Failure Based Variable Ordering Heuristics for Solving CSPs

Hongbo Li
School of Information Science and Technology, Northeast Normal University, Changchun, China

Minghao Yin
School of Information Science and Technology, Northeast Normal University, Changchun, China

Zhanshan Li[1]
College of Computer Science and Technology, Jilin University, Changchun, China

Abstract

Variable ordering heuristics play a central role in solving constraint satisfaction problems. In this paper, we propose failure based variable ordering heuristics. Following the fail first principle, the new heuristics use two aspects of failure information collected during search. The failure rate heuristics consider the failure proportion after the propagations of assignments of variables and the failure length heuristics consider the length of failures, which is the number of fixed variables composing a failure. We performed a vast experiments in 41 problems with 1876 MiniZinc instances. The results show that the failure based heuristics outperform the existing ones including activity-based search, conflict history search, the refined weighted degree and correlation-based search. They can be new candidates of general purpose variable ordering heuristics for black-box CSP solvers.

2012 ACM Subject Classification Computing methodologies

Keywords and phrases Constraint Satisfaction Problem, Variable Ordering Heuristic, Failure Rate, Failure Length

Digital Object Identifier 10.4230/LIPIcs.CP.2021.9

Category Short Paper

Supplementary Material *Software (Source Code)*: `https://github.com/lihb905/fbs/` archived at `swh:1:dir:2c2d84dba840b8f5299bf2bc2edc8df70031c82a`

Funding *Hongbo Li*: The National Natural Science Foundation of China under Grant NO. 61802056. *Minghao Yin*: The National Natural Science Foundation of China under Grant NO. 61976050. *Zhanshan Li*: Open Research Fund of Key Laboratory of Space Utilization, Chinese Academy of Sciences under Grant NO. LSU-KFJJ-2019-08.

1 Introduction

Constraint satisfaction problems (CSP) are a powerful framework to model and solve combinatorial search problems occurring in various fields. The challenge in a CSP is to determine an assignment of values to all variables that satisfies all the constraints, or otherwise, to prove there is no such an assignment. Backtracking search is a complete method that has been used to solve general CSPs. It performs a depth-first traversal of a search tree to solve CSPs. At each node of the search tree, an unassigned variable is selected to assign a value. The ordering in which the variables are assigned is crucial to the efficiency of backtracking algorithms for solving CSPs. It is a computationally difficult task to find an optimal ordering that results in a search tree exploring the fewest number of nodes [11], thus, the ordering is determined by variable ordering heuristics (VOH) in practice.

[1] Minghao Yin and Zhanshan Li are corresponding authors.

In the past decades, much effort has been done in developing efficient variable ordering heuristics. Many VOHs have been designed according to the fail first principle that "to succeed, try first where you are likely to fail" [6]. The aim is to first process the variables that belong to the most difficult part of a CSP. Modern VOHs are adaptive, which learn during search to find the variables that are most likely to cause a failure. To select the next variable to assign a value at a search tree node, the VOHs estimate how likely a variable causing a failure from various aspects. For instances, the minimum domain size VOH considers the variables with the smallest domain sizes [6], the weighted degree VOH considers the variables involved in the constraints causing more failures have a larger chance to cause a failure [2, 20], some VOHs consider the variables involved in the constraints with higher tightness have a larger chance to cause a failure [3, 10], the conflict history search considers the variables involved in the constraints causing recent failures have a larger chance to cause a failure [5].

In this paper, we propose failure based variable ordering heuristics pursuing the fail first principle from new aspects. The new VOHs do not consider which constraint leads to a failure but which variable causes the failure. They consider the straightforward information between an assignment of a variable and the propagation result of the assignment, failure or success. To estimate how likely a variable causing a failure, the failure rate based heuristic collects the information of proportion of failures caused by assignments of a variable, and the failure length based heuristic collects the information of length of failures caused by assignments of a variable, i.e., the depth of the search tree when a failure occurs. There is no parameter to set in the failure based heuristics. We employ the decaying strategy of the conflict history search to make the new VOHs favor the variables causing recent failures. The failure based VOHs behave like the last conflict based reasoning [9], so the difference between them is discussed. We perform experiments in the benchmark set of MiniZinc containing 1876 instances of 41 problems. Besides the naive VOHs, we compare the VOHs equiped with a geometric fast restart strategy and last conflict based reasoning. The results show that the failure rate based VOH with the decaying strategy solves the largest number of instances. It outperforms the state of the arts VOHs including ABS [12], CHS [5] and $dom/wdeg^{ca.cd}$ [20] and CRBS [18] and performs best in general.

