Improvements to the Kigen Solver: The Benefits of Increased Autonomy and A New Algorithm for Search-Based Model Transformation

Jordan W. Taylor
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
770-896-3043
jordan.taylor@gatech.edu

Sebastian J. I. Herzig
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
818-393-3059
Sebastian.J.Herzig@jpl.nasa.gov

***Abstract*— Most work in the field of multi-objective search-based software engineering has focused on the use evolutionary algorithms. Those that have researched multi-objective hill climbing have converted the multi-objective optimization problem into a single-objective optimization problem by weighting and summing of the objectives. Our contribution is a multi-objective hill climbing algorithm that uses Pareto dominance rather than a weighted sum as a success criterion. Also, empirical results imply combining traditional multi-objective genetic algorithms and our multi-objective hill climbing algorithms has the potential to closer approximate the Pareto frontier than can be done using genetic algorithms or multi-objective hill climbing.**

I. INTRODUCTION

Current methodologies for designing missions are severely limited. Mission design is a very manual process during which an engineer creates a few initial designs based on their expert knowledge and modifies them until all requirements are met. As a result, design is heavily biased by approximations from experienced engineers, which are heavily influenced by previous successful missions. This process limits the complexity of missions that can be considered, and, even if designs are found, they are likely suboptimal because only a small subset of the feasibility region can be explored manually. In fact, the oversimplified spacecraft design problem of finding a subset of a set of components with maximum value subject to a cost constraint is an NP-Hard[Lemma 1] problem. To cope with intractable design problems and expedite the design process, the Kigen tool suite has been developed.

Kigen computationally searches the design space for Pareto efficient solutions, defined as a set of solutions such that no solution performs better with respect to one of the objectives of another solution in the set without performing worse with respect to at least 1 of the other objective.

To use Kigen a user defines the meta-model, constraints and objectives in a custom modeling language, KigenML [1] that is then compiled to an Eclipse Modeling Framework [2] meta-model. Then, a starting model is instantiated from which to build based on the user specified initial model, and model-transformation rules are generated. Then, the solution space is searched. These solutions were represented as a string of simple graph transformations that either add a model element or a relationship between existing model elements implemented using the Henshin [3] model-to-model transformation framework. The space of solutions is searched using algorithms implemented using the Marrying Search-Based Optimization and Model Transformations (MOMoT) framework [4] and the MOEA framework [5] .

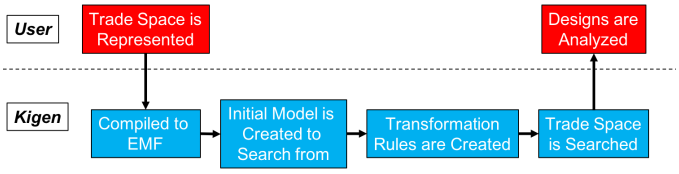

Fig. 1. High-Level Explanation of Kigen's Architecture

Toward the goals of decreasing the number of iterations needed to find a feasible solution and increasing the quality of solutions found, the following questions were explored:

1) Will starting the search from a small model that satisfies multiplicity requirements on the meta-model decrease the time needed to find a feasible solution?
2) Can autonomously generated rules perform comparably to those created by domain experts, making Kigen easier to use?
3) Can other algorithms search the model-space better than a genetic algorithm?

II. BACKGROUND

As discussed earlier, even an oversimplified mission design problem is NP-Complete. However, when designing a multi-spacecraft mission there is added complexity. How many spacecrafts should be used? Which spacecrafts should be used? How should spacecrafts communicate with one another?

Due to this added layer of complexity, the design of multi-spacecraft missions exemplifies the need for computational tools. Because of this, two multi-spacecraft missions were examined. The first is the design of a multi-spacecraft interferometry mission during which satellites orbit in low lunar orbit to detect signals below 30 Mhz, not strong enough to be detected on earth [1]. The second being the design of a multi-rover Mars cave exploration mission that are thought to be candidates for extraterrestrial life and future human settlement [6].

