An algorithm selection approach for QF_FP Solvers

Joseph Scott, Pascal Poupart, Vijay Ganesh

{joseph.scott,ppoupart,vganesh}@uwaterloo.ca

University of Waterloo, Ontario, Canada

July 11, 2019

Algorithm Selection

There are lots of SMT Solvers out there.

1. Alt-Ergo
2. AProVE
3. Boolector
4. Colibri
5. Ctrl-Ergo
6. CVC4
7. MathSAT
8. Minkeyrink
9. OpenSMT2
1. Q3B
2. SMTInterpol
3. SMTRAT
4. SPASS-SATT
5. STP
6. Vampire
7. veriT
8. Yices
9. Z3

It can be very intimidating to figure out which one to use and when!!

Algorithm Selection or Portfolio

In the presence of a surplus of algorithms and solvers, it is very natural to ask which SMT tool to use for a particular input!

Algorithm Selection (or Portfolio): Given a set of tools or algorithms which do we use and when?

The problem statement is a classification problem! But can be formulated as a regression problem.

SatZilla

Xu et al. implemented a very competitive SAT Solver, SatZilla, that uses algorithm selection.

SatZilla

Xu et al. implemented a very competitive SAT Solver, SatZilla, that uses algorithm selection.

1. Won five medals in 2007! (3 gold!)

SatZilla

Xu et al. implemented a very competitive SAT Solver, SatZilla, that uses algorithm selection.

1. Won five medals in 2007! (3 gold!)
2. Won gold in all major categories in 2009!

SatZilla

Xu et al. implemented a very competitive SAT Solver, SatZilla, that uses algorithm selection.

1. Won five medals in 2007! (3 gold!)
2. Won gold in all major categories in 2009!
3. Won the SAT Challenge in 2012!

SatZilla

Xu et al. implemented a very competitive SAT Solver, SatZilla, that uses algorithm selection.

1. Won five medals in 2007! (3 gold!)
2. Won gold in all major categories in 2009!
3. Won the SAT Challenge in 2012!
4. Eventually got banned from the main tracks.

Algorithm Selection is very powerful for SAT! How about SMT?

Why QF_FP?

1. Relative to other theories, QF_FP is fairly new and undeveloped
2. QF_FP SMT has a lot of interesting applications!
    1. Verifying Scientific Software
    2. Verifying Machine Learning Models
3. Variance in algorithms!
    1. Eager bit blasting approaches, with multiple bit blasters
    2. Lazy approaches!
        1. Abstract CDCL by D'Silva et al. (Implemented in MathSAT)
        2. Marre et al. use interval analysis and difference-bound matrices (Implemented in Colibri)
        3. Fragments of FP SMT can be reduced to optimization problems by Fu et al. (Implemented in XSat)
4. Local Interest!

Supervised Learning

1. Supervised learning is a branch of machine learning where a dataset of features is provided with labels.
2. **Classification:** Learn a function $f : X \to \mathcal{C}$
3. **Regression:** Learn a function $f : X \to \mathbb{R}$

We can use regression for algorithm selection by learning an empirical hardness model for each solver! Learn a function that predicts the (log) runtime of every considered solver, and take the argmin

Learning Algorithms (1/2)

- Linear Regression - learns a linear polynomial with an objective function of minimizing the mean square error over the training set.
- Linear Ridge Regression - is an extension to Linear Regression that adds the norm of coefficients of the learned polynomial to the objective function.
- Support Vector Machines (SVM) - Support Vector Machines is a classifier (with a regression formulation SVR) that learns a hyperplane to separate classes such that the margin between points and the hyperplane is maximized.
- (k) Nearest Neighbors - is a classification algorithm (with regression formulations) that makes classification decisions by sampling the k closest points of the training set.
- Logistic Regression - A classification algorithm that infers the probability of membership of a class given the features.

