v.9, n.3, p.345-361, dez. 2002

PLANEJAMENTO DE AÇÕES PARA AUTOMAÇÃO INTELIGENTE DA MANUFATURA

Flavio Tonidandel
Escola Politécnica,
Universidade de São Paulo,
Av. Luciano Gualberto, 158, trav. 3,
CEP 05508-900, São Paulo, SP, Brasil,
e-mail: flaviotoni@terra.com.br

Márcio Rillo
Centro Universitário da FEI,
Av. Humberto de A. Castelo Branco, 3972,
CEP 09850-901, São Bernardo do Campo, SP, Brasil,
e-mail: rillo@lsi.usp.br

Resumo

Este artigo investiga o uso do sistema FAR-OFF na área da Automação da Manufatura. O sistema FAR-OFF possui característica similar aos sistemas de planejamento baseado em busca heurística, os quais têm apresentado excelentes resultados nos últimos anos na área de planejamento. Entretanto, em vez de ser um sistema generativo, o FAR-OFF é um sistema de planejamento baseado em casos que garante estabilidade para solucionar problemas em tempo aceitável. Os resultados apresentados pelo sistema no domínio de logística mostram que este é um sistema promissor para automação inteligente.

Palavras-chave: ***planejamento inteligente de ações, raciocínio baseado em casos, planejamento baseado em casos.***

1. Introdução

A área de Inteligência Artificial (IA), sobretudo a área de planejamento de ações, avançou significativamente nos últimos anos. Este avanço possibilitou o aparecimento de sistemas de planejamento inteligente de ações que conseguem encontrar soluções para diversos problemas de forma rápida e automática.

Um sistema de planejamento inteligente de ações se caracteriza por tentar encontrar uma seqüência de ações que transforme, automaticamente, um estado inicial de planejamento em outro estado final desejado.

A maioria dos sistemas de planejamento é independente do domínio de aplicação, o que implica que podem, a princípio, ser aplicados em qualquer domínio de aplicação. Há diversos domínios de planejamento, que podem variar desde o mundo de blocos (Hoffmann, 2001) até o controle inteligente da manufatura, passando por domínios como logística e escalonamento (Bacchus, 2000). Entretanto, poucos são os sistemas de planejamento aplicados em sistemas de manufatura e controle de processos industriais. Uma das grandes razões para tal fato é que os sistemas de planejamento independentes do domínio, embora tenham se desenvolvido muito nos últimos anos, ainda apresentam falhas que os impossibilitam de encontrar soluções para todos os problemas de um certo domínio de aplicação (Hoffmann, 2001).

Neste artigo é analisada a aplicação de técnicas de raciocínio baseadas em casos, que permitem prover maior eficiência aos atuais sistemas de planejamento, diminuindo as falhas, e, ainda assim, mantê-los independentes do domínio de aplicação.

Neste artigo será apresentado o sistema de planejamento baseado em casos, chamado FAR-OFF (*Fast and Accurate Retrieval on Fast Forward*) (Tonidandel & Rillo, 2002), e sua aplicação no domínio da logística.

Na Seção 2 serão abordados os sistemas de planejamento baseados em casos e generativos. A Seção 3 trata da formalização em lógica de transações dos componentes do sistema FAR-OFF. As etapas do sistema serão detalhadas na Seção 4. Testes empíricos da aplicação do sistema FAR-OFF em logística são apresentados na Seção 5. A Seção 6 apresenta as considerações finais.

2. Os sistemas de planejamento

Planejamento de ações é uma área de pesquisa que se originou de trabalhos científicos que tentavam criar, principalmente com o uso da lógica de primeira ordem, solucionadores gerais de problemas, como é o caso do GPS (Ernst & Newell, 1969), cujo objetivo é encontrar, automaticamente, soluções para diversos problemas em diferentes domínios.

Os sistemas de planejamento são compostos, em sua maioria, por métodos de busca que visam encontrar uma seqüência de transformações que mude a descrição do estado de um sistema para uma descrição de um estado desejado. Em planejamento, tais transformações são realizadas por ações e se caracterizam por transições entre estados do mundo.

