Variable Ordering Selection for Cylindrical Algebraic Decomposition with Artificial Neural Networks

Changbo Chen[1,2(✉)], Zhangpeng Zhu[1], and Haoyu Chi[1,2]

[1] Chongqing Key Laboratory of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China
chenchangbo@cigit.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Cylindrical algebraic decomposition (CAD) is a fundamental tool in computational real algebraic geometry. Previous studies have shown that machine learning (ML) based approaches may outperform traditional heuristic ones on selecting the best variable ordering when the number of variables $n \leq 4$. One main challenge for handling the general case is the exponential explosion of number of different orderings when n increases. In this paper, we propose an iterative method for generating candidate variable orderings and an ML approach for selecting the best ordering from them via learning neural network classifiers. Experimentations show that this approach outperforms heuristic ones for $n = 4, 5, 6$.

Keywords: Cylindrical algebraic decomposition · Variable ordering · Machine learning · Neural network

1 Introduction

Cylindrical algebraic decomposition (CAD) was introduced by Collins for solving real quantifier elimination problems [14]. The original framework for computing CAD introduced by Collins is based on a projection and lifting scheme, which has now been gradually improved by many others [1,3,5,19–23]. In 2009, Moreno Maza, Xia, Yang and the first author [13] proposed a new way for computing CAD, which first computes a cylindrical decomposition of complex space and then transforms it into a CAD of real space based on the technique of triangular decompositions and regular chains [13]. Its efficiency was substantially improved in [7] based on an incremental algorithm, which can also take advantage of equational constraints. A complete and efficient algorithm for real quantifier elimination based on it was proposed in [12]. Moreover, it can utilize disjunctive equational constraints via computing a truth table invariant CAD [2].

Today, despite of its doubly exponential complexity [6,14], CAD has been efficiently implemented in many softwares such as QEPCAD, Mathematica, REDLOG and Maple, and found wide applications in geometry theorem proving,

A. M. Bigatti et al. (Eds.): ICMS 2020, LNCS 12097, pp. 281–291, 2020.
https://doi.org/10.1007/978-3-030-52200-1_28

stability analysis of dynamical systems, control system design, verification of hybrid systems, program verification, nonlinear optimization, automatic parallelization, and so on. Recently, it also finds applications on studying quantum nonlocality [8].

The choice of variable ordering has been shown to have a great impact on the performance of CAD, both theoretically [6] and in practice [15]. Several heuristic methods for variable ordering selection have been proposed. In particular, two heuristic strategies [4,9] are implemented in the SuggestVariableOrder (SVO) command of the RegularChains library in Maple. On the other hand, it also becomes a natural option to predict the best variable ordering by the approaches of artificial intelligence, among which machine learning is a natural choice [16–18,24,25].

Existing work for selecting variable ordering by machine learning focus on the trivariate case. For more than three variables, it becomes more difficult to obtain sufficient labelled data due to the doubly exponential behavior of CAD in terms of the number of variables n. Another difficulty is the exponential explosion of number of different orderings when n increases. In this paper, we first propose an iterative approach for generating a better variable ordering starting from the one given by SVO. Then we reduce the potential $n!$ number of classes to predict for the variable ordering problem to n by training a neural network classifier with data generated by the iterative approach. Experiments show that both the iterative approach and the machine learning approach outperform SVO for $n = 4, 5, 6$.

The organization of the paper is as follows. In Sect. 2, we briefly review the concept of CAD and the problem of variable ordering selection. In Sect. 3 and Sect. 4, we present respectively the iterative and the machine learning approaches. In Sect. 5, we show the effectiveness of our approaches by experimentation. Finally, we draw the conclusion in Sect. 6.

