Guiding Search by Singleton-Domain Variables for Constraint Satisfaction Problem

Yi Jun Liu*
Jilin University ,Changchun, Jilin Province, China.

* Corresponding author. Tel.: +8618604366655; email: 308718494@qq.com
Manuscript submitted May 26, 2016; accepted July 1, 2016.
doi: 10.17706/jsw.11.9.903-911

Abstract: Heuristic is an important subject in the field of constraint satisfaction problem(CSP). In the study, we found a kind of variables named *singleton-domain variables* have a great influence on the solution process. Based on this, we propose a heuristic method named *dom/wsdeg* and the corresponding calculating method and combine the method with a constraint network compression technology proposed in this paper. The results have shown that the new heuristics are more efficient than the classical heuristics.

Key words: Constraint satisfaction problem, compression method, singleton-domain variable, variable ordering heuristic.

1. Introduction

Constraint satisfaction problem(CSP)[1] is an important branch in the field of artificial intelligence, many practical problems can be modeled by constraint satisfied problem, such as the n-queens, job scheduling, figure coloring problem and symbolic reasoning. The existing algorithms mainly include three kinds of algorithms: backtrack search algorithm, local search method and dynamic programming [2]. Backtrack search algorithm with constraint propagation technique is applied widely at present. But because of the larger scale of problems and the need to constantly instantiate the variables and values in the process," Combination Explosion" phenomenon [2] often appears which makes the problem difficult to get an effective solution within the effective time. So how to improve the efficiency of the backtracking is one of the most important problem in the field of CSP.

In the process of solving CSP, backtrack search algorithm performs a depth-first traversal of a search tree and variables and values are selected continuously to instantiate. Experimental results show that the instantiation order of variables plays a vital role in efficiently solving CSP[2].The purpose of the heuristic algorithm is to get a better variables instantiation order. The thought of the heuristic also has important applications in many fields such as machine learning [3]. Many successful variable ordering heuristics have been proposed but there is no general heuristic method for value ordering.

When using the Maintaining Arc Consistency(MAC) algorithm to check each node in the tree, we need to maintaining arc-consistency in the current constraint network and incompatible values are not arc-consistent and can be safely deleted without losing any solutions. Deletion of inconsistent values is called filtering of the domains. We find a kind of variables whose domains have been cut down to only one value in this process. This kind of the variables is called singleton-domain variable. We find that the removal of these singleton-domain variables does not lose any solution of the problem and will improve the efficiency of CSP because of the close relationship between singleton-domain variables and the instantiated variable in the current constraint network.Based on this,we propose a new weighted-degree heuristic and a constraint network compression technology.We performed a number of experiments on a series of benchmark instances.The results show that the new heuristics are more efficient than classical ones especially for the problem with large data size and complex structure .

2. Background

Definition 1. Constraint Satisfaction Problem(CSP). A constraint satisfaction problem is a triple $P=(X,D,C)$,where $X=\{x_1,x_2,\cdots\cdots,x_n\}$ is a finite set of variable; $D=\{dom(x_1),dom(x_2),\cdots\cdots,dom(x_n)\}$ is a finite set of domains, where for $\forall x_i\in X$, $dom(x_i)\in D$ is a finite set of possible values for each of variable x_i ; $C=\{c_1,c_2,\cdots\cdots,c_k\}$ is a finite set of constraints (relations) over subsets of X . Each constraint c consists of two parts, an ordered set of variables $scp(c)=\{x_{i1},x_{i2},\cdots\cdots,x_{ir}\}$ and a subset of Cartesian product $dom(x_{i1})\times dom(x_{i2})\times\cdots\times dom(x_{ir})$ that specifies the disallowed (or allowed) combinations of values for the variables $\{x_{i1},x_{i2}\ldots x_{ir}\}$ .The arity of a constraint c is the number of variables involved in c ,i.e.$|scp(c)|$ .A constraint is binary if its arity is 2;non-binary if its arity is strictly greater than2. A non-binary constraint is usually transformed into a binary constraint in solving process, so we discuss the CSP of binary constraints in this paper.

