Journal of Computer Applications
计算机应用,2018,38(S2):38-41
ISSN 1001-9081
CODEN JYIIDU
2018-12-25
http://www.joca.cn

文章编号:1001-9081(2018)S2-0038-04

# 基于短子句的一种分支策略

胡忠雪*,徐 扬,胡 容,徐 颖
(西南交通大学 数学学院,成都 610031)
(*通信作者电子邮箱 1690213008@qq.com)

**摘 要**:针对目前大部分可满足性(SAT)问题算法中变量选择问题,提出一种基于短子句的分支策略。这个新思想的主要方法是:首先,赋予每个变量一个权重,依据变量的得分值来进行赋值;其次,在进行布尔约束传播过程时发生冲突后,依据对新产生的学习子句中变量所在决策层和冲突层的平均值的大小条件来对其增加得分,未参与冲突的变量分值保持不变;最后,选择得分最高的变量并赋值,重新进行布尔约束传播。分析结果表明,与变量状态独立衰减分值(VSIDS)策略相比,所提出新策略决策次数和平均运算时间较少,运算效率较高。新的策略优先满足短子句,能尽快找到一组可满足解,更快速解决问题,提高算法在解决实际问题中的效率。

**关键词**:分支策略;决策层;冲突层;子句长度;文字块距离
**中图分类号**:TP181 **文献标志码**:A

## Branching strategy based on short clause

HU Zhongxue*, XU Yang, HU Rong, XU Ying
(*School of Mathematics, Southwest Jiaotong University, Chengdu Sichuan* 610031, *China*)

**Abstract:** In view of the variable selection problem in most algorithms for SAT (SATisfibility) problems, a branching strategy based on short clauses was proposed. The main method of this new idea is: firstly, each variable was given to a weight; according to its score, the variable was assigned; secondly, when a conflict happens in the boolean constraint propagation process, the scores of the variable in new learnt clauses were increased according to the average of decision layer and conflict layer, and the values of variables not involved in the conflict remain unchanged; finally, the variable with highest score was chosen and assigned, and boolean constraint propagation was executed again. To compare with VSIDS (Variable State Independent Decaying Score) strategy, the analysis results show that the proposed new strategy spent fewer decision times and average operation time, thus the computing efficiency was improved. The new strategy can give priority to the short clauses, so a set of satisfiable solutions was found as soon as possible, the problem was solved more quickly, and the efficiency of the algorithm was improved in solving practical problems.

**Key words:** branching strategy; decision layer; conflict layer; clause length; literals block distance

## 0 引言

可满足性(Satisfiability, SAT)问题是一类典型的非确定性多项式(Non-deterministic Polynomial, NP)问题,此类问题的有关算法不断被学者们改进,为了能够在某些领域解决更多问题,如计算机自能设计、数据诊断逻辑推理、探索性数据分析(Exploratory Data Analysis, EDA)等多种领域,尽管现在还有许多有关现实生活的SAT问题还没有得到解决,但是随着SAT问题的算法不断改进,已经在许多领域取得了巨大的突破[1]。

完备算法与不完备算法是SAT问题的两大主要算法。"对于可满足性问题,能足以保证找到一组可满足真值赋值;对于不可满足性问题,但能证明此问题是不可满足"这就是完备算法。不完备算法是"对于可满足性问题,不能够保证一定能找到可满足的解;对于不可满足性问题,无法将其证明",完备算法[2]主要有DP(Davis-Putnam)算法、JW(Jeroslow-Wang)算法、DPLL(Davis-Putnam-Logemann-Loveland)算法以及CDCL(Conflict-Driven Clause Learning)算法等,现在以CDCL为主流,不断在此基础上进行改进。CDCL算法的求解过程的基本阶段为:1)初始处理过程;2)变量决策阶段;3)布尔约束传播过程(Boolean Constraint Propagation,BCP);4)非时序回溯;5)随机重启。分支策略是第二个阶段,如今这个阶段主要采用基于VSIDS(Variable State Independent Decaying Score)算法的改进算法。自从2001年在工程效率可满足算法(Chaff: Engineering an Efficient SAT Solver)中首次提出VSIDS,后来就有许多基于VSIDS改进的算法如NVSIDS(Normalized VSIDS)、ACIDS(Average Conflict-index Decision Score)等,在这篇文章里也是基于VSIDS改进。这个新策略主要是依据权重的思想,对每个变量在赋值之前都馈赠一个权重,选择权重最高的变量进行赋值,然后再进行布尔约束传播,一旦有冲突发生就对那冲突进行分析,将分析后学到的学习子句加入到子句库,对跟冲突相关的变量依据一些条件馈赠相应的值。采用一些基本的例子进行测试,如在某些方面有提升,那么实验成功。