2 Cylindrical Algebraic Decomposition

Consider a set of polynomials $F \subset \mathbb{Q}[x_1, \ldots, x_n]$ and a variable ordering $x_{i_1} > \cdots > x_{i_n}$. An F-invariant cylindrical algebraic decomposition (CAD) partitions $\mathbb{R}^n$ into disjoint and cylindrically arranged semi-algebraic subsets (called cells) such that the projection of any two cells onto $\mathbb{R}^k$ (with coordinate variables $x_{i_{n-k+1}}, \ldots, x_{i_n}$), $1 \leq k \leq n-1$, is either disjoint or identically equal. The variable ordering also specifies the order to eliminate variables and the order to construct CAD from projection factors or a complex cylindrical decomposition.

The algorithms presented in [2,10,13] for computing CADs based on regular chains have been implemented in the command CylindricalAlgebraicDecompose in the RegularChains library of Maple [11]. Its latest version is available from http://www.regularchains.org and we use the version from http://www.arcnl.org/cchen/software/cadorder. In the RegularChains library, the command SuggestVariableOrder (SVO for short) implements two different heuristic methods for variable ordering selection, namely the one by Brown [4] (with option "decomposition = cad", SVO(B) for short) and the one by Chen et al. [9] (default option, SVO(C) for short).

The variable ordering plays an important role in the efficiency of computing CAD, as illustrated by the following example.

Example 1. *Let* $F := \{68\, x_1{}^2 - 12\, x_3\, x_2 + 46\, x_3 - 126, -54\, x_2\, x_1 + 11\, x_1 + 92\, x_2 - 21, -60\, x_3\, x_1{}^2 - 42\, x_3\, x_2\, x_1 + 45\, x_4{}^2 - 35\}$. *Table 1 lists the computation times and number of cells for several variable orders. As we can see, for this example, current heuristic methods avoid picking the worst variable order, but also miss the best variable order.*

Table 1. Impact of different variable orders

Order	Method	Timing (seconds)	#cells
$x_4 \succ x_3 \succ x_2 \succ x_1$	–	5	3373
$x_3 \succ x_1 \succ x_4 \succ x_2$	–	93	43235
$x_2 \succ x_3 \succ x_4 \succ x_1$	SVO(B)	16	11953
$x_3 \succ x_2 \succ x_4 \succ x_1$	SVO(C)	14	9253

3 An Iterative Method

In this section, we present an iterative variable ordering selection method, called IVO, for cylindrical algebraic decomposition. It starts with an initial ordering $x_{i_1} > x_{i_2} > \cdots > x_{i_n}$ provided by SVO. Then it calls a subroutine, called RVO, to generate n orderings in a round-robin manner and picks the best by calculating the shortest time of computing CAD with them. Next, it fixes the largest variable and calls RVO again on the rest ones to select the second largest variable, and so on. Precise description of IVO and RVO are given as below. We denote by $||$ the concatenation of two sequences.

- **Algorithm** RVO.
- Input: a set of polynomials F; a sequence of variables O defining a descending ordering; the time t for running $\mathsf{CAD}(F, O)$, an integer k.
- Output: a new ordering O' and running time t' of $\mathsf{CAD}(F, O')$ such that $t' \leq t$.
- Steps:
  1. Let $P := O_1, \ldots, O_k$ and $Q := O_{k+1}, \ldots, O_n$.
  2. Let $Q^{(i)} := Q_i, Q \setminus \{Q_i\}$, $i = 1, \ldots, |Q|$.
  3. For each $Q^{(i)} \neq Q$, $i = 1, \ldots, |Q|$, call $\mathsf{CAD}(F, P||Q^{(i)})$, $Q^{(i)} \neq Q$ and record the running times.
  4. Compare these running times with t and return the shortest one and the corresponding order.

```
– Algorithm IVO.
– Input: a set of polynomials $F$; a sequence of variables $X$.
– Output: a permutation of $X$, which defines a descending variable order.
– Steps:
    1. If $X \leq 1$, return $X$.
    2. Let $O_B := \mathsf{SVO}(F, X, B)$ and $O_C := \mathsf{SVO}(F, X, C)$.
    3. Let $t$ be the shorter running time between $\mathsf{CAD}(F, O_B)$ and $\mathsf{CAD}(F, O_C)$
       and let $O$ be the corresponding order with shorter time (if equal, we use $O_B$).
    4. For $k$ from 0 to $n-2$ do
        (a) $O, t := \mathsf{RVO}(F, O, t, k)$.
    5. Return $O$.
```

