IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-5, May 2018 E-ISSN: 2347-2693

Size and Cost Optimization of AutoCAD Oil and Gas Control Flow Designs Using Constraint Satisfaction Problem and Machine Learning

H.A. Kore[1*], S.B. Mane[2], A. Madkaikar[3]

[1]Department of Computer Engineering and Information Technology, College of Engineering Pune(COEP), Pune, India
[2]Department of Computer Engineering and Information Technology, College of Engineering Pune(COEP), Pune, India
[3]Emerson Innovation Center Pune, (EICP), Pune, India

[*]*Corresponding Author: koreha16.comp@coep.ac.in*

Available online at: www.ijcseonline.org

Accepted: 20/May/2018, Published: 31/May/2018

Abstract— The aim of automating the task of generating flow control designs for oil and gas flow comes with the different dimension constraints and cost factors. The resulted designs should satisfy given dimension boundaries and pre- specified conditions. To make the module more efficient and automate we combine the machine learning and constraint satisfaction module which resulted in reduction of time complexity and how the accuracy gets maintained. The result shows how the separate module of machine learning and optimization module work and how the results get vary when we combine both modules. The constraint we want to optimize are size and cost of the design. The main factors we considered for measuring performance are time complexity and accuracy.

Keywords— Constraint Satisfaction Problem, Constraint Optimization, Optimization Engine, Machine Learning

I. Introduction

When we go for constrain satisfaction and optimization problem the first thing comes to our mind is what are the constraints, objective function and the boundary values we are going to dealt with? [2] Initially we were just generating single conventional design which was not so feasible and optimized. We wanted to do size and cost optimization of design. To overcome this problem, we proposed one solution where we generate number of different designs by doing variations in components positions. So, the size of designs get varies and the cost needed for each design will be different. As we no size and cost are directly related to each other. If we are reducing the size indirectly we are reducing the cost. We wanted the resulted design should be optimized one and should satisfy all the conditions. As per design the satisfying conditions for each design will get vary and the bound values too. In our case some boundary values are static. So rather than going for some heuristic algorithms we implemented Rule Based algorithm by setting our own rules. But the constraint satisfaction of each design and finding optimized one among them is very time taking process. Means every time we needed to go through all condition checking process for each design repeatedly.

To make it more efficient and less time taking we combine the concepts of constraint satisfaction and optimization with machine learning. CS (Constraint Satisfaction) techniques [1] can be used to (partly) solve tasks in Machine learning; Alternatively, one can use machine learning techniques to augment CS techniques. Yet, a proper approach in either direction requires good knowledge of both research fields. The results of both domains are contradictory [3]. If Constraint satisfaction algorithm gives good accuracy in finding optimized design, there might be possibility that this accuracy will get decrease after combing with machine learning algorithms. On the other hand, the time complexity will get reduced compared to module of constraint satisfaction.

The section I is the Introduction part of the whole project. Section II related work in light the recent algorithms get used for Constraint satisfaction. Followed by that Section III, Optimization Engine gives detail information of the optimization engine with the workflow and flowchart.

II. Related Work

Following are some algorithms which has been used for constraint satisfaction with their advantages and limitations.

A. Constraint Hierarchies

Constraint hierarchies (CHs) and namely hierarchical constraint logic programming belong to traditional frameworks for handling of over-constrained problems. They allow to express hard constraints which has to be satisfied and several preference levels of soft constraints which isolations are minimized level by level subsequently.

CHs define the so-called comparators aimed to select solutions via minimizing errors of violated constraints. Global comparators aggregate errors of violated constraints [1] at each level, basically we may compare weighted sum of errors or error of worst satisfied constraint (weighted-sum-better and worst-case-better comparators). Local comparator considers each constraint individually, regional comparator is able to select among assignments by individual comparison of constraints at some lower level even if they are incomparable at higher levels.

