Intelligent Narrative Technologies: Papers from the 2012 AIIDE Workshop
AAAI Technical Report WS-12-14

The Intentional Fast-Forward Narrative Planner

Stephen G. Ware
Liquid Narrative Group
Department of Computer Science
North Carolina State University
Raleigh, NC, 27695

Abstract

The Intentional Fast-Forward (IFF) planner is an attempt to apply fast forward-chaining state-space search methods to intentional planning—planning such that every action is directed toward some character's goal. The IFF heuristic is based on Hoffmann's original Fast Forward heuristic (2001), which solves a simplified version of the problem and uses that solution as a guide for the real problem. IFF incorporates constraints imposed by intentional planning to narrow down the set of steps which can be taken next, and it identifies fruitless branches of the search space early.

Introduction

One of the central challenges of narrative generation is to produce action sequences that are both understandable and believable. AI planning is frequently leveraged to produce understandable action sequences—ones which have a clear logical and causal progression. Riedl and Young (2010) proposed an extension to refinement planning which focused also on believable characters by ensuring that every action in the story is taken in service of some character's goals. Their planner, IPOCL, is an Intentional (I) planner which extends a Partial Order Causal Link (POCL) algorithm. POCL plans are appealing for narrative projects. A partial ordering of steps expresses minimal constraints on a story's discourse, and causal links explicitly model how the preconditions of a step are made true by earlier steps. POCL plans have been used to model narrative conflict (Ware and Young 2011), estimate the salience of past events in a discourse (Cardona-Rivera et al. 2012), and predict the inferences of narrative readers (Niehaus and Young 2009).

Despite the popularity of plan-space planners like POCL in the narrative community, the last two decades of classical planning research have have been dominated by state-space search. Since the first International Planning Competition in 1998, the fastest planners in the classical deterministic track have always used forward-chaining state-space search, with one exception in 2002 (Coles et al. 2011). Winners have traditionally been some variant of the Heuristic Search Planner (Bonet and Geffner 2001) with an improved heuristic.

IPOCL represents an important contribution to story modeling and generation because it defines the intentional planning problem and the set of solutions which represent believable stories. This paper describes the Intentional Fast-Forward system (IFF), a state-space planner which attempts to solve intentional planning problems faster by leveraging advances in classical planning. IFF is based on the Fast-Forward (FF) planner by Hoffmann (2001), which solves a relaxed version of the planning task in polynomial time and uses that relaxed solution to estimate the difficulty of the actual task. IFF utilizes constraints imposed by intentional planning to speed up the FF heuristic.

The usefulness of the IFF planner will depend on the context of the project. POCL plans provide many representational benefits and are more similar to *fabulae* as described by narratologists. However, the least-commitment nature of POCL planners means that exact details are often unknown until the search has finished. For example, a partial POCL plan may only state that "*someone* has money," and the variable *someone* may not get bound until later. This and the partially ordered nature of its steps can make it difficult to evaluate the narrative properties of a partial plan during search.

At any time during IFF planning, the exact current state is known, along with the exact past and an estimate of which steps will occur in the future. In practice, IFF planning should also be significantly faster than IPOCL planning. In some cases, it may be possible to live in the best of both worlds. The Re-POP planner (Nguyen and Kambhampati 2001) integrated state-space heuristics into POCL planning and enjoyed a significant speed increase. The architects of a narrative system should weigh the benefits and drawbacks of IPOCL and IFF based on their needs.

Related Work

To explain the IFF heuristic, I first introduce intentional planning in terms which can be applied to both IPOCL and IFF. I also review the original Fast-Forward heuristic.

Intentional Planning

A classical planning problem describes the initial state of a world and a set of goals that need to be achieved by applying a sequence of steps. A step has preconditions which must be true before it can be applied and effects which become true afterward.

Intentional planning extends this by annotating each step with a set of zero or more characters who must consent to the execution of a step. A plan is a valid solution to an intentional planning problem iff it achieves the goals of the problem and, for each step in the plan, we can identify some personal goal that each consenting character was working toward by taking that step.

Formally, a **step** is a 3-tuple $\langle P, E, C \rangle$, where P is a set of **preconditions**, ground predicate literals which must be true before the step can be applied, E a set of **effects**, ground predicate literals which become true after the step is applied, and C a set of constants representing **characters** which must consent to the execution of the step.

