Visualizations for User-supported State Space Exploration of Goal Models

Yesugen Baatartogtokh, Irene Foster, Alicia M. Grubb
Department of Computer Science
Smith College, Northampton, MA, USA
amgrubb@smith.edu

***Abstract*—Automated analysis has been used in goal-oriented requirements engineering (GORE) to evaluate scenarios and make trade-off decisions. For higher complexity problems (e.g., backwards analysis), using a search-based solver may be more efficient than custom algorithms. When these black-box solvers produce a single solution, users may be suspicious about whether the given answer is ideal or believable. Users would like to explore the potential solutions but are prevented from doing so because these inquiries often suffer from a state explosion problem.**

In this RE@Next! paper, we introduce the use of valuation-based filtering and coloring to assist users in understanding a solution space and selecting custom states from it. We use the concrete semantics of modeling requirements in the Evolving Intentions framework and its associated goal modeling tool, BloomingLeaf, to explore the application of these visualization techniques. In our initial evaluation, we demonstrate how these techniques can be used on a fully worked out example. We conduct initial measurements of the time savings and state space reduction created by the valuations and color filtering, and discuss future directions of this project.

I. Introduction & Motivation

Goal-oriented requirements engineering (GORE) provides stakeholders with analysis tools needed to make trade-off decisions in the early phases of a project [1], [2], [3]. GORE frameworks allow stakeholders to document their requirements and represent them through goal models, which consist of actors and intentions (e.g., goals) that are connected via relationships. Users create models for the purpose of considering how alternative project scenarios impact the needs of stakeholders, who depend on the system. Individual intentions in these models can then be given an evaluation label to describe whether the intention is fulfilled with respect to the current scenario under consideration. Goal model analysis enables stakeholders to understand and answer "what-if" questions by exploring alternate scenarios [4], including evolving scenarios [5], [6]. In this paper, we use the Evolving Intentions framework [6] and its associated tool, called BloomingLeaf [7], to look at how project scenarios evolve over time.

Depending on the approach, model analysis may be manual, semi-automatic involving user intervention, or fully automatic [2]. For automatic algorithms, results may be deterministic (e.g., forward propagation, from leaf to root nodes) or non-deterministic with multiple solutions (e.g., backward propagation, from root to leaf nodes). As previously demonstrated, solvers can be used to find a satisfying assignment for the evaluation labels in the model [4], [8], [9]. The model

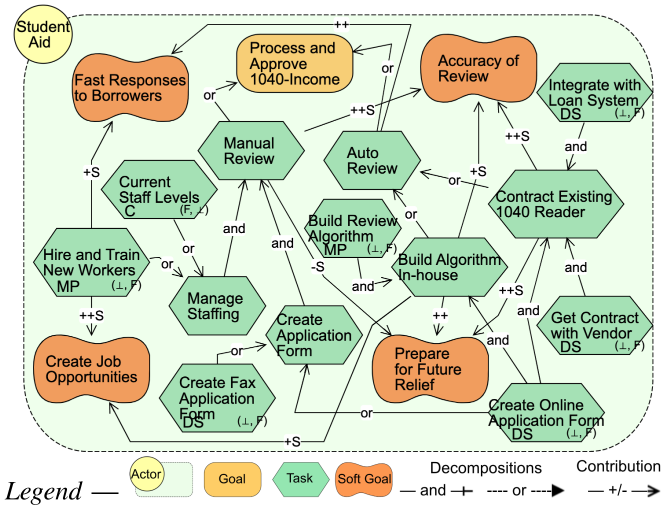

Student Aid
Fast Responses to Borrowers
Process and Approve 1040-Income
Accuracy of Review
Integrate with Loan System DS (⊥, F)
Manual Review
Auto Review
Contract Existing 1040 Reader
Current Staff Levels C (F, ⊥)
Build Review Algorithm MP (⊥, F)
Build Algorithm In-house
Hire and Train New Workers MP (⊥, F)
Manage Staffing
Create Application Form
Get Contract with Vendor DS (⊥, F)
Create Job Opportunities
Create Fax Application Form DS (⊥, F)
Prepare for Future Relief
Create Online Application Form DS (⊥, F)
++
or
++S
+S
-S
and
++
Legend —
Actor
Goal
Task
Soft Goal
Decompositions
— and —+
---- or ---▶
Contribution
— +/- —>

Fig. 1: Fragment of goal model for processing and approving income limits of student loan borrowers via IRS Form 1040.

structure, as well as the intention evaluation labels, act as the constraints placed upon the solver. When analyzing how the evaluation labels of a goal model change over time, a satisfying assignment for each intention must be found at each time point, creating a solution space consisting of multiple paths that the model can take [6]. In this paper, we investigate the problem of solution space exploration when the content of each state is the satisfying assignment for the entire model.

