Oil Sands Processability Analysis Using Symbolic Regression

Yixin Zhang, Qing Zhao, Zhenghe Xu*

Abstract—Hot or warm water based bitumen production process from mineable oil sands is extremely complex in nature and highly sensitive to variability of oil sands ores and process conditions. Understanding ore processability and developing sensible markers for ore processability are considered to be very challenging tasks. In addition to processing variables such as temperature, hydrodynamics, process water chemistry and chemical additives, ore characteristics, such as bitumen content, connate water content, fines content and more importantly types of fines play a decisive role in determining the processability of oil sands ores. It is therefore valuable to analyze the processability of oil sands ore using statistical modelling approaches. In this paper, a symbolic regression method based on genetic programming is applied to help understand oil sands ore processability, such as identifying sensible markers of ore processability. In this paper, the analysis is conducted based on using three input variables. The model is expressed analytically by a combination of these input variables (oil sands ore characteristics and operation conditions) and a given set of math operators and constants. This model provides a convincing prediction for the response variables, e.g, bitumen recovery. The results show an agreement with simulation and experiment data, highlighting the applicability of the Symbolic Regression (SR) method in identifying a mathematical model to describe the mechanisms involved in oil sands processability.

Index Terms—Symbolic Regression, Genetic Algorithm, Genetic Programming, Kernel Methods, Oil Sands Recovery Prediction, System Identification

1 Introduction

A majority of bitumen produced in Canada is from the mineable oil sands [1]. Hot water based bitumen production process from mineable oil sands is extremely complex in nature and highly sensitive to variability of oil sands ores, which involves essentially three key steps: *(1)* Extraction of bitumen from oil sands, where the solids and water are being removed; *(2)* Upgrading of bitumen to an intermediate oil product; and *(3)* Refining of the crude oil into the final products. Understanding ore properties and developing a sensible marker for ore processability have been proven to be highly desirable but challenging. Not only process variables but ore characteristics also play an important role in determining the processability of oil sands ores. In addition, there are three main contributing factors for uncertainties in oil sands extraction [2]: *(1)* lack of on-line determination of complex oil sands composition; *(2)* lack of advanced setup for process control; and *(3)* malfunctions or failures of mechanical equipment.

The current technology for improving oil sands processability is mostly based on single factor analysis or factorial design, which in many cases is limited and has not yielded satisfactory commercial results. On the other hand, in recent years there have been considerable efforts and extensive development of machine learning and data driven techniques for process modeling and analysis. It is expected that these new techniques and knowledge will provide new means for tackling challenging tasks of oil sands process analysis. As a result, there is an urgent need for the oil sands industry to review and investigate these new trends and techniques. Using plant or Batch Extraction Unit (BEU) data, algorithms based on probabilistic programming and/or statistical data analysis have been proposed for the improved modeling of oil sands ore processing [3].

Genetic Algorithm is a particularly unique and useful family of techniques in data analysis. In recent years, there has been a significant amount of research on genetic algorithm that focuses on particular characteristics of both the data itself and the resulting factors and their associations. Symbolic Regression (SR), for example, focuses on identification of the analytical mathematical description of a hidden system from experimental data. The applications of SR algorithms have grown significantly during the past years. It has been shown that they could be a successful solution to dimensionality reduction modelling and optimization in a variety of areas including, but not limited to, environmetrics [4], microarray data analysis [5], [6], [7], document clustering [8], face recognition [9], [10], [11], blind audio source separation [12] and more. What makes SR algorithms particularly attractive is the fitness function constraints imposed on the factors they produce, allowing for better interpretability.

In order to extend the applicability of SR in cases where the dataset is inconclusive and has missing attributes [13] [14], we introduce self-evolved blockers that

- *Y. Zhang (yixin6@ualberta.ca) and Q. Zhao (qingz@ualberta.ca) are with the Dept. of Electrical and Computer Engineering, Univ. of Alberta, Edmonton, Alberta, Canada; Z. Xu (zhenghe@ualberta.ca) is with the Dept. of Chemical and Material Engineering, Univ. of Alberta, Edmonton, Alberta, Canada. *: the Corresponding author E-mail: zhenghe@ualberta.ca*

impose non-negativity constraints on the input factors, but allows mixed signs in both data attributes and the generically generated procedure. This was motivated from a biology perspective, where SR descriptor represents an indicator for every data point, allowing genetic algorithm to learn new lower-dimensional features from the data.