O sistema STRIPS (Fikes & Nilsson, 1971) foi efetivamente o primeiro sistema de planejamento que se caraterizava por procurar uma seqüência de ações que permitia transformar o estado inicial do mundo em um estado no qual o objetivo era satisfeito. Os estados são, conforme o modelo STRIPS, conjuntos de predicados que ditam o que é ou não verdade.

Os predicados, em conjunto, descrevem a situação em que se encontra o estado do mundo. Por exemplo, na Figura 1, esses predicados descrevem que a mão do robô está vazia (*handempty*), que o bloco *A* está em cima do bloco *B* (*on*(*A,B*)), que o topo dos blocos *A* e *C* estão livres (*clear*(*A*),*clear*(*C*)) e que os blocos *B* e *A* estão sobre a mesa (*ontable*(*A*), *ontable* (*B*)). Considera-se, nesse sistema, que somente os predicados que estão na descrição do estado são realmente verdade no mundo ao qual se pretende representar.

Um estado do mundo pode ser modificado por ações que, no modelo STRIPS, removem e inserem predicados no conjunto que descreve o estado. Ações são operadores instanciados, ou seja, quando todas as suas variáveis possuírem valores definidos. Por exemplo, considere os seguintes operadores do domínio de blocos:

- **pickup(X):**

  Precondição: ontable (X), clear (X), handempty

  Lista de remoção: ontable (X), clear (X), handempty

  Lista de adição: holding (X)

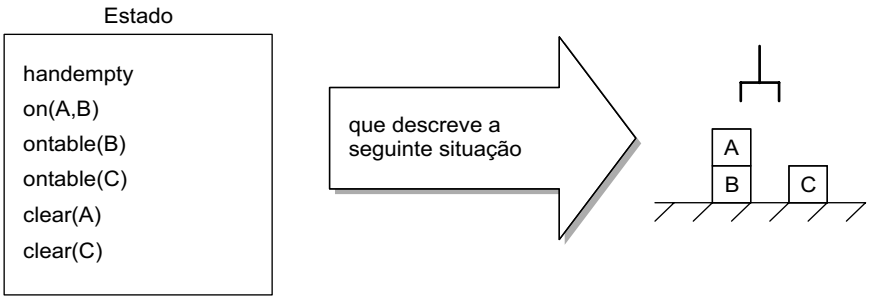

Figura 1 – Exemplo de descrição de uma situação no mundo dos blocos por conjuntos de predicados.

- **unstack(X,Y):**

  Precondição: handempty, clear (X), on (X,Y)

  Lista de remoção: handempty, clear (X), on (X,Y)

  Lista de adição: holding (X), clear (Y)

Esses operadores se tornam ações quando têm suas variáveis instanciadas. Ao aplicar uma ação em algum estado, alguns predicados são removidos (os indicados pela lista de remoção da ação) e outros são inseridos (os indicados pela lista de adição). Uma ação só pode ser aplicada se sua precondição for satisfeita no estado em que se pretende aplicá-la. Por exemplo, considere o estado da Figura 1 e a seguinte ação: *unstack*(*A,B*).

A ação *unstack*(*A,B*), conforme descrição anterior, somente poderá ser aplicada quando os predicados *handempty, clear*(*A*) e *on*(*A,B*) forem verdadeiros no estado, isto é, a precondição garante que, para a ação retirar o bloco *A* de cima do bloco *B*, o braço do robô deve estar vazio (*handempty*), não deve haver nenhum bloco sobre o bloco *A* (*clear*(*A*)) e o bloco *A* deve estar em cima do bloco *B* (*on*(*A,B*)). Satisfeita a precondição, a aplicação da ação retira os predicados *handempty, clear*(*A*) e *on*(*A,B*), contidos na lista de remoção, do estado e insere os predicados *holding*(*A*) e *clear*(*B*), contidos na lista de adição, que especificam que o robô está segurando o bloco *A* e o topo do bloco *B* está livre. A Figura 2 mostra a nova descrição de estado e a situação do mundo que ela descreve. Observe que a situação e a descrição são exatamente aquelas que se esperava após executar uma ação que desempilha o bloco *A* de cima do bloco *B*.

O sistema STRIPS trabalha escolhendo ações de modo a transformar o estado do mundo. Uma seqüência de ações define um plano. Para encontrar um plano é necessário definir um objetivo. Isto é feito definindo-se um subconjunto de predicados que descrevem quais são as características obrigatoriamente necessárias para o estado final desejável.