The Kigen solver has been used to search the trade space of interferometry mission designs using domain expert crafted exploration rules [1] and the trade space of the mars cave exploration using autonomously generated rules. Both of these searches were done using the genetic algorithm NSGA-II [7].We will revisit these solutions throughout this paper to evaluate the efficacy of new autonomously generated exploration rules and our new algorithm.

III. The Initial Model

Because solutions are represented in Kigen as a string of endogenous model-transformation rules applied to an initial model, the speed at which a solution can be found is heavily influenced by where the search starts from. Currently, a user must specify a starting location from which to search, but this adds more work for the user. Also, if a user does not specify an initial starting model, the search is done from an empty model.

Fig. 2. Starting Model of Mars Cave Exploration Mission using the Current Starting Methodology

Since Kigen attempts to find a set of Pareto efficient designs that satisfy all requirements, a feasible design, one that satisfies all requirements, must first be found. In Kigen are two types of requirements (constraints): well-formedness

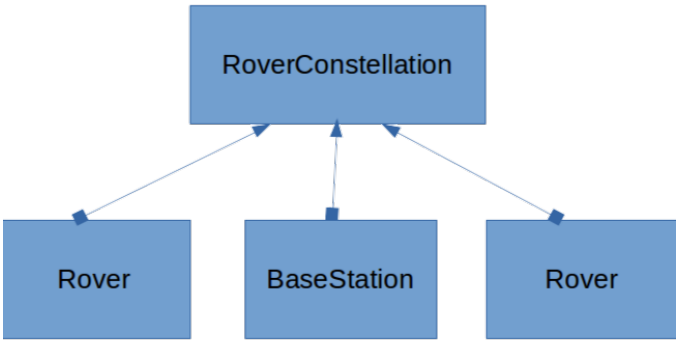

Fig. 3. Starting Model of Mars Cave Exploration Mission using the New Starting Methodology

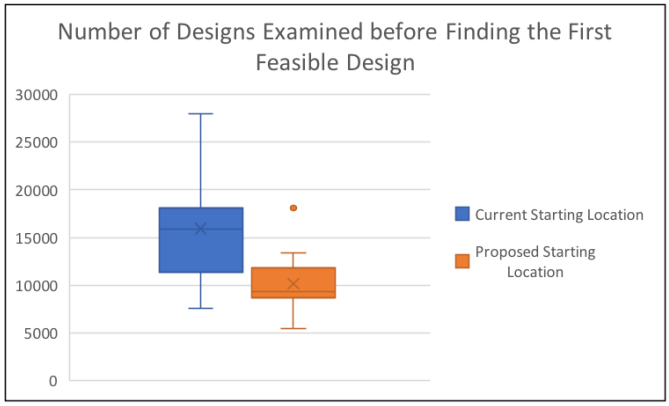

Fig. 4. Effect of starting location on the time to find a feasible solution

of the model with respect to the meta-model specified by the user, and those constraints explicitly defined by the user in KigenML [1]. The latter can be any function that evaluates to a boolean, but the former have a structure imposed on them by the Eclipse Modeling Framework. Let FR (the feasibility region) denote the set of models that satisfy all constants. Let FMS (the feasible model space) denote the set of models that satisfy the well-formedness constraints imposed by the meta-model. Because a model must satisfy the constraints imposed on the meta-model to be in the feasibility region, we know $FeasibilityRegion \subseteq FeasibleModelSpace$. This bounds the search space to a discrete space of models, which could be very large, but assuming the model does not allow for an unbounded number of model elements is finite.

Rather than starting from an empty model or a user defined model that is not necessarily in the feasible model space we

propose starting the search from the smallest model in the feasible model space by adding model elements to the lower multiplicity bounds of all composition relationships defined in the meta-model. As can be seen in Fig. 2, this proposed starting location decreased the number of solutions searched before the first feasible solution was found on the mars cave exploration case study by on average over $\frac{1}{3}$.

