{ARF}_\alpha(X) = \mathrm{ERF}(X) + \alpha\sqrt{\mathrm{VAR}(X)}.$$

Then, we choose as branching variable

$$X = \mathrm{argmin}_Y \mathrm{ARF}_\alpha(Y).$$

If we choose a large α, then we compare variables by their ability to shrink the search space which we can expect with some higher probability. On the other hand, if we choose a low value for α, we compare variables based on their potential to shrink the search space a lot.

Note that the idea to use a factor $\alpha < 0$ somewhat resembles the idea of *upper confidence trees (UCTs)* [10]. As a very high-level description, in the UCT method we probe a tree and achieve estimates of the quality of a subtree by the samples drawn from the probes over the different child nodes. The question arises which probes should be launched next. Based on a very nice theory it was proven that it pays off to optimistically consider subtrees first which combine a good current estimate and larger uncertainty [10].

Our situation is of course different, as each "probe" can incur a significant cost. Essentially, without nogood-learning, with each unfortunate choice of the branching variable we could multiply the minimally required search effort. Therefore, in this paper we compare risk-averse and optimistic impact-based search in an empirical study.

3.2 Computing a Variance-Estimate

To implement the approach outlined above we obviously need to assess the quantity $\mathrm{VAR}(X)$. We can achieve this based on the variance that we observe for the variable assignment impacts $I(X = v)$. Since the random variable ERF is based on the sum of these random variables, if we assume that the $I(X = v)$ (for various v) are independent, then the variance of the ERF will be simply the sum of the variances of the $I(X = v)$. In other words, the variance can be estimated as

$$\mathrm{VAR}(X) = \sum_{v \in D(X)} \mathrm{VAR}(X = v).$$

All that is left to develop now is a way for estimating the variance of the $I(X = v)$. We could of course keep a history of these values and compute the variance from scratch. However, there is a much more elegant way, namely, after a new value for $I(X = v)$ or $\mathrm{ERF}(X)$ is observed, we can update the variance *online*.

This is trivial for the mean of a sequence. Given numbers $a_1, \ldots, a_{n-1}$ and their mean $\mu_{n-1} = \frac{\sum_i a_i}{n-1}$, and a new number a_n, for the new mean it obviously holds

$$\mu_n = \frac{(n-1)\mu_{n-1} + a_n}{n}.$$

A similar update rule holds for the sum of square differences $\mathrm{SD}_{n-1} = \sum_i (a_i - \mu_{n-1})^2$ [9]:

$$\mathrm{SD}_n = \mathrm{SD}_{n-1} + (a_n - \mu_n)(a_n - \mu_{n-1}).$$

Therefore, we maintain three numbers for each $I(X = v)$: The number of times n we have observed a value, the current mean μ_n, and the current sum of square differences SD_n. Then, to choose a branching variable we use the unbiased [9] variance estimate $\frac{\mathrm{SD}_n}{n-1}$.

4 Numerical Results

We now present empirical results demonstrating the benefits obtained by incorporating variance information as well as impacts when branching. We implemented the new heuristic in IBM Ilog Solver, and studied a number of problems. The goal of our experiments is to compare the relative performance of these three different variable selection heuristics:

- Impact-based search (IBS).
- Impact-based search with addition of variable variance. That is, α is set to 1 or to 2, which means we favor variables with low variance.
- Impact-based search with subtraction of variable variance. That is, α is set to -1 or to -2, which means we favor variables with high variance.

To conduct a fair test of the different variable selection heuristics we use a randomized value selection strategy and perform multiple runs of the same instance with different random seeds. The initial mean and variance value are obtained by probing each value of the domain of the variable. This gives a first impact for each value. Then a second impact is computed by performing a few steps of a dichotomic search to approximate impacts as described in [12]. Thus we have enough values at the beginning to compute the impact mean and the unbiased variance. In the instances we consider here, the overhead of probing the whole variable domain is negligible compared to the solution time. All approaches ran on identical models with the DFS search implementation of IBM ILOG Solver. The experiments were run on dual processor dual core Intel Xeon 2.8 GHz computers with 8GB of RAM.

4.1 Quasigroup Completion

Problem Definition. *A quasigroup completion problem [5] is tasked with completing an $n \times n$ partially filled matrix such that the numbers from 1 to n exactly once in each row and column*

Quasigroup completion problems are a well-known combinatorial problem. These problems, unlike latin square or quasigroup with holes problems, are not strictly satisfiable. We consider two sets of 100 instances. One set with order 40 and 640 unassigned cells ("holes"), and another one with order 50 and 1250 unassigned cells. They are generated randomly using a standard tool provided by the authors of [5]. We used depth-first search to solve the problems and four standard deviation factors $\alpha = \{-2, -1, 1, 2\}$.

Table 1. Results for the Quasigroup Completion Problem (order = 40, holes = 640)

α value	-2	-1	0 (IBS)	1	2
Time	**302**	320	336	374	408
# Success	**596**	555	523	439	355

Table 2. Results for the Quasigroup Completion Problem (order = 50, holes = 1250)

α value	-2	-1	0 (IBS)	1	2
Time	**166**	187	184	247	333
# Success	**864**	825	805	739	567

When setting α to 0, the strategy is simply the standard impact-based strategy. We perform 10 runs for each instance with as many different random seeds. The time limit is 2000 seconds. We report the average running time in seconds and the number of instance solved (the maximum is 1000 for each set).