B. Using Heuristic Search Methods

Since the search space can be fairly large, the conventional enumerative [3] search may simply give unsatisfactory performance to find the globally optimal solution. Therefore, we can go for conventional branch-&-bound (B&B) search technique and two of heuristic search strategies which sacrifice global optimality for efficiency to tackle these real-life optimization problems.

- *The Branch-&-Bound Search Method*

In handling many optimization problems, which are always NP-complete [4], one of the frequently used heuristics is the branch-and-bound (B&B) [2] heuristic in which the exploration of any partial solution in a search tree will be abandoned immediately whenever the search cost of that partial solution, as represented by an arbitrary objective/heuristic function h, already 6exceeds the minimal cost Bound for the optimal solution found so far. The heuristic function h always tends to be an under-estimation of the cost for any partial solution in a minimization problem. For instance, in the PC configuration problem [3], we can assume h as the sum of costs for the assigned components plus the minimal costs of the remaining unassigned components. Whenever the value returned by h for any partial configuration exceeds the current Bound, we can readily prune off the subtree under that partial configuration. On the other hand, when the total cost of the newly found complete configuration is less than Bound, the algorithm will update the Bound and the Optimal configuration [1] accordingly. In general, the success of the B&B method applied to any particular application depends very much on a careful design of the heuristic function h and the finding of a “reasonably good” Bound at an early stage of the search process.

- *Beam-Search Based Optimizer*

The first proposal of a heuristic-based optimizer for solving the PC configuration [4] problems simply limit the search to focus on the best n possible values of every sorted domain to construct the partial solutions in each search step. Since the PC configuration problems are only sparsely constrained, our proposed strategy should be able to return a solution which is fairly close to the global optimal solution within a reasonable period of time. The two main factors which we used to control the search are Budget and Threshold as predefined by the user of the search strategies. Similar to the mechanism used in the B&B heuristic, the control parameter Budget will be used to filter out any partial configuration which already exceeds its allowed value. In addition, the Threshold value n will help to ensure the search will always return the best n.

C. Bucket Elimination

Bucket elimination is a generic technique suitable for many automated reasoning and optimization problems and, in particular for solving WCSP [5]. Probabilistic inference algorithms for belief updating, finding the most probable explanation, the maximum a posteriori hypothesis, and the maximum expected utility are reformulated within the bucket elimination framework. This emphasizes the principles common to many of the algorithms appearing in the probabilistic inference literature and carries the relationship of such algorithms to nonserial dynamic programming algorithms. A general method for combining conditioning and bucket elimination is also presented. For all the algorithms, bounds on complexity are given as a function of the problem structure.

III. OPTIMIZATION MODULE

A. Constraint Satisfaction and Optimization Module

The control flow designs generation comes with different components and dimensions values. We can go with the assorted designs like we can place components in horizontal way or vertical way and for each design the dimension values will vary. All constraint to be consider are the dimension values consist of length, height, width and the cost. The cost of the module is nothing but the some of the components used in the control flow design. The constraint optimization module named as Optimization Engine.

B. Optimization Engine

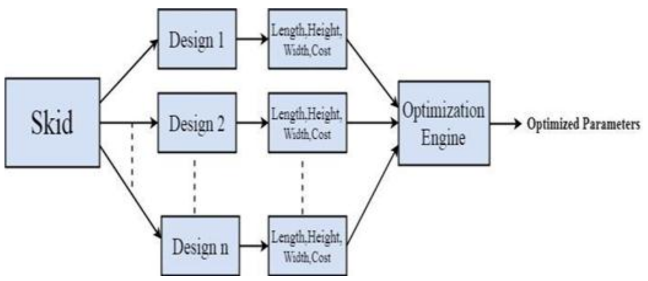
Skid
Design 1
Design 2
Design n
Length,Height, Width,Cost
Length,Height, Width,Cost
Length,Height, Width,Cost
Optimization Engine
Optimized Parameters

Fig 1. Block diagram of Optimization Engine

The block diagram shows we are taking input from users and generate different flow control designs. Then we calculate dimensions and cost of each design which will be input to the Optimization Engine which is nothing but the rules-based algorithm for the constraint satisfaction and optimization.