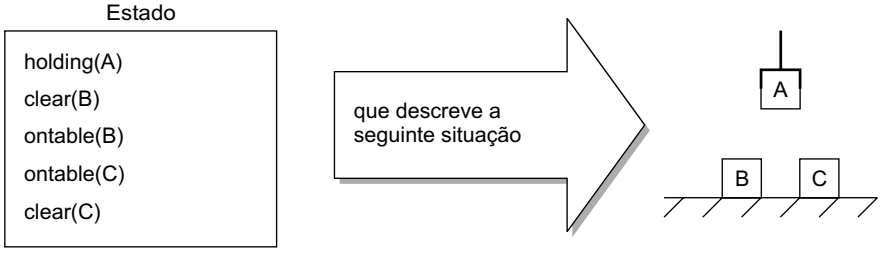

Figura 2 – Exemplo de descrição de uma situação no mundo dos blocos por conjuntos de predicados.

Dado o subconjunto, chamado de objetivo, o STRIPS procura ações para compor um plano que encontre um estado do mundo no qual esse objetivo seja satisfeito.

Após o sistema STRIPS, diversas outras técnicas foram empregadas nos mais diversos sistemas de planejamento. Surgiram então novos sistemas, como o ABSTRIPS, NONLIN, NOAH, SIPE e outros (Allen *et al.*, 1990). Esses sistemas empregam técnicas de abstração de planos e ações, de planejamento por ***compromisso-mínimo*** (*Least-Commitment*), técnicas como planejamento por ***ordem-parcial,*** entre outras (Allen *et al.*, 1990).

Da mesma forma que surgiram sistemas de planejamento, novas descrições de ações também foram sendo aprimoradas e desenvolvidas. Como uma alternativa ao modelo de ações STRIPS, Peadnult (1989) apresentou em seu trabalho uma extensão da descrição de ações, chamada de ADL, que incorporava efeitos condicionais, ou seja, que poderiam trabalhar com efeitos dependentes de contexto. Atualmente, observa-se que diversos sistemas de planejamento se baseiam nos modelos do sistema STRIPS, mas com o uso das ações do tipo ADL.

Mesmo com o aprimoramento de diversas técnicas de planejamento, foi o sistema GRAPHPLAN (Blum & Furst, 1997), que utilizava outras técnicas e métodos de IA, em vez das mesmas desenvolvidas e aplicadas durante anos na área de planejamento, que melhorou a performance dos sistemas de planejamento. Encontrou-se, então, um novo paradigma para a área: o planejamento baseado em busca heurística.

Esse tipo de sistema de planejamento, iniciado com o GRAPHPLAN e depois aperfeiçoado com o sistema HSP (Bonet & Geffner, 1999), baseia-se no uso de heurística para guiar o processo de busca. Embora a idéia de uso de heurística nos processos de busca seja dos primórdios da Inteligência Artificial, somente nos últimos anos foi possível encontrar heurísticas capazes de estimar a distância entre o estado inicial e o estado final de forma satisfatória.

Foi exatamente seguindo esse paradigma que o sistema FF (Hoffmann & Nebel, 2001), o mais rápido planejador da competição do AIPS'2000 (Bacchus, 2000), desenvolveu uma heurística que, baseada em um grafo parecido com o grafo gerado pelo GRAPHPLAN, possibilitava estimar a distância, em número de ações, entre o estado inicial e o estado final.

Os sistemas de planejamento baseados em busca heurística são rápidos e resolvem grande parte dos problemas nos mais diversos domínios, tais como logística, controle automático de elevadores e escalonamento, entre outros. No entanto, tais sistemas ainda apresentam dificuldades em alguns domínios simples, como é o caso do domínio chamado Mundo de Blocos. Isto ocorre em razão da busca heurística, que, por diversas vezes, se depara com pontos de mínimo local (*local minima*) no processo de busca que impedem tais sistemas de encontrarem solução (Hoffmann, 2001).

Uma forma de melhorar a performance dos atuais sistemas é por intermédio do uso das diversas técnicas desenvolvidas no passado e de novas técnicas que surgiram nos últimos anos.