The paper is organized as follows. Section 2 provides the background of CSPs. Some related works are mentioned in Section 3. The failure based VOHs are introduced in Section 4. Section 5 presents the experimental results. Finally, the conclusion is in Section 6.

2 Background

A constraint satisfaction problem (CSP) $\mathcal{P}$ is a triple $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where X is a set of n variables $\mathcal{X} = \{x_1, x_2 \ldots x_n\}$, $\mathcal{D}$ is a set of domains $\mathcal{D} = \{dom(x_1), dom(x_2) \ldots dom(x_n)\}$, where $dom(x_i)$ is a finite set of possible values for variable x_i, and $\mathcal{C}$ is a set of e constraints $\mathcal{C} = \{c_1, c_2 \ldots c_e\}$. Each constraint c consists of two parts, an ordered set of variables $scp(c) = \{x_{i1}, x_{i2} \ldots x_{ir}\}$ and a subset of the Cartesian product $dom(x_{i1}) \times dom(x_{i2}) \times \ldots \times dom(x_{ir})$ that specifies the disallowed (or allowed) combinations of values for the variables $\{x_{i1}, x_{i2} \ldots x_{ir}\}$. An assignment of a variable x is in the form $(x = v)$ where v is a value in $dom\ (x)$.

A solution to a CSP is an assignment of a value to each variable such that all the constraints are satisfied. Solving a CSP $\mathcal{P}$ involves either finding one (or more) solution of $\mathcal{P}$ or proving that $\mathcal{P}$ is unsatisfiable. Backtracking search performs a depth-first traversal of a search tree to solve general CSPs. In the context of a search tree, each edge is associated with an assignment, a node at level k is associated with a set of k assignments which are attached to the path from the root to this node. The root node at level 0 is an empty set.

We use *PastVar* to denote the set of fixed variables which have been assigned and *FutVar* to denote the set of future variables which have not been assigned. At each search tree node, a future variable is selected by a VOH and a new node is generated after the assignment to this variable, then a propagation algorithm filtering those inconsistent values from the domains of variables is applied. If the propagation leads to a domain wipe out, then a failure is encountered, one or more assignments must be canceled and a backtracking occurs.

3 Related Works

Following the fail first principle, the most popular variable ordering heuristic, minimum domain size, selects the variable with smallest domain size [6]. It has been combined with many efficient VOHs. The *dom/deg* [1] and *dom/ddeg* [16] combine minimum domain size with largest variable degree. The weighted degree associates a weight with each constraint, which records the number of failures caused by the constraint [2]. It identifies the variables involved in difficult parts of problems. Combined with minimum domain size, the *dom/wdeg* has been one of the most efficient general purpose VOHs and has been used as default VOH by some solvers, such as Choco [14]. Its variants use different strategies to update the weights of constraints, such as constraint tightness [10] and the explanation information of a failure [7]. Its recent refinement, $wdeg^{ca.cd}$, combines current arity and current domains to update the constraint weights, has been shown to outperform the classic weighted degree heuristic [20]. The impact-based search (IBS) estimates the search space reduction after the propagation of an assignment of a variable [15]. It prefers the variables which may lead to the greatest search space reduction. The count-based search (CBS) considers solution densities of constraints [13]. It prefers the variable-value pair with the largest solution density. The activity-based search (ABS) estimates how active a variable is, e.g., how often the variable is affected by the assignments of other variables [12]. It prefers the most active variables. The correlation based heuristic (CRBS) measures the possibility of having conflict between each pair of variables and estimates the degree of conflicts when choosing a variable [18]. It prefers the variable which is estimated to causing more conflicts. The conflict history search (CHS) considers the history of constraint failures [5]. It prefers the variables involved in those constraints causing recent failures. Given a set of candidate VOHs, the Multi-Armed Bandit(MAB) techniques have been used to estimate the best VOH on a CSP instance. It has been shown that the MAB-based methods are more efficient than any single candidate VOH [21, 19]. The last-conflict based reasoning can be combined with any VOH, called underlying VOH. Whenever an assignment of a variable x is canceled, such as the propagation of the assignment leads to a failure, x is stored as a last conflict variable. The strategy always selects the last conflict variable until its assignment succeeds. It makes the next selection by the underlying VOH if there is no last conflict variable stored.