IV. Autonomously Generated Rules

Unless otherwise specified by a user, as was done on the initial analysis of the interferometry mission [1], the set of model-transformation rules used to represent solutions are constructed using a one step lookahead method. Consider a mission with the following requirements from the meta-model:

1) A Rover must have 4 Wheels
2) A Wheel must have 8 bolts

With the one step lookahead, an "addRover" rule would add a Rover and 4 Wheels to the model. Similarly, an "addWheel" rule would add a Wheel and 8 Bolts to the model. Because these transformation rules don't consider requirements on a subsystem's structures to be satisfiable, the simple transformation rules that add an individual model element are likely instantiating invalid subsystems, as is the case with the "addRover" rule.

Our proposal is combining these simple rules to form more complex rules. For example, an "addRover" rule should add 4 Wheels and whatever is needed for each Wheel to be valid, namely 8 Bolts. Specifically, to improve the quality of solutions, model transformation rules by recursively searching the containment tree of each model class and instantiating model elements to the lower bounds of all composition multiplicities. We will refer to these as deep rules and the aforementioned rules as shallow rules. In situations were the class the relationship is specified for has children, such as there being more than one type of instrument, objects are instantiated uniformly from the set of concrete classes of the parent type.

To benchmark the deep transformation rules against the shallow rules, the NSGA-II algorithm, a genetic algorithm for multi-objective optimization, implemented in MOMoT was run 5 iterations with a population size of 1000, the maximum number of generations equal to 100, and a maximum solution length of 150 on a multi-spacecraft interferometry mission design problem used previously to benchmark Kigen. The objectives of this problem are to minimize cost, maximize percent coverage of the interferometry mission, and minimize mission duration. Tournament selection in threes was used to select from the population. One-point crossover is performed between "winning" solutions. With probability 0.05 a rule in a string is switched out for another. With probability 0.20 a numeric variable is changed to another value in the feasible range uniformly. With probability 0.05 a rule in the string is removed. The initial solutions are generated by uniformly selecting genes from the set of model transformation rules. Because population generation is probabilistic, different runs of NSGA-II will produce different solutions. To account for this, 5 iterations of the NSGA-II algorithm were performed.

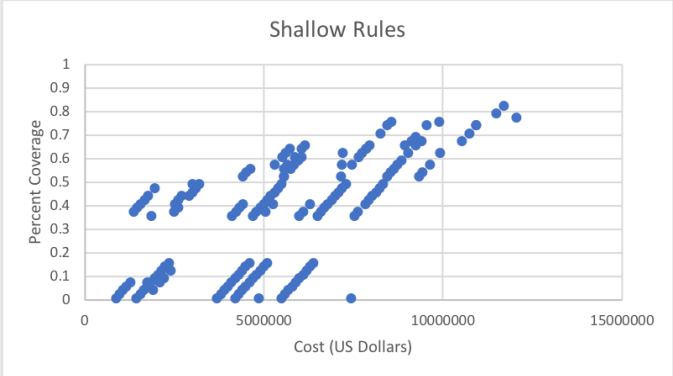

Fig. 5. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using the previous, shallow model transformation rules

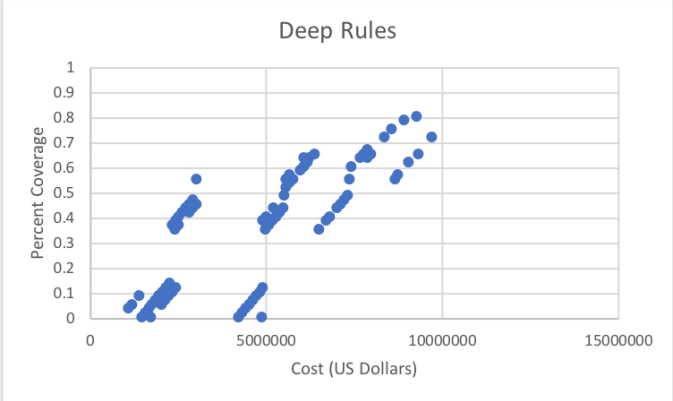

Fig. 6. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using the proposed deep model transformation rules

By examining Figure 5 and Figure 6, one can see the shallow graph transformation rules generated more solutions, and found solutions who's objective function values varied more than the solutions found using the deep rules alone with respect to coverage and cost. Using the shallow rules set for the representation of solutions, the NSGA-II algorithm found 158 Pareto efficient solutions, and using the deep rules 84 Pareto efficient solutions were found. Of the 158 solutions found using the shallow rules, 15, about 9%, of the solutions were dominated by solutions found using the deep rules. On the other hand, of the 84 solutions found using the deep transformation rules, 17, or 20%, were dominated by solutions found using the shallow rules.