It is easy to see that IVO calls CAD at most $2 + \sum_{k=1}^{n}(k-1) = (n^2 - n + 2)/2$ times. In the rest of this paper, if no confusion arises, we denote by $\mathsf{SVO}(*)$ an oracle that always returns the better ordering between $\mathsf{SVO}(B)$ and $\mathsf{SVO}(C)$ and by SVO either of the three.

4 A Machine Learning Approach

To train a useful machine learning model for predicting the best variable ordering for CAD, it is important to have a dataset of enough labelled examples and the size of the dataset cannot be too small. On the other hand, since computing CAD is expensive when the number of variables is larger than 3, a larger dataset demands more computing resources. To make the learned model useful, it is better that the training dataset contains diverse examples. On the other hand, if the data are too diverse, it will be hard to learn. The following table summarizes the information of the dataset we generate using random polynomials as input to IVO. The whole dataset is divided into three datasets, used respectively for training, validation and testing with ratio 9/1/1. The validation dataset is used for tuning the machine learning model while the testing dataset is treated as unseen data used only once for reporting experimental results in the paper and showing the generalization ability of the ML model.

Table 2. Dataset

n	Degree	#terms	#polynomials	Equations	#valid examples
4	2..3	2..5	2	No	10957
5	2..3	3..6	2	No	6875
6	2..3	4..6	2	No	3751

The data in Table 2 were generated on a cluster (4 compute nodes, each of which has two Intel E5-2620 CPU (6-core each) and 64 GB memory). On each node, 6 Maple sessions were run in parallel. The time limit is set as 15 min. The total time for generating the dataset is about 1 month. In Table 2, we only record the number of valid examples. An example is valid if CAD finishes the computation within the time limit for at least one ordering computed by IVO. Note that it is possible that SVO returns an ordering for which CAD times out.

Next, we recall the features to represent the polynomials introduced in our earlier work [24]. These features are generated based on a graph structure defined for polynomial systems. For a given variable x_i, $i = 1, \ldots, n$, an equivalent description of the features is summarized in the following Table 3. Let $E(i)$ be the features associated with x_i. Then the feature vector $E = \cup_{i=1}^{n} E(i)$.

Table 3. Features

Feature	Description
$E_1(x_i)$	$\|\{x_j : x_j, j \neq i,$ appears in the same polynomial as $x_i\}\|$
$E_2(x_i)$	$\|\{f \in F : x_i$ appears in $f\}\|$
$E_3(x_i)$	$\max_{f\in F}\{\deg(f, x_i)\}$
$E_4(x_i)$	$\sum_{f\in F} \deg(f, x_i)$
$E_5(x_i)$	$\max_{f\in F}\{\deg(\mathrm{lc}(f, x_i))\}$, where lc denotes for leading coefficient
$E_6(x_i)$	$\max_{f\in F}\{\|\{M : M$ is a monomial of f and $x_i\|M\}\|\}$
$E_7(x_i)$	$\max_{f\in F}\{\deg(M) : M$ is a monomial of f and $x_i\|M\}$
$E_8(x_i)$	$\sum_{f\in F}\sum_M$ is a monomial of f and $x_i\|M \deg(M, x_i)$
$E_9(x_i)$	$\sum_{f\in F}\{\deg(\mathrm{lc}(f, x_i))\}$
$E_{10}(x_i)$	$\sum_{f\in F} \|\{M : M$ is a monomial of f and $x_i\|M\}\|$

