A rejoinder to the review comments

We thank all reviewers for their interest in our work and for the very useful and stimulating comments. Below we summarize the notable changes made in the current version compared to the previous submission.

1. We totally deleted the parallelism Constraints which were negative tables and instead of that we added parallelism variables to the transition constraint tables to prevent conflicting actions from happening at the same time. Section 5.4 is rewritten.

2. Algorithm to decode the CSP solution back to a parallel plan is slightly changed and is described with more details.

3. The experiments are repeated for the new TCPP planner and the figures and tables are updated.

Below we address the concerns of each reviewer.

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

We have received all the reviews for the paper you submitted to JAIR, entitled "Encoding Domain Transitions for Constraint-Based Planning". While the reviewers all indicated the opinion of an acceptance with minor revisions, Reviewer 1 pointed out a potentially substantial flaw (which I quote (with some minor editing) as):

In the example of Figure 13, because actions \alpha and \alpha' are clearly conflicting the row corresponding to the action pair (\alpha,\alpha') is added in the negative table and, as stated "There is only one row in this table listing the values for the variables. During search this combination of variables and values will be avoided". Now suppose, in the domain, we have an additional action \alpha" that permits this transition, for instance an action \alpha":
preconds: v1=1, v2=5, v3=2, v5=6
effects: v3=3, v4=4, v5=7
I see a risk that the application of action \alpha" will always be forbidden (by the row in the negative table) for the wrong reason that there is somewhere else in the domain a conflicting pair of actions (\alpha and \alpha'). So is the formulation of parallelism constraints not over-constraining the problem?

>>>We don't have negative parallelism constraints in our new planner and instead we prevent conflicting actions by using parallelism variables. As explained in the paper, for the actions \alpha and \alpha' and the new action \alpha", we will have 3 different rows in the table with different values for parallelism variable. Now preventing \alpha and \alpha' from happening at the same time, does not prevent \alpha" from happening.

You and I have subsequently communicated about (see below for completeness) and you indicated that it was indeed a problem but that a fix was straightforward.

Given the seriousness of this error, even with your reassurance of a fix, I am not prepared to accept the paper at this time.

Furthermore, I (quickly) reviewed the current submission and am concerned about another point that I specifically raised in my previous decision that does not seem to have been fully addressed. Specifically in decoding the CSP solution (Algorithm 6) it is explicitly assumed that one action is selected and that that selection is correct. This point is raised by both Reviewer 1 and 3 but I see the issue as potentially more serious. Analogously to the negative example indicated by Reviewer 1, I am concerned with a situation where there are two or more actions that can achieve the same transition for a state variable with one action requiring a subset of the conditions of the other. For example:

alpha: preconds: v1 = 1; effects: v1 = 2
alpha': preconds: v1 = 1, v2 = 0; effects: v1 = 2, v2 = 1

Imagine that the valid plan requires alpha' and the CSP produces a solution. I do not see what prevents Algorithm 6 from selecting alpha given the "don't care" wrt v2. Further, if there were a number of actions that had various effects on {v3, v4, ...} within a hierarchy of subset relations, there would seem to be the need for a potentially expensive search to select the correct action.

>>>We answered this issue in the paper and corrected the decoding algorithm. In the new algorithm from each DTG we select all the actions that their preconditions and effects are satisfied. Our encoding algorithm guaranties that all the actions selected for one time step can happen at the same time and are not conflicting with each other. Therefore no search is needed to select the actions. However as we mentioned in the paper, the generated plan may include redundant actions that can be detected later and deleted from the plan. For this simple example, there will be two rows for alpha and alpha' in the table and since theses actions are conflicting (based on condition 6 of definition 13), the parallelism variable will have different values in these rows. So based on the value of parallelism variable, selecting the correct action is straightforward.

Perhaps, I am missing a detail of your paper that addresses this concern. If not, however, I believe it would be another serious flaw in your submission.

Another, related point arises in your explanation of the flaw indicated by Reviewer 1 and its fix. If you simply remove the negative table constraint as you suggest, what prevents the action selection procedure from selecting op1 and op2 instead of op3 when producing a plan and therefore generating an invalid plan?

>>>We totally deleted negative parallelism constraints and now conflicting actions are prevented by parallelism variables. Parallelism variables will have different values in the rows of conflicting actions.

Finally, in your correspondence, you pointed to the absence of the Reviewer #1's flaw in the problem instances in your experiment and therefore the continued correctness of your experiments. Logically, this is incorrect as a set of problem instances that do not include a particular characteristic do not provide any evidence as to the validity of the approach wrt that characteristic. If you want to use experiments to bolster your claims of correctness, you need to include the conditions that lead to the flaw. [Even then, while experiments can provide some reassurance that an approach may be correct, they cannot logically prove correctness].

