# Experiment Section

June 19, 2020

## 1 Architecture

**Multiphase textual description** The work flow of learning multiphase ranking functions utilizes nested learning as a sub-procedure. The algorithm takes given program and default/user-set depth bound as input, search for linear ranking functions and decreasing functions under the depth bound. Multiphase learning also maintains a linear templates set for learning. If searching for multiphase ranking functions fails, it backtracks and changes template used for new learning. The work flow output terminating and non-terminating statement similar to that of nested learning and return unknown result if all templates are tried in the searching.

When it comes to template setting, each template is encoded as a python array. Where each component represents a monomial and we also use python array whose length is the number of variable plus 1 to represent monomial, and the component represent the power of corresponding variable. For example, if there are two variable $x, y$. We can use $[[s_1, s_2, b_1], [t_1, t_2, b_2]]$ to represent polynomial $b_1(x^{s_1}y^{s_2}) + b_2(x^{t_1}y^{t_2})$. For nested learning, we generate different polynomial templates. For multiphase learning, the templates are only restricted to $\{ax_i + c_2, \sum_i^{\texttt{varNum}} b_i x_i + c_2\}$ where the set of variables is $\{x_i \mid i \in \texttt{varNum}\}$

For optimization, our efforts are mainly devoted to reduce the time in sampling and in Z3 solving. We provide some preset strategies to relieve the time consumption. To effectively find the sampling space, we also use constraint solving to find a sample point then sample randomly near that point. Besides, for multiphase learning, we add option `--template_strategy` for user to balance the time of searching and effectiveness of the algorithm.

**Optimization options and techniques**

## 2 Experimental Evaluation

After adding the multi-phase ranking function learning algorithm to SVMRanker, we conducted experiments on constructed dataset of loop programs to compare the

|  | SVMRanker |  |  | LassoRanker |
|---|---|---|---|---|
|  | Nested | 2-Multi | 4-Multi |  |
| Terminating | 54 | 72 | 78 | 72 |
| Non-teminating | 50 | 50 | 50 | 52 |
| Unknown | 126 | 108 | 102 | 106 |
| Total Time | 196s | 1183s | 4693s | 2053s |

Table 1: General Experiment Results

algorithms of learning nested ranking function, of multi-phase ranking function and algorithm in the state-of-the-art tool LassoRanker embedded in Ultimate Automizer. Our dataset are 230 Boogie programs, where 96 programs are converted from library of sv-comp**??** and 134 loop programs in Boogie are grabbed from repository of Ultimate Automizer. For the configuration of experments, we use a server with a 3.6 GHz Intel Core i7-4790 CPU and 16GB RAM and timeout is set to 300 seconds for each case.

## 2.1 Overview of the Experiments

As shown in Table 1, we use our 230 examples as inputs to SVMRanker and LassoRanker. For SVMRanker, we use “Nested, 2-Multi, 4-Multi” to represent the sythesising of nested ranking function, 2-phases-bounded and 4-phases-bounded multiphase ranking function respectively. lx: Add description of the bound if the phase bound is not mentioned previously :xl In the experiment, the algorithm tries different number of phases of nested ranking function according to the result of the learning. Furthermore, we use linear and non-linear templates for nested ranking function and use only linear templates for multiphase learning.

From Table 1, it is obvious our new algorithm for multiphase r.f. is more powerful and can solve more cases than that of nested r.f. learning. From the comparation between “2-Multi” and “4-Multi”, it is clear that larger bound on phases is given, more ranking functions can be found. Besides, since the number of terminating cases of multiphase learn is almost the same as LassoRanker, we can tell that our multiphsae learning algorithm enhance the capability of SVMRanker to deal with linear loops programs in our data set.

As for the total running time of this experiment. The total time of multiphase r.f. learning is much more than nested r.f. learning. This can be attribute to the backtracking and incremental learning of multiphase.lx: Add description of backtracking and incremental learning if the phase bound is not mentioned previously :xl Thought the long running time, we are still optimistic about our tool that it solves the same number of terminating cases as LassoRanker but only uses about half of their time.

|  | Nested | 2-Multi | 4-Multi |
|---|---|---|---|
| Sampling time | 14.9s | 59.6s | 170.3s |
| Training time | 7.2s | 65.1s | 186.9s |
| Z3 Solving time | 24.9s | 278.6s | 1965.7s |

Table 2: Detailed Time Cost of SVMRanker

|  | Non-linear loop programs |
|---|---|
| SVMRanker | 13 |
| LassoRanker | 0 |
| Total number | 19 |

Table 3: Solved Number of Non-linear Loops

## 2.2 Detailed Evaluation

### 2.2.1 *(a) Time cost of each stage.*

To better illustrate the time cost, we record sampling time, training time and Z3 sovling time of the experiments as shown in Table 2. Time costs all increase fastly when the depth bound of multiphase learning increases. Most of the time are used on solving Z3 formulae to check the satisfiability of requirements on learned functions. In the future, since our multiphase templates are all linear templates, we wish to improve the result by using more efficient LP tools like CPLEX and PuLP**??**.

### 2.2.2 *(b) Advantage on non-linear loops.*

According to Table 3, we summarized the non-linear cases solved by our tool and LassoRanker. We state that our tool performs better than LassoRanker on non-linear loops. This is because the latter only utilize linear templates while our algorithm use non-linear templates as well.

lx: exp section is slightly long, to cut:xl

## 3 Conclusion

: In this paper we give theoretical background, architecture and experimental results of our tool SVMRanker. The successful application of SVM on synthesising nested and multiphase ranking functions and the development of this tool demonstrate the promising future of learning more kinds of ranking functions such as compositional ranking functions mentioned in [**?**]. During the experiment, though we do not found significant advantage of our tool on linear loops, our tool is still a good sumplement to LassoRanker on learning of non-linear loops.

As future work, we plan to add more functionalities into our tool. For example, we wish to use recurrent set as a stronger way to find the non-termination and use LP tools to relieve the long solving time on linear templates. Besides, we also intend to enlarge the types of programs our tool supports, such as nondeterminism and nested loops. On theoretical aspect, whether we can use neural networks on termination verification is also worth exploring.