When a black-box solver produces only a single solution, users may be suspicious about whether the proposed answer is ideal or believable, given the provided constraints. As well, users may want to explore other results that satisfy the same criteria or create their own custom paths. Ideally, users would like to explore the potential solutions. However, depending on the size of the solution space (i.e., state explosion problem [10]) and the complexity of any individual state, current visualization techniques (e.g., [11]) are insufficient.

Illustrative Example. Consider the software team at the US Department of Education working to implement the *2022 Biden-Harris Administration Student Debt Relief Plan* [12]. Fig. 1 shows the goal model created in BloomingLeaf by analysts as part of their requirements process. As part of this temporary program, roughly 32 million borrowers are required to send in their most recent tax return (via IRS Form 1040) to verify they are within the income limit and qualify for the relief payment [12]. The team is evaluating trade-offs for processing the 1040 forms. This decision is illustrated by the top-level goal Process and Approve 1040-Income being or-decomposed into Manual Review and Auto Review, with each of these options being further decomposed into their sub-tasks.

In simulating this model over time, the analysts want to answer the question, "Is conducting a Manual Review, Auto Review, or a combination of the two best to ensure that the applications are processed correctly (i.e., satisfying Accuracy of Review) with fast turn-around for borrowers (i.e., satisfying Fast Responses to Borrowers)?". The solver in BloomingLeaf returns one possible path of eight steps (the first four steps are shown in Fig. 2), where each individual intention in the model changes over time according to its given constraints; yet, there are numerous additional valid solutions. In this initial solution, the Manual Review option becomes satisfied first (see t_1 in Fig. 2(b)), denoted by the dark blue or (F, $\perp$) as seen in the legend in Fig. 3. At this time, Accuracy of Review is satisfied and Fast Responses to Borrowers is conflicted, which prompts the team to want to know whether both these soft goals can be satisfied. Additionally, the team wants to explore creating two paths, one where Get Contract with Vendors is satisfied and the other where Build Algorithm In-house is satisfied. BloomingLeaf allows users to answer these types of questions through the *Next States* view, where users can get all possible states for a given time point in the model evolution. However, other than clicking through each possible state, it is unclear how the user can make an informed choice.

The state of the art in visualizing constraint programing looks at analyzing the decisions made by the solver [11]. Seeing a high-level view of the solution space does not enable users to interpret which elements have specific valuations. Verbeek et al. used Petri net models to give users a sense of the valuations of attributes in a solution space [13], [14], but this approach does not scale with more complex models with many attributes. Within GORE, Hu and Grubb proposed a simple set of filters to reduce the domain and solution space [15]. These filters allow state-reduction based on the whole model, such as removing any states that contain a conflicted satisfaction value, but users cannot filter by individual intentions. These filters are generic and eliminate unhelpful states, but do not provide insights or assist the user in selecting a future state.

Contributions. In this RE@Next! paper, we propose the use of valuation-based filtering and coloring to assist users in exploring and selecting states from a solution space more efficiently. We use the concrete semantics of the Evolving Intentions framework, to demonstrate the applicability of this approach, yet these visualization techniques can be adapted to other areas in RE.

In the remainder of this paper, we review relevant background in Sect. II, present our filtering and coloring techniques in Sect. III, and provide an initial evaluation in Sect. IV. We complete our presentation with a discussion of related work in Sect. V, and a description of our future development and validation plans in Sect. VI.