## 1 SAT问题的有关基础知识

SAT问题主要有两大类型,一个是子句集型,另一个是公式型,这两者在多项式时间内可以相互转换。下面是后面内容要用到的一些知识和相关概念[3]。

**收稿日期**:2018-01-10;**修回日期**:2018-03-14。 **基金项目**:国家自然科学基金资助项目(61673320)。

**作者简介**:胡忠雪(1991—),女,四川达州人,硕士研究生,主要研究方向:人工智能、自动推理; 徐扬(1956—),男,河南新乡人,教授,博士,主要研究方向:人工智能、自动推理、一阶逻辑和命题逻辑; 胡容(1992—),女,四川遂宁人,硕士研究生,主要研究方向:人工智能、自动推理; 徐颖(1993—),女,四川眉山人,硕士研究生,主要研究方向:优化与决策。

概念1　用 $S$ 表示若个变量的集合,那么 $S = \{x_1, x_2, \cdots, x_n\}$, $\forall x_i$ 为变量,与 $x_i$ 相对应的有两种形式文字存在,分别是 $x_i$ 与 $\neg x_i$,前者称之为 $x_i$ 的正文字,$\neg x_i$ 为负文字,每个变量都是以这两种形式出现在公式中。$\forall x_i$ 都有两种赋值情况,可赋为1或者赋为0,但是正负文字的赋值相反。例如 $x_i = 1$,相应地 $\neg x_i = 0$。

概念2　有限个文字的析取被称为子句,不含任何一个文字的子句是空子句,用□表示,在命题逻辑中□不可满足,然而空集在命题逻辑中却定为可满足。子句中所含文字的个数是子句的长度,用 $|C|$ 来表示子句的长度。例如 $C = x_1 \vee x_2 \vee \neg x_3 \vee x_4 \vee \neg x_6$,它的子句长度为 $|C| = 5$。

概念3　将有限个子句的合取称为公式,写作:$f = C_1 \vee C_2 \vee C_3 \vee \cdots \vee C_n$ 或者是集合形式 $\{C_1, C_2, \cdots, C_n\}$,任何一个子句集的真值指派,均可以看作一个映射,即:$T: S \rightarrow \{0,1\}$。

这里 $S = \{x_1, x_2 \cdots, x_n\}$,每个变量有两种赋值,所以这个公式集有 $2^n$ 种真值赋值。除此之外,还有一种表示公式的形式,这里就用一个例子来说明,譬如:

```
F  cnf  7  6
-1 2 4 0
3 -5 7 0
4 7 0
1 3 6 7 0
3 6 7 0
1 2 4 6 -7 0
%
0
```

此例中 F 代表公式,cnf 代表合取范式,7 表示 7 个变量,6 表示 6 个子句,% 和 0 均代表结束。这种表示法在后面例子中也会采用。

概念4　将一个子句中所含变量的决策层的平均值称为文字块距离(Literals Block Distance, LBD),后面就用 LBD 表示。例如:

```
F  cnf  4  3
-1 2 4 0
2 -3 4 0
1 2 0
%
0
```

在子句集中有 $C_2 = x_2 \vee \neg x_3 \vee x_4$ 这3个变量的决策层分别是1,2,4,所以这个子句的 $LBD = (1+2+4)/3 = 7/3$。

概念5　在进行布尔约束传播发生冲突后对所产生的冲突分析原因,导致产生冲突的子句就会被记录下来,这个子句就是学习子句。将学习子句记录下来可以避免在后面的搜索过程发生同样的冲突。如下例子是通过蕴含图介绍子句的产生过程。例如:

```
F  cnf  10  6
1 3 -2 0
-2 -4 6 0
-3 4 5 0
-5 8 9 0
-5 -7 -10 0
-8 7 0
%
0
```

现已决策赋值的变量是:

$\{x_1 = 0@1; x_2 = 1@3; x_6 = 0@1\}$

通过布尔布尔约束传播可得:

$x_3 = 1@3; x_4 = 0@3; x_5 = 1@3$

$x_7 = 0@3; x_8 = 1@3; x_9 = 0@2$

$x_{10} = 1@2$

接下来是一个蕴涵图来说明学习子句的产生过程[4-5]。

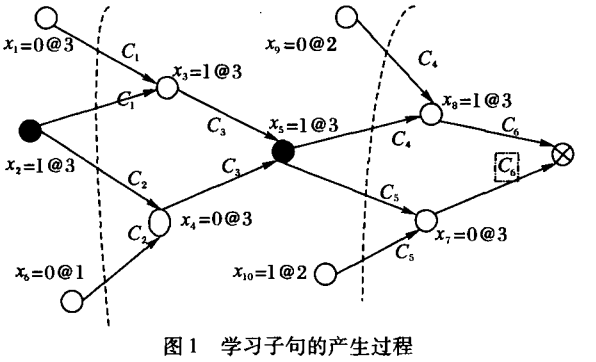


图1　学习子句的产生过程

从第一唯一蕴涵点 $x_5$ 与最后唯一蕴涵点 $x_2$ 以及两条蕴涵切割线,可以学习到两个学习子句:

$C^1 = x_1 \vee \neg x_2 \vee x_6$

$C^2 = x_9 \vee \neg x_5 \vee \neg x_{10}$

## 2　相关策略

新提出的策略与两种策略相关,所以后面就分别详细介绍这些策略。

### 2.1　VSIDS 策略

在 Chaff 中提出 VSIDS 后,基于 VSIDS 策略改进的策略有许多,正是这些改进才使得求解效率不断提高[6-7]。以下是 VSIDS 的详细步骤:

1)对子句集中的每一个文字都赋予一个计分器,记录文字在所给出的子句集中出现的次数,所以初始值设置为0;

2)当子句集中每次有子句增添入时,所增添的子句里的所有文字的得分数均增加1,并选分值最大的文字进行赋值;

3)每次进行分支选择时,选择得分值最大的文字进行赋值;

4)当布尔约束传播遇到冲突时,与冲突相关变量加1,反之没发生冲突就是可满足;

5)当几个文字的得分值出现相等的情况时,就随机任选一个文字进行赋值;

6)所有文字的得分数会周期性地除以一个常数。

VSIDS 策略是 SAT 算法中的一个巨大突破,也是质的飞跃,为生活中解决更多问题提供了巨大的帮助。

### 2.2　基于 LBD 分支启发式的奖励机制

1)对子句集中的每一个文字都赋予一个计分器,计录文字在所给出的子句集中出现的次数,所以初始值设置为0;

2)当子句集中每次有子句增添入,分数增加是文字个数;

3)每次进行分支选择时,选择得分值最大的文字进行赋值;

4)当布尔约束传播发生冲突时,将学习子句添加到子句库,与冲突相关的变量分值更新为:

①倘若学习子句的 $LBD \leqslant 2$,分值增加15;

② 倘若学习子句的 $3 \leqslant LBD \leqslant 10$,分值增加10;

③ 倘若学习子句的 $LBD \geqslant 10$,分值增加5;

与冲突无关的变量的分数保持不变。

5)当几个文字的得分值出现相等的情况时,就随机任选一个文字进行赋值;

6)所有文字的得分数会周期性地除以一个常数。

这个策略就是在布尔约束传播过程发生冲突时,对改变

冲突相关变量的分值,将学习子句的 LBD 相结合考虑[8]。新改进的策略也是受此策略的启发改变了一下,并且用例子和原始的 VSIDS 策略进行比较分析。

## 3 新策略

这个新策略是在前面两个策略的基础之上提出的一种新思想。由于在对变量进行赋值时长子句更容易满足,而短子句不容易满足,所以在选择变量赋值之前,对每个变量馈赠一个权重[9-11],使得能优先满足短子句。在布尔约束传播发生冲突时,与冲突相关的每个变量的分值也进行了一定改变,下面就是新策略的详细过程:

1)给每个变量一个计分器,记录变量在子句中的得分,如果一个变量没有在子句 $C$ 中出现则初始为 0,反之初始为 $\sum_{C_i \in J} 5/|C_i|$;

2)每次进行分支选择时,选择得分值最大的文字进行赋值;

