A New Adaptive Parameterized Consistency Algorithm

Yunpeng Wu
College of Computer Science and Technology
Jilin University
Changchun, China
E-mail: 2412466779@qq.com

Sibo Zhang
College of Computer Science and Technology
Jilin University
Changchun, China
E-mail: 187097385@qq.com

Yonggang Zhang *
College of Computer Science and Technology
Jilin University
Changchun, China
E-mail: zhangyg@jlu.edu.cn
* Corresponding Author

Abstract—Nowadays all kinds of constraint solver support the same level of local consistency on all problems. The authors proposed a new adaptive parameterized consistency algorithm which adjusted the level of consistency depending on the depth of their supports in there domain. To this end, the authors gave the notion of the improved distance to the end of a value, which is the dynamic distance, and it reflected the current state of the assignment. If the size of the variable domain reached its given threshold or the ratio to the original size reached its given threshold, the constraint solving process used the stronger consistency technique to propagate the assigned value. The faster deletion of invalid value, the easier cause backtracking. It is more likely to find the problem as soon as possible. Finally the authors carry on the experiments on some benchmark instances, the results showed that the new algorithm can outperform original parameterized consistency algorithm on many problems and is a promising method.

Keywords-Constraint Satisfaction Problem; Constraint Propagation; Adaptive; Backtracking; Parameterized Consistency

I. INTRODUCTION

Constraint programming (CP) is a powerful paradigm for solving combinatorial search problems that draws on a wide range of techniques from artificial intelligence, computer science, databases, programming languages, and operations research. Constraint programming is currently applied with success to many domains, such as scheduling, planning, vehicle routing, configuration, networks, and bioinformatics[1]. Constraint satisfaction problems (CSPs) are the central problems in CP, which are mathematical problems defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which were solved by constraint satisfaction methods. CSPs are the subject of intense research in both artificial intelligence and operations research, since the regularity in their formulation provides a common basis to analyze and solve problems of many unrelated families. CSPs often exhibit high complexity, requiring a combination of heuristics and combinatorial search methods to be solved in a reasonable time.

Backtracking and constraint propagation are two main and basic constraint solving methods[1]. Arc consistency is the oldest and most famous way of constraint propagation. It can be used to prune infeasible values in the preprocessing and backtracking phase. Arc consistency is the standard level of consistency supported in constraint solvers. In addition, several other local consistencies stronger than arc consistency have been proposed, such as max restricted path consistency[3] or singleton arc consistency[4] and their variants[6,7,8]. Unfortunately, these new consistency techniques do not play the important role in current constraint solvers because of the high computational cost of maintaining them during search.

Stergiou[9] found that choosing the right level of local consistency for solving a problem required making the trade-off between the ability of removing inconsistent values and the cost of the algorithm. And he also suggested take advantage of the power of strong consistencies to reduce the search space while avoiding the high maintaining cost in the whole network. His paper presented a heuristic approach based on the monitoring of propagation events to dynamically adapt the level of local consistency (arc consistency or max restricted path consistency) to individual constraints. This pruned more values than arc consistency and less than max restricted path consistency, but it was more efficient than the arc consistency and max restricted path consistency.

Recently, Balafrej[10] proposed an original

parameterized consistency method. The paper introduced the notion of stability of values for arc consistency, a notion based on the depth of their supports in their domain. The technique allowed defining levels of local consistency of increasing strength between arc consistency and a given strong local consistency. It has instantiated the generic parameterized consistency approach to max restricted path consistency. The experiments showed that the parameterized consistency technique is viable and adapting the level of local consistency during search using the parameterized consistency is a promising approach that can outperform both MAC and a strong local consistency on many problems.

In this paper the authors define the notion of the improved distance to end of a value. It was the improved notion of the notion in the Balafrej's paper. The authors have observe that the distance to end of a value keeping the same value during the whole constraint solving process, it can't reflect the accurate information guiding the weak or strong consistencies enforcement. The improved distance to end of a value and corresponding thresholds resulted the new p-maxRPC^{imp} algorithm could enforce the strong consistency when needed, so it could increase the constant solving efficiency. The experiments results showed that the new proposed method was more efficient than the original algorithm.

The remainder of this paper is structured as followes. In Section 2, the authors introduced the background of the CSP and constraint propagation. In section 3, the authors discussed the parameterized consistency notion and the algorithm. In Section 4, the authors define the notion of the improved distance to end of a value and presented the p-maxRPC^{imp} algorithm. In Section 5, the authors gave the results and analysis of the experiments. In Section 6, the authors brief summarized the whole paper and pointed out the future work.