4 Failure Based Search

The failure based variable ordering heuristics

Given a CSP, a CP solver applies a propagation algorithm F after an assignment of a variable x. If the propagation of the assignment of a variable x leads to a failure (a domain wipeout), then we say the failure is caused by x. Although the actual reason of the failure may contain other assignments, we consider only the last assignment as the reason here, because only the last assignment must be one of the reasons of the failure, whereas some of the previous assignments may not be the reasons.

▶ **Definition 1** (Failure Rate). *In the context of backtracking search, the Failure Rate of a variable x FR(x) is $\frac{failNum(x)}{assignNum(x)}$ where the $failNum(x)$ is the total number of failures caused by x and the $assignNum(x)$ is the total number of x being assigned (or selected) since the beginning of search.*

The failure rate based (FRB) variable ordering heuristic collects the information of $failNum(x)$ and $assignNum(x)$ to calculate the failure rate for each variable x. It selects the variable with the largest $\frac{FR(x)}{|dom(x)|}$. For each variable x, $failNum(x)$ and $assignNum(x)$ are initialized to 0.5 and 1 respectively. The initialization indicates that the default failure rate of each variable is 50% before the failure information is collected. After the searching starts, if a variable x leads to a failure soon, $FR(x)$ will increase. If the variable leads to a failure after being assigned several times, $FR(x)$ will decrease.

▶ **Definition 2** (Failure Length). *If the propagation of an assignment of a variable x leads to a failure, then the length of the failure is $|PastVar|$ + 1, where PastVar is the set of fixed variables before the assignment.*

Failure length is the depth of the search tree when a failure occurs. The length of a failure caused by a variable x is denoted by $|failure(x)|$. The failure length based (FLB) variable ordering heuristic associates an accumulated failure length with each variable x, denoted by $AFL(x)$. For each variable, $AFL(x)$ is initialized to 0. If a failure is caused by a variable x during search, then $AFL(x)$ is updated as follows.

$$AFL(x) = AFL(x) + \frac{1}{|failure(x)|} \tag{1}$$

The FLB variable ordering heuristic selects the variable with the largest $\frac{AFL(x)}{|dom(x)|}$. The heuristic prefers the variables causing more shorter failures, i.e., the variables causing failures at higher levels of the search tree.

Modern VOHs usually use some decaying strategies to give the recent information more priority, such as CHS and ABS. We employ the strategy of CHS to make the failure based variable ordering heuristics prefer the variables causing recent failures. For each variable x, the factor of the decaying strategy is defined as,

$$A(x) = \frac{1}{\#TotalFailure - LastFailure(x) + 1} \tag{2}$$

where *#TotalFailure* is the total number of failures detected since the beginning of search, and *LastFailure(x)* stores the *#TotalFailure* value of the last failure caused by x.

Based on the $A(x)$ factor, we propose a new VOH combining the FRB with $A(x)$, namely FRBA. We use addition to combine them here. The FRBA variable ordering heuristic selects the variable with largest $FRBA(x)$ defined as,

$$FRBA(x) = \frac{FR(x) + A(x)}{|dom(x)|} \tag{3}$$

Similarly, we propose a new VOH combining the FLB with $A(x)$, namely FLBA. The $AFL(x)$ may be much larger than $A(x)$ and the former may obscure the latter if we use addition here, so we use multiplication to combine them. The FLBA variable ordering heuristic selects the one with largest $FLBA(x)$ defined as,

$$FLBA(x) = \frac{AFL(x) \times A(x)}{|dom(x)|} \tag{4}$$

In backtracking search of CSPs, CP solvers usually use the binary branching (i.e., 2-way branching) strategy which has been shown to be more efficient than the non-binary branching strategy [8]. With binary branching, if the propagation of an assignment ($x = v$, a left branch) leads to a failure, a backtracking occurs and a propagation of the refutation ($x \neq v$, a right branch) is performed. Note that the failure based VOHs collect only the failure information of left branches, so the failures of right branches do not affect $FR(x)$ and $AFL(x)$. For $A(x)$, although $\#TotalFailure$ counts the failures of both left branches and right branches, $LastFailure(x)$ is updated only after a failure of left branch is detected.