3)当几个变量的得分值出现相等的情况时,就随机任选一个变量进行赋值;

4)当布尔约束传播过程发生冲突时,与冲突有关的变量得分就会增加,增加的分值为:

①当 $0 \leqslant (d + d_k)/2 < 10$ 时,得分数就增加 $10 + 5/|C_k|$;

②当 $10 \leqslant (d + d_k)/2 < 20$ 时,得分数就增加 $20 + 5/|C_k|$;

③当 $20 \leqslant (d + d_k)/2 < 30$ 时,得分数就增加 $30 + 5/|C_k|$;

④当 $(d + d_k)/2 \geqslant 30$ 时,得分数就增加 $40 + 5/|C_k|$;

其他变量的分数不变。

5)衰退过程与 VSIDS 一样(也就是 VSIDS 的第6)步)。

在第一步中赋予子句一个权重 $\forall C$ 的权重为 $5/|C|$,给予每个变量的权重为它所在的子句权重之和,即是 $\sum_{C_i \in J} 5/|C_i|$,之所以选择 $5/|C|$ 作为权重,为实现赋值时优先满足于短子句,子句越短权重就越大,所得到的分值越大,就越容易先被选择赋值。以下是一例子说明这样赋予权重的目的。

例如:

```
F cnf 10 6
2 3 0
-1 5 7 8 0
-2 4 0
3 6 -7 0
1 4 5 -9 0
2 10 0
%
0
```

计算各变量的权重结果在表1中。

表1 变量权重计算值

| 变量 | 权重值 | 变量 | 权重值 |
|---|---|---|---|
| $x_1$ | 5/2 | $x_6$ | 5/3 |
| $x_2$ | 15/2 | $x_7$ | 35/12 |
| $x_3$ | 25/6 | $x_8$ | 5/4 |
| $x_4$ | 15/4 | $x_9$ | 5/4 |
| $x_5$ | 5/4 | $x_{10}$ | 5/2 |

变量 $x_2$ 只出现在短子句中,经过计算它的权重值也是最大的,所以在选择变量赋值的时候就是依据权重的大小来选择的,这样的话就先满足短子句。在长子句中的变量较多,赋值使其满足的解也就比较多。为了让更多的子句得到满足,因此优先满足短子句[12]。

在布尔约束传播发生冲突时,此法以决策层与冲突层的平均值作为条件来考虑,因为在每次发生冲突时有个冲突层,与冲突相关的变量有决策层,所以以此来考虑,短子句优先保证可满足[13]。

## 4 实验分析

此实验中,采用的是实际小例子进行分析,此次实验在同一环境进行。所用实验数据是 SATLIB(SAT Little Information Bank)中的标准问题集。先后对不同例子在同一环境下对 VSIDS 策略和改进的新策略各进行了多次测试,只将求解出的例子算出多次测试的平均值,然后进行比较。表2就是本次测试的结果,其中 *num*(number)表示例子总数,*n* 为变量个数,*m* 为子句个数,*confs*(conflicts)代表冲突次数,*lits*(literatures)代表冲突文字数,*decs*(decisions)表示决策次数,*props*(propagates)表示布尔约束传播次数,表2是新策略测试结果,表3是 VSIDS 测试结果。

表2 新策略的实验结果

| *num* | *n* | *m* | *confs* | *lits* | *decs* | *props* |
|---|---|---|---|---|---|---|
| 100 | 90 | 300 | 0 | 1 | 11 | 42 |
| 100 | 500 | 3100 | 0 | 0 | 111 | 395 |
| 100 | 50 | 118 | 2 | 321 | 2 | 145 |
| 100 | 200 | 334 | 0 | 0 | 1 | 172 |
| 100 | 500 | 3000 | 0 | 2 | 101 | 400 |
| 150 | 150 | 545 | 22 | 66 | 12 | 99 |
| 150 | 150 | 544 | 22 | 66 | 12 | 99 |
| 150 | 150 | 545 | 0 | 0 | 9 | 71 |
| 150 | 2188 | 104856 | 0 | 0 | 232 | 11929 |
| 150 | 50 | 99 | 0 | 0 | 1 | 50 |
| 200 | 500 | 3100 | 0 | 0 | 109 | 387 |
| 200 | 500 | 3100 | 7519 | 11812 | 85 | 7806 |
| 200 | 21800 | 104856 | 0 | 0 | 232 | 1129 |
| 200 | 200 | 320 | 0 | 0 | 1 | 191 |
| 200 | 90 | 300 | 3 | 12 | 11 | 44 |