The results of our evaluation is presented in Table 1 and Table 2. We can see that a risk-optimistic approach outperforms both the classical and risk-averse impact-based search on the instances. The optimistic strategy solves considerably more instances in substantially less time. On the other hand, risk-averse strategy marks the worse performance. It is slower and solves the least number of instances.

4.2 Magic Square

Problem Definition: *A magic square [11] of order n is an $n \times n$ square that contains all numbers from 1 to n^2 such that each row, column and both main diagonals add up to the "magic sum" $n(n^2 - 1)/2$).*

Magic squares are a much studied problem in the domain of combinatorial solvers. Although polynomial-time construction methods exist for creating magic squares, the problem poses a challenge for constraint programming based approaches. The current best systematic approach was presented in [4] and can only construct magic squares of orders up to 18 efficiently.

We again evaluate the performance of impact based search (when α is 0) and impact based search incorporated with standard deviation using factors $\alpha = \{-2, -1, 1, 2\}$. We consider magic squares of orders between 5 and 16. We use 50 different seeds for each order. The time limit is set to 2000 seconds. We compare the average runtime and the average number of successful trials.

Table 3 summarizes our results for the magic square problem. We observe that risk-optimistic approaches and impact-based search perform similarly, while the best performance is achieved when $alpha$ is set to -2, i.e., when the most risk-optimistic strategy is used. On the contrary, risk-averse approaches depict an inferior performance in terms of both running time and number of successful trials.

Table 3. Results for the Magic Square Problem

α value	-2	-1	0 (IBS)	1	2
Time	**680**	691	686	705	735
# Success	**34.3**	34.2	34.2	33.8	33

Table 4. Results for the Costas array problem

α value	-2	-1	0 (IBS)	1	2
Time	224	**218**	220	234	234
# Success	46.8	**46.9**	46.8	46.4	46.8

4.3 Costas Array

Problem Definition: *A Costas array [6] is a pattern of n marks on a $n \times n$ grid, such that each column or row contains only one mark, and all of the $n(n-1)/2$ vectors between the marks are all different.*

Costas arrays are a mathematical structure that is studied in a number of domains. It is a combinatorial structure with links to number theory, and is used to provide a template for generating radar and sonar signals with ideal ambiguity functions [2,3].

We consider Costas arrays of orders between 10 and 19, and evaluate impact-based search, and impact-based search with standard deviation incorporated using factors $alpha = \{-2, -1, 1, 2\}$. We use 50 different seeds for each order. The time limit is set to 2000 seconds. We compare the average running time, and the average number of successful trials.

Our results are presented in Table 4. While the best performance is achieved with a risk-optimistic approach, it is better than impact-based search with only a small margin. The risk-averse approaches again perform worse than other strategies.

Overall, we tried to determine the best way to use variance information in practice on three different constraint satisfaction problems. We have shown that including variance in a risk-optimistic setting can improve the search performance in several cases. We attribute the improved performance of the optimistic strategy to its ability to select variables that have high potential to reduce the search space.

5 Conclusion

In this paper we presented a new search heuristic which is based on a simple modification to impact-based search. The modification is to take variance information into account as well as impact values when selecting a branching variable. We consider a risk-averse and a risk-optimistic version of impact-based search with different coefficients, and provide experimental results that compare their relative performance on

three different problems. Our findings suggest that a risk-optimistic approach can improve the search performance, hence it has potential to be a useful search heuristic, and, in general, variance information is an enhancement to be considered for impact-based search implementations.

References

1. Cambazard, H., Jussien, N.: Identifying and Exploiting Problem Structures using Explanation-Based Constraint Programming. Constraints 11(4), 295–313 (2006)
2. Costas, J.P.: A Study of a Class of Detection Waveforms Having Nearly Ideal Range-Doppler Ambiguity Properties. Proceedings of IEEE 72(8), 996–1009 (1984)
3. Freedman, A., Levanon, N.: Staggered Costas signals. IEEE Trans. Aerosp. Electron Syst. AES-22(6), 695–701 (1986)
4. Gomes, C.P., Sellmann, M.: Streamlined constraint reasoning. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 274–287. Springer, Heidelberg (2004)
5. Gomes, C.P., Shmoys, D.: Completing Quasigroups or Latin Squares: A Structured Graph Coloring Problem. In: Proceedings of Computational Symposium on Graph Coloring and Generalizations (2002)
6. Golomb, S.W., Taylor, H.: Two Dimensional Synchronization Patterns for Minimum Ambiguity. IEEE Trans. Informat. Theory IT-28(4), 600–604 (1982)
7. Haralick, R.M., Elliott, G.L.: Increasing Tree Search Efficiency for Constraint Satisfaction. Artificial Intelligence 14, 263–314 (1980)
8. IBM. IBM ILOG Reference manual and user manual. V6.4. IBM (2009)
9. Knuth, D.-E.: The Art of Computer Programming. seminumerical algorithms, vol. 2(3), p. 232. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1997)
10. Levente, K., Csaba, S.: Bandit Based Monte-Carlo Planning. Machine Learning ECML (2006)
11. Moran, J.: The Wonders of Magic Squares. Vintage, New York (1982)
12. Refalo, P.: Impact-Based Search Strategies for Constraint Programming. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 557–571. Springer, Heidelberg (2004)
13. Refalo, P.: Learning in Search. In: Hybrid Optimization. Springer, Heidelberg (2011)
14. Zanarini, A., Pesant, G.: Solution Counting Algorithms for Constraint-Centered Search Heuristics. Constraints 14(3), 392–413 (2009)