Discussion

The failure based VOHs behave like the last conflict based reasoning that tend to select the variable causing the last failure, but the new VOHs do not strictly select the last one. We have mentioned $FR(x)$, $AFL(x)$, $A(x)$ and $|dom(x)|$ as scores to design the new heuristics. Given a variable x causing the last failure, we discuss how these scores change after the last failure detected.

- $FR(x)$: $FR(x)$ increases because $\frac{failNum(x)+1}{assignNum(x)+1}$ is always larger than $\frac{failNum(x)}{assignNum(x)}$.
- $AFL(x)$: $AFL(x)$ is increased by $\frac{1}{|failure(x)|}$.
- $A(x)$: If x causes the last failure, then $\#TotalFailure$ equals to $LastFailure(x)$. $A(x)$ will be the largest one.
- $|dom(x)|$: After the failure is detected, the value of the assignment of x will be removed from $dom(x)$, so $|dom(x)|$ decreases.

From the analysis, we can see if x causes the last failure, then all the four scoring function give x a score better than that before the failure. Its score was the best one, so the better score has a large chance to be selected. Thus, both the failure rate based VOHs and the failure length based VOHs tend to select the variable causing the last failure.

In binary branching strategy, a failure of a left branch is followed by a propagation of a refutation. Although the propagation of the refutation does not affect $FR(x)$ and $AFL(x)$, it may reduce the domains of other variables, so some other variables may have much smaller domain sizes. In this case, the variable x gets a better score after the backtracking, but it may not be the best one due to the propagation of the refutation. In addition, if the propagation of the refutation leads to a failure, the search will backtrack to a higher level, then some fixed variables with a score better than that of x may become available for branching. Thus, the failure based VOHs do not strictly select the variable causing the last failure.

5 Experiments

The experiments were run in Choco 4.10.6 [14]. The environment is JDK8 under CentOS 6.4 with Intel Xeon CPU E7-4820@2.00GHz processor and 58 GB RAM.

To examine the robustness of the proposed VOHs, we tested the VOHs with MiniZinc benchmark from `https://github.com/MiniZinc/MiniZinc-benchmarks`. Solving a problem with different modelling may get different performances, so we tested all the MiniZinc models of CSPs. The instances are flattened offline. After eliminating some large instances which cannot be flattened in 1 hour and the problems where infeasibility is proved at the root node, we include 41 problems with 1876 instances of 46 MiniZinc models in the experiments.

The performance of searching for the first solution or proving unsatisfiable are measured by CPU time in seconds. Timeout is set to 1200 seconds. We have used a random seed 0 throughout the experiments. Besides the naive version of the VOHs, we compare these VOHs equiped with a geometric restart strategy [4, 17] and last conflict based reasoning

strategies [9] storing one (LC-1) and five (LC-5) conflict variables respectively. The restart strategy uses 10 as the initial cutoff of failure count and 1.1 as the growing factor. The ABS uses its default value selector and all other VOHs use lexicographic ordering as the value selector.

In the following tables, we present the number of instances solved (*#solved*) by each VOH, the accumulated CPU time of each VOH solving the instances solved by all the compared VOHs (all solved time, *ast*) and the total CPU time of each VOH solving the instances solved by at least one of the compared VOHs (total time, *tt*). The integer in the brackets after *ast* is the number of all solved instances, so is the one after *tt*. The time cost of a timeout run is count as 1200s and we eliminated the results of the instances where all compared VOHs are timeout. The best one in each row is in bold.

In Table 1, we compare the failure based VOHs. We have three observations from the table. Firstly, FRBA performs better than FRB and FLBA performs better than FLB, so the decaying strategy improves the original failure based VOHs. Secondly, FLBA performs best when the restart strategy is not equiped, and FRBA performs best when the restart strategy is equiped. This is because the effective FR scores may be quickly learned after several restarts, so FRBA could make good decisions after some restarts. Finally, the VOHs get better performance when equiped with last conflict based reasoning. In general, the FRBA VOH solves the largest number of instances, so we use it as the representative one of the failure based VOHs in the following experiments.