Uma dessas técnicas é o RBC – Raciocínio Baseado em Casos (*Case-based Reasoning*), que utiliza casos previamente armazenados para ajudar na solução de novos problemas. As técnicas de RBC em planejamento foram primeiramente utilizadas pelo sistema CHEF (Hammond, 1989), que introduziu o conceito de PBC – Planejamento Baseado em Casos (*Case-based Planning*). Os planejadores que não eram baseados em casos passaram a ser chamados de planejadores generativos, pois geravam os planos a partir do nada (*from scratch*).

Os sistemas de planejamento baseados em casos, de modo geral, possuem no mínimo três etapas distintas: Resgate (*Retrieval*), Adaptação (*Adaptation*) e Armazenamento (*Storing*). Essas três etapas são realizadas seqüencialmente. Primeiramente, é resgatado, da memória de casos, o caso mais similar ao novo problema a ser solucionado. Essa similaridade é medida por uma métrica que geralmente considera as diferenças

entre o antigo e o novo problema. Depois, o caso é modificado pela etapa de adaptação, que tenta adequá-lo à solução do novo problema. O novo caso pode, então, ser, eventualmente, armazenado na memória para uso futuro.

De forma genérica, um sistema de PBC, diante de um novo problema, retornará um ou vários casos semelhantes da base de casos, por intermédio do processo de resgate, e, então, escolherá o mais similar para ser adaptado para a solução do novo problema. A escolha do caso mais similar se faz pela criação de uma lista ordenada pela métrica de similaridade. Se o caso mais similar não puder ser adaptado, considera-se o segundo da lista e assim sucessivamente.

Entretanto, o problema de estabelecer uma métrica de similaridade apropriada tem sido estudado há anos na área de raciocínio baseado em casos. Uma métrica de similaridade pode variar consideravelmente entre mais ou menos dependente do contexto. Quando independente do contexto, a métrica de similaridade corre o risco de se tornar incompleta, isto é, não permite considerar todas as características pertinentes do domínio. Quando dependente do contexto, exige, muitas vezes, aplicação de técnicas de aprendizado (*learning*) ou mesmo regras específicas do domínio de aplicação.

Outro grande desafio dos sistemas de PBC, além do estabelecimento da métrica de similaridade, é determinar quais casos armazenar, pois uma base de casos muito grande aumentará o custo de resgate. Pode-se, para facilitar a procura por casos na base de casos, usar uma indexação dos casos armazenados como *ponteiros* para os casos similares. Entretanto, um dos grandes problemas do uso desse tipo de indexação, apontado por Kettler (1995), é que ela limita a busca e pode torná-la ineficiente para alguns tipos de consultas que não àquelas previamente determinadas pela indexação.

Para diminuir o tempo de resgate de casos é necessária a limitação do espaço de busca por casos similares, fazendo com que o processo de resgate analise apenas os casos que apresentem certa similaridade com o problema a ser solucionado. Uma abordagem interessante para esse problema é feita por Smyth & Mckenna (1999), que mantém o número de casos armazenados e evita usar a indexação do tipo *ponteiro*.

A maioria dos sistemas de planejamento baseado em casos são extensões de planejadores generativos, isto é, a etapa de adaptação é composta por um sistema completo e correto de planejamento. Isso permite que o sistema de planejamento baseado em casos não dependa única e exclusivamente dos casos armazenados, mas possibilita que o sistema encontre uma solução a partir do nada (*from scratch*), caso a etapa de resgate não descubra nenhum caso que seja similar ao problema proposto.

No entanto, embora os resultados dos sistemas de PBC possam ser melhores que os dos sistemas generativos, na análise de pior caso ambos os sistemas são *NP-hard* (Nebel & Koehler, 1995).

A análise de pior caso leva em consideração que, para solucionar um problema no RBC, é preciso resgatar um caso e modificá-lo. Tanto o resgate quanto a modificação feita na etapa de adaptação podem levar a resultados piores, no que se refere à performance computacional, do que a solução encontrada a partir do nada (*from scratch*). Ou seja, teoricamente, não se pode afirmar que haja vantagem no uso de PBC em relação aos planejadores generativos.

