A rejoinder to the review comments

We thank all reviewers for their interest in our work and for the very useful and stimulating comments. The paper has undergone huge improvements in terms of formal presentation, complexity analysis, and experimental results. Below we summarize the notable changes made in the current version compared to the previous submission.

1. All concepts such as domain transitions graphs(DTG), multi-valued parallel planning, and DTG-based planning are formally defined.

2. Algorithms to encode DTGs to CSP and to decode the CSP solution back to a parallel plan are described and their complexities are analyzed.

3. Proofs are provided to show the necessary and sufficient conditions for parallel conflicts between two actions.

4. Soundness of the whole constraint model is formally proved.

5. Our reconstruction of the PaP2 model and our improvements to the new planner named PaPR are explained in detail. The encoding and decoding algorithms are presented and their complexities are analyzed.

6. The experiments are significantly improved. PaP2 model is now reconstructed to run on a new version of Minion, which supports sets of values in the table constraints.

7. We showed the effects of using don't cares and mutexes on top of basic TCPP and PaP2 models.

8. The effect of Minion configurations (variable ordering heuristics and constraint propagation techniques) is examined on both models and and the difference in the behaviors is analyzed based on the statistics of the two encodings.

Below we address the concerns of each reviewer.

Dear Drs. Ghooshchi, Namazi, Newton, & Sattar,

We have received all the reviews for the paper you submitted to JAIR, entitled "Encoding Domain Transitions for Constraint-Based Planning". The reviewers agree that the paper presents original work that is relevant and interesting for the JAIR audience. However, the reviewers and I have also identified several difficulties, which preclude publication at this stage. Since JAIR's policy is to accept papers that only require minor revisions, we cannot currently accept the paper. However, we would be happy to see a revised paper on this subject--addressing the concerns below--published in JAIR. Thus, I strongly encourage you to resubmit the paper.

The central weakness of the paper is the lack of precision and formality in the presentation. In particular, there are at least two areas that must be treated formally. First, a proof of correctness of the encoding is needed. It is necessary to formally define and prove the soundness and completeness of the CSP model wrt the planning problem. This formalization should include the formal definition of a "transition-plan" (given that the encoding does not include action variables) that is the result of solving the CSP and a one-to-one mapping for transition-plans in the planning domain to solutions to the CSP (as this is presumably the case?). Second, the mapping between a transition-plan and an "action-plan" must be fully and formally shown. It is necessary to show that there exists an action plan for each transition plan (i.e., for each solution to the CSP) and the formal relationship between these two plans (is it a one-to-one mapping or could a transition plan map to multiple action plans (and presumably not vice versa)?). Furthermore, the algorithm for converting a transition-plan to an action-plan must be presented and proved correct, including a complexity analysis. This latter aspect is particularly important because it appears that relative to previous CSP and SAT encodings, the absence of action variables in the encoding is a key contribution and novelty of the current work. It is also necessary to include the CPU time required for this conversion in the empirical results as, to my understanding, the CSP model does not produce a solution (i.e., a set of actions) as is normally understood in the AI planning problem.

>>> All the concepts such as domain transition graphs (DTG), multi-valued parallel planning, and DTG-based planning are formally defined. Algorithms to encode DTGs to CSP and to decode the CSP solution back to a parallel plan are described. Soundness and completeness of the algorithms are proved and the complexity of them are analyzed.

I also would like to bring attention other important concerns:

- The experimental design can be improved:
  - It provides no insight as to why the proposed model results in better performance nor to explain the lack of such improvement with some parameter combinations (cf. Reviewer #1).

>>> The experiments are improved. PaP2 model is now reconstructed to run on a new version of Minion which supports sets of values in the table constraints. Thus, the PaP2 and TCPP models both are now compared on the same platform. Also, reconstructed PaP2 is significantly enhanced by using don't-care values and by adding optional mutex constraints. The effect of Minion configurations (variable ordering heuristics and constraint propagation techniques) is examined on both models and and the difference in the behaviors is analyzed based on the statistics.

  - As pointed out by Reviewer #2, the tuning of the CSP models is done on exactly the same problem instances as the testing. A much more robust design would be to use cross-validation as suggested by the reviewer.