The novelty of this work can be summarised as follows: *(1)* we outline a preliminary SR framework for genetic algorithm suitable for prediction of oil sands ore processability analysis (Figure 1). We present a greedy and fast genetic algorithm to optimize the factors of the oil sands recovery prediction. *(2)* we evaluate the statistical performance results obtained by SR. *(3)* We propose a modification to the SR algorithm to incorporate certain partially known information/constraints of the input attributes of given dataset, in order to extract features that are more meaningful. *(4)* We demonstrate the procedure to improve the performance of SR.

In this work, the main objective is to identify significant markers for processability of mineable oil sands ores and deliver the results that can be compared and even integrated into current industrial analysis system. In our approach, several sensible markers controlling the ore processability are identified, and analytical relationship between these markers and the oil recovery rate will be established.The remainder of this paper is outlined as follows: In Section 2, the main methodology used for oil sands ores processability analysis is described. In Section 3, the symbolic model construction by the GP algorithm is discussed in details. In Section 4, the SR descriptor built based on both simulated and experimental data for oil sands ore is shown and the statistical performance analysis is performed. Finally in Section 5, the concluding remarks are given and the future work is discussed.

2 Methodology

2.1 Symbolic Regression via Genetic Programming

In this paper, the symbolic regression (SR) via genetic programming (GP) is applied to the study of oil sands processability. A genetic programming is an optimization procedure. From a given population X, it seeks the item $x \in X$, which has the greatest fitness. A genetic programming searches for the best value by creating a small pool of random candidates, selecting the best candidates, allowing them to breed, with minor variations, and finally repeating this process over many generations. These ideas are all inspired by the analogy with the evolution of living organisms [4].

A genetic programming typically includes:

- a genetic representation of candidates;
- a way to create an initial population of candidates;
- a function measuring the fitness of each candidate;
- a generation step, in which some candidates “die”, some survive, others reproduce by breeding;
- a mechanism that recombines genes from breeding pairs, and mutates others.

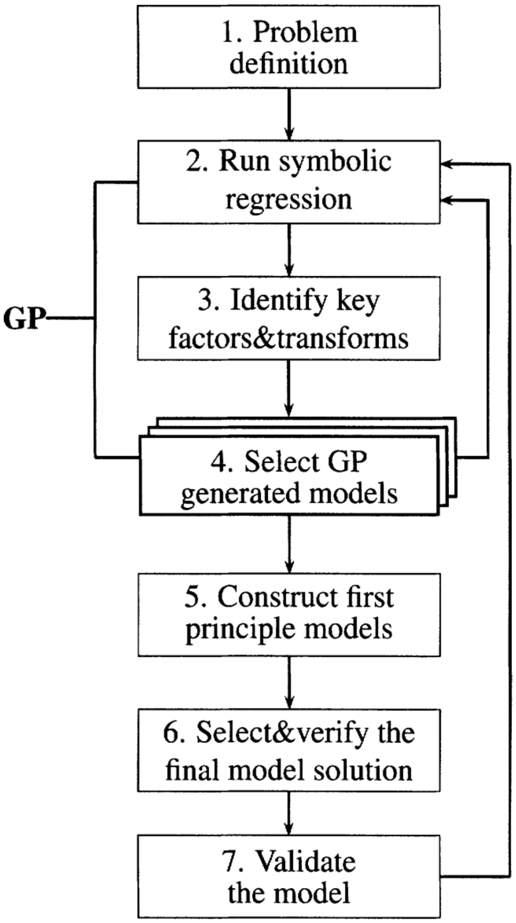
1. Problem definition
2. Run symbolic regression
3. Identify key factors&transforms
4. Select GP generated models
5. Construct first principle models
6. Select&verify the final model solution
7. Validate the model
GP

Fig. 1: Genetic Programming(GP) iteration procedure: The tree-based GP iterated procedure controls the evolved SR generation and destruct the unpromising offspring.

Symbolic regression is a method to search for a set of mathematical operators to identify an analytical description for the relationships among input and output attributes of the given data set. SR is an NP-hard problem but we can use genetic algorithm to find solutions. Moreover, a standard algorithm used for SR is genetic programming, which is specialized for evolving generation and tree structures, e.g, searching for a space of mathematical expressions and minimizing various error metrics. Both the parameters and the form of the equation are subject to search. In symbolic regression, many initially random symbolic equations compete via optimization to model experimental data in a most nontraditional way. New equations are formed by reorganizing and combining previous equations and probabilistically varying their sub-expressions. The algorithm leads to equations that model the experimental data and at the same time reject unpromising solutions. After an equation reaches a desired level of accuracy, the algorithm terminates, returning equations that may correspond to the intrinsic mechanisms of the initial dataset [15]. The overall procedure summarizing the above steps is shown in Figure 1.