Entretanto, no campo prático, com diversos avanços na área de RBC, como redução do espaço de busca de casos (Smyth & Mckenna, 1999) e manutenção da base de casos, entre outros, o uso de RBC em planejamento passou a ser novamente um atrativo para superar os resultados obtidos pelos atuais sistemas generativos de planejamento por busca heurística, como o HSP (Bonet & Geffner, 1999) e o FF (Hoffmann & Nebel, 2001).

3. A lógica de transações em planejamento

Para a definição de um sistema de planejamento, faz-se necessário o uso de uma estrutura formal que permita descrever ações, estados e objetivos de forma satisfatória. Para o sistema FAR-OFF, a Lógica de Transações (TR) (Bonner & Kifer, 1995) será usada para tal fim.

A TR é uma lógica que especifica formalmente fenômenos de atualização de base de dados, caminhos de execução e procedimentos de transições seqüenciais. A TR, em sua versão Horn-serial, é uma extensão da lógica de primeira ordem com a introdução de um novo operador chamado de conjunção serial (⊗). Este operador representa uma transação, em que $\alpha \otimes \beta$ significa: "primeiro execute α e então execute β".

Para descrever uma transação é considerada a seguinte notação: $P,D_0,...,D_n \models \phi$, em que ϕ é uma fórmula em TR e P é chamado de base de transações, que é um conjunto de fórmulas em TR. Cada D_i é um conjunto de fórmulas de primeira ordem que representa a constituição da base de dados. Intuitivamente, P é um conjunto de transações descritas em TR, ϕ é a invocação de algumas dessas transações e $D_0,...,D_n$ é a seqüência da base de dados representando todos os estados da execução de ϕ. Por exemplo, executando formalmente a transação ϕ, a base de dados vai do estado inicial D_0 ao estado final D_n.

Entretanto, se ϕ for somente uma consulta, ou seja, não modificar a base de dados, não haverá caminho de execução e a representação se dará apenas pelo estado em que ϕ é verdadeiro: $P,D \models \phi$. Uma consulta é descrita pelo predicado *qry(_)*. Por exemplo, considere o literal *nochão* que está presente no estado D_k. Então, $P,D_k \models qry(nochão)$.

A TR também trabalha com atualizações, e a especificação de transições é feita por intermédio do oráculo de transição O^t, que é uma função de mapeamento entre pares de estados e um conjunto de fórmulas atômicas. Neste artigo, uma transição será definida por meio dos predicados ins(_) e del(_). Por exemplo: uma possível satisfação para ins(c) e del(d) pode ser $P,D,D_1,D_2 \models ins(c) \otimes del(d)$. Isto ocorre se, e somente se, $ins(c) \in O^t(D,D+\{c\})$; $del(d) \in O^t(D,D-\{d\})$; $D_1 = D+\{c\}$; e $D_2 = D+\{c\}-\{d\}$.

Em razão de todas essas características formais, a semântica da TR se torna equivalente à semântica dos componentes de um sistema de planejamento (Tonidandel & Rillo, 1999), mostrando que a TR é uma lógica apropriada para formalização de ações, estados e planos em planejamento.

Diversos outros trabalhos já haviam utilizado TR na formalização dos componentes e do sistema de planejamento (Santos & Rillo, 1997; Tonidandel & Rillo, 1998; Tonidandel & Rillo, 2000).

Considerando ***L*** uma linguagem definida em versão Horn-serial da TR, pode-se definir os componentes de um sistema de planejamento como a seguir:

Definição 1: *O estado D é um conjunto finito de predicados de primeira ordem e é representado em TR como uma base de dados. Cada d ∈ D é denominado de fato.*

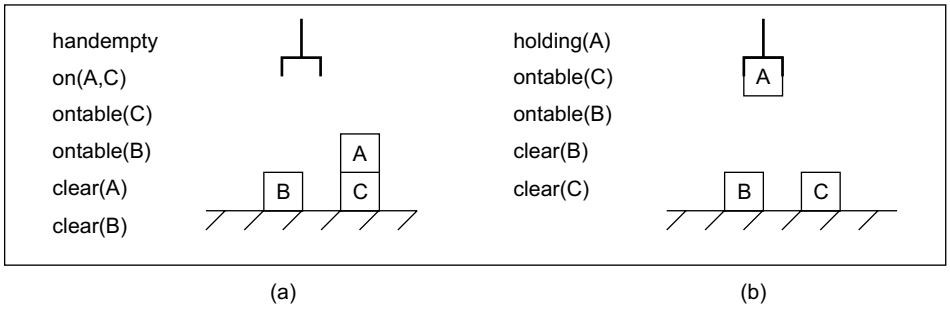

Figura 3 – Dois exemplos de estados no domínio do mundo de blocos.