We aim to train a model which can predict variable orders for $n = 4, 5, 6$. Instead of treating a variable order as a class, which may lead to huge number of classes for a fixed n, we would like to train a multiclass classifier M_n, which only predicts the largest variable in an ordering. To achieve this, for each example in the dataset, we will call SVO to return an initial ordering, and then call RVO once to get a hopefully better ordering. Then the first variable in the ordering

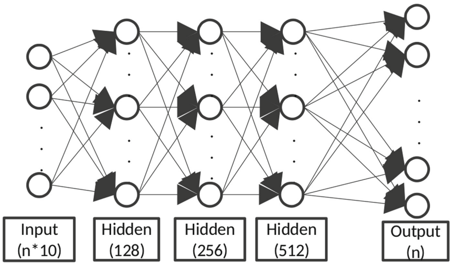

Fig. 1. The neural network classifier

is set as the label of the example. For each $n = 4, 5, 6$, we train an artificial neural network classification model implemented in TensorFlow. The structure and parameters of the neural network are illustrated in Fig. 1. Each M_n is a full-connected neural network with one input layer, three hidden layers and one softmax output layer. The activation function used is ReLu. Hyperparameters of the network are hand-tuned to maximize validation accuracy.

Suppose that we have obtained the well-trained models M_n, $n = 4, 5, 6$. We then employ the following procedure PVO to predict the variable ordering.

```
– Algorithm PVO
– Input: a set of polynomials $F$; a sequence of $n$ variables $X$.
– Output: a permutation of $X$, which defines a descending variable order.
– Steps:
    1. Compute a sequence of feature vectors $E = E(1), \ldots, E(n)$ for $F$.
    2. Let $O := \mathsf{SVO}(F, X)$.
    3. Let $x_i := M_n(E)$.
    4. Return $O \setminus \{x_i\}$.
```

The overall training and predicting process is depicted in Fig. 2.

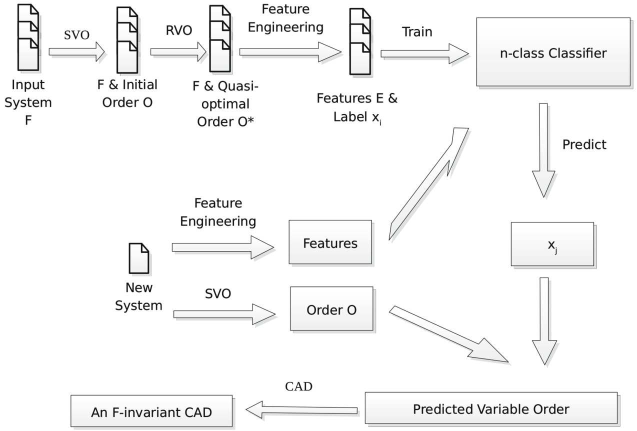

Fig. 2. The flow graph for finding variable order based on an artificial neural network.

5 Experiments

In this section, we report on the experimental results of the iterative method and the machine learning approach for selecting variable orderings.

Note that when we call PVO to predict the variable ordering, we have three options. One can use SVO(B) or SVO(C) to get an initial ordering without calling CAD. Or one can use SVO($*$) to get the better one between SVO(B) and SVO(C), but this requires calling CAD twice. Nevertheless, for $a = B, C, *$, we use IVO(a), RVO(a) and PVO(a) to denote the iterative method, the first iteration of the iterative method and the ML-based method for variable ordering selection which uses SVO(a) as an initial ordering. As a result, the testing dataset for $a = B, C, *$ is respectively the set of examples, for which SVO(a) provides the initial ordering in the original testing dataset. Table 4 summarizes the size of the datasets. Note that the dataset for $a = C$ does not contain timeout examples. This is because whenever CAD times out with the ordering given by SVO(C), IVO will use the ordering given by SVO(B). Although the testing datasets and inference procedures for $a = B$ and $a = C$ are different, for a given n, both $a = B$ and $a = C$ use the same classifier trained with the dataset in Table 2, where the examples are labelled by RVO(*) with an initial variable ordering provided by SVO(*). Table 5 summarizes the average computation times (in seconds) and timeout rates of IVO(*) and RVO(*) for the whole datasets. Note that the datasets only contain examples on which IVO(*) succeeds.