In SR, the represented symbolic expressions are the combination of the genes often represented as a binary tree of algebraic operations with numerical constants and symbolic variables as its leaves. Others include acyclic graphs and grammars [16]. The fitness of a particular equation is a numerical measure of how well it agrees with the data, such as the correlation of equations or performance measurement of the experimental data.

The operations can be varied among *abs*, *exp* and *log*, or binary operations, *add*, *sub*, *multiply*, and *divide*. If prior knowledge of the initial value is known, which is the so-called domain knowledge, the types of operations available can be chosen ahead of time. The terminal values consist of input attributes of function variables and the constant values.

Mutation in a symbolic expression can change an operator in the binary tree, e.g. it can change the *add* functions into *sub*, change the arguments of an operation, e.g. it can change $x + c$ into $x + x$, delete an operation, e.g. it can change $x + x$ into x or add an operation, e.g. change $x + x$ to $x + (x + x)$. If the operator is changed from a binary operation to a constant, one of the two sub-tree branches is discarded and those branches can be chosen randomly.

Crossover of a symbolic expression exchanges subtrees in the binary trees of the initial expressions. For example, crossing $f(x) = x^2+c$ and $f(x) = x^2+sin(x)+x$ could produce a sub tree with $f(x) = x^2 + sin(x)$, from which the leaf node c was exchanged with the $sin(x)$ term.

2.2 Fitness Prediction and Constraint Optimization

Fitness function and constraint optimization should be performed by treating every constant as the parameter, which are tuned by an automatic differentiation algorithm [20]. The constraint optimization can also be treated as a fitting problem and a customized model with high performance is presented by minimizing an objective function $Q(\alpha)$, which is the sum of squared errors between the experimental data y and the prediction of the SR expression $f(x, \alpha)$ (Equation 1).

$$Q(\alpha) = \sum_{i=0}^{m}(y_i - f(x_i, \alpha))^2 \tag{1}$$

One of the conditions for SR to work is that differentiable functions such as logistic functions must be incorporated in the SR parsing tree, which encodes formula expression using the combination of the identified markers. Otherwise optimization solutions with one or multiple sets cannot be achieved by the gradient calculation. The gradient of this SR model can generate a continuous search direction, whose information can be used for accelerating this entire process.

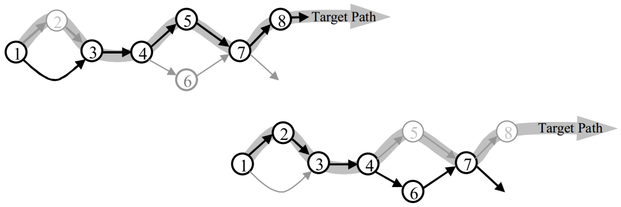

Fig. 2: The execution of the first individual (covering the six nodes 1, 3, 4, 5, 7, and 8 on the target path), will obtain a high approximation level if all identical path sections are considered for the fitness evaluation. If only the first matching path section is measured, the second individual (covering five nodes 1, 2, 3, 4, and 7) will obtain a higher approximation level than the first one [19]

.

The constraint optimization is an iterative procedure which gradually improves the SR model quality using the gradient calculation starting from initial parameters. As a start, the Jacobian matrix has been calculated from which all numerical values of all initial observed data are derived. Afterwards, the Jacobian matrix is used to update the parameter vector for the iteration procedure to continue until a specified stopping criterion is reached.

For the calculation of all partial derivatives for the parameter vector α, automatic differentiation has been used. In terms of modeling performance and accuracy, automatic differentiation becomes especially competent for constraint optimization. Therefore, all constant values have to be extracted and replaced by an appropriate parameter α_i. (Equation 2)

$$\nabla f = \left(\frac{\partial f}{\partial \alpha_1}, \frac{\partial f}{\partial \alpha_2} ... \frac{\partial f}{\partial \alpha_i}\right) \tag{2}$$

Additionally, artificial nodes are introduced in the SR model regarding nonlinear scaling terms. In one iteration, the start values for the SR model are the previously extracted constraint values in terms of additive and multiplicative scaling terms. The length of all path sections has been used as an approximation level. Because a high fitness value can be achieved by covering the desired path to the convergence criterion from the target initial start, the combination of two such nodes may lead to considerably better offspring. This is shown in Figure 2. The algorithm is stopped after a customized number of iterations. The quality of individual parameters for optimized constants would be updated. Meanwhile, the optimization starts from the same values are automatically retrieved in the model. This procedure is repeated many times according to tuning the parameter vectors from which the individuals are flagged with a boolean value.