This is likely because for a string of equal length, the deep rules will instantiate many more model elements than the shallow rules. This will cause the solutions found using only the deep rules to be biased toward model designs with more model elements in them, which will have higher cost. Hence, the region of the model space closer to the lower bounds of relationships was searched more thoroughly by the shallow

rules than the deep rules. Also, solutions found using the set of deep rules might underperformed those found using the set of shallow rules because the number of rules that add aggregation relationships is the same in both sets. Despite the model constructed using the deep rule set satisfying the multiplicity constraints from the containment tree, more model elements were instantiated than with the shallow rule set without adding enough aggregation relationships to make the model satisfiable.

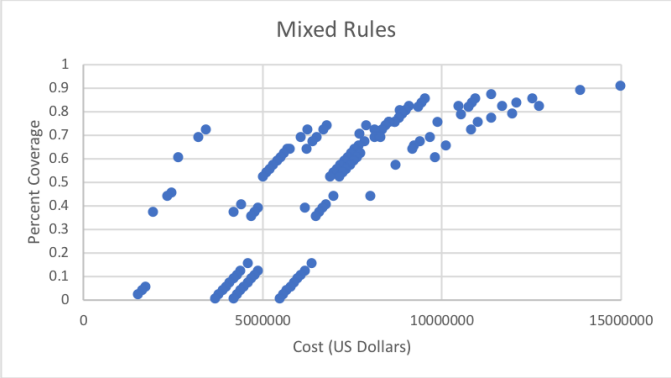

Fig. 7. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using both deep and shallow model transformation rules

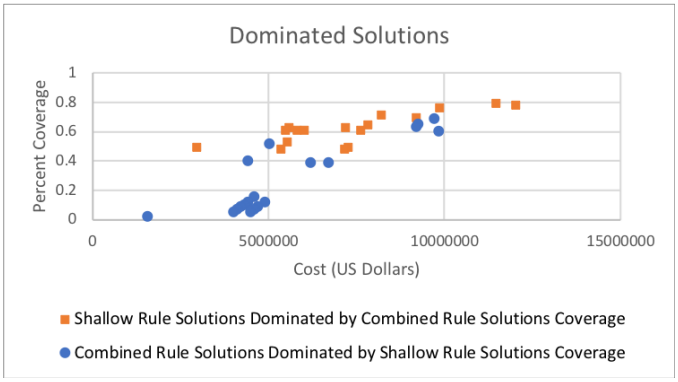

Fig. 8. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using both deep and shallow model transformation rules

When used alone, the set of deep transformation rules underperformed the shallow rules. However, as can be seen in Figure 7, by making the set of rules the union of the deep rule set and the shallow rule set, the design space is searched more thoroughly than if using only shallow or deep rules. Of the solutions found using the shallow rules, 17, or 11%, of the solutions were dominated by solutions found using both rules. Whereas, 19 of the 126 solutions, or 15%, of the solutions found using the mixed method were dominated by the solutions found using the shallow method. Looking at these numbers alone, this looks to be a small improvement on the solutions found using only the deep rule set, but that is not the only metric by which to compare solution sets.

In addition to looking at the number of solutions one set of solutions dominates from another, a set of solutions are ”better” if they closer approximate the entire Pareto-front. This is because Kigen is trying to augment human decision making by providing a set of solutions that satisfy given constraints such that no solution in the set is mathematically superior to another. Then, given this approximation of the solutions along the Pareto frontier a designer can use their domain knowledge and preferences with regard to the objective functions to make a rational design decision. In this specific example, a set of Pareto efficient solutions with a higher maximum percent coverage is better approximating the entire Pareto frontier than a set of Pareto efficient solutions that all have very low coverage because low coverage implies the set of points are close to one another on the Pareto frontier.