A solution to a CSP is an assignment of a value to each variable,if $\forall x_i\in X$ can find $x_i=a_i$, $a_i\in dom(x_i)$,tuple $T=\{a_1,a_2,\cdots\cdots,a_n\}$ .If there is a T satisfies all the constraints, T is a solution of the CSP and the CSP is satisfiable. If there is no T satisfies all the constraints, the CSP is unsatisfiable.

Definition 2. Generalized Arc Consistency (GAC) [4].

- A constraint c is generalized arc-inconsistent,or GAC-consistent iff $\forall x\in scp(c)$, $\forall a\in dom(x)$,there exists a support for (x,a) on c .
- A constraint network is P generalized arc-inconsistent iff every constraint of P is generalized arc-inconsistent.

For binary constraints, this property is classically known as *arc consistency*(AC).

Maintaining arc consistency(MAC) is certainly the most popular systematic search algorithm for solving instances of the constraint satisfaction problem. After each branching decision, enforcement of some kind of local consistency prunes some parts of the search space that contain no solution. Dynamic variable heuristic is to select the appropriate variable to instantiate and MAC enforces GAC after each decision taken.

3. Integrating Singleton-Domain Variables Information in a Conflict-Based Heuristic

According to whether the instantiation order of the variables can be changed in the solving process, the heuristic algorithm is divided into two types. One is that the instantiation order of variables is determined before the start of search and cannot be changed in the solving process, which is called static heuristic. The other is that the instantiation order of variables is determined or constantly changing in the process of the search, which is called dynamic heuristic. At present, dynamic heuristic is the most widely used in the practical application. Many successful variable ordering heuristics have been proposed, such as *dom*[5] heuristic based on variable domain size, *dom/ddeg*[6] heuristic which combines the number of constraints related to the variables with *dom* heuristic, *dom/wdeg*[6] heuristic which combines weighted degree with the *dom* heuristic.

Dom/wdeg is the most widely used based on the underlying fail-first principle(*FFP*)[6].According to *FFP*, the variables with smaller current domain sizes and the variables with high weighted degree(*wdeg*) should be instantiated first. The *dom/wdeg* heuristic computers a score for each variable and orders the variables according to their scores. It associates a weight(c)(initialized to 1) for each constraint c,when the domain of a variable becomes empty (so-called *domain wipe-out*),the weight of constraint that leads to the wipe-out is incremented. The weighted degree of a variable xi is defined as:

$$wdeg(X_i) = \sum_{R_{ij} \in C} weight(R_{ij}) \middle| X_j \in FutVars$$

Combining the weighted degree with domain size, the *dom/wdeg* heuristic computes the ordering score for each variable x_i by the ratio of the current domain size to wdeg(x_i),denoted by $\frac{|dom(x_i)|}{wsdeg(x_i)}$,and selects the variable with the smallest score.

Dom/wdeg heuristic has a certain learning ability that can give higher weights to the constraints difficult to handle with. So in most cases, *dom/wdeg* can find hard part of a CSP, and select the variables which have a higher probability of belonging to this region. When the scale of a CSP is complex and the structure is complex,*dom/wdeg* selects variables located in the hard part because of *FFP*, and variables related to the variables are usually located in the same region. As a result, conflict-based heuristic only collects information in this local area. This process may cause too much dead nodes and reduce efficiency.

At each step of backtrack search, MAC enforces GAC to reduce domains inferentially after each decision taken. In the process, the domains of some variables have been cut down to only one value.This kind of variables is called singleton-domain variable and defined as follow:

Definition 3. Singleton-Domain Variable. A variable x_i is a singleton-domain variable iff $|dom(x_i)| = 1 | x_i \in FutVars$.

Definition 4. The number of new singleton-domain variables after GAC.After x_i instantiated,MAC enforces GAC to reduce domain. The number of singleton-domain variable generated in this process is recorded $varsin(x_i)$.