Table 1 Comparing the failure based VOHs.

			FRB	FRBA	FLB	FLBA
no restart	LC-0	*#solved*	702	711	818	**870**
		ast(596)	9,401	9,344	18,740	**8,140**
		tt(972)	348,961	340,532	224,463	**162,034**
	LC-1	*#solved*	738	741	828	**879**
		ast(618)	**11,574**	12,558	22,436	15,572
		tt(980)	320,280	325,257	222,613	**168,985**
	LC-5	*#solved*	767	763	862	**894**
		ast(638)	15,494	16,119	15,777	**12,316**
		tt(1,012)	339,881	344,336	217,003	**184,585**
restart	LC-0	*#solved*	949	985	984	**1000**
		ast(864)	21,357	**11,730**	19,218	15,504
		tt(1,087)	201,929	162,862	160,376	**145,543**
	LC-1	*#solved*	1,024	**1,037**	969	992
		ast(911)	14,895	**11,813**	17,361	17,014
		tt(1,094)	132,819	**119,706**	180,152	165,624
	LC-5	*#solved*	1,027	**1,037**	958	973
		ast(910)	14,676	**13,990**	21,305	29,482
		tt(1,097)	128,496	**119,595**	200,505	200,291

In Table 2, we compare FRBA with the state of the art VOHs. The ABS, CHS and $dom/wdeg^{ca.cd}$ have been implemented in Choco and we implemented the CRBS and the proposed VOHs[2]. We use the default parameters for these VOHs, which are recommended in

[2] The source code is available at `https://github.com/lihb905/fbs/`.

the literatures [12, 5, 20, 18]. For each MiniZinc model, we calculate the rate of the number of instances solved by each VOH to the number of instances solved by at least one VOH. The aggregated 46 rates are shown in the *solvedRates* rows. The results show that, when the restart strategy is equiped, FRBA gets the best performance. When the restart strategy is not equiped, FRBA is competitive with the existing ones and the CHS with LC-5 solves more instances than the other VOHs. The FRBA equiped with the restart strategy and last conflict based reasoning solves instances of largest number and its total time cost is less than that of the existing ones. The FRBA has the largest aggregated solved rate in most of the rows.

Table 2 Comparing the failure based VOHs with the existing VOHs.

			ABS	CHS	$dom/wdeg^{ca.cd}$	CRBS	FRBA
no restart	LC-0	*#solved*	530	665	596	616	**711**
		ast(415)	7,104	6,306	16,860	13,007	**5,603**
		tt(801)	350,965	194,178	274,969	250,625	**135,332**
		solvedRates	36.65	35.96	32.31	34.4	**37.47**
	LC-1	*#solved*	557	744	647	**770**	741
		ast(427)	10,073	5,362	11,038	11,859	**3,964**
		tt(961)	528,055	303,136	415,091	**274,381**	302,457
		solvedRates	38.42	36.92	34.09	35.32	**38.50**
	LC-5	*#solved*	583	**796**	704	790	763
		ast(464)	10,279	9,213	18,073	17,000	**6,998**
		tt(998)	530,479	**287,299**	391,992	291,255	327,536
		solvedRates	**39.90**	38.88	36.19	35.88	38.73
restart	LC-0	*#solved*	630	730	700	864	**985**
		ast(443)	9,687	9,276	22,666	18,829	**8,360**
		tt(1,106)	618,393	492,387	536,574	356,115	**185,662**
		solvedRates	39.82	38.49	33.39	36.88	**40.48**
	LC-1	*#solved*	655	905	883	911	**1,037**
		ast(474)	13,022	8,552	14,409	18,480	**8,493**
		tt(1,134)	630,983	362,924	349,635	320,169	**167,706**
		solvedRates	40.37	39.74	36.22	37.47	**41.32**
	LC-5	*#solved*	674	960	929	879	**1,037**
		ast(475)	9,618	**5,810**	17,482	20,766	7,935
		tt(1,162)	661,177	300,729	331,411	397,650	**197,595**
		solvedRates	40.33	40.47	37.37	37.45	**40.55**