II. Evolving Goal Model Representation

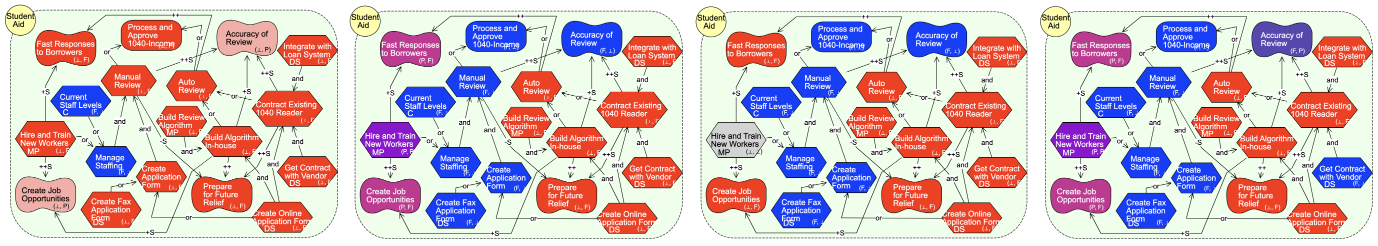
In this section, we define an evolving goal model and describe how it is captured as a constraint satisfaction problem (CSP) [16]. Within the Evolving Intentions framework [6], an *evolving goal graph* M is a tuple $\langle A, G, R, \mathit{EF}, C, \mathit{maxTime}\rangle$, where A is a set of actors, G is a set of intentions (e.g., goals, tasks, and soft goals, see legend in Fig. 1), R is a set of relationships over intentions, *EF* is a set of evolving functions, C is a set of constraints over the time points in the graph, and $\mathit{maxTime} \in \mathbb{N}^+$ is the maximum absolute time over which any time point is defined (adapted from [6]). For the purposes of our work, M is a graph (henceforth called a *model*) consisting of intentions $g \in G$ and these intentions are assigned evidence pairs as valuations. An *evidence pair* is a pair (s, d) where $s \in \{\mathit{Full\ (F)}, \mathit{Partial\ (P)}, \mathit{None}\ (\perp)\}$ is the level of evidence *for* and $d \in \{F, P, \perp\}$ is the level of evidence *against* the fulfillment of g. The cross product of s and d results in nine valuations, denoted as the universe $\mathcal{E}$ (see legend in Fig. 3). The evidence pair assignments for intentions are determined by the model constraints defined in R, EF, and C.

For example, the model fragment in Fig. 1 illustrates an evolving goal model M with one actor and 18 intentions in G. The Current Staff Levels intention has the evidence pair (F, $\perp$) to indicate that it is fully satisfied. Other letter markings (e.g., C, DS) denote evolving functions that constrain the valuation of the intentions in *EF*. Intentions are also constrained by and/or decomposition and contribution links (e.g., ++S) in R.

The model can then be simulated over a series of time points Π, known as a time point path [6]. For a given time point $t \in \Pi$, the evaluation of g at t, is a mapping $G \times \Pi \rightarrow \mathcal{E} \cup \{\perp\}$. Thus, a complete evaluation of M at t is the total mapping $G \times t \rightarrow \mathcal{E}$ resulting from the repeated application of all constraints within R, *EF*, and C. Simply put, a complete evaluation path is the complete evaluation of M at each time point in Π, which we represent as a CSP.

A CSP is defined as a triple $\langle V, D, Q\rangle$, where V is a set of variables, D is a set of the respective domains of values, and Q is a set of constraints [16]. When constructed as a CSP, our aim is to find an evaluation for each intention in the graph ($g \in G$) at each time point ($t \in \Pi$). Thus, the CSP variables V are a set of intentions at all time points (i.e., $|V| = |G| * |\Pi|$). Initially, the domain d of each variable v is the set of evidence pairs ($\mathcal{E}$). Each element in R, EF, and C forms constraints Q over the elements in V. Thus, the valuation of all variables in $|V|$ is a complete evaluation path of model M.