Given two steps s and t, we say that s is an **intentional parent** of t for character c if and only if:

- s is ordered before t.
- $c \in C$ for s and $c \in C$ for t.
- There exists some effect $p \in E$ for s, and
- There exists a precondition $p \in P$ for t.
- No step that comes after s but before t has effect $\neg p$.

In other words, step s may have been taken by character c to enable future step t. When s is the intentional parent of t, t is the **intentional child** of s. A step's **intentional ancestors** and **intentional descendents** are the transitive closures of the parent and child relationships.

A **motivation** is a ground modal predicate literal of the form `(intends c g)`, where c is some character and g is a **character goal**, a ground predicate literal that character c wishes to be true. Motivations can also appear in the effects of a step. A step s is said to be **motivated** if and only if the following are true for all characters $c \in C$:

- There exists some goal g such that `(intends c g)` is true before step s.
- There exists a step t with effect $g \in E$ such that $t = s$ or t is the intentional descendent of s.
- `(intends c g)` is true until t is executed.

Informally, a step is motivated when it achieves a character's goal or contributes to achieving a character's goal. When a step is not motivated, it is an **orphan**. A step with no consenting characters is always motivated.

A planning **problem** is a 2-tuple $\langle I, G \rangle$, where I is the **initial state,** a set of ground predicate literals and motivations which fully describe the world at the beginning of the problem, and **G** is a set of **author goals**, ground predicate literals which must be true at the end. A planning **domain** defines a set of steps which can be taken during the course of a problem. A **plan** Π is a set of steps which can be applied in order from the initial state such that, in the final state, all goals in **G** are true and every step is motivated.

IPOCL Performance

For a full description of the IPOCL algorithm, readers are referred to Riedl and Young (2010). In short, each node in the IPOCL search space is a partial plan annotated with flaws. Flaws are iteratively repaired until a flawless plan is produced. Flaws are repaired by adding new steps, adding new causal links, and grouping steps into frames of commitment to ensure that each is motivated.

Two major factors affect IPOCL's performance. Firstly, each time a new step is added to a partial plan, IPOCL branches for every way that the step could be the last intentional child in a sequence of steps. Where POCL branches only once per new step, IPOCL branch as many as $2^{|C|}$ times. Secondly, IPOCL cannot directly make an orphan step motivated. All steps begin as orphans and become motivated as a side-effect of other repairs. This means a partial IPOCL plan can have orphans even when it has no flaws; such nodes are dead-ends in the search space. If orphans which will never become motivated are detected early, the search can be made more efficient. IFF attempts to do this.

Haslum showed that intentional planning problems can be solved faster by compiling them into classical planning problems and leveraging state-of-the-art planners (Haslum 2012). His solution involves generating one step for each way the step could be used to achieve a goal, and introducing new predicates to ensure that only intentional plans are produced. IFF leverages some of the same constraints described by Haslum, but it does not increase the size of the problem and works more directly with its intentional semantics.

The Original Fast-Forward Heuristic

Whereas POCL searches the space of plans, FF searches the space of states. It begins at the initial state and applies steps whose preconditions are met until it reaches a state in which all goals are met.

For each state in its search space, the original FF planner (Hoffmann 2001) works by reducing the remaining problem down to an easier (but incomplete) problem, and then using the solution to this easier problem as an approximation of the solution to the true problem. The relaxed problem assumes that literals are only added to the current state, never deleted. Each time FF's search process generates a new state, it uses this heuristic to estimate how far that state is from the goal. Specifically, FF builds a plan graph (Blum and Furst 1997) which starts at the current state, and then it extracts an incomplete solution plan from that graph.

An **FF heuristic plan graph** is a directed, layered graph composed of step nodes and fact nodes. A **step node** exists for step s at layer i iff all the fact nodes for the preconditions of s exist at layer $i - 1$. Layer 0 has no step nodes. A **fact node** exists for a ground literal p at layer 0 iff p is part of the initial state. A fact node exists at layer $i > 0$ if it exists at layer $i-1$ or if p is the effect of a step whose node appears in layer i. Directed edges go from step nodes to the fact nodes that represent their effects and from fact nodes to the step nodes for which they are preconditions (see Figure 1).