It has been shown that the VOHs equiped with restart and LC-5 is the best strategy in general, so we present the detailed results of the strategy in Table 3. The table includes all the 46 MiniZinc models of 41 problems. The integer in the brackets after each problem name is the total number of instances of the problem. In each cell, we present the number of solved instances and the number in the brackets is the total time cost of the instances solved by at least one VOH. The last row shows the numbers of problems where the corresponding VOH performs best. To decide which VOH performs best in a problem, we give a rule that considers the VOH solving instances of largest number as the best one. If a tie exists, we further compare the total time cost. It is shown that, the VOHs get best performance in different problems. Both ABS and FRBA get best performance in 13 problems, which is the largest number.

Table 3 Detailed results of all problems.

Problems	ABS	CHS	$dom/wdeg^{ca.cd}$	CRBS	FRBA
alpha(1)	1(0.13)	**1(0.11)**	1(0.17)	1(0.31)	1(0.35)
amaze2(47)	**10(1,117)**	7(4,820)	4(7,693)	7(4,034)	7(4,102)
amaze3(47)	**45(391)**	44(1,242)	44(2,393)	43(2,727)	43(3,038)
areas(4)	4(0.21)	**4(0.06)**	4(0.1)	4(0.08)	4(0.07)
bibd(16)	15(805)	13(2,509)	9(7,617)	15(127)	**15(15)**
black-hole(21)	**21(24)**	21(51)	21(28)	21(72)	20(1,213)
carseq(79)	**23(7,795)**	0(27,600)	0(27,600)	0(27,600)	0(27,600)
cars(79)	**37(17,480)**	10(46,048)	14(42,313)	5(54,652)	18(37,850)
CostasArray(10)	8(1,880)	7(2,683)	8(1,642)	8(1,769)	**9(1,270)**
step1-aes(7)	2(2,417)	3(1,882)	3(1,739)	**3(1,658)**	2(2,408)
debruijn-binary(11)	4(3,678)	7(1,088)	7(1,094)	7(1,180)	**7(1,013)**
elitserien-noseasonal(10)	10(184)	10(56)	**10(26)**	10(113)	10(48)
eq20(1)	1(0.05)	1(0.17)	1(0.09)	1(0.12)	**1(0.04)**
fillomino(20)	**17(170)**	16(2,372)	16(3,976)	16(2,970)	16(2,983)
golfers1(9)	6(794)	**6(117)**	6(1,089)	6(259)	6(142)
golfers1b(9)	6(223)	**6(15)**	6(20)	6(47)	6(31)
golfers2(9)	5(2,347)	**5(1,338)**	2(4,820)	2(5,876)	3(3,775)
kakuro(6)	6(0.18)	6(0.15)	6(0.12)	**6(0.11)**	6(1.38)
knights(4)	4(2.18)	4(0.61)	**4(0.53)**	4(1.13)	4(6.79)
langford(25)	20(119)	**20(87)**	20(88)	20(112)	20(616)
latin-squares-fd(7)	**7(24)**	6(1,291)	5(2,594)	4(3,893)	5(3,640)
latin-squares-fd2(7)	7(1.66)	**7(0.91)**	7(0.97)	7(1.90)	7(10)
latin-squares-lp(7)	**7(161)**	7(1,294)	4(3,626)	4(3,684)	6(1,909)
magicseq(9)	7(2,503)	9(386)	**9(191)**	7(2,408)	7(2,468)
market-split(60)	**39(9,858)**	33(10,567)	30(15,614)	33(10,749)	36(11,169)
mknapsack(7)	**7(2,035)**	5(3,299)	2(6,717)	4(3,924)	5(3,388)
nmseq(20)	9(6,293)	13(3,687)	11(5,407)	11(6,266)	**14(2,290)**
non(26)	21(11,151)	23(4,783)	15(17,988)	15(16,631)	**24(3,847)**
nsp-1(200)	56(95,939)	71(66,452)	55(78,245)	18(130,871)	**114(10,801)**
nsp-2(200)	6(18,424)	0(22,800)	6(19,884)	0(22,800)	**13(12,459)**
oocsp-racks(6)	6(898)	6(46)	**6(29)**	6(86)	6(45)
pentominoes-int(7)	7(252)	7(96)	7(181)	**7(41)**	7(193)
QCP(60)	**60(4,385)**	59(5,488)	55(8,914)	51(13,722)	53(11,441)
quasigroup7(10)	5(16)	**5(11)**	5(14)	5(38)	5(40)
queens(7)	**7(1.11)**	7(129)	7(108)	7(4.10)	7(1.49)
rect-packing(56)	56(56)	56(12)	56(12)	**56(10.02)**	56(10.24)
rect-packing-mznc2014(56)	56(54)	56(13)	**56(9.16)**	56(12)	56(10)
rubik(5)	4(2,298)	**5(541)**	1(4,826)	3(2,771)	3(2,458)
schur(3)	3(0.07)	**3(0.05)**	3(0.08)	3(0.06)	3(0.06)
search-stress2(1)	1(0.63)	1(0.59)	1(0.56)	**1(0.46)**	1(0.48)
search-stress(3)	2(3.00)	2(3.97)	2(40)	2(3.30)	**2(2.91)**
slow-convergence(10)	10(8.57)	10(9.39)	10(519)	10(355)	**10(8.30)**
solbat(39)	**36(6,491)**	32(10,813)	28(18,975)	24(22,157)	26(19,404)
tents(3)	3(0.33)	3(0.19)	3(0.24)	3(0.08)	**3(0.07)**
wwtpp-random(251)	0(168,000)	125(20,864)	133(11,650)	134(13,203)	**138(5,298)**
wwtpp-real(401)	7(292,879)	218(56,216)	226(33,711)	223(40,806)	**232(20,576)**
Sum of bests	**13**	10	5	5	**13**