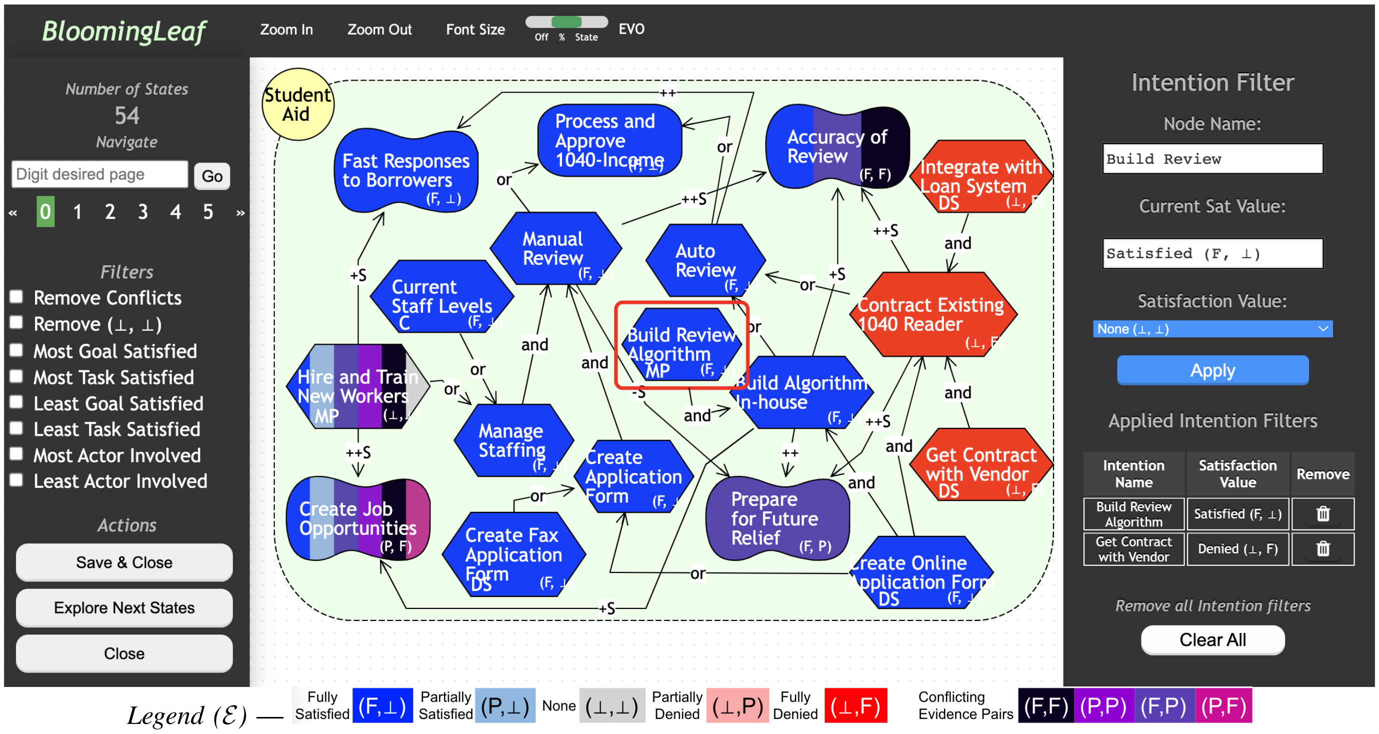
For example, the CSP solver in BloomingLeaf generates a simulated time point path Π for the model M in Fig. 1, consisting of eight timepoints t_0–t_7. As introduced in Sect. I, the team wants to build a path where Build Algorithm In-house is satisfied. They agree with the selected results for t_0–t_2 (see Fig. 2), but want to find an alternative solution for the assignments in t_3. By constraining the acceptable values for the model M at t_0–t_2, we update our CSP solver request in BloomingLeaf to return all possible solutions for t_3, discussed in the next section.



Student Aid
Fast Responses to Borrowers
Process and Approve 1040-Income
Accuracy of Review
Integrate with Loan System DS
Current Staff Levels C
Manual Review
Auto Review
Contract Existing 1040 Reader
Build Review Algorithm MP
Build Algorithm In-house
Hire and Train New Workers MP
Manage Staffing
Create Application Form
Get Contract with Vendor DS
Create Job Opportunities
Create Fax Application Form DS
Prepare for Future Relief
Create Online Application Form DS

(a) t_0 (b) t_1 (c) t_2 (d) t_3

Fig. 2: Fragment of one possible simulation path of the Student Aid model, with EVO color visualization applied (see legend in Fig. 3), showing the evolution of intentions over time points t_0 to t_3.



Node Name: Build Review

Current Sat Value: Satisfied (F, ⊥)

Satisfaction Value: None (⊥, ⊥)

Intention Name	Satisfaction Value	Remove
Build Review Algorithm	Satisfied (F, ⊥)	🗑
Get Contract with Vendor	Denied (⊥, F)	🗑

Fig. 3: BloomingLeaf's Next States view of Student Aid model, with EVO % mode selected and two intention filters applied.

III. Filters & Visualizations

In this section, we describe two techniques to assist users in creating their own path from a solution space of an evolving model. Fig. 3 shows a screenshot of our extension of BloomingLeaf in the state exploration view (called the *Next States* window). BloomingLeaf invokes the JaCoP constraint solver [17] to find satisfying assignments for each variable and displays the first option for the selected time point (t_3) on the center canvas of the Next State window. The left panel shows the existing filters, as presented in [15]. The user can select from the many possible states or incrementally navigate through each state with arrows in the left panel. The new Color Visualization is shown in the top toolbar (called EVO, '%' selected) and the valuation filtering is shown in the right panel.

Color Visualization. Each evidence pair (see Sect. II and legend in Fig. 3) is assigned a color, where blue denotes satisfied, red denotes denied, and purple denotes conflicting values with both evidence for and against the fulfillment of the intention [18]. The more saturated (or darker) the color shade, the stronger the evidence (e.g., F is darker than P).