Fact and step nodes increase monotonically at successive layers until a fact node for every goal is present. Then a relaxed solution is extracted using Algorithm 1[1]. Extending an FF graph and extracting a relaxed solution takes only

[1]Relaxed solution extraction starts at the goal and works backwards, and is thus backward-chaining. However, the overall search process of IFF begins at the initial state and works toward the goal, so the planner as a whole is forward-chaining.

polynomial time, whereas solving the true planning problem is PSPACE-complete (Hoffmann 2001). In addition, the FF heuristic suggests promising successors to the current state. A **helpful step** is one whose node appears at layer 1 and which achieves one of the goals assigned to layer 1 during relaxed solution extraction (Hoffmann 2001). Applying a helpful step in the current state is more likely to lead to the goal than other steps.

Algorithm 1 Relaxed Solution Extraction

Input: problem $\langle I, G\rangle$ and plan graph with layers $L_0 ... L_m$
Output: a set of steps Π

1: Let $\Pi = \emptyset$.
2: $\forall g \in G$ assign g as a goal to layer L_m.
3: **for** each layer L_i, starting at L_m going back to L_1. **do**
4: **for** each goal g assigned to L_i **do**
5: **if** a node for g exists at L_{i-1} **then**
6: Assign g as a goal to L_{i-1}.
7: **else**
8: Choose a step s with effect g.
9: Add s to Π.
10: Assign preconditions of s as goals to L_{i-1}.
11: **end if**
12: **end for**
13: **end for**

The Intentional Fast-Forward Heuristic

The IFF heuristic extends FF in two simple ways: a step node only appears at a layer in the graph once it can be motivated, and the graph is used to detect unmotivatable steps.

Before going into detail, consider a simple problem and domain which illustrates how IFF works. In a village lives a rich baker and a poor thief. The baker has money and cake, and he can bake new cakes to sell at his shop whenever he pleases. The thief has no money and wants cake. He cannot steal a cake directly, but he can pickpocket the baker and use this illicitly-obtained money to buy a cake from the store. The baker can also give the thief a cake for free. The FF plan graph for this problem is shown in Figure 1, and the IFF plan graph for this problem is shown in Figure 2.

Potentially Motivated Steps

Before the planner begins its search, we can analyze the problem to discover when and how steps can be motivated. Conceptually, a motivation `(intends c g)` can motivate any step s which requires the consent of character c and has the effect g, or any step which can be the intentional ancestor of such a step. Technically, this definition permits certain undesirable corner cases and needs to narrowed slightly, but it will serve to introduce an example.

Consider the BUY step in the example domain. It causes the baker to have money (BM) and the thief to have cake (TC). These effects can directly satisfy the baker's intention B:BM and the thief's intention T:TC, so those motivations can motivate BUY. None of the effects of STEAL satisfy any motivations, however STEAL can be the intentional ancestor of BUY because STEAL causes the thief to have money (TM), BUY requires TM, and both steps require the consent of the thief. Therefore, T:TC can also motivate STEAL (B:BM cannot; the baker is not a consenting character for STEAL).

According to this definition of a potential motivation, B:BC could motivate GIVE because GIVE has effect ¬BC, which is a precondition for BAKE, which has effect BC. In words, this plan would translate to "the baker can obtain cake by giving his cake away so that he can bake a new one." To avoid nonsensical situations like this, we use the following formal definition of a **potential motivation**: A motivation `(intends c g)` can motivate a step s which requires the consent of character c if and only if:

- There exists a path of alternating steps s_i and facts f_i like $\langle s_0, f_0, s_1, f_1, ...g\rangle$ such that $\forall i$ s_i has effect f_i and s_{i+1} has precondition f_i.
- c is a consenting character for every step in the path.
- No step in the path has g as a precondition.
- The path never contains both a fact and the negation of that fact.

When constructing an IFF graph, a step appears at a layer iff its preconditions appear at the previous layer *and* at least one of its motivations appears at the previous layer for every consenting character. The IFF plan graph in Figure 2 demonstrate the difference this makes relative to the graph in Figure 1. GIVE does not appear in the IFF graph, even though its preconditions are met, because there is no possible explanation for why the baker would give away his cake. The relaxed solution to the FF graph is $\Pi = \langle$GIVE$\rangle$, whereas the relaxed solution to the IFF graph is $\Pi = \langle$STEAL, BUY$\rangle$. The relaxed IFF solution is more similar to the problem's actual solution both in length and content.