As can be seen in Figure 6, most of the combined rule solutions dominated by the shallow rule solutions had low coverage and the shallow rule solutions dominated by the combined rule solutions had much higher coverage. This was expected because solutions of a fixed length constructed using shallow transformation rules are only able to search a small subset of the feasible model space, which in this case corresponded to one with low coverage. It follows that genetic algorithm using the shallow rules to construct solutions are searching a smaller space than the space of solutions that can be constructed using the combined rule set; hence, the solutions found in that small space will be closer to the Pareto frontier than those found by searching a much larger space using both the deep and shallow rules. This implies the set of solutions generated using both the shallow and deep rules is better able to approximate the Pareto frontier when compared to solutions of equal length generated by using the set of shallow rule transformations.

V. Multi-Objective Hill Climbing

Having improved the representation of solutions, I sought a way to better search the space of solutions. Because solutions are represented as a string of graph transformations, one can search the space using a genetic algorithm. This is because one can ”mate” two solutions using a simple one point crossover operation, where a random pivot is selected and the genes in the string to the right of the pivot are swapped between parents. In the previous publication on Kigen, the NSGA-II genetic algorithm was used to search the space of solutions. Genetic algorithms have been successfully applied in the field of search-based engineering to design antennae [8], infrared filters [9], and the design of analog circuits [10]. However, these problems involved small subsystems.

Unlike the aforementioned design problems, the mission architecture design problems involve many objectives, and missions have multiple subsystems that can all affect the objective functions and the performance of other subsystems. This added layer of complexity makes the crossover operation on our solution strings unlikely to result in satisfiable solutions. When designing a mission architecture, or any system with a deep containment tree, the addition of a single model element can cause a large shift in the model space. It would follow that the addition of a model element to a system

with a deep containment hierarchy can cause a large shift in the objective function. This is why I decided to explore hill climbing as an alternative to genetic algorithms for searching the solution space.

Input : Initial Solution s
Output: Best Solution Found
while *!(Stopping Condition)* **do**
  Generate Neighborhood Set from s_t;
  $s := SelectaSolutionfromtheNeighborhood$
end
return s;

Algorithm 1: High-Level Local Search Algorithm Description [4]

Algorithm 1 shows the basic pseudocode for local search. Currently, the MOMoT framework only has an implementation of single-objective hill climbing, which is a local search algorithm that selects the best performing solution with respect to a single objective function at each iteration and terminates when a better solution can not be found. This is unable to solve problems with more than one objective, such as mission design.

In the first paper on applying multi-objective local search to search-based model transformation, Bill et al. [11] extended the MOMoT framework by implementing a multi-objective local hill climbing algorithm. Using this implementation [11], Bill et al. provided experimental evidence that, as I hypothesized, when an objective function is sensitive to shifts in the model space because of a deep containment hierarchy, multi-objective local search outperforms the NSGA-II genetic algorithm. However, their algorithm used a randomly generated set of weights to convert multiple objective values into a single objective equal to the weighted sum of the objectives. This is not effective at informing a designer by estimating the Pareto front because objective values with larger scale, such as cost in the previous example, will have a larger influence on the single weighted-sum objective value. This encodes a preference for solutions that perform well according to the objectives with larger scale, regardless of the suer's preferences with respect to the objective functions.

Even if objective function values are normalized such that the scale of an objective doesn't affect the weighted-sum objective more than another, weighting functions implies preferences and by randomly assigning weights to the objective function values encodes a preference ordering of solutions randomly. Even if every objective value is both normalized and weighted equally, one is still representing the ordinality of the objectives. In this case ordering the objectives equally. To show the importance of properly representing solution preferences, consider a multi-objective airplane design problem [12] with 5 objectives ranging from 0 to 20 to be maximized, one of which is safety. By converting the objectives into a single equally weighted sum and performing hill climbing, a solution which scores a 0 on safety but a perfect 20 in every other category, a total score of 80, would be preferred over a solution with a score of 15 on every objective, a total score of 75. However, one would hope an airplane designer would prefer the latter to the former.