Learning Algorithms (2/2)

- Linear Perceptron - A biologically inspired classifier that learns a linear hyper-plane that separates two classes. This can be generalized to multi-class by training one class against all for each considered class.
- Random Forests - Uses an ensemble learning approach over a 'forest' of several decision trees. Each decision tree votes on a class or regressed value and is propagated up to a final decision.
- Neural Networks - A biologically inspired algorithm that emulates a directed acyclic graph of neurons firing messages to one and another.

Features

Name	Description
N	Total number of occurrences of terms in the input (constants, variables, operators, predicates, assertions.)
N_c	Total number of constants
N_v	Total number of variables
N_{op}	Total number of operators
N_{pred}	Total number of predicates
N_{assert}	Total number of assertions
$32-bit?$	If input contains at least one 32-bit float: 1.0, otherwise: 0.0
$64-bit?$	If input contains at least one 64 bit float: 1.0, otherwise: 0.0
$128-bit?$	If input contains at least one 128 bit float: 1.0, otherwise: 0.0
Variant	If the input contains at least one float that is not 32-bit, 64-bit, or 128-bit: 1.0, otherwise 0.0
$fp.abs\%$	$N_{fp.abs}/N_{ops}$, the percentage of $fp.abs$ over the total number of operands
$fp.neg\%$	$N_{fp.neg}/N_{ops}$, the percentage of $fp.neg$ over the total number of operands
$fp.add\%$	$N_{fp.add}/N_{ops}$, the percentage of $fp.add$ over the total number of operands
$fp.mul\%$	$N_{fp.mul}/N_{ops}$, the percentage of $fp.mul$ over the total number of operands
...	...
$fp.eq\%$	$N_{fp.eq}/N_{pred}$, the percentage of $fp.eq$ over the total number of predicates
$fp.lt\%$	$N_{fp.lt}/N_{pred}$, the percentage of $fp.lt$ over the total number of predicates
...	...

Considered Solvers

We will exclusively consider the following list of solvers:

1. **Z3** v4.8.0 - A multi-theory open source SMT solver by Microsoft Research. Z3 implements FP SMT by a reduction to arithmetic over bit-vectors for each FP operator.
2. **MathSAT5** v5.5.3. A multi-theory SMT solver from FBK-IRST and DISI-UniTN. MathSAT5 implements an Abstract Conflict Clause Driven Learning (ACDCL) algorithm for their FP solver. MathSAT5 additionally provides bit-blasting approaches, but in this paper we only consider the ACDCL configuration.
3. **CVC4** v1.7 - A multi theory open source SMT Solver by Stanford. CVC4 implements FP SMT similarly to Z3 by bit blasting FPU circuits.
4. **Colibri** v2070 - A proprietary CP Solver with specialty in FP SMT developed by CEA LIST.

We use a global timeout of 2500 seconds. If a solver has a runtime error of any kind the solver-input pair is labeled as a timeout.

Training and Evaluation

1. We train and evaluate over the 40,300 benchmarks from SMT-LIB.
2. Same benchmark set used in the SMT-COMP
3. We randomly partition into two sets with 50% of all data going into a training set and 50% into a testing set.
4. Training set features are scaled to zero mean and unit deviance.
5. 20% of the training set is initially reserved as a validation dataset to determine the hyperparameters of the algorithm. Then retrained over the entire training set with the highest scoring hyper parameters.

Solvers were ran on the Compute Canada (SHARCNET) service. CentOS V7 Intel Xeon Processor E5-2683 at 2.10 GHz. Each run of a solver was configured to be restricted to 8GB sequentially. Otherwise, solvers were run as close to their default configurations as possible. We observe prediction times to take a few milliseconds at most and are not included in timing analysis.