II. BACKGROUND

A constraint network N is defined as set of n variables $X=\{x_1,x_2, \cdots ,x_n\}$, a set of ordered domain $D=\{D(x_1),D(x_2), \cdots ,D(x_n)\}$,and a set of e constraints $C=\{c_1,c_2, \cdots ,c_e\}$.Each constraint c_k is defined by a pair$(var(c_k),r(c_k))$, where $var(c_k)$is an ordered subset of X, and $r(c_k)$ is a set of combinations of values satisfying c_k.In the following, the authors restrict binary CSP.A binary constraint c between x_i and x_j will be denote by c_{ij}, and $C(x_i)$ denote the set of variables x_j involved in a constraint with x_i.

A value $v_j \in D(x_j)$ is called an arc consistent support(AC support)for $v_i \in D(x_i)$ on c_{ij} if $(v_i,v_j) \in r(c_{ij})$.A value $v_i \in D(x_i)$is arc consistent(AC) if and only if for all $x_j \in C(x_i)$,v_i has an AC support $v_j \in D(x_j)$. A domain $D(x_i)$us arc consistent if it is non empty and all values in $D(x_i)$are arc consistent. A network is arc consistent if all domains in D are arc consistent. If enforcing arc consistency on a network N leads to a domain wipe out, the authors say the N is arc inconsistent. A tuple $(v_i,v_j) \in D(x_i) \times D(x_j)$is path consistent(PC) if and only if for any third variable x_k there exists a value $v_k \in D(x_k)$ such that v_k is an AC support for both v_i and v_j, and v_k is called witness for the path consistency of (v_i,v_j).

A value $v_j \in D(x_j)$is a max restricted path consistent (maxRPC) support for $v_i \in D(x_i)$ on c_{ij} if and only if it is an AC support and the tuple$(v_i,v_j)$is path consistent. A value $v_i \in D(x_i)$is max restricted path consistent of a constraint c_i if and only if there isoo a $v_j \in D(x_j)$) maxRPC support for v_i on c_{ij}. A value $v_i \in D(x_i)$ is max restricted path consistent if and only if for all $x_j \in C(x_i)$ v_i has a maxRPC support $v_j \in D(x_j)$on c_{ij}.A domain $D(x_i)$ is maxRPC if it is non empty and all values in $D(x_i)$ are maxRPC. A constraint network N is maxRPC if all domains in D are maxRPC.

The problem of finding the solution of a constraint network is called constrain solving, and it is NP-complete. The basic solving method is backtrack search combined with the some level of consistency.

III. PARAMETERIZED CONSISTENCY

Parameterized consistency was based on the concept of stability of values. Balafrej introduce the notion of the distance to end of a values in a domain.This captured how far a value was from the last in its domain. In the following, the definition and the theorem came from[10].

Definition 1.(Distance to end of a value).The distance to end of a value $v_i \in D(x_i)$ is the ratio

$$Dis(x_i,v_i)=(|D_0(x_i)|-rank(v_i,D_0(x_i)))/|D_0(x_i)| \quad (1)$$

where $D_0(x_i)$ is the initial domain of x_i.

Definition 2.(p-stability for AC) A value $v_i \in D(x_i)$ is p-stable for AC on c_{ij} iff v_i has an AC support $v_j \in D(x_j)$ on c_{ij} such that $Dis(xj,vj) \geqslant p$.A value $v_i \in D(x_i)$ is p-stable for AC iff for all $x_j \in C(x_i)$,v_i is p-stable for AC on c_{ij}.

Theorem 1. Let LC be a local consistency stronger than AC for which the LC consistency of a value on a constraint is defined. Let p_1 and p_2 be two parameters in [0..1]. If $p_1 < p_2$ then AC <constraint-based p_1-LC< constraint-based p_2-LC < LC.

Theorem 2. Let LC be a local consistency stronger than AC. Let p_1 and p_2 be two parameters in [0..1]. If $p_1 < p_2$ then AC < value-based p_1-LC < value-basedp_2-LC < LC.

Definition3. (p-maxRPC). A value and a network are p-maxRPC if and only if they are constraint-based p-maxRPC.

IV. P-MAXRPC^{IMP} ALGORITHM

Definition 4. (Improved Distance to end of a value).The Improved distance to end of a value $v_i \in D(x_i)$ is the ratio

$$Dis(x_i,v_i)=(|D_{cu}(x_i)|-rank(v_i,D_{cu}(x_i)))/|D_{cu}(x_i)| \quad (2)$$

Where, $D_{cu}(x_i)$ is the current domain of x_i.