Since Kigen solves problems without knowledge about the objective preference ordering of a designer, any weighting schema of the objective functions to convert a multi-objective optimization problem into a single-objective optimization problem would be making an assumption about the designers preferences. Instead of using this weighting schema, Kigen is objective agnostic: the solver does not consider any Pareto efficient solution to be "better" than another. In order to represent this preference, I extended the MOMoT framework by implementing a multi-objective local hill climbing algorithm, by maintaining a set of Pareto efficient solutions rather than a single "best" solution and uses Pareto dominance to determine whether to add a solution to the set of best solutions. A hybrid of depth first search and breadth first search is used to generate the neighborhood by generating the same number of neighbors for each each solution in the set of current Pareto efficient solutions until during an iteration solutions are found in the neighborhood that changes the Pareto set. Then, only the neighbors of the new solutions added to the Pareto set are searched. This multi-objective hill climbing algorithm is further described in Algorithm 2. The algorithm terminates when the minimum number of iteration is exceeded and the maximum number of attempts to find an improvement is exceeded.

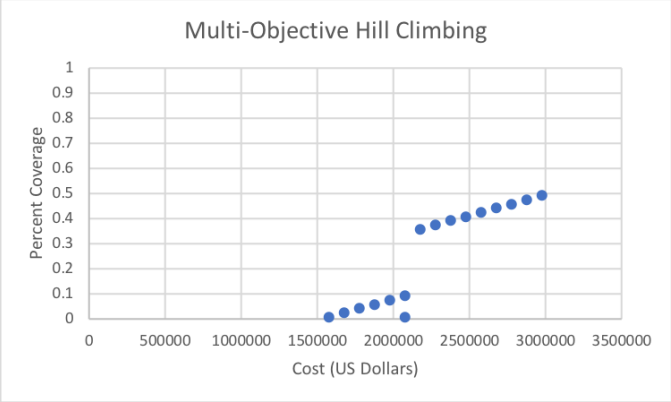

Fig. 9. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using multi-objective hill climbing and the set of both deep and shallow model transformation rules

It was hypothesized that multi-objective hill climbing (MOHC) would better approximate the Pareto frontier than a genetic algorithm on complex system design problems because a single rule change can instantiate many model elements. In Figure 9 one can see that the solutions found using MOHC performed worse than those found using a genetic algorithm in Figure 7. In fact, of the 16 solutions found using MOHC, 5, or 31%, were dominated by solutions found using a genetic algorithm, and of the 126 solutions found using a genetic algorithm, only 1 was dominated by solutions found using MOHC. Despite MOHC approximating the Pareto frontier poorly, the bands in Figure 7

Input : Initial Set of Pareto Efficient Solutions S, Maximum Neighborhood Size n, Minimum Number of Iterations $minimumIterations$, Maximum Number of Attempts $maxAttempts$
Output: Set of Pareto Efficient Solutions found

```
currentSolutions :=
    (|S| > 0) ? S : {aRandomSolution};
visitedSet := {};
iterationCount := 0;
attempts := 0;
numberOfImprovements := 0;
while (attempts < maxAttempts) OR
 iterationCount < minimumIterations do
    notExploredSet := {s ∈ S|s ∉ visitedSet};
    nuberOfNeighborsPerSolution :=
     n / |notExploredSet|;
    foreach s ∈ notExploredSet do
        neighbors := Generate
         nuberOfNeighborsPerSolution Neighbors
         of s;
        foreach neighbor ∈ neighbors do
            if neighbor ∪ currentSolutions is
             Pareto-efficient then
                Add neighbor to currentSolutions;
                numberOfImprovements++;
            else if neighbor Pareto dominates one or
             more solutions in currentSolutions then
                Add neighbor to currentSolutions;
                Remove dominated solutions from
                 currentSolutions;
                numberOfImprovements++;
        end
        visitedSet.add(s);
    end
    if numberOfImprovements == 0 then
        visitedSet := {};
        attempts++;
    else
        attempts := 0;
    numberOfImprovements := 0;
    iterationCount++;
end
return currentSolutions;
```