>>>We ran new TCPP planner again for all the benchmark problems and the new planner can now solve more problems. Figures and tables are updated.

As you know, JAIR has a limit of two submissions. However, given the otherwise positive opinions of the reviewers, your claim that the flaw pointed out by Reviewer 1 was fixable, and the possibility that I am mistaken in the flaws I have indicated above, I will waive this restriction and thus would be willing to consider a further (and final) resubmission of the paper. For this subsequent submission to be acceptable, I will need to be confident in the correctness of your approach. Please endeavour to more formally deal with this aspect of your submission.

>>>We greatly appreciate for allowing us a further resubmission. By fixing the flaw pointed out by reviewer 1, the performance of our planner is improved.

Attached you will find the reviewers' comments and suggestions. While I have not addressed them here in detail, they contain a number of valuable pieces of advice that will improve your paper. Please address them conscientiously.

Thank you for submitting your paper to JAIR.

Regards,

Chris Beck
Associate Editor

Reviewer 1

_____
Overall evaluation: Accept with minor revisions

_____
Comments:

The article is a resubmission of an original version submitted earlier this year. As I reviewed the original version (Reviewer 1), the present review is mostly incremental.

In general, I think the authors satisfactorily addressed my questions/issues. The new version is more formal on different aspects of the approach, in particular the encoding of the problem as CSP and the decoding of the CSP solution to build a plan.

There are two things that still puzzle me but I'm quite convinced that they are minor and can be fixed quite easily.

1) The first thing is that you seem to consider it as evident that two different actions will lead to two different rows in the CSP tables. I have the feeling that this is true because the table considers all preconditions and effects of the action thus if two actions result in the same row it probably implies that they have the same preconditions and effects (thus they do not need to be differentiated) but even so it would be better to say it explicitly. It is an important point because the decoding implicitly assumes 1 row = 1 action.

>>>In our modified model, if there are two actions with the same preconditions and effects, there will be two rows in the table. If these actions change the value of a variable from real value k to a real value k', they are conflicting and there will be a parallelism variable in the table that takes two different values in the rows for these actions. This parallelism variable prevents these actions from

happening at the same time and also helps in decoding phase to select the action for the final plan. But if these action change the values of variables from don't-care to a real value k', they are not conflicting and both will be selected by decoding algorithm while one is redundant. We have mentioned in the paper a post-processor can be used to find the redundant actions and remove them. We however did not do that for the time-being since it has no effect on the makespan.

2) The second point is in a certain sense related to the first one as it is also about the possible loss of information while ruling out the actions from the CSP. It is about the parallelism constraints: in the example of Figure 13, because actions \alpha and \alpha' are clearly conflicting the row corresponding to the action pair (\alpha,\alpha') is added in the negative table and, as stated "There is only one row in this table listing the values for the variables. During search this combination of variables and values will be avoided". Now suppose, in the domain, we have an additional action \alpha" that permits this transition, for instance an action \alpha":
preconds: v1=1, v2=5, v3=2, v5=6
effects: v3=3, v4=4, v5=7
I see a risk that the application of action \alpha" will always be forbidden (by the row in the negative table) for the wrong reason that there is somewhere else in the domain a conflicting pair of actions (\alpha and \alpha'). So is the formulation of parallelism constraints not over-constraining the problem?

>>>We don't have negative parallelism constraints in our new planner and instead we prevent conflicting actions by using parallelism variables. As explained in the paper, for the actions \alpha and \alpha' and the new action \alpha", we will have 3 different rows in the table with different values for parallelism variable. Now preventing \alpha and \alpha' from happening at the same time, does not prevent \alpha" from happening. Thank you very much for pointing out this flaw.

Some more minor comments/suggestions:

- The first time DTG is mentioned in the introduction, it would be good to explain the acronym (it is explained in the abstract but doing it in the introduction would not hurt)

>>>Done

- In the end of section 2.4 on Constraint-Based planners, the comparison with CPT is not really fair because CPT is a temporal planner that can handle more general problems. By the way, this raises the question if (and how) the proposed encoding could be extended to temporal planning or at least a limited form of temporal planning as the one handled in CPT ?

>>>This comparison is based on what is reported in the paper [Gregory et al., 2010] and particularly in the context of classical planning.

- For self-containedness reasons, it could be good to recap / say a bit more how mutex groups are computed in helmert 2006) as mutex groups seem to help in the approach and I suppose one can dedicate more or less effort to compute more complete mutex groups

>>>Addressed partly and a further reference “Concise finite-domain representations for PDDL planning tasks” by Malte Helmert is added.