>>> We are not performing any artificial learning here and not developing any knowledge-based planner. Performing learning would need a thorough statistical study of characteristics of the planning problems using machine learning methods and cross-validation is relevant in that context. While a portfolio learning would be interesting, in this paper, our aim is just to explore the effect of different available configurations of Minion. We run a number of variants of our planner and not all of them could be presented nicely in a chart or table. Our selection was mainly to take the best planner variants. We now carefully avoided wording that could someway imply learning.

- Some important aspects of the Minion solver are not well defined. As with Reviewer #3, I was surprised that SAC is sometimes the best propagation method, given its expense: a more detailed explanation of just what this option does is necessary.

>>> The reason about why SAC method is the best propagation method for TCPP is explored and explained based on the statistics of the planning benchmark problems. SAC is better because it could efficiently prune the search space particularly in the unsatisfiable CSP instances.

- Investigate whether (as indicated by Reviewer #3) SICStus Prolog can represent "don't cares" by using the complete set of values. Using this solver would help to further elucidate the solver/model interactions.

>>> Unfortunately we are not familiar with SICStus Prolog but to explore the interaction between the solver and models, we reconstructed PaP2 model and run on a new version of Minion which supports sets as cell in the table constraints.

- Reviewer #3 provides a number of comments about the details of your encoding from the lack of a formal definition of a DTG to the inclusion of details that do not appear on the edges of the DTG (i.e., the conditions for tau + 1). These need to be clearly addressed and formalized especially if you are departing from "standard" preprocessing to convert the PDDL problem description into your model.

>>> Our Domain Transition Graphs are now formally defined and the algorithm to generate them from the SAS+ representation is provided.

- The descriptions of the PaP2 and PaPr encodings need to be improved (cf Reviewer #3).

>>> PaP2 encoding and our improvements to this model named PaPR are explained in detail. Encoding and decoding algorithms are presented and their complexities are analyzed.

- A number of typos, grammatical errors, and incomplete references.

>>> Typos, Grammatical errors and missing references are corrected to our best.

Attached you will find the reviewers' comments and suggestions. Please address them in any subsequent resubmission to JAIR.

Please note that papers can be resubmitted to JAIR only once. If you choose to revise the paper, please submit it through our standard web process, and include a cover letter describing the changes and responding to the reviewers' criticisms.

Thank you for submitting your paper to JAIR.

Regards,

Chris Beck
Associate Editor

Review 1

Title: Encoding Domain Transitions for Constraint-Based Planning
Author(s): Nina Ghanbari Ghooshchi, Majid Namazi, M.A. Hakim Newton and Abdul Sattar

Overall evaluation
==========

[ ] Accept as is
[ ] Accept with minor revisions
[X] Reject with resubmission encouraged
[ ] Reject

Comments and suggestions
==========

The article describes an encoding of classical planning problems into a CSP. In the proposed encoding the only variables are the values of state variables. The preconditions and effects of actions are compiled into table constraints describing the compatibility/incompatibility between state variables values in a transition. The approach was implemented using a classical CSP system (Minion) and evaluated against both other CSP based encoding and some state-of-the-art planners.

The article is clearly written with a good balance between general description of encoding principles and illustration on several examples. To the best of my knowledge, the proposed encoding approach is original.

The proposed encoding is attractive because it completely rules out the actions from the CSP model: it only deals with state variables and their possible values. In a sense, it results in a more "homogeneous" CSP compared with encodings that refer to both state variables and actions. Maybe that can explain (at a very high level) why the proposed encoding behaves better as, in general, CSP engines tend to be more efficient at solving "homogeneous" problems.

I think there are two issues that should be fixed/clarified before the article can be accepted.

1. Feasibility of decoding

Completely ruling out the actions from the CSP model seems to be a good idea but on condition that a feasible solution to the CSP can always be decoded into an action plan. Stated otherwise, it relies on the fact that the set of negative constraints is "complete" and captures all the possible incompatibilities between actions. This condition is never clearly stated in the paper. If it is clear that the 6 conditions enumerated in section 4.3.2 are sufficient conditions for two actions to be inconsistent, it does not say that this set of conditions is necessary. If that was already proved by Bartak&al or other authors, then you should provide a reference to this result. Otherwise you should provide a proof.

>>> We revised the 6 conditions for parallel conflict and proved that theses are the necessary and sufficient conditions for two actions to be inconsistent

In fact the decoding step is not described in the article and I think it would deserve more attention.

>>> Decoding algorithm is now included in the paper and its time complexity is analyzed.