The designed algorithm describing necessary steps for optimization the constraint of the proposed SR model is given as the Algorithm 1. In the literature, the SR method

	Linear Regression	Neural Networks	Random Forests	Symbolic Regression
Knowledge about explicit model structure required	Yes	No	No	No
Parametric or Non-parametric	Param.	Param.	Non-param.	Non-param.
Possibility for Local Adaptation	No	No	Yes	No
Model complexity depends on the # of data samples	No	No	Yes	No
Potential to create compact models irrespectively of data size and structure	High	Limited	Limited	High
Can final models provide insight into the problem, and increase system understanding	Yes	Hardly	Hardly	Yes
Complexity control possible	Yes	Yes	Yes	Yes
Danger to over-fit the data without explicit complexity control?	No	High	Limited	No
Danger of having insignificant variables in final models	Present	Present	Present	Not Present, or Heavily Reduced

Table 1. Comparison of different perspectives with respect to SR and three modeling techniques(Linear Regression, Neural Networks and Random Forests) which belong to three distinct culture in modeling prediction family [18]
.

has been utilized in many application areas. It is able to understand the data and build a descriptive model. Most importantly it provides certain insights about the data and the model. In Table 1, a summary of the comparison among several methods including SR is provided. In section 3, we demonstrate the application of SR to the analysis of oil sands ore properties.

Algorithm 1 This proposed algorithm for training a SR model: Initially we approximate the factors greedily using the automatic differentiation algorithm [14] and then we fine-tune the factors until the convergence criterion is satisfied.

```
Input: $\boldsymbol{X} \in \mathbb{R}^{p \times n}$, a list of initial constraint values
Output: weight matrices $\boldsymbol{Y}_i$

Add scaling tree nodes
Transform the tree for constraint optimization
while Stopping criterion not reached do
    Calculate the gradient with automatic differentiation
end while
Calculate quality with optimized constraint
repeat
    if Update Constraint then
        Retrieve optimized constraint
    end if
until Stopping criterion is reached
```

3 Implementation Procedure

3.1 Problem Setup

To implement the SR based genetic programming, we will perform the following procedure:

1) Binary representation conversion.
2) Initial population setup.
3) Fitness measurement.
4) Death, breeding, and mutation.

For this study, we chose six decimal digits of accuracy to create the possible solutions, by converting the dataset using binary representation.

Six decimal digits of accuracy corresponds to about 22 binary digits. A 22 digit binary string b can be converted to an integer k:

$$k = \sum_{i=1}^{22} b_i \cdot 2^{i-1} \tag{3}$$

The integer k is then converted to a real number u between 0 and 1 by:

$$u = k/(2^{22} - 1) \tag{4}$$

and u is again converted to a real number r between -1 and +2 by:

$$W = -1 + 3 \cdot u \tag{5}$$

so now we have a mapping between genetic information b and the objective W.

3.2 Implementation Details

Each candidate is a string of 22 binary digits, which can be treated as an integer vector.

We choose a population of $n = 50$ candidates, hence a 2 dimensional array of size 50×22 is created.

A random initial population can be selected as an array of 0's and 1's.

```
i   ----------b-----------      ----W----
#1  1000101110110101000111  =  0.637197
#2  0000001110000000010000  = -0.958973
...
#50 1110000000111111000101  =  1.627888
```

The best candidate W is search that optimize the quantity $R(T)$.

In this study, the initial fitness function is chosen as follows:

$$R(T) = T\sin(10\pi T) + 1 \tag{6}$$

Second, the iteration begins by measuring the fitness of each candidate:

```
i   -----------b----------       --R(T)--

#1  1000101110110101000111    => 1.586345
#2  0000001110000000010000    => 0.078878
...
#50 1110000000111111000101    => 2.250650
```

Based on the fitness results, the following steps are performed for the next generation:

- Out of the 50 candidates, remove 10 candidates with the lowest fitness;
- Let the 10 candidates with the highest fitness breed in pairs, creating 10 new candidates;
- Randomly select 2 of the nonbreeding, nondying candidates for mutation;

The size of the new population remains unchanged when it contains the best candidates from the previous population. The 10 candidates with lower score are replaced by new offspring with respect to higher fitness score. For the intermediate 30 candidates, we modify two randomly by mutation and keep 28 candidates unchanged.

For mutation, a candidate i can be picked to mutate. It is known that each candidate has 22 bits of genetic information. For example, one can pick index j between 1 and 22 and flip that bit:

```
b(i,j) = 1 - b(i,j)
```

For breeding, assume that parents i_1 and i_2, create children i_3 and i_4. To achieve this, an index j between 1 and 21 is chosen and parental information can then be spliced together:

```
b(i3,1:j) = b(i1,1:j)& b(i3,j+1:22)
b(i4,1:j) = b(i2,1:j)& b(i4,j+1:22)
```

Given the candidate #50 which is:

```
i  -----------b----------     --R(T)--

#50 1110000000111111000101 => 2.250650
```

If the fifth gene in this candidate is mutated, then the result is

```
1110100000111111000101  =>  -0.082257
```

but if the 10th gene is mutated, then the following result is obtained:

```
1110000001111111000101  =>  2.343555
```

Thus, a mutation process can be controlled to improve the fitness value.