Typos / formatting:

- Section 4: in items 1 and 2 of Definition 12, symbols k and k' are not defined, I suppose they denote p_\alpha(v) and e_\alpha(v) respectively ?

>>>corrected

- Algorithm 3, line 9: the comment should better be shifted to the left for readability

>>>corrected

- p25: Fiqure 12

>>>corrected

- p27: Not sure with the procedure is (consistently) called "PrallelismConstraints", is it a typo?

>>>This section is rewritten

- p31: nevertehless

>>>corrected

- Figure 24: The y-axis read "Time in Seconds" whereas it should be "Number of Instances"

>>>corrected

Reviewer 2

The authors have satisfactorily addressed my comments. My main original concern was an aspect of the experimental design. However, the experiments are thorough and provide sufficient evidence for the proposed approach. Just as a comment, not a concern: I did understand that you were not doing learning in your approach. The point that I was getting at in my original comment is that it is common to tune constraint solvers to the benchmarks by exhaustively finding the best settings for constraint propagation, heuristics, and the other parameter settings. However, it is also well-known that "tuning to the benchmarks" is not the best approach and statistical learning/machine learning has long known a better way to set parameters: cross-validation. In your case, it could be done by tuning your parameters on k-1 benchmarks and evaluating on the k'th benchmark. If it turns out that the parameters chosen (level of constraint propagation, variable ordering heuristic, value ordering heuristic, and all of the other parameter settings) were robust---i.e., roughly the same settings are chosen no matter which k-1 benchmarks you tune on---then you can also argue that you have found very good parameter settings overall. In my experience, this often doesn't happen. In other words, by tuning on k-1 and then evaluating on a k'th benchmark the parameter settings chosen on the k-1 can sometimes perform very poorly on the k'th benchmark. What does this tell you then? It suggests that

the same thing may happen when your planner is applied to a new benchmark that you didn't consider in your paper. That is the point behind the cross-validation methodology for setting/tuning parameters: to discover this and not report overly optimistic results.

Reviewer 3

Paper: Encoding Domain Transitions for Constraint-Based Planning

Recommendation: Accept with minor revisions

This is a revised version of already submitted paper so this review is basically an extension of my original review. The authors did a very good job in addressing the comments from previous reviews and they improved the paper considerably, mainly in the section giving technical details about the models and theoretical justifications of their soundness. There are also extensive experiments though deeper understanding of relation between problems and models is left for future work (which is fine with me). I have only a few minor comments/suggestions that should be easy to include in the final version. In general, I suggest acceptance of the paper.

Further comments:

Page 4: In Definitions 1 and 2 you used the notion of a goal state. I think it is better to talk just about a goal, because there could be more goal states. What you describe is not really a state, but a property of any goal state.

>>> Addressed!

Page 5, Figure 2: The variable t-occ in SAS+ model seems superficial as the same information is encoded in d-loc, but perhaps you use the variables found automatically. Also, the rigid preconditions in drive-truck (road) and driver-walk (path) seem missing. I understand that they are satisfied in the grounded representation, but the example model uses operators and these conditions are somehow missing there.

>>>The SAS+ representation of this example is exactly the same as what the translator used in (Helmert, 2006) generates.

Page 16, Algorithm 2: As the possible output of selectParallelActions, you use symbol that looks like phi, but it is probably an empty set. Perhaps do not use italics for this symbol to make it clear.

Addressed.

Page 27, Algorithm 5: For some reason, you call the procedure PrallelismConstraints instead of ParallelismConstraints. It is used consistently in the text, so formally it is correct, but it looks strange.

>>>This section is rewritten

Page 29: You mentioned that when decoding the solution of a CSP to a plan, there is exactly one action selected for each DTG (only one row from each table is selected). It is not fully clear to me, if there is really just one action satisfying all the constraints in a solution to the CSP or there could be more such actions. Obviously, there could be more arcs between two values and perhaps more actions annotating these arcs can be used to do the transition which are consistent with actions selected in other DTGs. Can you comment on this?

>>>We corrected the issue in the new version of the paper. As you mentioned more than one action may be selected from each DTG but these actions need to be able to happen at the same time meaning that they should not conflict with each other. We have addressed this issue in the paper.

Page 31, Section 6, r. 2-: Nevertehless -> Neverthless

>>>corrected

Page 32, Section 6.3: Is there any reason to use table constraints for encoding the initial state and goal condition rather than simply using singleton domains for selected variables?

>>>In MINION the domain of variables is represented by a range {-min,...,max} of consecutive values while the set of values for the constraints for initial and goal state are list of values. We could use an index for actions but this would need using the same index in other tables as well. To simplify the encoding we used table constraints.