从结果可以看到,基本上新策略的冲突次数少于原始策略冲突次数,决策次数也比原始策略的少。在每次例子中倘若冲突次数越少,运行效率也就越高,决策次数越少,为运算节省了时间。从这些例子多次测试求解的平均结果中可以看出新策略对这些例子的求解效率提高不少。

对同一例子用两种不同策略测试,新策略的结果更好。

## 5 结语

本文介绍了 VSIDS 策略以及基于 LBD 分支启发式的奖励机制,由于新提出的算法中与子句长度有关,所以用小例子介绍了选择子句长度作为初始值的目的,最后采用 SATLIB 例子分别用 VSIDS 策略和新策略各进行多次测试,将求出的例子结果求得平均冲突次数、平均冲突文字数、平均决策层等,最后进行了比较,分析了结果。

在后期计划中打算把这个新策略嵌入到 CDCL 求解器[14],放入比较大的 SAT 问题实例运行,从求解时间等多方面分析对比,求解更多大例子[15]。

表 3　VSIDS 策略的实验结果

| num | n | m | confs | lits | decs | props |
|---|---|---|---|---|---|---|
| 100 | 500 | 3 100 | 1 | 5 | 117 | 396 |
| 100 | 500 | 3 100 | 2 | 1 | 120 | 428 |
| 100 | 50 | 118 | 4 | 329 | 9 | 166 |
| 100 | 200 | 334 | 4 | 9 | 9 | 178 |
| 100 | 500 | 3 000 | 11 | 129 | 167 | 800 |
| 150 | 150 | 545 | 6 | 41 | 20 | 239 |
| 150 | 150 | 544 | 6 | 41 | 20 | 239 |
| 150 | 150 | 545 | 2 | 11 | 10 | 108 |
| 150 | 2 188 | 104 856 | 0 | 0 | 263 | 11 929 |
| 150 | 50 | 99 | 0 | 0 | 1 | 50 |
| 200 | 500 | 3 100 | 2 | 1 | 120 | 428 |
| 200 | 500 | 3 100 | 1 | 5 | 117 | 396 |
| 200 | 21 800 | 104 856 | 0 | 0 | 262 | 11 929 |
| 200 | 200 | 320 | 0 | 0 | 1 | 191 |
| 200 | 90 | 300 | 3 | 10 | 20 | 85 |

**参考文献:**


[1] WU G, XU Y, CHANG W, et al. Parallel genetic algorithm for SAT problem based on the coarse-grained model[C]// FLINS 2016: Proceedings of the 12th International Conference on Uncertainty Modelling in Knowledge Engineering and Decision Making. Roubaix: World Scientific Publications, 2016: 489 - 495.

[2] MARQUES-SILVA J P, SAKALLAH K A. GRASP: A search algorithm for propositional satisfiability[J]. IEEE Transactions on Computers, 1999, 48(5): 506 - 521.

[3] 陈稳. 基于 DPLL 的 SAT 算法的研究及应用[D]. 成都: 电子科技大学, 2011: 1 - 61.

[4] SABHARWAL A, BEAME P, KAUTZ H. Using problem structure for efficient clause learning[C]// SAT 2003: Theory and Applications of Satisfiability Testing, LNCS 2919. 2003: 242 - 256.

[5] RYAN L. Efficient algorithms for clause-learning SAT solvers[D]. Burnaby, Canada: Simon Fraser University, 2004: 428 - 432.

[6] MOSKEWICZ M W, ZHAO Y, MADIAN C F, et al. Chaff: engineering an efficient SAT solver[C]// DAC 2001: Proceedings of the 38th International Conference on Design Automation. Washington, DC: IEEE Computer Society, 2005: 530 - 535.

[7] LIANG J H, GANESH V, ZULKOSKI E, et al. Understanding VSIDS branching heuristics in conflict-driven clause-learning SAT solvers [C]// HVC 2015: Proceedings of the 11th International Haifa Verification Conference on Verification and Testing of Hardware and Software. Haifa: Springer International Publishing, 2015: 225 - 241.

[8] AUDEMARD G, SIMON L. Predicting learnt clauses quality in modern SAT solver[C]// Proceedings of the 21st International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann, 2009: 399 - 404.