2. Experimental results

Experimental results seem to suggest that the proposed TCPP encoding outperforms PaP-R but you do not provide any analyze of the *reasons* why it is so. Furthermore, as illustrated on Fig 15, the comparison between TCPP and PaP-R highly depends on the CSP solver configuration (variable selection heuristics, propagation level) as for instance with "shortest domain first" and "generalized arc-consistency", the proposed encoding seems to be worse than PaP-R. It would really be useful to provide some explanations for this.

A possible way to go could be to take particular problem instances and see what are the decisions/propagations occurring during the CSP search in the different encodings in order to explain why some heuristics/inferences are better in one encoding or the other. You could start answering some questions like:
- why "shortest domain first+generalized arc-consistency" seems to be the worse configuration for both TCPP and PaP-R: does it really do something bad when translated into the original planning problem?
- why is PaP-R performing so well on Airport problems compared to TCPP, what is the characteristic of the Airport problem that explains it ?
- similar question for Block's World for TCPP …

>>> The experiments are improved. The PaP2 model is now implemented with the new version of Minion which supports sets as cells in table constraints . TCPP and PaP2 models are now compared on the same platform. Also, PaP2 is improved by using don't-care values and also by adding optional mutex constraints. The effects of Minion configurations (variable ordering heuristics and constraint propagation techniques) is examined on both models and the difference of planner behavior is analyzed based on the statistics.

Deciding which encoding is better in which domain needs in-depth understanding of the domain dynamics (which is out of scope of the paper) and some detailed knowledge about the solver as well (which is not available to us). Besides, characteristics of many CSP techniques are not well-understood even in the CSP research area. Our focus is to look at the overall behaviour and we can explain the difference of the planner behavior based on the domain size of the resulting CSP variables in the encodings and some other factor such as numbers of negative tables. We addressed these issues in the experiment section.

In the experimental section, the part on "plan length comparison" is, from my point of view, not very useful because except for Symba2, the other planners do not explicitly try to minimize plan length so any impact on plan length is purely incidental (unless shown otherwise but again, this would mean digging into the details of the resolution of the CSP to find some explanation).

>>> The plan length comparison is done just to show that in classical planning where actions do not have cost, SAT-based and CSP-based planners that optimize the horizon, can get reasonable overall plan length. Also we showed that PaP2 plans are shorter because of the effect of value selection

heuristic. We do not claim any contribution here, we just report the plan length to show the side effects, knowing that it is incidental.

Some more detailed comments

- In the introduction a couple of sentences are not clear:
* "Two possible ... and thus might improve variable and value choices during search": without more context, it is difficult to understand what is the "solution depth" and the "long and entangled chains"
* "In this paper we investigate the second direction ...": what are the 1st and 2nd directions here?

>>> This sentence is corrected.

- In the review of the state of the art in the introduction, you do not mention the POCL planners that also use some Constraint Programming techniques (like CPT [1]). Or do you focus here only on using CSP techniques (so table constraints only) as opposed to CP in general?

[1] V. Vidal and H. Geffner, 2006. Branching and pruning: An optimal temporal POCL planner based on constraint programming. Artificial Intelligence 170(3):298-335.

- On the same line, you say in section 2.4 that "Constraint satisfaction techniques are first applied to AI planning in (van Beek & Chen, 1999). Constraint satisfaction techniques were used in planning earlier than that and in an domain-independent automated planners, see for instance [2].

[2] M. Ghallab and H. Laruelle, 1994. Representation and Control in IxTeT, a Temporal Planner. In AIPS, 61-67.

>>> The references are added.

- In the section on "preliminary knowledge" or even earlier in the introduction, you should make it explicitly clear that the described approach is restricted to "classical planning". Your references to the PDDL language may suggest that your encoding is able to handle the complete PDDL language which is clearly not the case.

>>> In the preliminary knowledge we formally defined the classical planning

- On Fig 3, it is not clear what is the meaning of the numbers (example 2:) on the edges.