Limiting the planner's next moves based on potential motivations can reduce the branching factor of the search. This insight can also be applied to any form of forward-chaining state-space intentional planning, not just IFF.

Relaxed Solution Extraction in IFF Building an IFF graph using potential motivations adds one small wrinkle to the solution extraction process given in Algorithm 1. When a step is added to the relaxed plan, its preconditions are assigned as goals to the previous layer (line 10). We must also assign any one of the step's motivations that appears at the previous layer as a goal to the previous layer. This maintains the property that the relaxed solution will contain at least one step from every layer of the graph and make the relaxed solutions more accurate.

Detecting Unmotivatable Steps

Not every step taken in an intentional plan satisfies a character goal directly; some are taken to enable future steps which do satisfy a goal. This makes intentional planning difficult, because it is hard to know which currently unmotivated steps will become motivated later by their intentional descendents. Including only potentially-motivated steps in an IFF graph can reduce the number of orphans in a plan but will not eliminate them entirely. We can further improve performance by detecting unmotivatable steps early and pruning these branches from the search space.

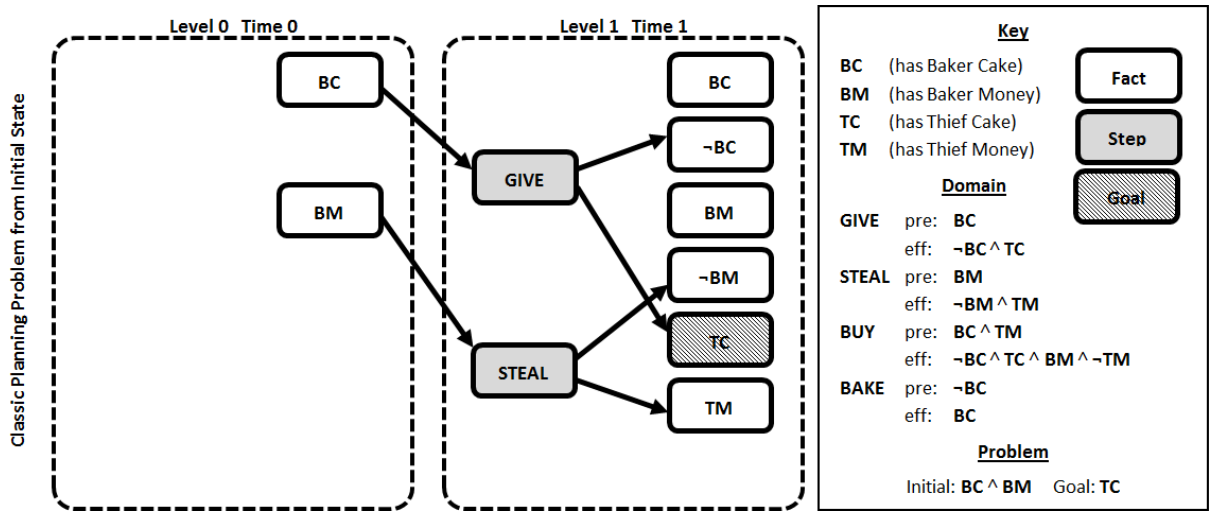

Figure 1: A Fast-Forward heuristic plan graph for the classical planning example problem.