We introduce two color overlays to assist users in understanding the model and selecting future states. State mode (not shown) colors the background of each intention in the graph with the color associated with the assigned evidence pair. This view is similar to the coloring applied to the model path in Fig. 2. Seeing the colors for each intention allows users to more quickly understand the valuations in the viewed state without reading each of the evidence pairs. The second is the percentage mode, which is selected in the EVO slider on the top toolbar and shown on the canvas in Fig. 3. When percent mode is selected, the background of each intention is colored with the percentage of states in the solution space where the intention has each evidence pair assignment.

Filtering Intention Valuations. To allow users to sort through and find desired solutions, we added an additional panel on the right-side of the window that enables the user to add and remove filters (i.e., Intention Filters). Unlike the filter on the left panel (from [23]), it filters solutions based on the valuation of individual intentions, returning only solutions where intentions have the specified evidence pair. Users are able to add separate filters to different intentions, as well as filter multiple evidence pairs for a single intention. This act reduces the solution space, allowing users to focus only on the options that meet their aims without having to manually evaluate all permutations of an intention. For example, when Build Review Algorithm is selected on the center canvas in Fig. 3 (see red box), then the Intention Filter panel is populated with the information for this element and allows the user to add a new filter.

Usage Scenario. While each of these filters may seem simplistic on their own, taken together they form a powerful mechanism for users to create and validate their own paths. In the illustrative example, after generating the path shown in Fig. 2 (see Sect. II), the user is unsatisfied with the model at t_3. Specifically, the user wants to find a path where Get Contract with Vendors is not the next step, but perhaps Build Review Algorithm is satisfied instead. When the user explores all possible states for t_3, there are 864 states (not shown). First, we recommend turning on EVO % mode to see a high-level view of the solution space. Since the user is looking for a state where Build Review Algorithm is satisfied, they select the node and choose Satisfied $(F, \perp)$ in the right panel, limiting the number of states to 108. With a second valuation filter applied to Get Contract with Vendors, which is shown in Fig. 3, there are 54 possible states. From Fig. 3 the user can see that much of the variability in the remaining states exists within a few elements. If the user had applied the valuation of None $(\perp, \perp)$ to Hire and Train New Workers, then only 9 similar states remain. Finally, the user turns on EVO in State mode, and clicks through the remaining states and selects #5 as their preferred one. By selecting Save & Close (see Fig. 3) the CSP solver generates the remainder of the path given this updated state. Alternatively, the user could click Explore Next States to generate all possible solutions for the next time point (t_4), given the updated choices for t_3.

IV. Discussion & Validation

Benefits. Our work allows for the more efficient exploration of a solution space. The intention valuation filters reduce the number of states in the solution space that need to be manually considered. Valuation-based coloring through EVO State mode may allow for faster processing of evidence pairs in states since users do not need to read each evidence pair label. The EVO % mode shows an overview of the composition of the state space and allows users to verify the absence of a state. For example, given the filters applied in Fig. 3, no state exists where Accuracy of Review is Denied (i.e., colored red).

Limitations. Our current implementation is limited to the Evolving Intentions framework and needs to be expanded to work with other CSPs. Within this framework, we require users to generate a full path before exploring the state space. Another design choice would allow users to explore states

TABLE I: Rate of state review with and without EVO.

Group	Task	Rate of Review (states per second)				
		0	1	2-Close	2-Distant	3
Author First	Base	5.37	4.08	4.03	2.08	0.66
	EVO	5.78	5.64	4.89	4.98	4.69
EVO First	Base	5.12	4.69	3.76	3.16	3.68
	EVO	5.27	5.00	4.30	3.42	3.63
Base First	Base	5.83	3.52	2.53	1.91	1.94
	EVO	5.19	3.8	3.7	3.28	3.00

from the initial model. Further, the color palette may not be intuitive across an international audience. We are extending our implementation to enable users to select the palette colors, including a palette for individuals with a color vision deficiency. Finally, there is an upper bound on the number of states our solver will generate before timing out. The illustrative example without any MP and DS functions produces 1,740,288 states if strong conflict (F, F) is removed from the domain d of each variable v and a memory error if kept. In these situations, it is often the case where the scenario has not been fully explored by the users. We investigate a methodology to help users explore state spaces more efficiently.

Initial Evaluation. Our initial evaluation work focused on demonstrating the feasibility and effectiveness of our approach. We took measurements of the time savings and state space reductions created by the valuations and color filtering. Supplemental information for this evaluation is available at: *https://doi.org/10.35482/csc.003.2023*.