In the study,we find that singleton-domain variables have an special guiding role to solve CSP. So we integrate the singleton-domain variable information in *dom/wdeg* heuristic. The directest strategy is to combine the $varsin(x_i)$ with the degree-based heuristic, so we proposed a new degree named weighted singleton-domain variable degree(*wsdeg*).The *wsdeg* of a variable x_i is defined as:

$$wsdeg(x_i) = varsin(x_i) + \sum_{c \in C} weight(c) \middle| x_i \in scp(c) \wedge futScp(c) > 1$$

Combining the *wsdeg* with domain size, the *dom/wsdeg* heuristic computes the ordering score for each variable x_i by the ratio of the current domain size to $wsdeg(x_i)$,denoted by $\frac{|dom(x_i)|}{wsdeg(x_i)}$ and selects the variable with the smallest score. We performed some experiments on a series of benchmark instances to compare the efficiency of the new heuristics and the classical conflict-based heuristic and the results and analysis are shown in Sect.5.

4. A Compression Method of the Constraint Network

In the process of solving CSP, backtrack search algorithm performs a depth-first traversal and with the increase of the number of instantiated variables and the decrease of the domains of uninstantiated variables, the constraint network is simplified. Constraint network can improve the efficiency of solving CSP through simplification, such as *filtering of the domain*, reduction of the record items [8], etc..With the deepening of the search tree depth, the number of singleton-domain variables is also increased. Classical conflict-based heuristics did not consider about the decline of efficiency caused by singleton-domain variables. This leads to useless computation, useless records, etc., even branching on the singleton-domain variables due to weighted degree increase and reduces the efficiency of recording and recovery. The deletion of the singleton-domain variables can not only have no affect on the satisfaction of CSP [8], but also simplify the constraint network improve the efficiency.

A stable heuristic method named *MAC_singleHalf* is proposed in the paper [9].In the solving process,when calculating the *dom/wdeg* of an uninstantiated variable considers the weights of constraints between new generated singleton-domain variables an the variable, ignores all previous related singleton-domain variables and uses weight/2 as *wdeg* to calculate. Based on the [9], this paper proposes a new compression method of the constraint network named *MAC_Com*. Different from *MAC_singleHalf*, *MAC_Com* considers both new generated and previous singleton-domain variables, but the weights of constraints between previous related singleton-domain variables and uninstantiated variables cannot change any more.

Mac_Com method can reserve the affect of constraints between singleton-domain variables and uninstantiated variables and prevent that multiple backtracks make some weights of constraints too high which may cause the constantly strengthened impact of singleton-domain variables.We performed some experiments on a series of benchmark instances to compare the efficiency of *MAC_singleHalf* and *Mac_Com* method and the results and analysis are shown in Sect.5.

5. Experiments

We performed some experiments on a series of benchmark instances to examine the performances of the new heuristics. Benchmark is currently the international general standard use cases to test the solving efficiency of CSP include a variety of academic issues and a variety of typical practical use cases. In this paper, the XML description documents of the test cases can be downloaded online from [10].Experiments use MAC3rm[4] as a solution algorithm and MAC3rm is the most efficient solving algorithm in the existing MAC algorithms. Among the existing heuristics, the *dom/wdeg* heuristic is the most widely used and works best. Therefore, we compare the *dom/wdeg* with the new heuristics. he performances of finding the first solution or proving unsatisfiable are measured in terms of CPU time marked t in millisecond(ms),the number of explored search tree nodes marked n and the number of constraint checks marked cc .In order to improve the efficiency of the experiments, in the pre-processing phase, an arc-consistency check will be performed in the constraint network first to delete incompatible values and the number of constraint checks will not be included in the results.

5.1. MAC_Com versus MAC_singleHalf

In Section 4, this paper analyses and illustrates the *MAC_singleHalf* method and the compression method of the constraint network named *MAC_Com* proposed in this paper. In this section, We compared four kinds of solving strategy, including classic *dom/wdeg* heuristic, *MAC_singleHalf* proposed in paper [9], *MAC_com* proposed in this paper and a heuristic named S for comparison. S deletes all of singleton-domain variables in constraint network in order to avoid branching in the singleton-domain variables.While calculating the weighted degree of a candidate variable, S ignores all of singleton-domain variables in the network.We selected a variety of benchmark instances to examine the performances of the four solving strategy and selected representative results shown in Table 1.