Dois exemplos de descrição de estados são apresentados na Figura 3.

Como dito anteriormente, estado é o conjunto de fatos que são verdadeiros na situação que se pretende descrever. Desta forma, se algo mudar na situação, pelo menos um fato será removido ou inserido. Essa transição entre estados, com remoção e/ou inserção de fatos, é realizada pelas ações, que podem ser definidas como:

Definição 2: *Considerando* $\boldsymbol{A \subseteq L}$ *como um conjunto de definições de ações, cada* α, $\alpha \in \boldsymbol{A}$, *possui a seguinte estrutura:*

$$\alpha \leftarrow pre(\alpha) \otimes delete(\alpha) \otimes add(\alpha);$$

em que:

- *pre(*α*) é uma fórmula TR composta por predicados qry(_) que representam as precondições da ação* α.
- *delete(*α*) é uma lista de remoção que representa os fatos que serão removidos da base de dados pela ação.*
- *add(*α*) é uma lista de adição que representa todos os fatos que serão inseridos na base de dados pela ação.*

O resultado da execução de uma ação é um estado atualizado *D'* gerado após a remoção e a inserção de predicados em *D* conforme, respectivamente, as listas de remoção e adição. Qualquer ação somente pode ser executada a partir de algum estado *D* se, e somente se, $pre(\alpha) \subseteq D$. Por exemplo, considere a seguinte ação no domínio do mundo de blocos:

$unstack(x,y) \leftarrow qry(handempty) \otimes qry(clear(x)) \otimes qry(on(x,y)) \otimes del(clear(x)) \otimes del(handempty) \otimes del(on(x,y)) \otimes ins(holding(x)) \otimes ins(clear(y))$.

Essa ação tem por objetivo transformar o estado de tal modo que ela represente retirar o bloco *x* de cima do bloco *y*. Para isto, a ação é dividida em precondição, lista de adição e lista de remoção, como a seguir:

$pre(unstack(x,y)) = qry(handempty) \otimes qry(clear(x)) \otimes qry(on(x,y))$.

$delete(unstack(x,y)) = del(clear(x)) \otimes del(handempty) \otimes del(on(x,y))$.

$add(unstack(x,y)) = ins(holding(x)) \otimes ins(clear(y))$.

Para exemplificar o uso de uma ação, considere o estado D_0 como descrito na Figura 3a e considere o estado *Df* como descrito na Figura 3b. Assim, ao aplicar a ação *unstack(A,C)*, o estado D_0 se torna o estado *Df*. Formalmente, isto pode ser descrito em TR como:

$$P, D_0 ... Df \models unstack(A,C)$$

em que *P* é um conjunto de ações e $<D_0 ... Df>$ é um caminho de estados resultante da aplicação dos predicados *del(_)* e *ins(_)* contidos na fórmula *unstack(A,C)*.

Com as definições de ações e estados é possível, conforme Tonidandel & Rillo (1999), definir planos com o uso da lógica de transações:

Definição 3: *Um plano é uma fórmula TR da seguinte forma:*

$$\delta = \alpha_1 \otimes ... \otimes \alpha_n,$$

em que $\alpha_i \in A$; $1 \leq i \leq n$.

Um plano pode ser vazio, isto é, sem ação em sua composição, denotado por δ_0. É exatamente um plano δ ou um plano δ_0 que o planejador deve encontrar como resposta a um objetivo proposto. Define-se, então, uma instância *plan* para representar, formalmente, o plano retornado pelo planejador:

Definição 4: *Uma instância plan pode ser um plano* δ *ou um plano vazio* δ_0.