Color Visualization: In a recent investigation, we found that using EVO increased the speed for users to answer questions about goal models [19]. We hypothesized that using EVO in the Next States window enables users to click through states faster than without using EVO. Thus, we collected initial measurements of the time it took individuals to physically click through the states in search of a solution with and without EVO State mode enabled. Tbl. I lists the averaged number of states that were reviewed per second for three groups. Author, see first column in Tbl. I, represents expert clicking by one of the authors of this paper. The other two groups are averages of three undergraduate lab members with one group using EVO First, while the other group (i.e., Base First) were tested without EVO first. For each group, we measured finding a solution given specific evidence pairs for zero, one, two, or three intentions, including two intentions that were close and distant from one another. We surmised that we may observe cognitive processing delays by comparing the speed of searching for close and distant intentions.

Our observations are inconclusive. The Author group was faster with EVO than without EVO in all cases; yet, this has the obvious problem of unconscious researcher bias. The two student groups were often faster using EVO; however, there was significant variation between individuals. Future studies should account for individual variability, while controlling for fatigue and carryover effects [20], as well as the interaction between intention filters and state selection with/without EVO.

TABLE II: Best/worst case percentage reduction in states.

Model	1-Worst	1-Best	2-Worst	2-Best
1. Grad	80.0%	98.9%	99.2%	99.6%
2. WME	6.3%	98.8%	30.4%	99.7%
3. Ready4Work	37.7%	96.1%	68.8%	98.7%
4. Course	88.4%	99.0%	98.0%	99.6%
5. Debt	50.0%	96.1%	75.0%	98.7%

TABLE III: Average user percentage reduction in states using one, two, and three valuation filters.

Model	1 Filter	2 Filters	3 Filters
1. Grad	96.4%	99.6%	99.9%
2. WME	58.7%	87.6%	96.2%
3. Ready4Work	80.5%	91.2%	96.9%
4. Course	97.5%	98.9%	99.5%
5. Debt	81.9%	91.4%	97.7%

Intention Valuation Filtering: To test the effectiveness of applying intention valuation filters, we measured the percentage reductions in the size of the solution space after applying one to two filters on five different models, with a maximum model size of 20 intentions. Depending on the number of constraints in Q, the number of initial states could be anywhere between one hundred and a million. Tbl. II lists the percentage of states that are removed after applying one and two filters in the best case and worst case, based on our own expert judgement. We chose these cases based on visual inspection of the EVO % mode. From Tbl. II, we observe that with a variety of models, in the best case, we reduce the number of states by 96-98% with one intention and 98-99% with two intentions. The worse case reductions were as low as 6%, with averages between 50 and 75%, which indicates that much of the reduction comes from the user choosing an appropriate intention.

To get a sense of how users would apply filters, we asked five lab members to apply three filters to the same five models, based on their intuitions after examining each model. Tbl. III lists the average percentage reductions after applying one to three filters. These average percent reductions are consistent with those in Tbl. II. While state space reduction depends on the users' appropriate choice of a filter, on average, valuation-based filtering provides a 90% reduction in the solution space after two filters have been applied. In summary, intention valuation filtering reduces the state space significantly by removing undesired states, though its efficacy depends on the user's goals and choice in applying an appropriate filter.

Threats and Future Validation: Our initial data collection was never intended as empirical validation. Our main threat is that our evaluation was completed by the authors of this paper and members of our lab. With three people per group for the EVO evaluation and five people total for the filtering evaluation, our sample size was insufficient to make conclusions. In future work, we plan a large-scale study with plausible subjects in a setting that is reflective of how stakeholders would learn to use and apply these methods in the "real world". Further experiments with different populations, problem domains, and contexts will be needed, as well as validating EVO Percent mode. For EVO State mode, further exploration is required to determine whether measuring clicking speed over states per second is an appropriate measurement, as subjects' fatigue and attention span may make their clicking inconsistent.

V. Related Work

Goal Modeling. Prior work in GORE investigated improving the interpretability of goal model analysis. Horkoff and Yu first investigated highlighting root/leaf nodes and conflicting alternatives in goal graphs to assist users in understanding analysis tasks [21]. Reddivari et al. investigated using visual analytics techniques to help in requirements negotiation [22]. More recently, Oliveira et al. used RGB values to color nodes in non-functional requirements models based on their quantitative valuations [23]. Amyot et al. used colors to visualize analysis results in GRL using the jUCMNav tool [2], while TimedGRL used color in heat maps to visualize evolving GRL goal models [5]. Varnum et al. proposed coloring nodes within qualitative goal models to assist users in interpreting the results of path-based analysis [18]. In this paper, we adopt the initial color scheme proposed by Varnum et al. as we use the same universe of evidence pairs. As already stated in Sect. I, we reimagine the work of Hu and Grubb, who reduced the size of a CSP domain and solution space with generic filters [15].