During the process of solving the variable universe, the authors have the priority to implement the principle of priority in the process of the current domain size to reach a certain limit. According to the experimental data summarized the threshold set to a and b. That $|D_{cu}(x_j)| \leqslant a$ or $|D_{cu}(x_i) / |D_o(x_i)| \leqslant b$, the b is measure the current on the domain size of the standard, the current on the domain size and initial on the domain size, the ratio is less than beta determine when the domain of the variable value up to standard, does not have to perform comparison (x_i, v_i) with P, direct use of strong consistency checking algorithm, thus ensuring more backtracking to the premise with

minimal overhead were judged. Therefore, it is easier to produce a dead node so as to lead to backtracking and improve the efficiency of solving. The followed were the pseudo code of p-maxRPC3^{imp} algorithms.

```
Algorithm1 Initialization(X, D, C, Q)
1  Begin
2      foreach x_i ∈ X do
3          foreach v_i ∈ D(x_i) do
4              foreach x_j ∈ Γ(x_i) do
5                  p-support ← false
6                  foreach v_j ∈ D(x_j) do
7                      if (v_i, v_j) ∈ c_ij then
8                          LastAC_{xi, vi, xj} ← v_j
9                          USEmaxRPC ← false
10                         if |D(x_j)| ≤ 3 OR |D(x_j)|/ |D_0(x_j)| ≤ 0.3 OR Δ'(x_j, v_j) < p then
11                             USEmaxRPC ← true
12                         if ¬ USEmaxRPC then
13                             p-support ← true
14                             LastPC_{xi, vi, xj} v_j
15                             break;
16                         if searchPCwit(v_i, v_j) then
17                             p-support true
18                             LastPC_{xi, vi, xj} v_j
19                             break;
20                 if ¬p-support then
21                     remove v_i from D(x_i)
22                     Q ← Q∪{x_i}
23                     break;
24         if D(x_i) = ∅ then return false
25     return true
```

Figure 1. Algorithm1 Initialization

```
Algorithm2 checkPCsupLoss(v_j, x_i)
1  begine
2      if LastAC_{xj, vj, xi} ∈ D(x_i) then b_i ← max(LastPC_{xj, vj, xi} | 1, LastAC_{xj, vj, xi})
3      else
4          b_i ← max(LastPC_{xj, vj, xi}+1, LastAC_{xj, vj, xi}+1)
5      foreach v_i ∈ D(x_i), v_i ≥ b_i do
6          if (v_j, v_i) ∈ c_ji then
7              if LastAC_{xj, vj, xi} ∉ D(x_i) & LastAC_{xj, vj, xi} > LastPC_{xj, vj, xi} then
8                  LastAC_{xj, vj, xi} ← v_i
9              if |D(xj)| ≤ 3 OR |D(xj)|/ D0(xj)| ≤ 0.3 OR Δ'(xj, vj) < p then
10                 USEmaxRPC ← true
11             if ¬ USEmaxRPC then
12                 LastPC_{xj, vj, xi} ← v_i
13                 return true
14         if searchPCwit(v_j, v_i) then LastPC_{xj, vj, xi}−1 < v_i return true
15     return false
13     return false
```

```
Algorithm3 checkPCwitLoss(x_j, v_j, x_i)
1.  begin
2.      foreach x_k ∈ Γ(x_j) ∩ Γ(x_i) do
3.          witness ← false
4.          if v_k → LastPC_{xj, vj, xk} ∈ D(x_k) then
5.              USEmaxRPC ← false
6.              if |D(x_j)| ≤ 3 OR |D(x_j)|/ |D_0(x_j)| ≤ 0.3 OR Δ'(x_j, v_j) < p then
7.                  USEmaxRPC ← true
8.              if ¬ USEmaxRPC then
9.                  witness ← true
10.             else
11.                 if LastAC_{xj, vj, xi} ∈ D(x_i) & LastAC_{xj, vj, xi} = LastAC_{xk, vk, xi}
12.                 OR LastAC_{xj, vj, xi} ∈ D(x_i) & (LastAC_{xj, vj, xi, vk}) ∈ c_ik
13.                 OR LastAC_{xk, vk, xi} ∈ D(x_i) & (LastAC_{xk, vk, xi, vj}) ∈ c_ji
14.                 then witness ← true
15.                 else
16.                     if searchACsup(x_j, v_j, x_i) & searchACsup(x_k, v_k, x_i) then
17.                         foreach v_i ∈ D(x_i), v_i ≥ max(LastAC_{xj, vj, xi}, LastAC_{xk, vk, xi}) do
18.                             if (v_j, v_i) ∈ c_ji & (v_k, v_i) ∈ c_ki then
19.                                 witness ← true
20.                                 break;
21.     if witness & checkPCsupLoss(v_j, x_k) then return false
22.     return true
```