A instância *plan* definida servirá para apontar ao planejador onde deve ser feito o planejamento. A instância *plan* é utilizada na definição do objetivo como a seguir:

Definição 5: (*Objetivo*) *Um objetivo é uma fórmula TR com a seguinte composição:*

$$\boldsymbol{Obj}: plan \otimes Df$$

em que Df é um grupo de consultas que representam o estado final desejado.

Desta forma, dado um objetivo, o planejador deve encontrar um plano que substituirá a instância *plan* de modo que haja um caminho de estados entre o estado inicial D_0 e o estado final D_k, tal que $Df \subseteq D_k$.

Um caso é um plano realizado anteriormente e armazenado para uso futuro. É preciso, então, definir a estrutura de um caso para armazenamento:

Definição 6: *(Caso) Um caso* η *é uma fórmula TR que segue a seguinte estrutura:*

$$\eta = \boldsymbol{Wi} \otimes \alpha_1 \otimes ... \otimes \alpha_n \otimes \boldsymbol{Wf}$$

em que:

- $\alpha_i \subseteq A;\ 1 \leq i \leq n$, *seqüência de ações definidas pelo planejador para satisfazer um objetivo proposto.*
- ***Wi*** *é um conjunto de consultas em TR que representam a precondição do caso.*
- ***Wf*** *é um conjunto de consultas em TR que representam a pós-condição do caso.*

Intuitivamente, ***Wi*** é um conjunto de consultas para os predicados que devem estar presentes no estado inicial, de modo a permitir a execução do plano. Já ***Wf*** é o conjunto de consultas que representam os fatos que serão inseridos e que estarão, necessariamente, presentes no estado final após a execução do plano.

4. O Sistema FAR-OFF

O sistema FAR-OFF (*Fast and Accurate Retrieval on Fast Forward*) (Tonidandel & Rillo, 2002) é um sistema de planejamento baseado em casos que usa novos e precisos métodos de resgate de casos. A versão apresentada neste artigo é uma versão do tipo STRIPS, ou seja, não permite a descrição de ações do tipo ADL. Isto significa que, no sistema FAR-OFF, as ações, planos e casos são definidos conforme seção anterior.

O sistema é constituído de três fases distintas: a fase de resgate de casos, a fase de adaptação do caso resgatado e, finalmente, a fase de armazenamento, que verifica se o plano solução encontrado será ou não armazenado. O sistema, ao tentar solucionar um problema de planejamento, primeiramente resgata um caso que seja similar à possível solução do problema pela fase de resgate e depois faz as adaptações necessárias no caso resgatado de modo que ele possa se tornar uma solução. Após isto, a nova solução, que é um plano, se torna um caso, conforme Definição 6, a ser armazenado para usos futuros.

Nas seções posteriores serão detalhadas as etapas de resgate e de adaptação de casos, que são as responsáveis pela solução dos problemas de planejamento.

4.1 Resgate de casos

O resgate de casos é uma importante fase de um sistema de planejamento baseado em casos, pois o esforço de adaptação para encontrar a solução para um certo problema depende da qualidade do caso que é resgatado.

Para resgatar um caso é utilizada uma métrica de similaridade que escolhe, entre diversos casos armazenados, aquele que mais se assemelha à possível solução do problema. No entanto, a utilização de uma métrica de similaridade imprecisa pode tornar o sistema ineficiente, pois exigirá esforço maior da fase de adaptação.

Diante disso, diversos métodos foram propostos com regras de similaridade baseadas no esforço da etapa de adaptação (Leake *et al.,* 1997; Smyth & Keane, 1998). Esses métodos incorporam características específicas do domínio ao qual foram propostos. Pelo fato de as regras de similaridade baseadas em esforço de adaptação existentes não poderem ser aplicadas a sistemas independentes do domínio foi proposta uma regra de similaridade baseada na estimativa da distância entre estados para ser usada nos diversos sistemas de PBC.

Essa regra de similaridade, denominada Similaridade Guiada pela Distância-Ação (*ADG – Action-Distance Guided*) (Tonidandel & Rillo, 2001a), surgiu da relação entre a quantidade de ações e o esforço de geração de um plano. Quanto mais ações um planejador precisar encontrar, maior será o tempo gasto para gerá-lo. Esta é exatamente a idéia na qual os sistemas de planejamento por busca heurística se

basearam. Esses sistemas utilizam uma heurística baseada na distância entre estados para guiar a busca no espaço de estados. Essa distância entre estados é medida pela estimativa do número de ações necessárias para a transição entre os dois estados.