Algorithm 2: Multi-Objective Hill Climbing Implementation

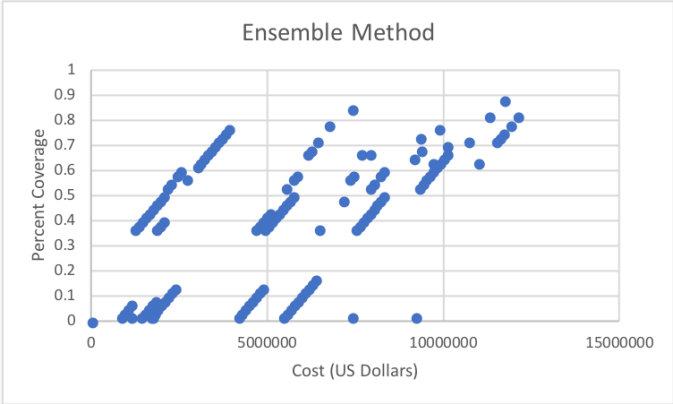

Fig. 10. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using multi-objective hill climbing and the set of both deep and shallow model transformation rules

shows the area around the initial randomly chosen solution was thoroughly searched. This implies MOHC found local extrema near the starting location. Since the success of a solution depended significantly on the starting location of the search, NSGA-II was run with the same parameters stated before and used this Pareto efficient solution set as the input to the MOHC algorithm. The resulting solutions from this ensemble search method are shown in Figure 10. As one can see, compared to the results in Figure 7, the set of solutions formed using this ensemble search algorithm are further to the left. Of the 126 solutions found using only a genetic algorithm 51, or 40%, were dominated by solutions from the ensemble set. Also, of the 132 solutions found using the ensemble method, only 2 solutions were dominated by solutions found using the genetic algorithm alone.

This implies using a genetic algorithm to find the starting solution set for MOHC much better approximates the Pareto frontier than using a genetic algorithm or randomized hill climbing alone. This was expected because genetic algorithms are able to make larger jumps in the search space than hill climbing and don't easily fall into local minima like hill climbing does, but they do not always converge. On the other hand, hill climbing makes small changes hoping to improve on the solution so it is easier to converge to a local extrema. By running the genetic algorithm first, one is able to generate a diverse set of solutions and by running MOHC from these solutions, one is able to push them closer to the Pareto front.

The results from alternating between NSGA-II and MO Hill Climbing twice was examined. As can be seen from Figure 11, using this new proposed algorithm and the new transformation rules significantly improved the quality of solutions found using expert crafted rules and only the NSGA-II algorithm. Using this ensemble method, we were able to improve upon 79% of the solutions found using expert crafted rules and the NSGA-II algorithm. Of the solutions found using an equal number of iterations we were able to improve upon 83% of the solutions found using the autonomously generated rules and the NSGA-II algorithm by

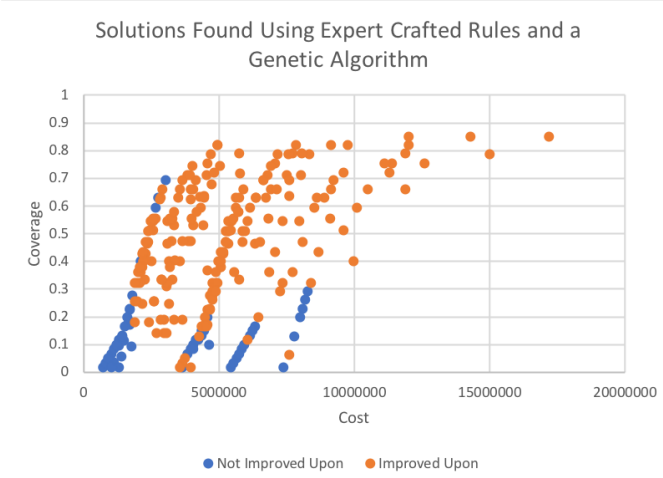

Fig. 11. Graph of cost vs percent coverage for the interferometry mission of the set of Pareto-efficient solutions found using the NSGA-II algorithm and expert crafted rules dominated by those solutions found using an ensemble method

itself.