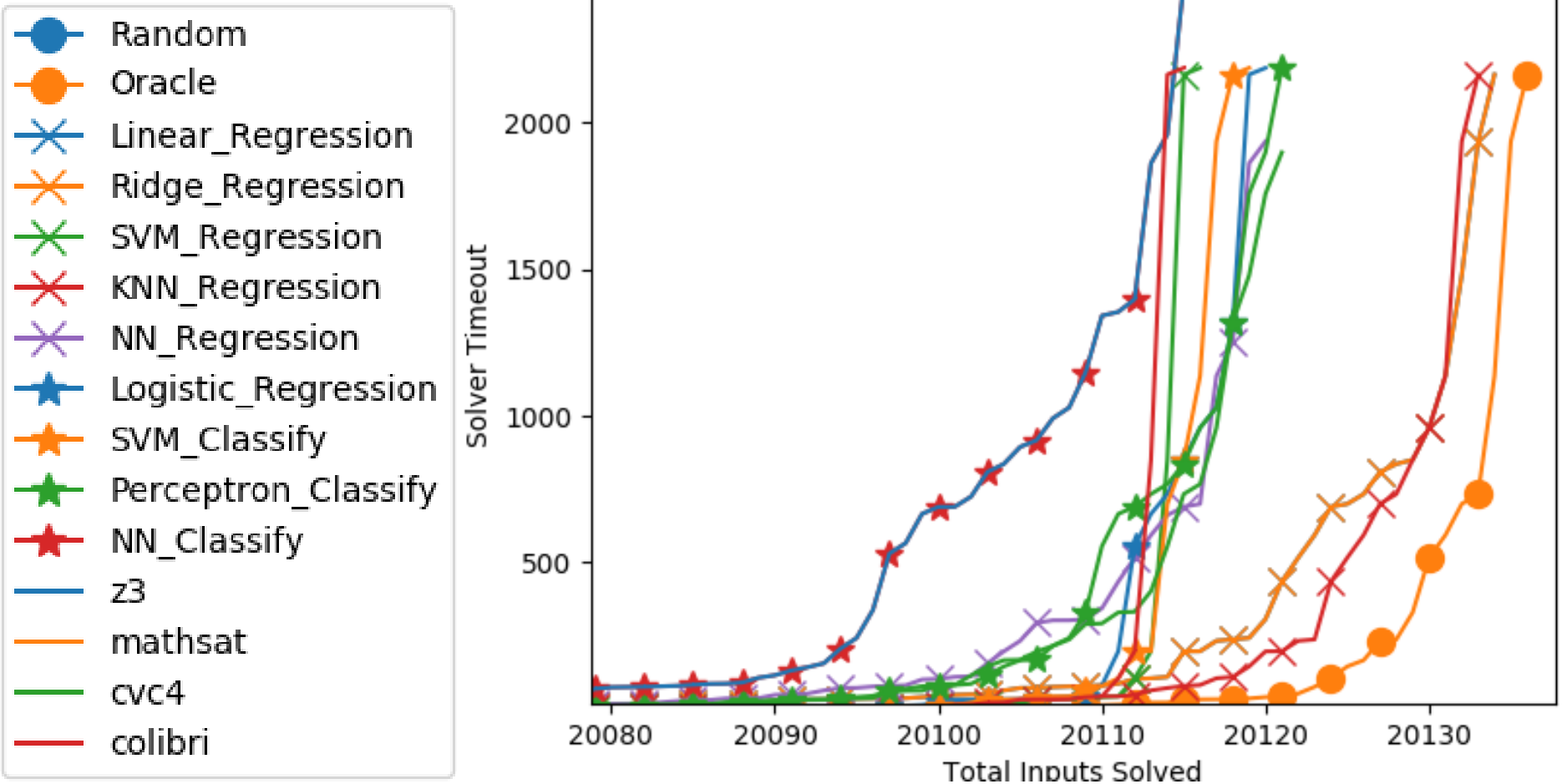
Baselines

We consider the following baselines to the considered algorithm selection models:

1. Each solver individually
2. A uniformly random algorithm selector
3. An Oracle that always picks the best solver

A learned algorithm selection model should improve on all individual solvers and random algorithm selection while being competitive with an Oracle.