In Table 4, we compare each pair of the VOHs according to the previous rule. The number of problems where each VOH performs better is present in the table. We can see that FRBA performs better than the others in general.

Table 4 Comparing FRBA with the existing VOHs by pairs.

		ABS	FRBA	CHS	FRBA	$dom/wdeg^{ca.cd}$	FRBA	CRBS	FRBA
no restart	LC-0	21	**25**	20	**26**	17	**29**	19	**27**
	LC-1	22	**24**	21	**25**	13	**33**	15	**31**
	LC-5	**26**	20	**31**	15	21	**25**	16	**30**
restart	LC-0	**24**	22	**24**	22	13	**33**	18	**28**
	LC-1	20	**26**	22	**24**	13	**33**	9	**37**
	LC-5	21	**25**	**23**	**23**	18	**28**	15	**31**

The numbers of instances of each MiniZinc model vary greatly. If a VOH works well in some models containing a large number of instances, it may get a good overall performance. To balance the effect of instance set size, we randomly select $\frac{1876}{46} = 41$ instances from the instance set of each MiniZinc model to generate a smaller benchmark set (46 is the number of MiniZinc models). If an instance set contains less than 41 instances, we select them all. The smaller benchmark set contains 832 instances. The results are present in Table 5. We can see that, when the restart strategy is not equiped, FRBA clearly outperforms the others. When the restart strategy is equiped, FRBA is competitive with the existing ones. In general, FRBA gets the best performance in the smaller benchmark set.

Table 5 Comparing FRBA with the existing VOHs in the smaller benchmark set.

			ABS	CHS	$dom/wdeg^{ca.cd}$	CRBS	FRBA
no restart	LC-0	*#solved*	451	455	420	430	**478**
		ast(351)	6,504	5,476	15,569	11,961	**4,630**
		tt(538)	127,769	119,427	167,554	150,743	**89,434**
	LC-1	*#solved*	461	467	431	454	**490**
		ast(360)	9,579	4,594	9,509	9,833	**3,219**
		tt(552)	142,492	124,518	175,928	143,740	**92,591**
	LC-5	*#solved*	484	491	466	461	**497**
		ast(391)	8,600	7,692	16,355	15,180	**6,156**
		tt(565)	121,385	110,707	148,158	148,961	**104,365**
restart	LC-0	*#solved*	505	488	427	477	**521**
		ast(371)	**7,116**	8,002	22,176	12,087	7,574
		tt(598)	142,153	161,329	239,873	177,064	**120,197**
	LC-1	*#solved*	508	510	470	482	**527**
		ast(389)	10,017	**5,897**	13,847	11,099	8,068
		tt(590)	133,624	126,346	169,283	155,500	**106,790**
	LC-5	*#solved*	514	525	489	480	**533**
		ast(400)	6,316	**3,890**	16,542	12,876	7,399
		tt(608)	150,353	124,973	176,728	181,416	**122,106**