VI. Conclusions, Limitations, and Future Work

We have shown the time required to find a feasible solution can be decreased by starting search in the feasible model space and by reasoning about the meta-model we are able to autonomously generate a starting point in the feasible model space. This makes the Kigen system easier to use because a user doesn't need to find an initial model to search from. However, autonomously generating a model in the feasible model space to search from will be sensitive to restarts because in situations where more than one class can satisfy a relationship, objects are instantiated uniformly from the set of classes that satisfy the relationship. This added stochasticity will change the results from one trial run to the next. This is a trade users of the system need be aware of. By decreasing the workload on the user to define an initial model the results of the solver become more sensitive to restarts.

Additionally, we have shown autonomously generated transformation rules can perform comparably to those manually designed by experts because using the same number of iterations we were able to Pareto dominate 61% of the solutions found using hand crafted rules by solutions found using autonomously generated transformation rules. This implies a user does not need to know the inner workings of the Kigen solver to find quality solutions, decreasing the wok necessary for a user to find high quality solutions. Future work could be done on selecting graph transformation rules according to different probability distributions. For example, because a model element higher in the containment tree will cause a larger jump in the model space, one could decrease the probability of selecting that transformation rule over time in the style of simulated annealing [13].

Also, we have shown that ensemble algorithms are able to better search the model space than a genetic algorithm because with all else equal solutions found using the ensemble algorithm were able to dominate 83% of solutions found using only a genetic algorithm. This is likely a result of the representation of solutions. Future work should be done analyzing alternative methods of solution representation.

VII. Appendix

Lemmas

1) Consider the problem of finding a subset of a set of components with maximum value, where each element in the subset has a weight. Knapsack is a specific instance of this problem where the value of a subset is the sum of the values of the components of the set. Because any instance of knapsack is an instance of the problem described, the problem described is NP-Hard.

References

[1] S. J. I. Herzig, S. Mandutianu, H. Kim, S. Hernandez, and T. Imken, "Model-transformation-based computational design synthesis for mission architecture optimization," in *2017 IEEE Aerospace Conference*, pp. 1–15, March 2017.

[2] Eclipse Foundation, "Eclipse modeling framework."

[3] T. Arendt, E. Biermann, S. Jurack, C. Krause, and G. Taentzer, "Henshin: Advanced concepts and tools for in-place emf model transformations," in *Model Driven Engineering Languages and Systems* (D. C. Petriu, N. Rouquette, and Ø. Haugen, eds.), (Berlin, Heidelberg), pp. 121–135, Springer Berlin Heidelberg, 2010.

[4] M. Fleck, J. Troya, and M. Wimmer, "Search-based model transformations with momot," in *Theory and Practice of Model Transformations* (P. Van Gorp and G. Engels, eds.), (Cham), pp. 79–87, Springer International Publishing, 2016.

[5] D. Hadka, "Multi-objective optimization algorithms."

[6] E. J. Wyatt, K. Belov, S. Burleigh, J. Castillo-Rogez, S. Chien, L. Clare, and J. Lazio, "New capabilities for deep space robotic exploration enabled by disruption tolerant networking," in *6th International Conference on Space Mission Challenges for Information Technology (SMC-IT 2017)*, (Alcala de Henares, Spain), pp. 1–6, September 2017.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, Apr 2002.

[8] G. Hornby, A. Globus, D. Linden, and J. Lohn, *Automated Antenna Design with Evolutionary Algorithms*. American Institute of Aeronautics and Astronautics, 2018/07/09 2006.

[9] T. Cwik and G. Klimeck, "Genetically engineered microelectronic infrared filters," in *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pp. 242–246, 1999.

[10] J. R. Koza, D. Andre, F. H. Bennett, III, and M. A. Keane, "Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming," in *Proceedings of the 1st Annual Conference on Genetic Programming*, (Cambridge, MA, USA), pp. 132–140, MIT Press, 1996.

[11] R. Bill, M. Fleck, J. Troya, T. Mayerhofer, and M. Wimmer, "A local and global tour on momot," 12 2017.

[12] G. A. Hazelrigg, *Fundamentals of Decision Making for Engineers*. NEILS CORP, 2012.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.