Other work in GORE has used constraint programming for model optimization. Anda combined GRL models and cyber-physical systems to integrate social concepts into the requirements activities for these systems [24]. Alwidian proposed union models to improve the efficiency of analysis by simultaneously analyzing related elements in goal models [9].

Exploring and Visualizing State Spaces. There has been significant work on visualizing and debugging constraint programs [25]. Researchers investigated visualizing search trees (e.g., [26]) and global constraints (e.g., [27]), with later work focusing on explaining and comparing various algorithms for constraint programming (CP). Freuder et al. generated explanations for solving methods in the form of trees [28]. Dooms et al. created a generic approach to visualize constraint-based local-search [29]. Li and Epstein provided high-level visuals of the search space to inform the guided search of CSPs [30]. Simonis et al. created a generic visualization of CP problems for postmortem analysis [11].

Color has been used to visualize and explore decision trees. Rojas and Villegas used color to enhance the weights of nodes in 2D and 3D decision trees, allowing users to visually identify weights associated with a node [31]. Closer to our investigation, Verbeek et al. enabled users to explore state spaces via the attributes of the system and "get a feeling" for their behavior [13], [14]. In addition to visualizing the graph of states, Verbeek et al. generated Petri net models from the state space to give users a deeper understanding of the valuations of attributes. Thus, while there has been a consistent effort to visualize the decisions of the solver, there is limited work on visualizations for the purpose of making human decisions, a significant area for future exploration.

VI. Summary and Future Plans

In this paper, we introduced the use of valuation-based filtering and coloring to assist users in exploring a solution space and selecting custom states from it. We demonstrated the use of these techniques through our illustration of the Student Aid model. Our initial measurements with authors and lab members showed a greater than 90% reduction in the number of states viewable by the user with valuation filtering after two filters have been applied, although this depends on the users' choice. Given the threats described in Sect. IV and mixed EVO results, further study is required to validate our initial observations. In this work, we mitigate the issue of navigating explosive state spaces when the content of each state is an entire model. We see our visualizations as new avenues for interpreting solutions in other areas of GORE.

Our current implementation is limited to the Evolving Intentions framework, which requires users to generate a full path before exploring the state space. Additionally, there is an upper bound on the number of states our solver will generate before timing out due to incomplete scenario specifications. Future work will develop a methodology to assist users in fully describing scenarios and tooling to generate paths from the initial state. The models used in this paper were not reflective of "real world" scenarios. Future validation includes evaluating the scalability of these visualization techniques.

We also intend to conduct a user-validation study to measure the efficiency gains of these filters and visualization techniques on the Evolving Intentions framework, as well as the usability of our extensions to BloomingLeaf. This study will include EVO Percent mode to fully understand the efficacy of EVO. Further, we are extending our EVO color scheme to enable users to select from multiple color palettes including a palette for individuals with a color vision deficiency. Other GORE frameworks support actor evaluations [32]. Adding actor-level filtering may make our methods more applicable across different frameworks. Finally, we make intention filtering more expressive, by adding boolean and comparison operators.

Acknowledgments. We thank our lab members for their assistance. This material is based upon work supported by the National Science Foundation under Award No. 2104732.

References

[1] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented Requirements Analysis and Reasoning in the Tropos Methodology," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 159–171, 2005.

[2] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating Goal Models Within the Goal-Oriented Requirement Language," *International Journal of Intelligent Systems*, vol. 25, no. 8, pp. 841–877, 2010.

[3] J. Horkoff, F. B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos, and P. Giorgini, "Goal-oriented Requirements Engineering: An Extended Systematic Mapping Study," *Requirements Engineering*, vol. 24, no. 2, pp. 133–160, 2019.

[4] J. Horkoff and E. Yu, "Interactive Goal Model Analysis For Early Requirements Engineering," *Requirements Engineering*, vol. 21, no. 1, pp. 29–61, 2016.

[5] Aprajita, "TimedGRL: Specifying Goal Models Over Time," Master's thesis, McGill University, 2017.

[6] A. M. Grubb and M. Chechik, "Formal Reasoning for Analyzing Goal Models that Evolve over Time," *Requirements Engineering*, vol. 26, no. 3, pp. 423–457, 2021.

[7] ——, "BloomingLeaf: A Formal Tool for Requirements Evolution over Time," in *Proc. of RE'18: Posters & Tool Demos*, 2018, pp. 490–491.

[8] G. Mathew, T. Menzies, N. Ernst, and J. Klein, "Shorter Reasoning About Larger Requirements Models," in *Proc. of RE'17*, 2017.

[9] S. Alwidian, "Union Models: Support of Variability Modeling and Efficient Reasoning About Model Families Over Space and Time," Ph.D. dissertation, University of Ottawa, 2020.