Algorithm Selection over SMT-LIB Benchmarks



	Z3	MathSAT5	CVC4	Colibri
Z3	20010	13	0	6
MathSAT5	4	8	2	36
CVC4	8	4	6	16
Colibri	1	4	2	30

Comments on First Experiment

1. Highly accurate! 99.97%!
2. But the problem is very polarized. Z3 solves 99.2% within 2 seconds of the 2,500 second timeout.
3. A large chunk of the benchmarks are unit tests.

A new randomly generated benchmark set

To further study algorithm selection over QF_FP we create an additional randomly generated data set.

1. The input will be comprised of entirely 32-bit floats or 64-bit floats and is selected at the start uniformly at random.
2. The number of variables is chosen uniformly at random in the interval $[1, 20]$.
3. The input consists of $[1, 20]$ assertions with each asserting an Abstract Syntax Tree (AST) over floating-point arithmetic.
    1. The root node consists of one of the predicates selected uniformly at random. (fp.eq, fp.lt, fp.isSubnormal, etc.)
    2. Depending on the arity of the selected predicate, the required number subtrees are generated with roots of floating-point operators chosen uniformly at random (fp.abs, fp.mul, fp.sqrt, etc.)
    3. This process is repeated for a fixed net depth of $[2, 6]$ chosen uniformly at random to which floating point variables selected uniformly at random fill out the leaf nodes of the AST.