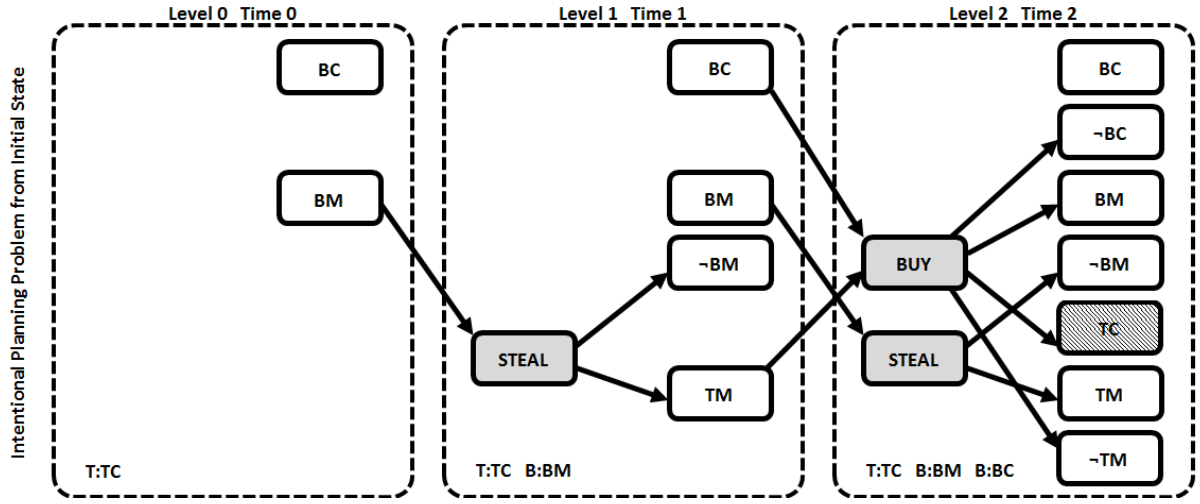

Figure 2: An Intentional Fast-Forward heuristic plan graph for the example problem, starting from the initial state.

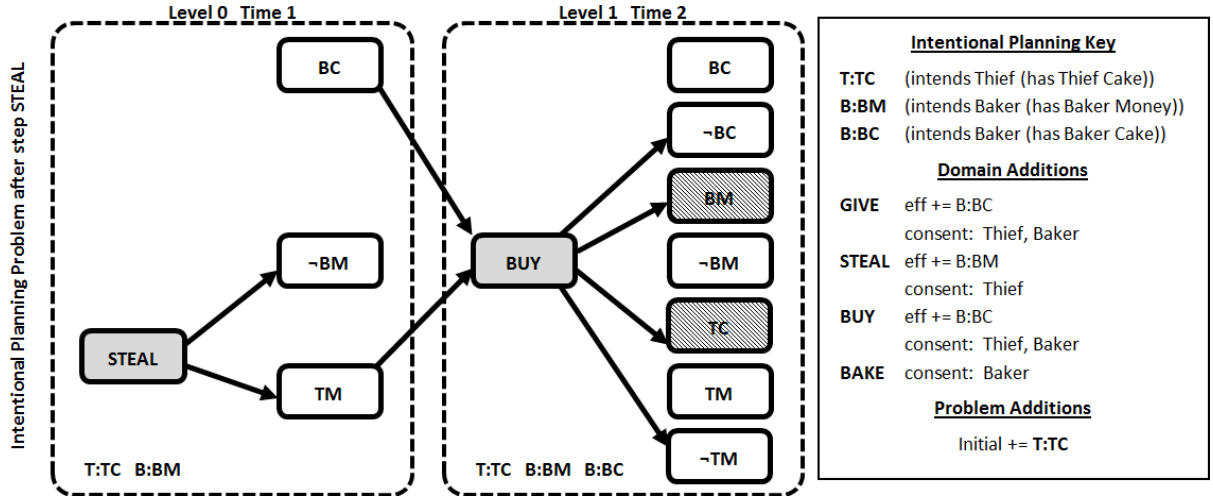

Figure 3: An Intentional Fast-Forward heuristic plan graph for the example problem, starting from time 1, after the STEAL step.

As an IFF plan grows, the planner keeps track of which steps are intentional ancestors of which other steps using a history graph. A **history graph** has one step node for each step that has occurred previously in the plan. An edge $n_s \xrightarrow{c,p} n_t$ leads from a step node n_s (for step s) to a step node n_t (for step t) via character c and fact p if s is the intentional ancestor of t for character c because of p. In addition, an edge $n_s \rightarrow n_f$ leads from a step node n_s in a history graph to a fact node n_f (for fact f) of an IFF heuristic graph if step s has effect f, $\neg f$ is never true before the current state, and n_f appears at layer 0 of the IFF graph. The history graph is linked to the heuristic plan graph. While this may sound complicated, a simple example can be seen in Figure 3. After STEAL is taken as the first step of a plan, it shows up in the history graph henceforth and gets linked to the heuristic graph via all of its effects which have remained true: namely TM and $\neg$BM.

Maintaining a history graph requires more overhead than simply tracking which facts are true and false, as FF does, but the history graph allows us to quickly detect when an orphan has become motivated. It can also be used to detect when an orphan can never become motivated.