C. Flowchart

The flowchart shows how the Rule Based Algorithm works. Dimensions will be the input to the Optimization
Engine. It will first check for the hard constraints if it is getting satisfy then only it will go for soft constraint checking. Suppose in count we have 'n' conditions which each design should satisfy. Then we will keep count of how many conditions each design satisfying. And will choose two designs with two highest probabilities. And will check for other component level conditions. And finally, will select the optimized one design.

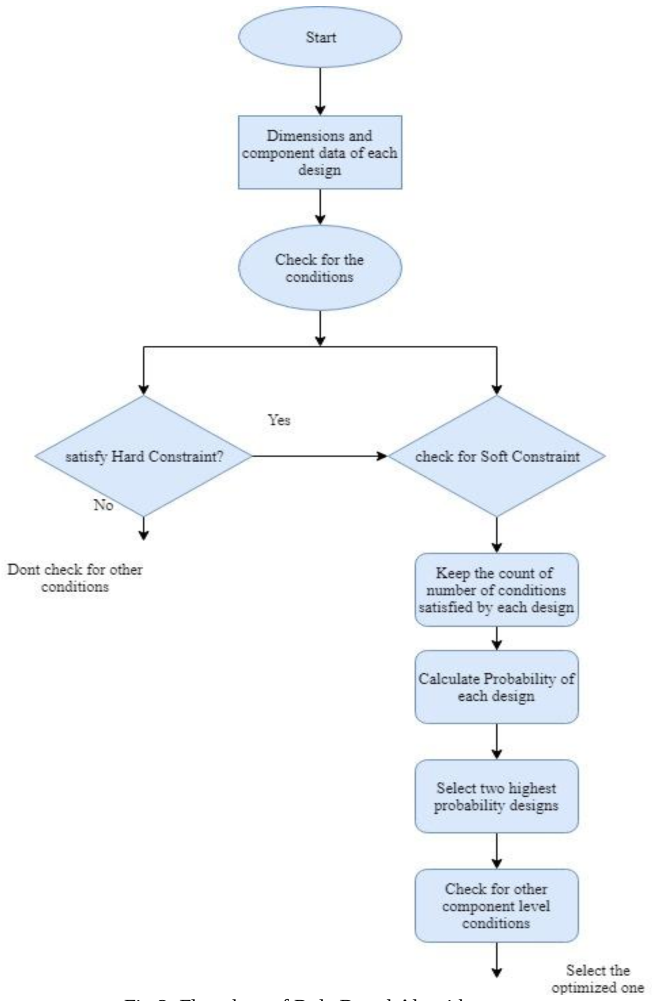
Start
Dimensions and component data of each design
Check for the conditions
Yes
satisfy Hard Constraint?
check for Soft Constraint
No
Dont check for other conditions
Keep the count of number of conditions satisfied by each design
Calculate Probability of each design
Select two highest probability designs
Check for other component level conditions
Select the optimized one

Fig 2. Flowchart of Rule Based Algorithm

So, the Rule Based Algorithm is the set of rules where the bound values set by us. We wanted to set our own rules and bounds that why we preferred to go for Rule Based Algorithm.

D. Mathematical Formula

The objective function which should be satisfied by the design to be optimized one is given as below:

$$F(x) = f(n)/n \qquad (1)$$

The Equation (1) shows the objective function where f(n) is the total count of conditions satisfied by the design and n stand for total number of conditions. The design with highest probability will be the optimized one.

IV. MACHINE LEARNING MODULE

A. Machine Learning Module

So, the machine learning module works like it will take the dimension parameters of each design and pass it to the algorithm to find out which design is optimized one. The dataset contains 6 features like label, length, height, width, cost, container, component number and the classes are Optimized, Not Optimized and No Optimized Design Found. We tried different algorithm and check the accuracy of each algorithm. The result section contains the details how the accuracy gets vary with algorithms and the size of the dataset.