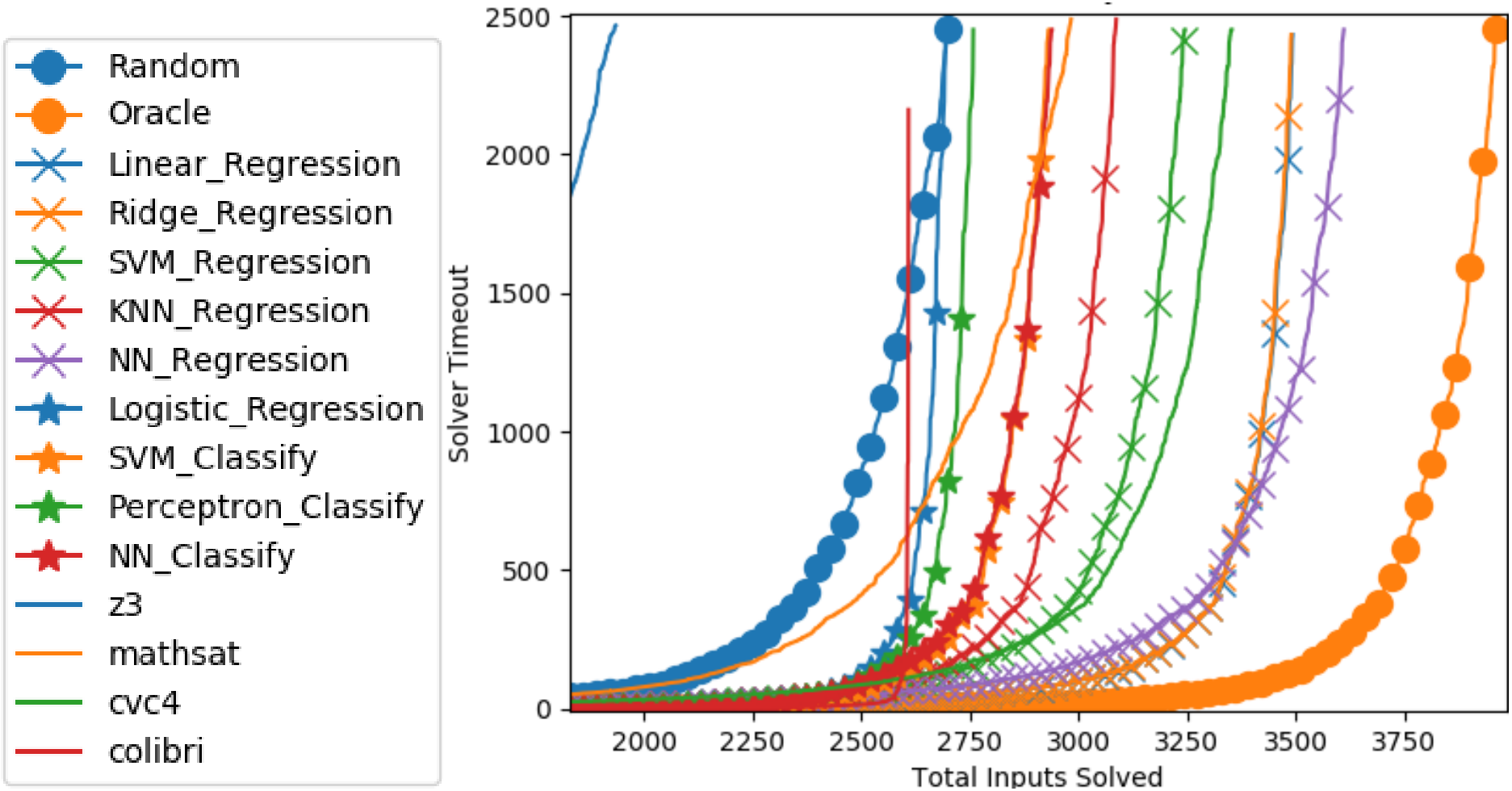
Randomly generated benchmark notes

1. The following were banned:
   1. fp.fma
   2. fp.rem
   3. RNA

   as they are not supported by all solvers.
2. The following remained turned on:
   1. fp.min
   2. fp.max
   3. fp.isPositive
   4. fp.isNegative
   5. fp.roundToIntegeral

   But inconsistent outputs were observed amongst solvers.
3. 10,000 inputs were generated but 2,095 were discarded as they were not solved by any considered solver.
4. Same Training and Analysis process as before!

Algorithm Selection over Randomly Generated Benchmarks



	Z3	MathSAT5	CVC4	Colibri
Z3	7	8	21	32
MathSAT5	14	142	205	479
CVC4	57	445	599	1007
Colibri	24	133	169	610