For every character c who must consent to step s of step node n_s in the history graph, there must exist a path from n_s to any fact node in the heuristic graph for one of c's goals that visits only step nodes whose step requires the consent of c. This path can traverse nodes and edges in both the history graph and the heuristic graph. In other words, the orphan step must be able to have an intentional descendent which achieves one of the character's goals. There must be some way to explain (in the future) why c took that step. In Figure 3, there is a path from STEAL to TC. Even though STEAL is currently unmotivated, it might still become motivated by T:TC. If, for any consenting character, no such path exists, the plan's distance to the goal is ∞ because it contains an unmotivatable step.

Longer Plan Graphs? The astute reader may foresee a problem with the above method. A heuristic plan graph is only extended until all the goals appear, even if the graph has not yet leveled off. Consider a situation where an orphan step exists, the graph has extended such that all goals appear, no path can be found from the orphan to a character goal, but additional extension of the plan graph would create such a path. We dub this a "graph too short" condition.

To avoid these cases, an IFF graph is extended until all problem goals appear *and* at least one path exists for each orphan. If the graph levels off before then, no solution exists. This means IFF graphs may need to be extended farther than FF graphs, but when we consider this problem at a higher level of abstraction, we see that it is an inherent property of intentional planning.

The FF heuristic has these two properties: if no relaxed solution exists then no solution exists, and if a relaxed solution exists in an FF graph with n layers then any real solution will contain at least n steps (Hoffmann 2001). These properties still hold for IFF. In a graph too short case, even if all the problem goals appear and a relaxed solution exists, there cannot exist a real solution. In any such plan, the orphan step would not be motivated, and thus the plan would not be a solution. So even though IFF may sometimes require longer graphs than FF, it never extends farther than it must.

IFF Summary

Now we can summarize the iterative process of IFF:

1. Choose the next partial plan[2] to evaluate. If it is a solution, return it. Otherwise...
2. Create a heuristic plan graph from the current state and extend until the problem's goals and all current character goals are present (or until the graph levels off).
3. Check for unmotivatable orphans. If any are found, prune this branch of the search.
4. Extract a relaxed solution and use it to estimate the remaining distance to the goal.

The FF heuristic is effective because it uses a problem solvable in polynomial time (based on the size of the problem) to estimate a very difficult P-SPACE complete planning problem (Helmert 2006). The IFF heuristic can be calculated in polynomial time based on the size of the problem *and* the size of the current plan (due to the unmotivatable step check). In practice, plan length is not a limiting factor.

Implementation

I have implemented the IFF planner in Java. The source code is available as an Eclipse project at my website:

`http://stephengware.com/projects/iff`

This implementation assumes a slightly different intentional planning formalism which better suites my future work and the future versions of this software. I distinguish between a desire that a character wishes to remain true always (an intention) and a desire which a character wishes to make true only once (a goal). I also impose the restriction that when a motivation appears, a character must act on it.

The software has a framework in place to support more expressive action descriptions in future versions, including disjunctive preconditions, universally and existentially quantified preconditions, conditional effects, universally quantified effects, conjunctive disjunctive and quantified intentions, and domain axioms. I hope the narrative and multi-agent planning communities will find this software helpful in their projects.

IFF Performance

In order to demonstrate the usefulness of IFF, I implemented a version of the Aladdin narrative planning problem described by Riedl and Young (2010). It is not an exact replica of theirs because IPOCL allows characters to delegate intentions to other characters. This features is not yet implemented in IFF.

[2]Because IFF maintains a history graph, a node in the IFF search space is more than just a state; it is a partial plan, though it lacks much of the representational richness of a POCL plan.

Specifically, I modified the original domain by (1) merging the COMMAND and LOVE-SPELL actions into a single COMMAND-LOVE-SPELL action and (2) compiling the ORDER action to delegate any end goal of the King, which resulted in 12 possible uses of ORDER. Comparing IFF's performance on my version of the problem to IPOCL's performance on the original will not be an entirely fair comparison.

Using a domain-specific heuristic that was hand-authored for the Aladdin problem, IPOCL finds a 13 step solution in 12.3 hours on an Intel Core2 Duo 3GHz system with 3GB of RAM and 100GB of virtual memory (though 11.6 hours were spent in garbage collection). It generates 1,857,373 nodes and visits 673,079. Using a domain-independent heuristic, IPOCL runs out of virtual memory before finding a solution (Riedl and Young 2010). I attempted to replicate this result, but was unable to obtain a professional license for Allegro Common Lisp. The free version of Allegro does not allow sufficient heap space to replicate this result or test my version of the problem.