6 Conclusion

In this paper, we propose failure based variable ordering heuristics for solving CSPs. The new VOHs consider failure rate and failure length to estimate how likely an assignment of a variable causing a failure. All the heuristic information are collected during search, so the new heuristics are parameter-free. The experiments in the MiniZinc benchmark set show that the failure based VOHs outperform the state of the art VOHs in general. They can be new candidates of general purpose variable ordering heuristics for black-box CSP solvers.

References

1 C. Bessière and J. C. Règin. Mac and combined heuristics: two reasons to forsake fc (and cbj?) on hard problems. In *Proc. CP'96*, pages 61–75. Springer, 1996.

2 F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proc. ECAI'04*, pages 146–150, 2004.

3 I. P. Gent, E. MacIntyre, P. Prosser, B. M. Smith, and T. Walsh. An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem. In *Proc. CP'96*, pages 179–193. Springer, 1996.

4 C. P. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *Proc. AAAI'98*, pages 431–437. AAAI, 1998.

5 D. Habet and C. Terrioux. Conflict history based search for constraint satisfaction problem. In *Proc. of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1117–1122. ACM, 2019.

6 R. Haralick and G. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.

7 E. Hebrard and M. Siala. Explanation-based weighted degree. In *Proc. CPAIOR'17*, pages 167–175. Springer, 2017.

8 J. Hwang and D. G. Mitchell. 2-way vs d-way branching for csp. In *Proc. CP'05*, pages 343–357. Springer, 2005.

9 C. Lecoutre, L. Saïs, S. Tabary, and V. Vidal. Reasoning from last conflict(s) in constraint programming. *Artificial Intelligence*, 173(18):1592–1614, 2009.

10 H. Li, Y. Liang, N. Zhang, J. Guo, D. Xu, and Z. Li. Improving degree-based variable ordering heuristics for solving constraint satisfaction problems. *Journal of Heuristics*, 22(2):125–145, 2016.

11 P. Liberatore. On the complexity of choosing the branching literal in dpll. *Artificial Intelligence*, 116(1):315–326, 2000.

12 L. Michel and P. Van Hentenryck. Activity-based search for black-box constraint programming solvers. In *Proc. CPAIOR'12*, pages 228–243. Springer, 2012.

13 G. Pesant, C. G. Quimper, and A. Zanarini. Counting-based search: Branching heuristics for constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 43:173–210, 2012.

14 C. Prud'homme, J-G. Fages, and X. Lorca. *Choco Documentation*. TASC - LS2N CNRS UMR 6241, COSLING S.A.S., 2017. URL: `http://www.choco-solver.org`.

15 P. Refalo. Impact-based search strategies for constraint programming. In *Proc. CP'04*, pages 557–571. Springer, 2004.

16 B. M. Smith and S. A. Grant. Trying harder to fail first. In *Proc. ECAI'98*, pages 249–253, 1998.

17 T. Walsh. Search in a small world. In *Proc. IJCAI'99*, pages 1172–1177, 1999.

18 R. Wang, W. Xia, and R. H. C. Yap. Correlation heuristics for constraint programming. In *Proc. ICTAI'17*, pages 1037–1041. IEEE, 2017.

19 H. Wattez, F. Koriche, C. Lecoutre, A. Paparrizou, and S. Tabary. Learning variable ordering heuristics with multi-armed bandits and restarts. In *Proc. ECAI'20*, pages 371–378. IOS Press, 2020.

20 H. Wattez, C. Lecoutre, A. Paparrizou, and S. Tabary. Refining constraint weighting. In *Proc. of ICTAI'19*, pages 71–77. IEEE, 2019.

21 W. Xia and R. H. C. Yap. Learning robust search strategies using a bandit-based approach. In *Proc. AAAI'18*, pages 6657–6665. AAAI, 2018.