Comments on Second Experiment

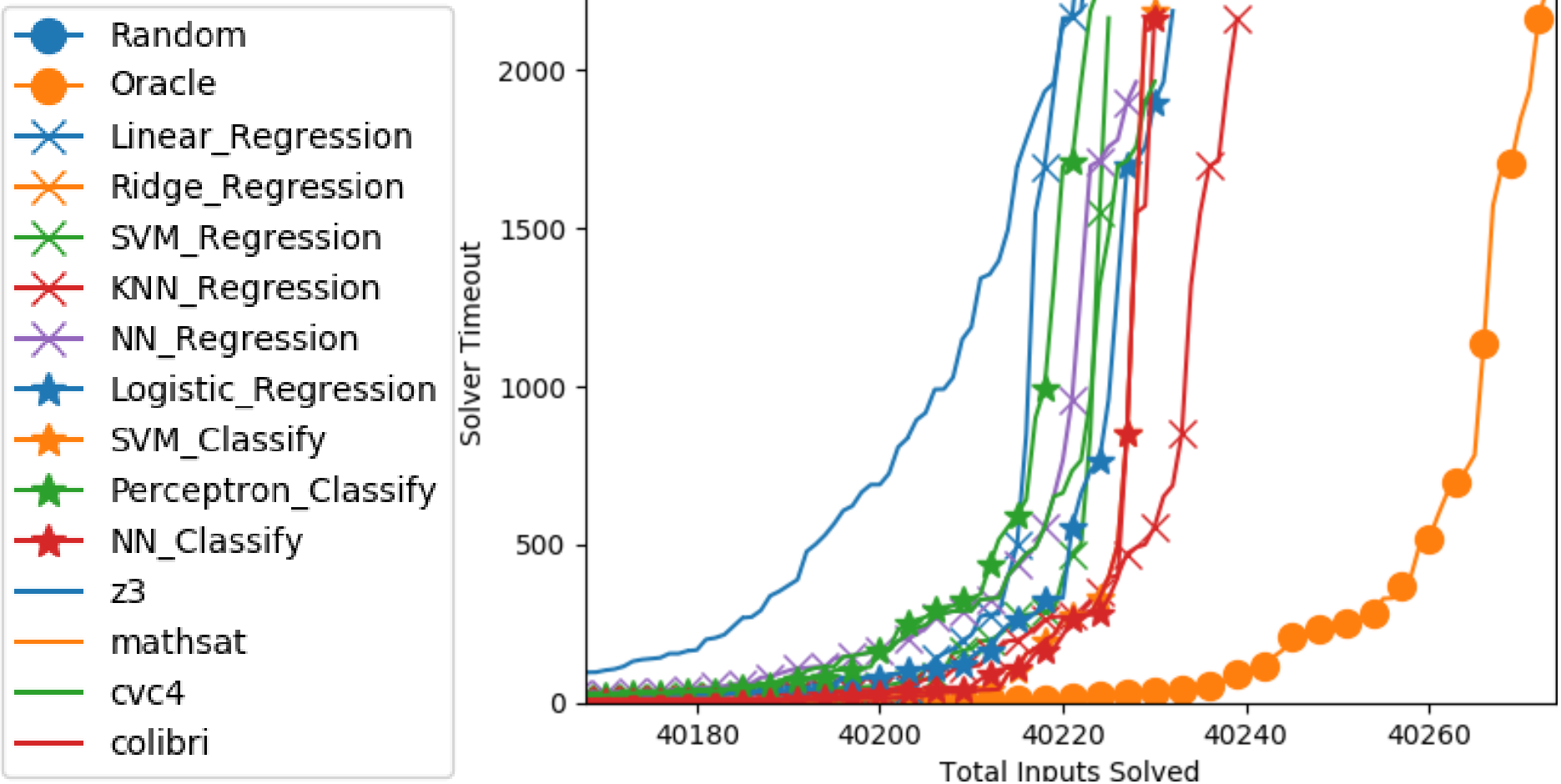
1. High performance dropoff! 36.4% accuracy!
2. Z3 went from being the best solver by a large margin to the worst by a notable margin.
3. Low accuracy, but still improves on any single solver! Fair margin of improvement to be competitive with an Oracle.

Comments on Second Experiment

1. High performance dropoff! 36.4% accuracy!
2. Z3 went from being the best solver by a large margin to the worst by a notable margin.
3. Low accuracy, but still improves on any single solver! Fair margin of improvement to be competitive with an Oracle.

Next experiment, Train over the entire randomly generated benchmark set, test over the SMT-LIB benchmark set.

Train: Randomly Generated Set, Test: SMT-LIB Set



	Z3	MathSAT5	CVC4	Colibri
Z3	34365	15	2	0
MathSAT5	5	9	1	22
CVC4	2878	10	12	33
Colibri	2801	30	17	100

Comments on Second Experiment

1. 85.6% accuracy!
2. See improvement over any individual, but once again, a notable margin away from an oracle.
3. Low accuracy, but still improves on any single solver! Fair margin of improvement to be competitive with an Oracle.

Conclusions

1. QF_FP SMT solvers have several interesting applications and algorithm selection can help in reducing runtimes!
2. Discussed algorithm selection models can situationally be close to an Oracle selector, but there still remains an observable margin
3. Ridge Regression remains dominant for algorithm selection!
4. Success of machine learning is very dependent on having good features! What else could we use?
5. Other SMT-LIB theories?
6. How would an algorithm selection solution perform in the SMT Competition?
7. I am happy to provide the randomly generated benchmark set (or fragments thereof) for the contest next year!

The end!

Thanks for your attention!

Any questions?

Email: joseph.scott@uwaterloo.ca