Table 4. Size of testing datasets

n	B	C	*	n	B	C	*	n	B	C	*
4	593	404	997	5	359	266	625	6	214	127	341

Table 5. Comparison between IVO(*) and RVO(*)

n	RVO(*) (time)	IVO(*) (time)	RVO(*) (timeout)	IVO(*) (timeout)
4	7.46	2.25	0.5225	0
5	38.45	9.91	0.3025	0
6	63.69	18.36	0.1954	0

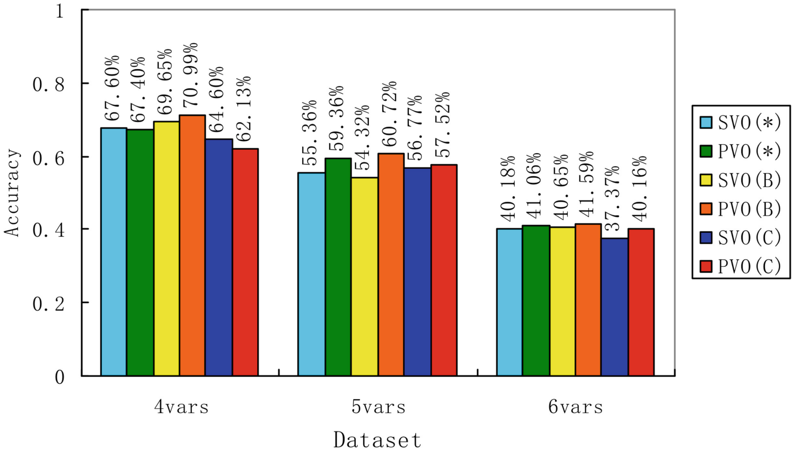

Fig. 3. Accuracies of different order selecting methods

Figure 3 summarizes the accuracies of six ordering selecting methods. Here the ground truth for SVO(a) and PVO(a) is given by RVO(a), for $a = B, C, *$. The accuracy is defined as the percentage of best orderings chosen by each method over the total number of test examples, given in Table 4. We observe that the accuracy of PVO(a) is higher than SVO(a) for $n = 5, 6$, for all $a = B, C, *$. For $n = 4$, the accuracy of PVO(a) is slightly lower than SVO(a) for $a = C, *$ and higher than SVO(a) for $a = B$.

Figure 4 provides the average running time of CAD with the variable orderings predicted by different methods. If there is a timeout, the time is counted as twice the time limit, that is half an hour. For $n = 4, 5, 6$ and $a = *, B$, the average computation time of PVO(a) is considerably less than SVO(a). The average computation time of PVO(C) is less than SVO(C) for $n = 4, 6$ but greater than SVO(C) for $n = 5$.

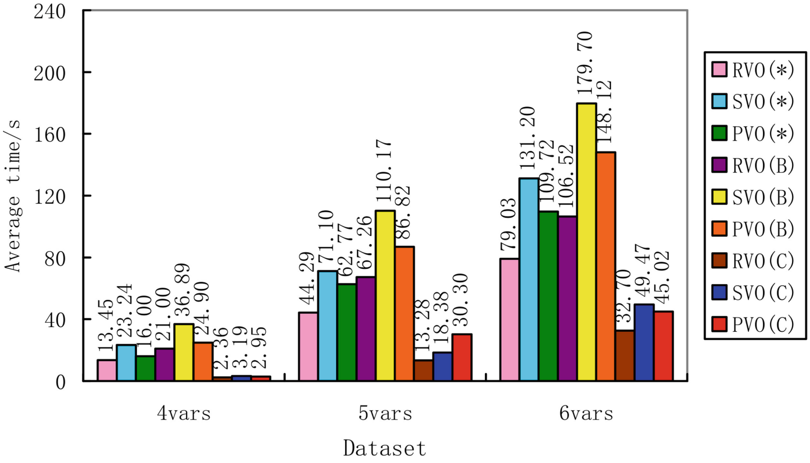

Fig. 4. Average running time of CAD with different variable orderings

Figure 5 gives the percentage of timeout examples for CAD with the variable orderings predicted by different methods. The phenomenon here is similar to the computation time as illustrated in Fig. 4.