Table 1. Results of Binary Instances

instance		*dom/wdeg*	S	*MAC_singleHalf*	*MAC_Com*
BH-4-4 （10）	t	**24518**	26280	27240	25884
	n	6293	**5039**	**5039**	**5039**
	cc	**3695465**	3738483	3923677	3927926
Lard （10）	t	31041	47776	466418	**25836**
	n	4946	3770	28422	**3245**
	cc	3018439	2979794	23519855	**2630105**
geom （100）	t	1845	1046	655	**429**
	n	119	103	103	**102**
	cc	43384	43680	43828	**43003**
composed-25-10-20 (10)	t	393	353	**326**	360
	n	154	143	**142**	156
	cc	10499	**10692**	10695	12355
e0ddr1-10-by-5-1-donum49-D 145-v50-r265-c265	t	798620	2408201	2470726	**1057008**
	n	142082	442351	425126	**192774**
	cc	60592317	193231171	186840613	**86949726**
ehi-85-297-5_ext	t	17467	**10453**	21932	36189
	n	363	**87**	239	247
	cc	210380	**215318**	557500	725257
composed-75-1-80-1_ext	t	2276	2115	2626	**1349**
	n	283	240	**240**	256
	cc	53064	52578	52578	**2678**
composed-75-1-80-9_ext	t	1418	1872	2520	**1310**
	n	241	238	238	**206**
	cc	**44808**	55060	55060	47156
driverlogw-01c-sat_ext	t	68	82	118	**56**
	n	73	73	73	**71**
	cc	**338**	393	393	416
driverlogw-05c-sat_ext	t	34378	10840	10504	**7281**
	n	991	422	422	**370**
	cc	79697	79908	79586	**33101**
driverlogw-09-sat_ext	t	1152067	656557	587656	**515943**
	n	10783	2099	2024	**2127**
	cc	**1681885**	2025222	1907892	1736249
frb50-23-1_ext	t	1030	1061	1081	**1105**
	n	179	130	130	**129**
	cc	59531	60584	60278	**58884**

The first four test results shown in Table 1 are average values of a number of test instances from four kinds of benchmark problem and after eight are the representative test results selected from other benchmark problem for analysis.

Through the experimental results shown in Table 1, we can see that S, *MAC_Com* and *MAC_singleHalf* are more efficient than the *dom/wdeg* heuristic in most cases. In the *BH-4-4* problem, although the number of search tree nodes is maximum, the *dom/wdeg* heuristic still costs least time to solve the problem because of the minimum number of constraint checks. In the *driverlogw-09-sat_ext* problem, although the number of constraint checks of *MAC_Com* is bigger than the number of *dom/wdeg* heuristic, the efficiency of solving problem by *MAC_Com* method is still the highest because the number of search tree nodes is reduced by 5 times. Through the analysis of the above two examples and combining with the data in Table 1, we can come to the conclusion preliminarily that increase in the efficiency of the CPU is co-determined by reducing the number of search tree nodes and the number of constraint checks and any single factor can be a decisive factor. At the same time, we can also find that when the number is same, the effect of reducing of tree nodes to the improve efficiency is far greater than the effect of reducing the number of constraint checks.In *Ehi-85-297-5_ext*,efficiency of S heuristic is the highest.Through the analysis of the structure of the instance and variables instantiation order, we found that the singleton-domain variables are most in the complex region, and simply avoid branching in the singleton-domain variables and delete related constraints can greatly reduce the complexity of the complex region.Through the analysis of the four heuristic methods and the results shown in Table 1, we can confirm that the *MAC_Com* method is more efficient and the stabler in most of the problems.

5.2. New Heuristics versus Classical Conflict-Based Heuristic

Dom/wsdeg heuristic uses $\frac{|dom(x_i)|}{wsdeg(x_i)}$ as the score of variable x_i, and selects the variable with the minimum score to give priority to instantiate. *Dom/wsdeg2* heuristic combines the *dom/wsdeg* heuristic and the constraint network compression technique-*MAC_Com* proposed in this paper . We selected a variety of benchmark instances to examine the performances of the new heuristics and selected representative results shown in Table 2.