I implemented both the original FF heuristic and the IFF heuristic for comparison using my version of the Aladdin problem. At each iteration, FF extends its heuristic graph to the problem goals and all current character goals, but does not attempt to handle motivations or orphans intelligently. Both the FF and IFF implementations use the same data structures and methods whenever possible. Complete A* search is used at the top level. Both tests occurred on a Mac Pro with 8 2.39GHz Intel processors and were given 2GB of RAM.

FF runs out of memory without finding a solution after 70 seconds, generating 742,682 nodes and visiting 59,331. IFF, on the other hand, finds a 10 step solution in 0.07 seconds, generating only 147 nodes and visiting only 26. These results demonstrate that the extensions made to IFF can dramatically improve its performance on intentional problems. Though the comparison to IPOCL is tenuous, the dramatic difference in their run times suggests that IFF is faster.

During its search, IFF finds and prunes 7 plans with unmotivatable orphans. Because it finishes so quickly, the benefits of that pruning never bare fruit, but this demonstrates that on a larger problem the unmotivatable step check could be helpful.

Future Work

This work is a precursor to a forward-chaining state-space implementation of the conflict planning algorithm described by Ware and Young (2011), which was originally based on IPOCL. As it stands, IFF only finds a solution when every character's goals can be satisfied. This is rarely the case in narrative domains, so the inclusion of conflict should significantly enhance the expressive capabilities of this planner.

Adapting the Aladdin problem raised some interesting questions about how the intentional framework can be extended. Obviously, it could benefit from the delegation feature of IPOCL. The definition of motivation may also need to be relaxed. As it stands, a step is only the intentional ancestor of another when there exists a path that travels only along steps intended by the same character. However, it seems reasonable that one character would be willing to take an action that enables another character's action; i.e. Aladdin might bring the king the lamp so the king can SUMMON the Genie (even though Aladdin is not a consenting character for SUMMON). The original IPOCL can reason like this indirectly.

Conclusion

Partial-order causal-link planning can be well suited to narrative generation because of the representational benefits of POCL plans and the least-commitment nature of the algorithms. However, POCL planning is often to slow for practical use, and evaluating the properties of a partially-instantiated plan can be difficult. Forward-chaining state-space search is usually much faster and provides access to the complete, fully-instantiated past, so it may be preferred in some cases.

This paper proposed the Intentional Fast Forward heuristic for guiding such a state-space planner. It extends the original Fast Forward heuristic by including only steps which can be motivated and by discovering orphan steps which can never be motivated. By incorporating the constraints of an intentional planning problem into the heuristic, IFF can facilitate faster, more intelligent narrative planning.

Acknowledgments

This research was supported by the United States National Science Foundation, award IIS-0915598.

References

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial intelligence* 90(1-2):281–300.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1):5–33.

Cardona-Rivera, R. E.; Cassell, B. A.; Ware, S. G.; and Young, R. M. 2012. Indexter: A computational model of the event-indexing situation model for characterizing narratives. In *Computational Models of Narrative Workshop*.

Coles, A. J.; Coles, A. I.; Garca Olaya, A.; Jimnez, S.; Linares Lopez, C.; Sanner, S.; and Yoon, S. 2011. A survey of the seventh international planning competition. *AI Magazine*.

Haslum, P. 2012. Narrative planning: Compilations to classical planning. *JAIR* 44:383–395.

Helmert, M. 2006. New complexity results for classical planning benchmarks. In *ICAPS*, 52–61.

Hoffmann, J. 2001. Ff: The fast-forward planning system. *AI magazine* 22(3):57.

Nguyen, X. L., and Kambhampati, S. 2001. Reviving partial order planning. In *IJCAI*, volume 17, 459–466.

Niehaus, J., and Young, R. M. 2009. A computational model of inferencing in narrative. In *AAAI Spring Symposium*.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: balancing plot and character. *JAIR* 39:217–268.

Ware, S. G., and Young, R. M. 2011. CPOCL: A Narrative Planner Supporting Conflict. In *AIIDE*.