[9] PIPATSRISAWAT K, DARWICHE A. On the power of clause-learning SAT solvers with restarts[C]// CP 2009: Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming. Heidelberg: Springer Berlin, 2009: 654 - 668.

[10] 刘志明, 吴明芬, 许勇. 一种基于遗传算法的权重的确定方法[J]. 五邑大学学报(自然科学版), 2006, 20(3): 45 - 48.

[11] 凌应标, 吴向军, 姜云飞. 基于子句权重学习的求解 SAT 问题的遗传算法[J]. 计算机学报, 2005, 28(9): 1476 - 1482.

[12] ZHAO Y, SONG Z. A new branching heuristic for propositional satisfiability[C]// iFuzzy 2016: Proceedings of the 2016 International Conference on Fuzzy Theory and its Applications. Piscataway: IEEE, 2017: 109 - 112.

[13] SILVA J, MARQUES O P. The impact of branching heuristics in propositional satisfiability algorithms[C]// EPIA 1999: Proceedings of the 9th Portuguese Conference on Artificial Intelligence, LNCS 1695. Heidelberg: Springer Berlin, 1999: 62 - 74.

[14] LIU Y, WANG J, XU C, et al. An effective branching strategy based on structural relationship among multiple forbidden induced subgraphs[J]. Journal of Combinatorial Optimization, 2015, 29 (1): 257 - 275.

[15] SILVA J P M, LYNCE I, MALIK S. Conflict-driven clause learning SAT solvers[J]. Frontiers in Artificial Intelligence & Applications, 2009, 185(1): 343 - 351.


(上接第 15 页)


[10] ZHANG Y L, LI L, ZHOU J, et al. Anomaly detection with partially observed anomalies[C]// Companion of the the Web Conference 2018. New York: ACM, 2018: 639 - 646.

[11] 林金钏, 艾浩军. 噪声可容忍的标记组合半监督学习[J/OL]. 计算机工程, 2018[2018 - 06 - 25]. https://doi.org/10.19678/j.issn.1000-3428.0050398.

[12] LIU B. Research Projects[EB/OL]. [2018-06-02]. https://www.cs.uic.edu/~liub/NSF/PSC-IIS-0307239.html.

[13] LIU B, LEE W S, YU P S, et al. Partially supervised classification of text documents[C]// Proceedings of the 2002 Nineteenth International Conference on Machine Learning. San Francisco: Morgan Kaufmann Publishers, 2002: 387 - 394.

[14] PLESSIS M C D, NIU G, SUGIYAMA M. Analysis of learning from positive and unlabeled data[J]. Advances in Neural Information Processing Systems, 2014, 1: 703 - 711.

[15] CHEN T, GUESTRIN C. XGBoost: a scalable tree boosting system [C]// Proceedings of the 2016 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2016: 785 - 794.

[16] ELKAN C, NOTO K. Learning classifiers from only positive and unlabeled data[C]// Proceedings of the 2008 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2008: 213 - 220.

[17] PEDREGOSA F, GRAMFORT A, MOCHEL V, et al. Scikit-learn: machine learning in Python[J]. Journal of Machine Learning Research, 2012, 12(10): 2825 - 2830.

[18] BUITINCK L, LOUPPE G, BLONDEL M, et al. API design for machine learning software: experiences from the scikit-learn project [J]. Eprint ArXiv, 2013, 2013: arXiv1309.0238.

[19] GitHub Inc. dmlc/xgboost[EB/OL]. [2018-05-27]. https://github.com/dmlc/xgboost/.

[20] YAYANA S. ODDS Library[EB/OL]. [2018-06-02]. http://odds.cs.stonybrook.edu.

[21] LÉCUN Y, BOTTOU L, BENGIO Y, et al. Gradient - based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278 - 2324.

[22] DUARTE M F, HU Y H. Vehicle classification in distributed sensor networks[J]. Journal of Parallel and Distributed Computing, 2004, 64(7): 826 - 838.

[23] PROKHOROV D. Slide presentation in IJCNN[EB/OL]. [2018-06-01]. http://www.geocities.ws/ijcnn/nnc_ijcnn01.pdf.

[24] DUA D, EFI K T. UCI Machine Learning Repository[EB/OL]. [2018-06-14]. http://archive.ics.uci.edu/ml.