The following shows the results of breeding. For example, the candidate #2 and #50 breed,

```
i  -----------b----------         --R(T)--
#2  0000001110000000010000   =>   0.078878
#50 1110000000111111000101   =>   2.250650
```

The crossover point is defined as a selection of parental information to generate offspring. If the crossover point is after $j = 5$, the two children will be as follows.

```
-----------b----------      --R(T)--
0000000000111111000101  =>  0.940865
1110001110000000010000  =>  2.459245
```

It is obvious that one child has a significant increase in the fitness function.

The GP procedure is terminated after 441 iteration. For the simulated data, intermediate results from the symbolic regression algorithm in several steps are shown in Table 2. The final results and the performance validation are included in Section 4.

4 SR DESCRIPTOR RESULTS

The oil sands data set contains three inputs variables: Temperature(T), pH, and Clay fines(Cf). The output is oil sands processability recovery rate(R). Due to lack of industrial data, we use data from laboratory experiments

Gen	$\hat{R}(T)$
1	$T\sin(10\pi T)+1$
6	T
8	$16.2+T$
9	$34.3+1.08T$
10	$61.7+29\sin(4.14+0.0519T)$
12	$61.7+43.9\sin(4.23+0.0517T)$
39	$18.3+73logistic(0.465T-16.6)$
40	...
51	...
99	$18.2+0.000165T^2+72.3logistic(0.468T-16.7)$
441	...

Table 2. The fitness function generation results, for a variable number of combinations of input attributes from the simulation data.

together with the generated simulation data. The simulation data set contains 1000 data points, which are generated based on known and empirical knowledge about relations among variables. Furthermore, it is noted that symbolic regression requires repeated simulation from multiple data generation. This can be achieved by adding noises and changing noise profiles to repeat the simulation.

By applying SR to the simulated oil sands dataset and after 441 iterations, we obtain a descriptor (i.e. a mathematical expression) for the factor analysis of oil sands recovery rate. In this section, we will construct and evaluate how well the fitness function is performed by calculating the root mean square error for the difference between the original data and the reconstruction by the descriptor for all input factors. Note that, in order to have comparable results, all of the fitness functions have the same stopping criterion rules. We have set the maximum amount of iterations to 1000 (usually ~100 iterations are enough). As explained in Section 3, the final descriptor obtained by the symbolic regression is shown as (Equation 7), which presents several sensible markers and their relationships constructed from the data used in this study.

The SR descriptor shows that the SR method renders a clear reconstruction for the simulation data than the other methods, in the sense that an analytical expression is obtained. This descriptor, to a certain degree, demonstrates sensible features of several input attributes. Such an descriptor can also be improved by integrating physical insights as constraints, such as function candidate blockers or generators during SR iterations.

$$\begin{aligned}\boldsymbol{R} = {} & 0.39\cdot T\cdot pH + 25.5\cdot logistic(0.0597\cdot pH)\\ & + 0.00783\cdot pH^2\cdot Cf^{-3} + 5.79\cdot logistic(0.001\cdot pH^3)\\ & + 0.049\cdot T^2 - 0.035\cdot T^3 - 0.002\cdot T\cdot pH\cdot Cf^{-2}\\ & + 0.00369\cdot Cf^{-4}\cdot logistic(0.001\cdot Cf^{-3})\\ & + \ldots\end{aligned} \tag{7}$$

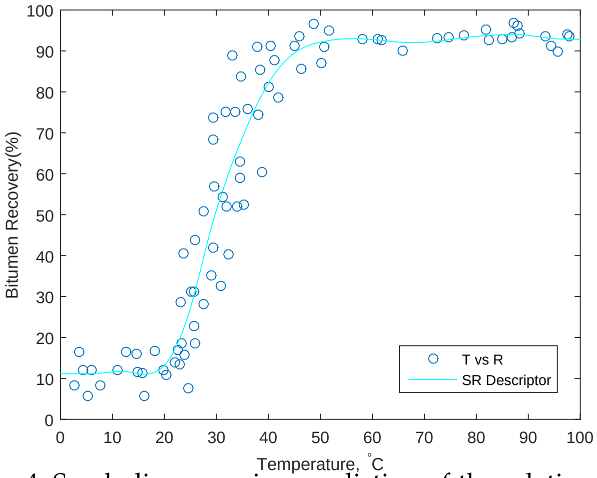

Fig. 4: Symbolic regression prediction of the relationship between recovery and temperature.