[10] E. M. Clarke, W. Klieber, M. Novacek, and P. Zuliani, "Model Checking and the State Explosion Problem," in *Tools for Practical Software Verification*, ser. LNCS 7682. 154-163, 2012, pp. 1–30.

[11] H. Simonis, P. Davern, J. Feldman, D. Mehta, L. Quesada, and M. Carlsson, "A Generic Visualization Platform for CP," in *Principles and Practice of Constraint Programming*, 2010, pp. 460–474.

[12] Federal Student Aid, An Office of the U.S. Department of Education, "One-Time Student Loan Debt Relief," Online at *https://studentaid.gov/debt-relief-announcement/one-time-cancellation*, 2022, accessed 10/01/2022.

[13] H. Verbeek, A. Pretorius, W. van der Aalst, and J. van Wijk, "On Petri-net Synthesis and Attribute-based Visualization," in *Proc. of PNSE'07 Workshop*, 2007, pp. 127–141.

[14] H. Verbeek, A. Pretorius, W. Van der Aalst, and J. van Wijk, "Visualizing State Spaces with Petri Nets," *Computer Science Report*, vol. 7, no. 01, 2007.

[15] B. C. Hu and A. M. Grubb, "Support for User Generated Evolutions of Goal Models," in *Proc. of MiSE'19 Workshop*, 2019, pp. 1–7.

[16] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010, ch. 6 Constraint Satisfaction Problems.

[17] K. Kuchcinski and R. Szymanek, "JaCoP - Java Constraint Programming solver," http://jacop.osolpro.com, 2016, accessed: 2016-02-21.

[18] M. H. Varnum, K. M. B. Spencer, and A. M. Grubb, "Towards an Evaluation Visualization with Color," in *Proc. of iStar'20 Workshop*, 2020, pp. 79–84.

[19] Y. Baatartogtokh, I. Foster, and A. M. Grubb, "An Experiment on the Effects of using Color to Visualize Requirements Analysis Tasks," in *Proc. of RE'23*, 2023.

[20] S. Vegas, C. Apa, and N. Juristo, "Crossover Designs in Software Engineering Experiments: Benefits and Perils," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 120–135, 2016.

[21] J. Horkoff and E. Yu, "Visualizations to Support Interactive Goal Model Analysis," in *Proc. of REV'10 Workshop*, 2010, pp. 1–10.

[22] S. Reddivari, S. Rad, T. Bhowmik, N. Cain, and N. Niu, "Visual Requirements Analytics: A Framework and Case Study," *Requirements Engineering*, vol. 19, no. 3, pp. 257–279, 2014.

[23] R. F. Oliveira and J. C. S. do Prado Leite, "Using Colorimetric Concepts for the Evaluation of Goal Models," in *Proc of MoDRE'20*, 2020, pp. 39–48.

[24] A. A. Anda, "Combining Goals and SysML for Traceability and Decision-Making in the Development of Adaptive Socio-Cyber-Physical Systems," Ph.D. dissertation, University of Ottawa, 2020.

[25] P. Godefroid, "Model Checking for Programming Languages Using VeriSoft," in *Proc. of POPL'97*, 1997, pp. 174–186.

[26] H. Simonis and A. Aggoun, "Search-tree Visualisation," in *Analysis and Visualization Tools for Constraint Programming*. Springer, 2000, pp. 191–208.

[27] P. Deransart, M. V. Hermenegildo, and J. Małuszynski, Eds., *Analysis and Visualization Tools for Constraint Programming: Constraint Debugging*, ser. LNCS 1870. Springer, 2000.

[28] E. C. Freuder, C. Likitvivatanavong, and R. J. Wallace, "Deriving Explanations and Implications for Constraint Satisfaction Problems," in *Proc. of CP 2001*, 2001, pp. 585–589.

[29] G. Dooms, P. Van Hentenryck, and L. Michel, "Model-Driven Visualizations of Constraint-Based Local Search," in *Proc. of CP 2007*, 2007, pp. 271–285.

[30] X. Li and S. L. Epstein, "Visualization for Structured Constraint Satisfaction Problems," in *Proc. of AAAI'10 Workshops*, 2010.

[31] W. A. C. Rojas and C. M. Villegas, "Graphical Representation and Exploratory Visualization for Decision Trees in the KDD Process," in *Proc. of IC-ININFO'12*, 2012.

[32] X. Franch, G. Grau, and C. Quer, "A Framework for the Definition of Metrics for Actor-Dependency Models," in *Proc. of RE'04*, 2004, pp. 348–349.