Figure 2. Algorithm 2 checkPCsupLoss and Algorithm 3 checkPC witLoss

The line 9 of the algorithm 1 initialization (X, D, C, Q) (Fig. 1), line 9 of the algorithm 2checkPCsupLoss (v_j, x_i) and the line t of the algorithm 3 checkPCwitLoss (x_j, v_j, x_i), the authors add the the $|D(x_j)| \leqslant 3$ and $|D(x_j)| / |D_0(x_j)| \leqslant 0.3$ judgment, namely, to determine the current domain size, if on a smaller domain, use the

delete value ability strongly consistent algorithm, so more easily produced dead node as soon as possible so as to lead to backtracking, improve the efficiency of the algorithm.

V. EXPERIMENTS

In this section, the authors will show the experimental evaluation of the proposed algorithm. All the problem instances come from the website (http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html#) maintained by Lecoutre. All algorithms have been implemented in C++ and embedded into the constraint solving platform designed by ourselves. In this platform, the authors use dom/deg variable ordering because it is easy to implement and very efficient. The experiments were carried out on a DELL Intel i7-3770 3.40GHz CPU/8GB RAM with Windows 7.0/Visual C++6.0.

TABLE I. PERFORMANCE (CPU TIME, NODES AND CONSTRAINT CHECKS) OF MAXRPC, VARIABLE-BASED AP-MAXRPC(APX-MANRPC), CONSTRAINT-BASED AP-MAXRPC(APC-MAXRPC) AND THE CORRESPONDING IMPROVED VERSION ON QWH INSTANCES

Instances (sat)	**Measure**	**maxRPC**	**p-maxRPC**	**apc-maxRPC**	**apx-maxRPC**
			p-maxRPC^{inp}	**apc-maxRPC^{inp}**	**apx-maxRPC^{inp}**
qwh-15-106-0_ext	cpu(s) #ccks #nodes	307.551 4348296 1856	**Time-out**	479.477 2857255 3893	302.539 3945798 1856
	cpu(s) #ccks #nodes		302.173 4299125 1879	310.1 4296012 1879	275.95 4358267 1856
qwh-15-106-2_ext	cpu(s) #ccks #nodes	522.875 7976700 2801	3585.64 28010711 24135	1179.71 5660938 9287	524.15 7091034 2801
	cpu(s) #ccks #nodes		566.563 8136404 2990	565.207 8133621 2990	516.075 8030984 2801
qwh-15-106-3_ext	cpu(s) #ccks #nodes	26.949 480361 230	56.097 518713 471	34.947 124863 359	23.508 367257 230
	cpu(s) #ccks #nodes		73.039 409856 230	27.368 409847 230	22.564 446423 230
qwh-15-106-4_ext	cpu(s) #ccks #nodes	1.859 118680 104	1.675 60896 114	1.822 23392 138	1.652 97603 104
	cpu(s) #ccks #nodes		1.571 71664 104	1.573 71920 104	1.685 107256 104
qwh-15-106-6_ext	cpu(s) #ccks #nodes	69.254 1115837 421	179.123 2538845 1517	101.846 571642 690	62.659 952177 422
	cpu(s) #ccks #nodes		61.54 1038729 421	66.831 1039289 421	66.828 1092199 421
qwh-15-106-7_ext	cpu(s) #ccks #nodes	536.533 6693875 2896	**Time-out**	1256.66 7291572 10069	543.576 6156600 3003
	cpu(s) #ccks #nodes		533.014 6549992 2874	533.657 6553730 2874	534.168 6672117 2900

All the instances in TABLE I are belong to the SAT (satisfaction type), where each instance including 100 variables and 900 constraints. The experiments showed that the improved algorithms have advantage both in solving time and visited nodes. Compared with the classical maxRPC algorithm, the adaptive version of maxRPC algorithms were not as good as the classical maxRPC on the most instances. It showed that the adaptive version algorithm increase the computational cost though they decreased the strong consistency cost. The authors could observe the data about p-maxRPC, apc-maxRPC, apx-maxRPC algorithm, and could find that apx-maxRPC was the best one.It was because that this algorithm adjusted threshold p using variable-based method, the value of p became bigger and bigger with the domain reduced, thus it enforced strong consistency more frequently and got the solution more quickly.The improved p-maxRPC, apc-maxRPC, apx-maxRPC had stable advantage both on the CPU time and number of the visited node. The execution time is generally less than the original algorithms, and the number of nodes is the same as those of the original algorithms.