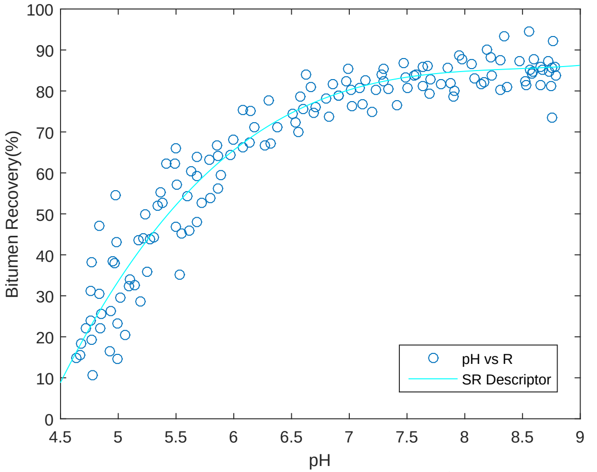

Fig. 5: Symbolic regression prediction of the relationship between recovery and pH.

It is noted that the terms shown in Equation 7 are considered to be the key factors, and ... represent the other expression terms of lower importance.

We first discuss the results obtained by SR descriptor (Equation 7). In Figure 4 we display the descriptor curve together with the training data. The data points surrounding the curve represent the real laboratory data and a few selected simulation data. As one would expect, the few number of laboratory data alone do not contain enough information to detect recovery rate trending. However, with the simulation data, the SR descriptor has the possibility of reviving the recovery rate trending based on temperature.

In the following we discuss the results for the best SR descriptor for each input variables. Figure 5, we can clearly see how the bitumen recovery varies with respect to the pH value: first the starting point is 10%, then a rapid increase is observed followed by a slow increase and finally the recovery rate settles around the 85%. Similarly in Figure 6, the relation (from the constructed SR descriptor) between the recovery rate and the Cf is

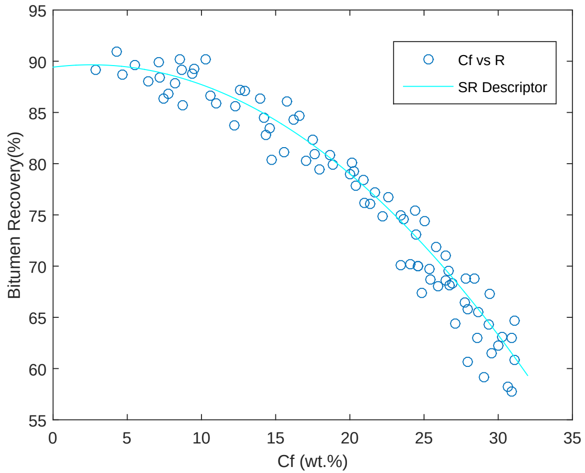

Fig. 6: Symbolic regression prediction of the relationship between recovery and Cf.

depicted.

Note that the shape of the curve strongly depends on the fitness function $Q(\alpha)$ in Equation 1. General speaking, more information from data points should not lead to a decrease in predictability. Thus, the SR descriptor from the simulation data set has richer interpolation in features domain. Counterintuitively, the descriptor from the mixed dataset becomes more reliable by changing parameter α_i in Equation 2. Thus, we can use α_i to influence the shape of the SR curve. Despite that, the most accurate model of the mixed data set is still the most accurate model overall.

To assess the quality and accuracy of the descriptor, comparison of the predicted results from the descriptor and the original laboratory data is performed, and the comparison results are shown in Figure 7-9. Figure 7 shows the constructed relationship of the recovery rate versus the temperature, compared to the original experiment data used. By increasing the clay fines from 20 wt% to 30 wt%, two curves are generated by SR descriptor and yet still share the same trending with respect to the bitumen recovery affected by temperature from 0 to 100 degree.

Two other input attributes, such as pH value and the clay fines, and how they affect the recovery are also evaluated. Figure 8 and Figure 9 show the comparison in prediction accuracy when changing temperature on the feature representations. From these figures, it is shown that the SR descriptor presents meaningful interpretation and more clear understanding of the data.

In Figure 10, the RMS errors between the SR result and the two original datasets, which are the training dataset and the test dataset, are compared respectively. The RMS scores are 0.24534 and 0.2443, respectively.

Finally, we evaluate the residuals between the observed value of the dependent variable and the predicted value (Figure 11). Both the training and testing data indicate a non random patterns and show a good fit for the SR model.