>>> corrected

Typos:
---

p3, section 1: "Secion 3 ..."

p9, section 2.3: "... in our model contain don't care values ...". I would suggest to use some typographical effect (e.g. italicize) the "don't care" in all occurrences in the article for slightly easier reading. Also when you first introduce this value here, it would be nice to say a bit more about its exact semantics.

p 31, acknowledgements: "... We are garteful …"

>>> corrected

Summary:

The article is clearly written. The proposed encoding is original and attractive. I think the article should be resubmitted by addressing two issues:
1- A more detailed description of the decoding step and in particular a proof that it is feasible
2- An experimental evaluation that *explains* (and not only *notice*) the advantages of the proposed encoding

---
Review 2

Title: Encoding Domain Transitions for Constraint-Based Planning
Author: N. Ghanabri, M. Namazi, M.A. Newton, A. Sattar

Synopsis:

The authors present a new constraint model and a variation on an existing constraint model for solving planning problems. The constraint models are then solved using an off-the-shelf constraint solver called Minion. The approaches are evaluated and compared against existing approaches on an extensive set of benchmarks.

General Comments:

The paper is adequately written and makes good use of examples to illustrate the ideas. The problem of improving automatically generated constraint models for planning is well motivated and has been the subject of several papers by different authors in the past. The key improvement here is in a new encoding into constraints of the state transitions in the planning graph and the use of don't cares/wild cards in succinctly specifying the constraints (which relies on a capability in the Minion constraint solver). I found the results in the paper to be, although mixed, still interesting.

Perhaps the one criticism I have of the paper is of the experimental design. The best configurations for the proposed planners TCPP and PaPr are first found (Figures 14 & 15) and then these best configurations are compared against the current state-of-the-art planners. The issue, a common one, is that this conflates training and testing. I.e., we are training on the same set that we are evaluating on to learn the best configuration of the planners. This is known, of course,

to give overly optimistic estimates of how, in this case, the planners would behave on problems that have not been seen before. Can you show the same results by carefully dividing the planning instances into mutually exclusive training and testing data sets? Perhaps by doing something like 10-fold cross-validation, where you pick the best solvers using 9 of the folds and evaluate the solvers on the 10'th fold. The folds could be either over the entire set of planning instances or over the benchmarks (e.g., learn on all but one of the planning domains and evaluate on the planning domain left out). Also, for the competing planners such as PaP2, did you similarly attempt to find the best configuration of this planner over the same problem sets? If not, why not?

>>> We do not intent to perform a portfolio learning or any learning in this paper. Performing learning would be itself an interesting work and would need a complicated statistical study of characteristics of planning problems using machine learning methods. Our aim is just to investigate the effect of different available configurations of Minion. Since the performance of CSP-based planners are much worse compared to the state-of-the-art planners, for each CSP-based planner, we just tried to select the best performing variant. Moreover, we now carefully avoided wording that could someway imply learning.

>>> Since original PaP2 has no input for selecting the configuration, we could not explore the effects of configuration on that planner. However, we reconstructed the PaP2 model to a planner named PaPR and run it on Minion. We investigate the effect of different Minion configurations on PaPR and its enhanced versions.

Specific Comments:

- p.1: "While many key characteristics of a planning problem are overlooked in this way, ...". Please elaborate on what these key characteristics are.
- p.8: "and theses tables" should be "and these tables"
- p.8: "singleton-arc-consistency", elsewhere you consistently use "singleton arc-consistency".
- p.9: "satisfiability(SAT)", insert a space before left parenthesis. Similarly, elsewhere in the paper you omit a space before a left paren.

>>> Corrected

- p.12: first use of "negative constraints" and p.18: first use of "negative table constraints": The idea of negative table constraints isn't widely known outside of constraint programming, so it should be explained.

>>> Negative table is defined

- p.19: "X and Y" then transitions into "A, B".
- p.22: "Lastly, The"
- p.23: "Minion supports two options for variable ordering": I believe Minion supports seven options; see page 63 of the

Minion manual.
- p.23: "shortest domain first (sdf)" should be "smallest"
- p.25, Fig. 15: "PaP-R" in caption.

>>> Corrected

- For PaP2, this is run using SICStus Prolog on your hardware, correct? How much of the difference in performance of the planners is explained by the difference in performance of the SICStus Prolog constraint solver versus the highly optimized Minion solver? I understand that you are presenting a planning system (TCPP = constraint model + Minion constraint solver), but it would still be interesting to quantify the improvement due to changing the underlying constraint solver. Are there experiments that you could design that would quantify this? Can you solve PaP2's constraint model using Minion, for example, or does the PaP2 constraint model rely on SICStus Prolog features that are too far away from what is available in Minion?