Table 2. Results of Binary Instances

instance		*dom/wdeg*	*dom/wsdeg*	*dom/wsdeg2*
geom（100）	t	1845	**131**	93
	n	119	**53**	**53**
	cc	43384	**9248**	9326
BH-4-4（10）	t	24325	23066	**19844**
	n	6293	6116	**5045**
	cc	**3695465**	4102430	4115515
driver（10）	t	48811	33011	**25425**
	n	1569	991	**432**
	cc	102488	**79697**	92441
rand-2-23（10）	t	3426406	458818	**316642**
	n	497584	75095	**61812**
	cc	257815659	38389742	**37544872**
graph（5）	t	2219	1687	**1547**
	n	**300**	**300**	**300**
	cc	85955	86628	**85032**
frb50（10）	t	5469	5628	**4734**
	n	789	791	**521**
	cc	268243	270038	**175404**
frb59（10）	t	82631	146085	**45439**
	n	9624	13445	**5455**
	cc	4377886	7576879	**4225938**
frb35	t	31221	28374	**28073**
	n	4946	4106	**3722**
	cc	3018439	**2528983**	2910084
e0ddr1-10-by-5-1-donum49-D145-v50-r265-c265	t	717331	445115	**257891**
	n	142082	85055	**63255**
	cc	60592317	37559823	**28628358**
geo50.20.d4.75.1-donum1-D20-v50-r472-c472	t	2208102	1779445	**1665938**
	n	214052	169411	**139117**
	cc	249244749	198445851	**198321376**
geo50-20-d4-75-1_ext	t	2289077	1728936	**1436276**
	n	214052	169411	**139117**
	cc	249244729	198445851	**198321376**

The first six test results shown in Table 1 are average values of a number of test instances from six kinds of benchmark problem and after four are the representative test results selected from other benchmark problem for analysis.

As can be seen from Table 2, for most of the instances, the efficiency of *dom/wsdeg* and *dom/wsdeg2* heuristic is both higher than the efficiency of *dom/wdeg* heuristic. Combining Table 1 and Table 2 to analysis, results indicate that in most of the problems the efficiency of the single use of *dom/wsdeg* or method is lower than the efficiency of *dom/wsdeg2* heuristic which combines the advantages of the above two. Furthermore, both of these heuristics have improved the efficiency of the classical heuristic in varying degrees. In *rand-2-23* problem ,the solving efficiency of the new heuristics has improved ten times. The results imply that there are a large amount of singleton-domain variables generated in the solving process. In the *graph* problem, three heuristic methods generate the same number of nodes and the number of constraint checks is almost the same. Through the analysis of the structure of the instances and variables instantiation order, we find that the number of singleton-domain variables generated is small and this kind of variables is located in the marginal region , so they did not play a guiding role in this problem.

In some instances, the number of constraint checks of *dom/wdeg* and *dom/wsdeg* is less than *dom/wsdeg2*.For example, in the *driver* problem,*dom/wsdeg* heuristic performed minimum constraint checks and the generated twice as many nodes as *dom/wsdeg2* did.Because backtracking whenever a dead-end occurs and the CPU time cost by a backtracking is more than that of a constraint check. So in this problem *dom/wsdeg2* heuristic is still more efficient than *dom/wsdeg* heuristic.

As for instances of large data size and complex structure, the efficiency of new heuristics is obviously improved compared with the classic one, such as *rand-2-23*,time efficiency enhanced in multiples,search tree nodes exponentially reduced.

Through analysis on the retrieval process and the experimental results, we can find that due to the complexity of the structure, *dom/wdeg* always chooses variables located in the hard part to instantiate because of FFP, and variables related to the variables are usually located in the same region. As a result, conflict-based heuristic only collects information in the local area and cannot correctly jump away from sub-network. This process may cause too much dead nodes generated and reduce efficiency. New heuristics based on the singleton-domain variables can increase the weights of the boundary variables in the process of backtracking and constraint propagation can reach multiple regions, avoid constraint propagation confined to a part of the problem, accumulate the heuristic information in a broader space.

Overall, in solving process. if there is no or only a very small amount of singleton-domain variables, the efficiency of new heuristics and the classical heuristic is about the same; when the scale of the data is big or a large number of singleton-domain variables generated, the efficiency of new heuristics will be greatly enhanced compared with the classical one.