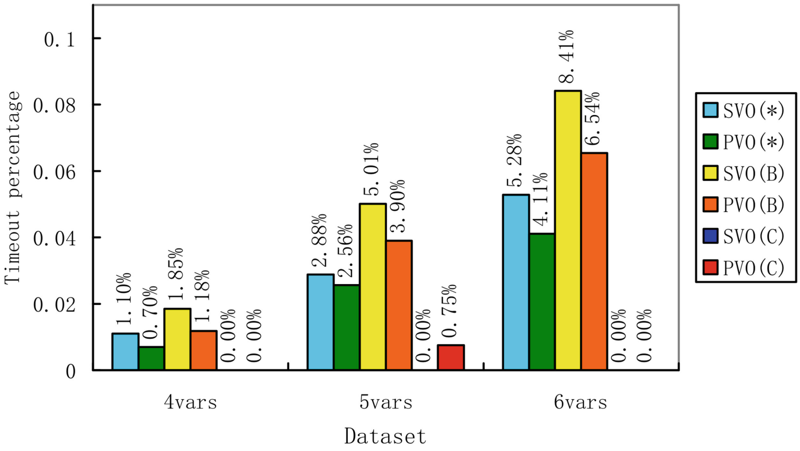

Fig. 5. Percentage of timeout examples for CAD with different variable orderings

6 Conclusion and Future Work

In this paper, we propose a machine learning based approach for predicting the best variable ordering for CAD targeting on $n > 3$. The experiments show that it outperforms traditional heuristic approaches for $n = 4, 5, 6$ on randomly generated datasets. The CylindricalAlgebebraicDecompose command can compute CAD for even larger n, say $n \leq 10$ in reasonable time if there are several equational constraints in the system [7]. It will be interesting to extend the model for $n > 6$, test it on CAD-QE problems from real applications and finally make the ML-based variable ordering selection method a useful option for SuggestVariableOrder. The data and code used in this paper is available at http://doi.org/10.5281/zenodo.3818086.

Acknowledgments. The authors would like to thank anonymous referees for helpful comments. This research was supported by NSFC (11771421, 11671377, 61572024), CAS "Light of West China" Program, the Key Research Program of Frontier Sciences of CAS (QYZDB-SSW-SYS026), and cstc2018jcyj-yszxX0002 of Chongqing.

References

1. Arnon, D.S., Collins, G.E., McCallum, S.: Cylindrical algebraic decomposition I: the basic algorithm. SIAM J. Comput. **13**(4), 865–877 (1984)
2. Bradford, R., Chen, C., Davenport, J.H., England, M., Moreno Maza, M., Wilson, D.: Truth table invariant cylindrical algebraic decomposition by regular chains. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) CASC 2014. LNCS, vol. 8660, pp. 44–58. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10515-4_4
3. Bradford, R.J., Davenport, J.H., England, M., McCallum, S., Wilson, D.J.: Truth table invariant cylindrical algebraic decomposition. J. Symb. Comput. **76**, 1–35 (2016)