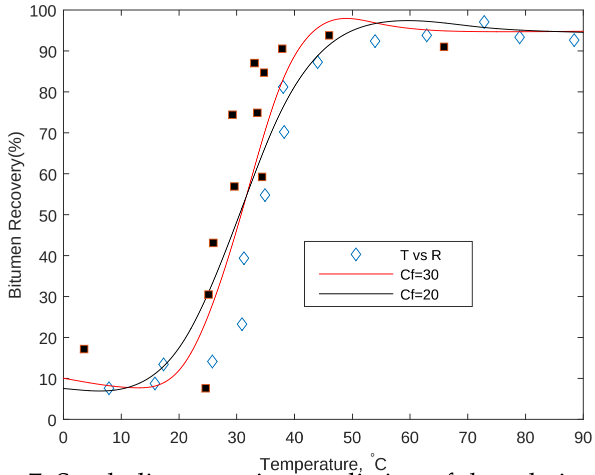

Fig. 7: Symbolic regression prediction of the relationship between recovery and temperature. Blue hollow square indicate actual data points. By altering clay fines amount to 10%, red solid curve is visualized compared with original dark solid curve.

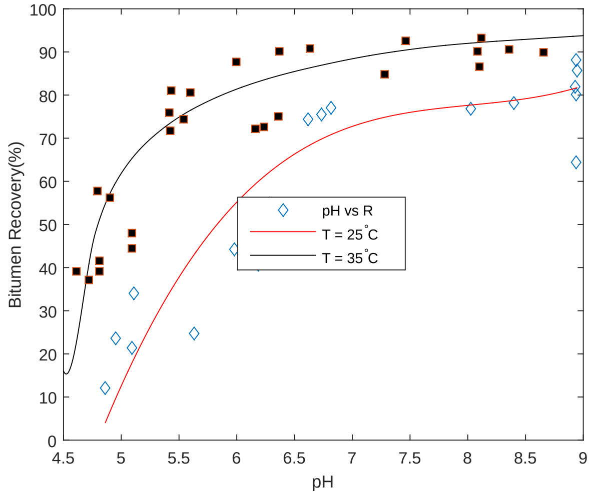

Fig. 8: Symbolic regression prediction of the relationship between recovery and pH. Blue hollow square indicate actual data points. The difference between red and dark visualized curves is 10 degree alteration of temperature.

5 Conclusion

We have introduced a novel symbolic regression model for oil sands recovery prediction, fitness function blockers, which are able to select an optimum combination of function candidates and a set of mathematical operators so that the data can be described by the constructed mathematical descriptor. Furthermore, we show that the proposed technique is able to understand a combination of representations for oil sands recovery with respect to the sensible markers obtained using a simulation oil sands data.

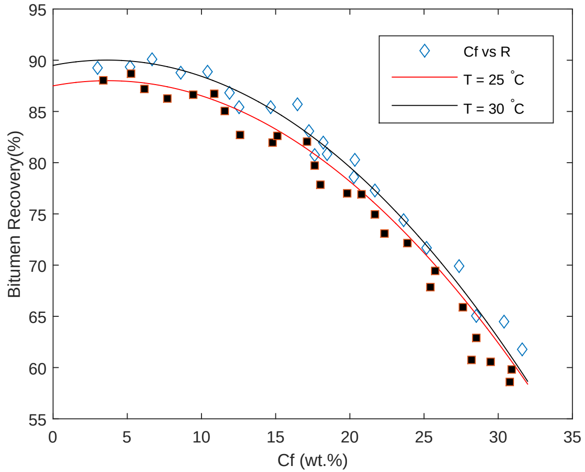

Fig. 9: Symbolic regression prediction of the relationship between recovery and clay fines. Blue hollow square indicate actual data points. The difference between red and dark visualized curves is 5 degree alteration of temperature.

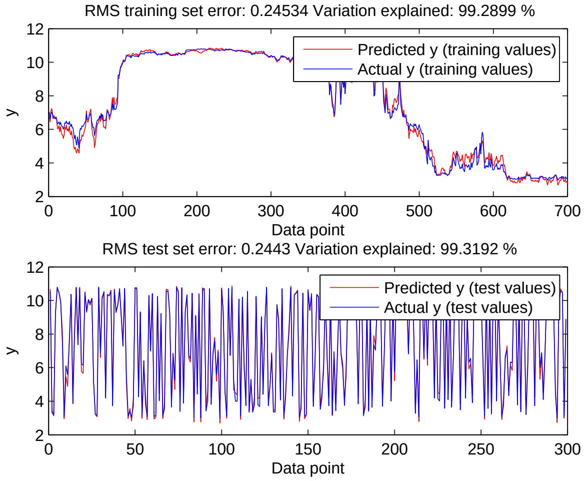

Fig. 10: Symbolic regression model performance.