5.3. An Special Guiding Role on Composed Problem

In the experimental process, we also found that the new heuristic has an special guiding role to solve the *Composed* problem and the results are shown in Table 3.

Table 3. Result of Composed Problem

instance		*dom/wdeg*	*dom/wsdeg*	*dom/wsdeg2*
composed-75-1-80 （10 all unsat）	t	1122	1186	**1110**
	n	283	179	**168**
	cc	53064	**46787**	48401
composed-25-1-80 （10 all unsat）	t	528	473	**393**
	n	165	149	**96**
	cc	38006	33642	**28157**
composed-75-1-40 （10 all unsat）	t	1255	969	**807**
	n	369	297	**222**
	cc	56853	42901	**41387**
composed-75-1-40-6	t	779	**30**	**30**

(unsat)	n	260	**4**	**4**
	cc	29401	**829**	**829**

In this paper, three kinds of composed problem are tested, each of them contain 10 instances, the test results are shown in Table 3. The results in Table 3 are the average values of the instances eliminate interfering values from special instances such as the result of *composed-75-1-40-6*. Test results from Table 3 indicate that the new heuristics are more efficient than *dom/wdeg*. At the same time, we also found that in about 20% of the instances appear the same results as *Composed-75-1-40-6.* In *Composed-75-1-40-6,*the number of nodes is greatly reduced and the solving efficiency is greatly improved. We found that all of the instances are unsatisfied,a lot of singleton-domain variables appear in the solving process and new heuristics give priority to instantiate the nodes which is decisive and the new heuristics greatly reduce the death node.

6. Conclusion

In this paper, we propose two new variable ordering heuristics and a compression method of the constraint network based on singleton-domain variables.The experimental results show that the new heuristics improve the classical one on some series of benchmarks where the conflict-based heuristics are efficient. It also proves the conjecture that the singleton-domain variable plays an important guiding role in the process of solving CSP. And as dynamic heuristic methods, the new heuristics also have certain universality.

References

[1] Freuder, E. C., & Mackworth, A. K. (2006). Constraint satisfaction: An emerging paradigm. *Handbook of Constraint Programming*.

[2] Van, B. P. (2006). Backtracking search algorithms. *Handbook of Constraint Programming*.

[3] Silver, D., Huang, A., Maddison, C. J., Guez. A., Sifre, L., *et al.* (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529(7587)*,484-489.

[4] Lecoutre, C., & Hemery, F. (2007). A study of residual supports in arc consistency..

[5] Golomb, S.W., & Baumert, L. D. (1965). Backtrack programming. *Journal of the ACM, 12(4),* 516-524.

[6] Bessiere, C., & Regin, J. C. (1996). MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. *Proceedings of the 2nd Int. Conf. on Principles and Practices of Constraint Programming,* 61-75.

[7] Haralick, R. M., & Elliott, G. L. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence, 14(3),* 263-313.

[8] Lecoutre, C., Sais, L., Tabary, S., & Vidal, V. (2007). Transposition tables for constraint satisfaction.

[9] Zhang, L. (2014). Heuristics for solving constraint satisfaction problems. Master's degree thesis of Jilin University, Changchun.

[10] Lecoutre. Retrieved from: http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html

[11] Boussemart, F., Hemery, F., Lecoutre, C., & Sais, L. (2004). Boosting systematic search by weighting constraints.

[12] Mackworth, A. K . (1977). Consistency in networks of relation. *Artificial Intelligence, 8(1),* 99-118.

[13] Sabin, D., & Freuder, E. C. (1994). Contradicting conventional wisdom in constraint satisfaction.

[14] Lppez, A., & Bacchus, F. (2003). Generalizing graphplan by forumlating planning as a CSP. *Proceedings of the International Joint Conference on Artificial Intelligence*.

Yi Jun Liu was born in Baicheng, Jilin, China. She received her bachelor degree in software from Northeast Normal University, China in 2014. Now she is studying for a master degree in the Institute of Computer Science and Technology in Jilin University. Her research interests are heuristics for solving constraint satisfaction problems and consistency algorithms for non-binary constraint.