B. Combining Constraint Satisfaction with Machine Learning

There is also an increasing interest in using machine learning to improve the solving of constraint problems, as well as improving the modelling and the embedding of learned constraints and objectives into the model [4]. As the independent module were working well, the constraint satisfaction module takes more time compare to machine learning module, but it gives 100% accuracy where machine learning modules accuracy is not so good compared to rule-based algorithm. As for making the module more efficient and accurate we needed to combine both the modules.

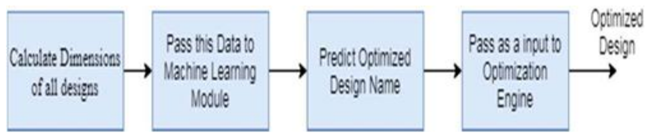
Calculate Dimensions of all designs
Pass this Data to Machine Learning Module
Predict Optimized Design Name
Pass as a input to Optimization Engine
Optimized Design

Fig 3. Combined Module

First, the calculated dimensions will be given to the machine learning module where it will classify which design is optimize design and which is not-optimized design. So once the module will predict the optimized design names which will pass to the rule-based algorithm for checking that the optimized design suggested by machine learning module is optimized or not. Like this way, we will get accuracy and will reduce the time complexity also. But there are some possibilities which can happen:

- *What if we get more than one optimized designs?*

We will check for those designs and find the high probability design and select that as an optimized design otherwise go for the conventional design.

- *What if the optimized design found by machine learning module is not satisfying rule-based conditions?*

That time we will go for the conventional design.

- *What if it doesn't find any optimized image?*

Go for the conventional design.

V. Experimental Result

First, we will see results of the both modules. How the both modules works independently and then combinedly.

A. *Rule Based Algorithm*

Table 1. Time and Accuracy of Rule Based Algorithm

Design Count	Time Taken	Accuracy
2	1-3 sec	89-93%
5	5-8 sec	86-91%
7	10-13 sec	85-89%

This table shows the results of optimized engine. Design count is the number of distinctive designs we used to find optimized among them. We can see the time taken by this module is very high.

B. *Machine Learning Module*

We used Weka tool to check how different algorithms perform on our dataset.

Table 2. Accuracy of Different Algorithm With 150

Algorithm	Dataset Count	No of Correctly classified instances in %
OneR	150	73.22
Random Tree	150	71.43
Random Forest	150	82.50
J48	150	75
Kstar	150	50
ZeroR	150	35.71
Naïve Bayes	150	65
Logistic Regression	150	39.29
Multilayer Perceptron	150	48.22

Table 2 shows the accuracy of different machine learning algorithm with dataset of count 150.

Table 3. Accuracy of Different Algorithm With 5000 Records

Algorithm	Dataset Count	No of Correctly classified instances in %
OneR	5000	69.20
Random Tree	5000	62.42
Random Forest	5000	71.23
J48	5000	61.27
Naïve Bayes	5000	59.32
SVM	5000	70.02

Table shows the accuracy of algorithms with data set of count 5000.We eliminate some algorithms based on accuracy performance of those algorithms.

Table 4. Accuracy of Different Algorithm With 25000

Algorithm	Records Size	No of Correctly classified instances in %
Random Forest	25000	70.01
SVM	25000	68.045
OneR	25000	60.123
Random Tree	25000	59.56

So, after training and testing with different algorithms we come to conclusion that Random Forest is performing well with our dataset.

C. *Result in different Dataset Ratios*

We tried the training and testing to with different data ratios to find out in which ratio it will perform well and give more accuracy. All the results are shown below:

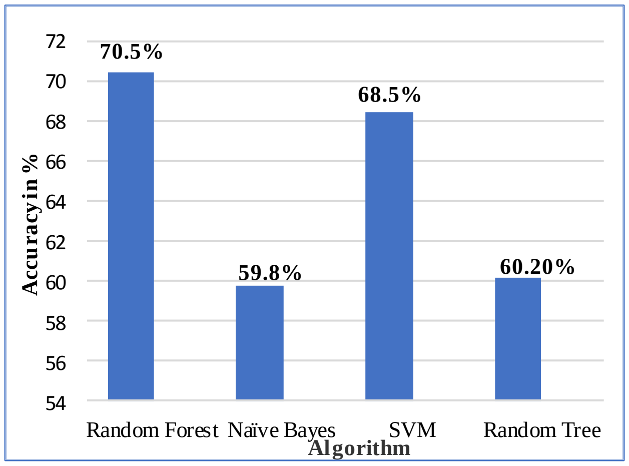

Fig 3. Accuracy in Split Ratio 60:40

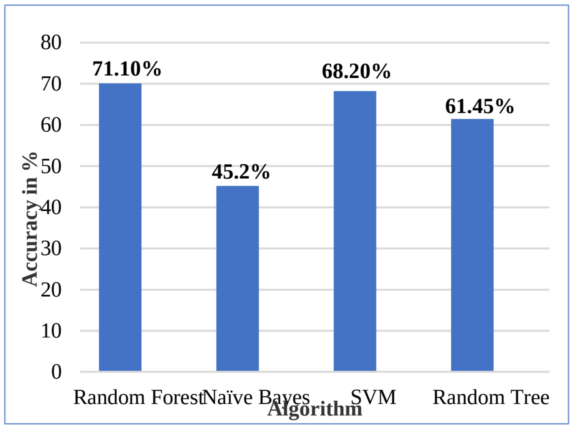

Fig 4. Accuracy in Split Ratio 80:20

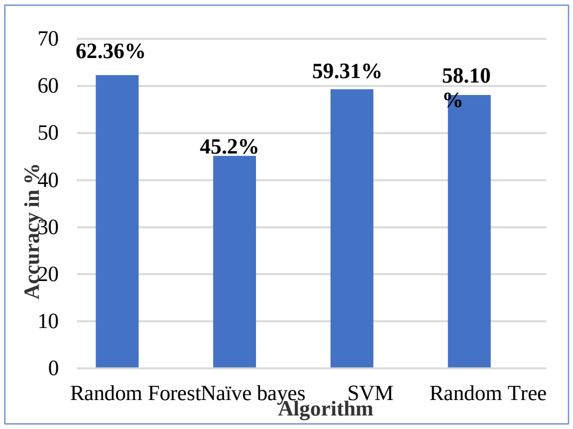

Fig 5. Accuracy in Split Ratio 30:70

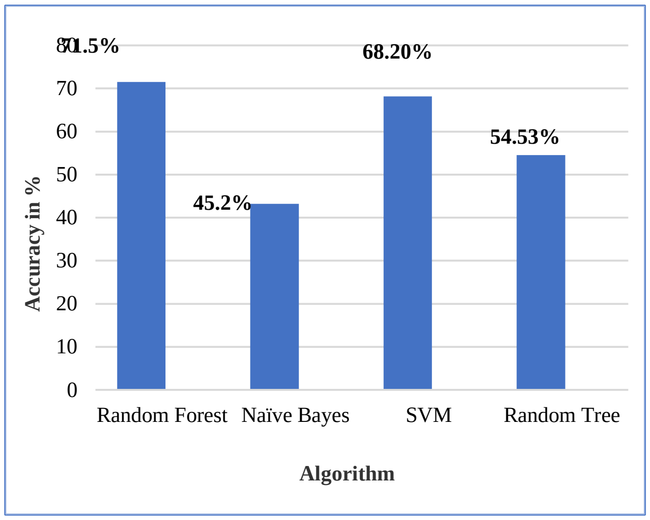

Fig.5. Accuracy in Split Ratio 50:50

D. Results After Combining Both Modules

Table 5. Time and Accuracy of Optimization Module with Random Forest Algorithm

Design Count	Total Time Taken	Accuracy
2	0.828 sec-1.989 sec	81-90%
5	2.124 sec -3.458 sec	82-90%
7	3.968 sec to 5.236 sec	83-88%

The result shows that the time complexity is get reduced as compared to initial time taken. And we tried to maintain accuracy by passing the result of machine learning module to the Rule Based algorithm. It will give accuracy that the result of machine learning module is right or wrong and the final output will always be right one. If our module failed to predict any optimized design that time we will go for conventional design.