One important future work is to experiment with real world recovery datasets, especially the industrial plant data.

References

[1] Masliyah, J, et al. ”Understanding waterbased bitumen extraction from Athabasca oil sands.” The Canadian Journal of Chemical Engineering 82.4 (2004): 628-654.

[2] N. Fong, S. Ng and K. Chung. ”A Two Level Fractional Factorial Design to Test the Effect of Oil Sands Composition and Process Water Chemistry on Bitumen Recovery from Model Systems”. The Canadian Journal of Chemical Engineering. Volume 82, Issue 4, 782-793,2004

[3] Q. Zhang, R. Sawatzky, E. Wallace, M. London, and S. Stanley, ”Artificial neural network modelling of oil sands extraction processes” Journal of Environmental Engineering and Science, 2003.

[4] Z. Michalewicz, ”Genetic Programming + Data Structures = Evolution Programs” Third Edition, Springer, 1996.

[5] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, “Metagenes and molecular pattern discovery using matrix factorization,” *P-NAS*, vol. 101, no. 12, pp. 4164–4169, 2004.

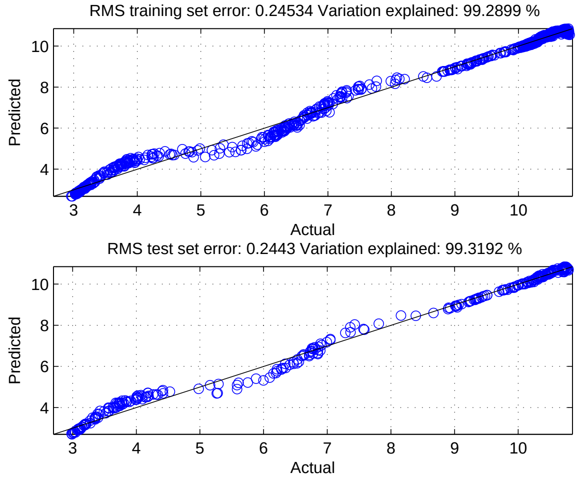

Fig. 11: Symbolic regression prediction performance.

[6] K, Devarajan, “Nonnegative matrix factorization: an analytical and interpretive tool in computational biology,” *PLoS computational biology*, vol. 4, no. 7, p. e1000029, 2008.

[7] Koza,R. ”Genetic programming II: Automatic discovery of reusable subprograms.” Cambridge, MA, USA (1994).

[8] M. W. Berry and M. Browne, “Email surveillance using nonnegative matrix factorization,” *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 249–264, 2005.

[9] Freitas, Alex A. ”A genetic programming framework for two data mining tasks: classification and generalized rule induction.” Genetic programming (1997): 96-101.

[10] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas, “Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification,” *TNN*, vol. 17, no. 3, pp. 683–695, 2006.

[11] I. Kotsia, S. Zafeiriou, and I. Pitas, “A novel discriminant nonnegative matrix factorization algorithm with applications to facial image characterization problems.” *TIFS*, vol. 2, no. 3-2, pp. 588–595, 2007.

[12] F. Weninger and B. Schuller, “Optimization and parallelization of monaural source separation algorithms in the openBliSSART toolkit,” *Journal of Signal Processing Systems*, vol. 69, no. 3, pp. 267–277, 2012.

[13] C. Houck, J. Joines, and M. Kay, ”A Genetic Algorithm for Function Optimization: A Matlab Implementation” NCSU-IE Technical Report 95-09, 1996.

[14] C. H. Ding, T. Li, and M. I. Jordan, “Convex and semi-nonnegative matrix factorizations,” *IEEE TPAMI*, vol. 32, no. 1, pp. 45–55, 2010.

[15] M. Schmidt, H. Lipson, ”Distilling Free-Form Natural Laws from Experimental Data”, Science, 324: 81-85, pp.2009

[16] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” *International Journal of Data Warehousing and Mining*, vol. 3, pp. 1–13, 2007.

[17] J. Herrero, A. Valencia, and J. Dopazo, “A hierarchical unsupervised growing neural network for clustering gene expression patterns.” *Bioinformatics (Oxford, England)*, vol. 17, pp. 126–136, 2001.

[18] E, Vladislavleva,”Model-based problem solving through symbolic regression via pareto genetic programming.” *PhD thesis, (Tilburg University,Tilburg)*, the Netherlands,2008.

[19] Andr Baresel, Harmen Sthamer, and Michael Schmidt, Fitness Function Design To Improve Evolutionary Structural Testing, *Proceedings of the Genetic and Evolutionary Computation Conference*, 1329-1336,2002.

[20] Neidinger, D. ”Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming.” *SIAM Review 52 (3): 545563*,2010.