4. Brown, C.: Tutorial: Cylindrical algebraic decomposition, at ISSAC (2004). http://www.usna.edu/Users/cs/wcbrown/research/ISSAC04/handout.pdf
5. Brown, C.W.: Improved projection for cylindrical algebraic decomposition. J. Symb. Comput. **32**(5), 447–465 (2001)
6. Brown, C.W., Davenport, J.H.: The complexity of quantifier elimination and cylindrical algebraic decomposition. In: Proceedings of ISSAC, pp. 54–60 (2007)
7. Chen, C., Moreno Maza, M.: An incremental algorithm for computing cylindrical algebraic decompositions. In: Feng, R., Lee, W., Sato, Y. (eds.) Computer Mathematics, pp. 199–221. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43799-5_17
8. Chen, C., Ren, C., Ye, X.J., Chen, J.L.: Mapping criteria between nonlocality and steerability in qudit-qubit systems and between steerability and entanglement in qubit-qudit systems. Phys. Rev. A **98**(5), 052114 (2018)
9. Chen, C., et al.: Solving semi-algebraic systems with the RegularChains library in Maple. In: Proceedings of MACIS, pp. 38–51 (2011). in the long version
10. Chen, C., Moreno Maza, M.: Algorithms for computing triangular decomposition of polynomial systems. J. Symb. Comput. **47**(6), 610–642 (2012)
11. Chen, C., Moreno Maza, M.: Cylindrical algebraic decomposition in the `RegularChains` library. In: Hong, H., Yap, C. (eds.) ICMS 2014. LNCS, vol. 8592, pp. 425–433. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44199-2_65
12. Chen, C., Moreno Maza, M.: Quantifier elimination by cylindrical algebraic decomposition based on regular chains. J. Symb. Comput. **75**, 74–93 (2016)
13. Chen, C., Moreno Maza, M., Xia, B., Yang, L.: Computing cylindrical algebraic decomposition via triangular decomposition. In: Proceedings of ISSAC, pp. 95–102 (2009)
14. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 134–183. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07407-4_17
15. Dolzmann, A., Seidl, A., Sturm, T.: Efficient projection orders for CAD. In: Proceedings of ISSAC, pp. 111–118. ACM (2004)
16. England, M., Florescu, D.: Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition. In: Kaliszyk, C., Brady, E., Kohlhase, A., Sacerdoti Coen, C. (eds.) CICM 2019. LNCS (LNAI), vol. 11617, pp. 93–108. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23250-4_7
17. Florescu, D., England, M.: Improved cross-validation for classifiers that make algorithmic choices to minimise runtime without compromising output correctness. In: Slamanig, D., Tsigaridas, E., Zafeirakopoulos, Z. (eds.) MACIS 2019. LNCS, vol. 11989, pp. 341–356. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43120-4_27
18. Huang, Z., England, M., Wilson, D.J., Bridge, J.P., Davenport, J.H., Paulson, L.C.: Using machine learning to improve cylindrical algebraic decomposition. Math. Comput. Sci. **13**(4), 461–488 (2019)
19. Lazard, D.: An improved projection for cylindrical algebraic decomposition. In: Bajaj, C.L. (ed.) Algebraic Geometry and Its Applications, pp. 467–476. Springer, New York (1994). https://doi.org/10.1007/978-1-4612-2628-4_29
20. McCallum, S.: An improved projection operator for cylindrical algebraic decomposition. In: Caviness, B., Johnson, J. (eds.) Quantifier Elimination and Cylindrical Algebraic Decomposition. Texts and Monographs in Symbolic Computation, pp. 242–268. Springer, Vienna (1998)

21. McCallum, S., Parusiński, A., Paunescu, L.: Validity proof of Lazard's method for CAD construction. J. Symb. Comput. **92**, 52–69 (2019)
22. Strzeboński, A.: Solving systems of strict polynomial inequalities. J. Symb. Comput. **29**(3), 471–480 (2000)
23. Strzeboński, A.: Cylindrical algebraic decomposition using validated numerics. J. Symb. Comput. **41**(9), 1021–1038 (2006)
24. Zhu, Z., Chen, C.: Variable order selection for cylindrical algebraic decomposition based on machine learning. J. Syst. Sci. Math. (2018, accepted). (in Chinese)
25. Zhu, Z., Chen, C.: Variable ordering selection for cylindrical algebraic decomposition based on a hierarchical neural network. Comput. Sci. (2020, accepted). (in Chinese)