In the original three parameters of the algorithm can be seen based on the adaptive variable is a absolute advantage, the reasons above have given analysis. By comparison, the new three parameter apx-maxRPC^{imp} algorithm is more efficient than the algorithm based on variable in the same optimization algorithm. Although the overall execution time is less than before optimization, most of the algorithms and the visit to the number of nodes is minimized, but apx-maxRPC^{imp} in based on the method of adaptive variable optimization is not perfect. This is mainly because the idea is to optimize the value of the idea is to update the value of the end of the distance, while the variable is also the variable adaptive and it is worth to cut and continue to change. Based on the variation of the threshold parameters of the variable is by all variables where the median constraint deletion caused, but terminal distance changes indeed within the limited scope changes, this range is the initial domain size. Therefore, change from the end of distance speed to change the parameters of the threshold p value range. The implementation of the algorithm is less than the value of the p value of the implementation of the strong compatibility check. In front of the initial domain value because of distance is greater than the threshold; it is not strong compatibility check. However, due to the rapid increase of P value, it is not required to perform the strong compatibility check of the value of the end of the scale is less than P, which makes a partial value of the early implementation of the strong compatibility check. As a result, the strong consistency checks led to the increase in the implementation of the time.

VI. CONCLUSIONS

State-of-the-art constraint solvers uniformly maintain the same level of local consistency on all the instances. The authors proposed a new parameterized local consistency based on the notion of the improved distance to the end of a value, which is the dynamic distance, and it reflected the current state of the assignment. If the size of the variable domain reached its given threshold or the ratio to the original size reached its given threshold, the constraint solving process used the stronger consistency technique to propagate the assigned value. The faster deletion of invalid value, the easier cause backtracking. It is more likely to find the problem as soon as possible. Finally the authors carry on the experiments on some benchmark instances, the results showed that the new algorithm can outperform original parameterized consistency algorithm on many problems and is a promising method.

ACKNOWLEDGMENT

The authors of this paper express sincere gratitude to all the anonymous reviewers for their hard work. This work was supported in part by NSFC under Grant No. 61170314, No. 61373052.

REFERENCES

[1] F.Rossi, P.van Beek, T.Walsh, Handbook of Constraint Programming.Elsevier, 2006

[2] T. Balafoutis, A.Paparrizou, K.Stergiou, T.Walsh, "New algorithms for max restricted path consistency," Constraints, volume16, pp.372-406,August 2011,doi: 10.1007/s10601-011-9110-y.

[3] C.Bessiere, J.C.Regin, R.H.C.Yap, Y.Zhang, "An optimal coarse-grained arc consistency algorithm," Artificial Intelligence, volume 165, pp.165-185,Feb.2005,doi: 10.1016/j.artint.2005.02.004.

[4] C.Bessiere, K.Stergiou, T.Walsh, "Domain filtering consistencies for non-binary constraints,". Artificial Intelligence, volume172,pp.800-822,Feb.2008,doi: 10.1016/j.artint. 2007. 10. 016.

[5] R.Debruyne, C.Bessiere, "Domain Filtering consistencies", Journal of Artificial Intelligence Research.volume 14, pp.205-230,Oct.2001,doi: 10.1613/jair.834

[6] K.Stergiou, "Heuristics for dynamically adapting propagation in constraint satisfaction problems," AI Commun,volume22, pp.125-141,August 2009,doi: 10.3233/AIC-2009-0450

[7] I.Katriel, P. Van Hentenryck, "Randomized filtering algorithms,"Technical ReportCS-06-09, Brown University ,June 2006.

[8] M.Sellmann, "Approximated consistency for knapsack constraints," Proc. International Conference on Principles and Practice of Constraint Programming 2003(CP03), Springer Press, 2003, Oct.2003,pp.679-693,doi: 10.1007/978-3-540-45193-8_46.

[9] A. Balafrej, C. Bessiere, R. Coletta, E.-H. Bouyakhf, "Adaptive Parameterized Consistency,". Proc. International Conference on Principles and Practice of Constraint Programming (CP13), Springer Press, 2013, Sept.2013,pp.143-158,doi: 10.1007/978-3-642-40627-0_14.

[10] R. J. Woodward, A. Schneider, B.Y. Choueiry, C. Bessiere, "Adaptive Parameterized Consistency for Non-binary CSPs by Counting Supports," Proc. International Conference on Principles and Practice of Constraint Programming (CP14), Springer Press, 2014, Sept.2014,pp.755-764,doi: 10.1007/978-3-319-10428-7_54.