>>> PaP2 model is reconstructed using a new version of Minion which supports sets of values as cells in the table constraints and the reconstructed planner PaPR is compared with the original PaP2. The behavior is different on instances but overall the numbers of solved problems are almost the same. To compare the encodings regardless of the solver used, we compared PaPR with TCPP both running on Minion. We observed that don't-cares are a key to efficiency for both planners but still TCPP outperforms PaPR in most planning domains. The reasons are explained in the paper.

- Which constraint-based planner to use varies a lot depending on the planning domain. Do you have anything to say about using portfolios of constraint-based planners? Or about predicting which constraint-based planner to use given a planning domain?

>>> Deciding to use which planner in which domain needs a detailed statistical study of features of domains and the behavior of CSP models in each domain which needs machine learning techniques and is out of scope of this paper. It is an interesting subject for future work.

Review 3

Review of paper Encoding Domain Transitions for Constraint-Based Planning

Overall evaluation: reject with encouragement to resubmit

The paper is about solving planning problems by encoding them as a sequence of constraint satisfaction problems with progressively increasingly makespan. Two constraint models are proposed in the paper; one that directly encodes a slightly extended domain transition diagrams for state variables using extensionally defined (table) constraints and another one rewrites existing encoding from the PaP2 planner using domain transition diagrams. The first encoding was already presented at AAAI 2015, the second encoding is a new contribution of this paper. Extensive

experimental evaluation shows that both encodings are more efficient than the current state-of-the-art PaP2 planner though not yet competitive with domain-independent planners.

I believe that the presented material is worth for journal publication, but in my opinion the current text requires significant rewriting. The first part is reasonably clear though it requires some extension (see below), but the part on PaP2 and PaPr must be rewritten as it does not describe PaP2 accurately and does not describe PaPr precisely enough so the reader can reconstruct it. This is critical as PaPr is a new encoding proposed at this paper. This is the major reason for my overall evaluation. The section with experiments is fine but could be deeper in the explanation of behavior of planners. As accept with major revision is not an option in JAIR, I suggest rejection but encourage re-submission after the below-mentioned problems are resolved.

>>> PaP2 encoding and our improvements to this model named PaPR are explained in detail. Encoding and decoding algorithms are presented and the complexity of them are analyzed. The resultant planner could be reconstructed by reading the paper. The experimental section is improved and more explanation on certain planners' behavior is added.

Particular comments (in the order of appearance):

Page 6, Figure 3: It would be useful to define DTG formally, especially because an extended version is used there. Examples are nice to demonstrate the concept but there should be a formal definition in the journal paper. In particular, explain formally when the node '-' is used and how the constraints are designed.

>>> The redefined domain transition graph is formally defined

Page 7, row 8: Each row of ... is an assignment to variables X_1, X_2, and X_3 that satisfies the inequality.

>>> corrected

Page 7, para 2, row 3: You wrote that value is GAC if there exists a row in the table. You explain only later that you modify the table in such a way that it contains only the values from current variables' domains. This is how Minion treats it but, a more typical "standard" way is that the table is given and not changing so I would write that the row in the table (the support) must contain values from current variables' domains. I mean the definitions of notions should be independent of particular implementation.

>>> addressed as deemed appropriate

Page 8: I think that it is not necessary to explain GAC using a long example. It does not hurt, but formal definition is enough as this is not a paper about GAC; GAC and SAC is only used there as a tool.

>>> Not all people in the planning community are familiar with CSP concepts. Since SAC method is one of the factors behind the efficiency of TCPP planner, we wanted to explain it by a simple problem.

Page 8, para 2, row 2: there are no supports for pairs

Page 9 para 3, row 11: What is equivalency checking? It was never explained in the paper.

>>> Corrected

Page 11, Preprocessing+Decoding: While you discussed extensively GAC, you later omit many details of your method. As mentioned above, it would be useful to have a formal definition of DTG version that you use and also the algorithm how you get it from PDDL. Similarly, as you do not use actions directly in the model, it would be useful to have the description of method, how you generate the plan from the solution of the CSP.

>>> Our domain transition graphs are formally defined. Algorithms to generate DTGs and also to encode DTGs to CSPs and decode CSP solutions to plans are explained in detail. The time and space complexities are analyzed. Also the proofs of soundness and correctness are provided.