VI. CONCLUSION

So, the result shows how initially the optimization engine module was time taking and how it gets decrease after combining it with machine learning module. The machine learning module reduces the unnecessary task of checking for each design to find optimized design among them. So, our approach works properly, and it helps in maintaining accuracy and time complexity both.

There are different number of constraint satisfaction algorithms which we can us with machine learning algorithms to make it more automotive.

VII. ACKNOWLEDGMENT

I would like to express my sincere gratitude to my guide Dr. Sunil B. Mane and Mr. A. Madkaikar for the continuous support for my MTech study and research, for their patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.
Also, I thank my friends in College of Engineering Pune for constantly directing me to the right thing and suggesting me the way of doing research.

VIII. REFERENCES

[1] Vincent Tam and K.T. Ma, *"Optimizing Personal Computer Configurations with Heuristic-Based Search Methods*," 129–140, 2012.
[2] Igor Rivin and Ramin Zabih *"An Algebraic Approach to Constraint Satisfaction Problems,"* 2011.
[3] Guido Tack, Tias Gunsand*, "Introduction to the special issue on Combining Constraint Solving with Mining and Learning,"* March 2017 Artificial Intelligence 244:1-5.
[4] P.C. Fourie,A.A. Groenwold, *"The particle swarm optimization algorithm in size and shape optimization,"*2001.
[5] A. Javier Larrosa and B. Rina Dechter, "Boosting Search with Variable Elimination in Constraint Optimization and Constraint Satisfaction Problems," MIT Press, 2007.
[6] Luc De Raedt1, Siegfried Nijssen2, Barry O'Sullivan3 , and Pascal Van Hentenryck4, "*Constraint Programming meets Machine Learning and Data Mining,"* Report from Dagstuhl Seminar 11201.
[7] Hana Rudová *"Constraint Satisfaction with Preferences,"* IEEE, 2014.
[8] L. De Raedt, T. Guns, and S. Nijssen ,"Constraint Programming for Itemset Mining In ACM SIGKDD", Int. Conf. KDD'08, Las Vegas, Nevada, USA, 2008.
[9] L. De Raedt, T. Guns, and S. Nijssen, "Constraint programming for data mining and machine learning," In Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI10), pages 1671–1675, 2010.
[10] M. Khiari, P. Boizumault, and B. Crémilleux, "Local constraint-based mining and set constraint programming for pattern discovery", ECML/PKDD-09 Workshop, pages 61–76, Bled, Slovenia, 2009.
[11] M. Khiari, P. Boizumault, and B. Crémilleux *"Constraint programming for mining n-ary patterns,"* In 16th Int. Conf. on Principles and Practice of Constraint Programming (CP'10), volume 6308 of LNCS, pages 552–567. Springer, 2010.
[12] A. Javier Larrosa, B. Rina Dechter, *"Boosting Search with Variable Elimination in Constraint Optimization and Constraint Satisfaction Problems,"* 2002 Kluwer Academic Publishers.
[13] Frost, D. and R. Dechter: 1994, *"In Search of the Best Constraint Satisfaction Search,"* In: Proceedings of the 12th AAAI. pp. 301–306.
[14] Larrosa, J.: 2000, *"Boosting Search with Variable Elimination,"* In: Proc. of the 6th CP,Singapore, pp. 291–305.
[15] Dechter, R., K. Kask, and J. Larrosa: 2001, *"A General Scheme for Multiple Lower Bound Computation in Constraint Optimization,"* In: Proc. of the 7th CP. pp. 346–360.
[16] Meseguer, P., J. Larrosa, and M. Sanchez: 2001, *"Lower Bounds for Non-Binary Constraint Optimization Problems,"* In: Proc. of the 7th CP.pp.3