Page 11, para 4, row 1: As it is mentioned; row 3. The solver also has
Page 11, para 5, row 3: state variables at times

>>> corrected

Page 13, row 8: It is a bit surprising to see that you add the variables from conditions also for time tau+1. Note that these variables do no appear at edges of DTG so you cannot get their values from DTG but you need to go back to the description of actions. I think it requires more detailed clarification. For example, do you need to include these variables at all (the transitions will be also described in their DTGs)? If yes, why? If not, what is the effect on performance if they are used vs not used?

>>> In the formal definition of redefined DTGs, we conceptually have the variables both in the preconditions and effects of actions on edges but in the implementation we just use the actions and extract this information whenever needed. These variables are needed to ensure that whenever a transition is made in a DTG, the action on the edge also satisfies the preconditions and executes the effects on other variables.

Page 18: Are the mutex constraints necessary in the model or are they redundant? Why do you express them as a table? It looks like this is a classical disjunction. Is there a difference between using disjunction and this table?

>>> The mutex constraints are optional but using them, we have a major improvement in the efficiency of the planner; which is explained in the paper. Also, we added mutex constraints to PaP2 model and improved its performance as well. It is mentioned in the paper that the mutex tables with similar scope are then merged to have a compact representation. Therefore using tables is much more suitable than using disjunction.

Anyway, the paper misses a more formal clarification of the model. It is important to show correctness of the model, that the model really describes the planning problem. It is important to distinguish between the required constraints and the redundant constraints (if any) that are used to strengthen propagation. From the text it is not clear if some constraints are redundant.

>>> We provided a formal description of the model and all needed concepts. We also provided proofs of correctness of the model. We clarified which constraints are necessary and which are optional.

Page 22, Section 5: This is the weakest part of the paper. First, I believe the description of PaP2 is not correct, or perhaps Figure 13 is misleading. The PaP2 constraint model is not based on the

planning graph. The action variable is there for each state variable and describes which action (if any) is changing that variable. Also, there is no single no-op operation, but for each value of the state variable there is one no-op describing that that value is used in the transition. The action succession constraints are not derived from a single automaton as all pairs of consecutive edges there. These constraints assume all effects and preconditions of these actions so even the actions might be consecutive in the automaton they might not follow each other due to incompatible value of another state variable that they share.

>>> The section describing the PaP2 model is totally revised. PaP2 encoding and our improvements to this model named PaPR are explained in detail. Encoding and decoding algorithms are presented and the complexity of them are analyzed.

Page 22, para 3: You wrote that you cannot use SICStus Prolog as it does not support “don’t cares”. This is not true. The table constraint in SICStus Prolog allows using a set of values in the table so if you put all the values to this set, you get the “don’t care” value. PaPr constraint model should be described there more precisely. Honestly, from the text I do not understand how the model looks like.
Do I understand right that in your experiments you used SICStus Prolog to run PaP2 and Minion to run PaPr? It might be that the speed difference is because of the constraint solver not because of the model. Hence it is critical to encode all models in a single system and then compare their efficiency. Otherwise it is not clear what the reason for the speed-up is. Is it because of the solver or because of the model?

>>> The PaP2 model is now implemented with new version of Minion which supports sets as cells in table constraints. We compared PaPR with TCPP both running on the same platform. We observed that the don't-care feature is a key to efficiency for both planners, but still TCPP outperforms PaPR in most planning domains. The reasons are explained in the paper.

Page 23: By the shortest domain first do you mean the classical min-dom heuristic (smallest domain first)?

>>> Corrected

When you mentioned using SAC, does the system really use SAC during search? SAC is very expensive and hence it is not usually used in nodes of the search tree so it is surprising to see that is is used there. Please clarify it.

>>> SAC is used during search. Although it is costly, it is worthwhile in planning problems because of its ability to prune the search space effectively. The cost of SAC in TCPP model is less than that of PaP2 because of its smallest domains.

Page 29: You showed that different planners behave differently for different domains. Do you know why? Is there some property of the domain that makes one planner faster than another planner? At section 6.4 you mentioned some encoding statistics, is there any relation between them and between performance of the planners? It would be really interesting to understand how properties of the domain influence efficiency of particular encoding.

>>> Understanding which planner works better on which domain needs complicated statistical analysis using machine learning techniques. Performing learning is out of scope of this paper. We explained in the paper how performance varies with different encoding statistics.