Com a mesma heurística usada pelo sistema FF (Hoffmann & Nebel, 2001), a *ADG* encontra uma medida precisa para a similaridade entre casos e problemas. Isto é feito por dois processos. O primeiro é a determinação da distância entre o estado inicial D_0 e o *Wi* de um caso. O outro é a determinação da distância entre o *Wf* do caso e o estado final desejado, *Df*, do objetivo. As duas estimativas estão apresentadas na Figura 4.

A estimativa da distância entre o estado inicial e o *Wi* é a aplicação direta do cálculo da heurística do sistema FF, denominada *FF-heurística*. Essa heurística constrói um grafo de possibilidades de ações e estados a partir do estado inicial, ignorando a lista de remoção (Hoffmann & Nebel, 2001). Ignora-se a lista de remoção das ações para criar um grafo finito sem interações entre os diversos predicados.

Esse grafo é constituído por camadas intercaladas de fatos e ações. A primeira camada de fatos é o estado inicial. A primeira camada de ações é composta por todas as ações que podem ser aplicadas na primeira camada de fatos, ou seja, ações cujas precondições são satisfeitas no estado inicial. A segunda camada de fatos é composta pela primeira camada de fatos mais os predicados das listas de inserção das ações da primeira camada. A partir da nova camada de fatos criada, uma nova camada de ações é criada, e assim por diante.

O grafo vai sendo construído até que não haja mais nenhuma camada de fatos que possa ser diferente de outra já determinada. A partir desse grafo, algumas informações são obtidas:

Definição 7: *level(d) := min {i | d ∈ F_i, em que F_i é a i-ésima camada de fatos}*

Definição 8: *level(α) := min {i | α ∈ O_i, em que O_i é a i-ésima camada de ações}*

As Definições 7 e 8 determinam qual foi a primeira camada em que cada ação e cada predicado apareceu no grafo. Essas informações são necessárias para estimar a distância.

Para obter a estimativa da primeira distância, entre o estado inicial e o *Wi*, determina-se em direção reversa, isto é, do *Wi* para o estado inicial, quais ações satisfazem os predicados de *Wi*, considerados como objetivos, em cada camada do grafo (Figura 5). Cada objetivo satisfeito é marcado como *verdadeiro* e cada predicado da precondição da ação é posto como um novo objetivo a ser cumprido. O processo termina quando todos os objetivos a serem satisfeitos estiverem no estado inicial. A estimativa da distância é dada pelo número de ações usadas para satisfazer os objetivos no processo (Hoffmann & Nebel, 2001) e o algoritmo é representado pela função ***estimativa_inicial(Wi)*** da Figura 5.

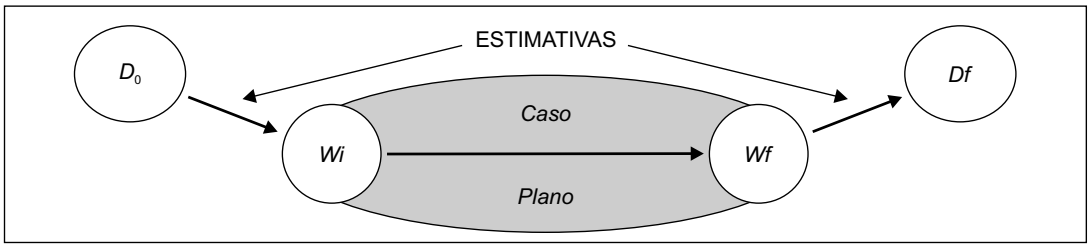

Figura 4 – As duas estimativas usadas para determinação da similaridade *ADG*, em que D_0 é o estado inicial, D_f é o objetivo final e *Wi* e *Wf* são as características do estado inicial e final em que o caso foi aplicado.

O cálculo da distância entre *Wf* e *Df* é um pouco mais complicado. Como a *FF-heurística* calcula qualquer distância apenas a partir do estado inicial, não é possível aplicar diretamente a *FF-heurística* para *Wf* e para *Df* e determinar a diferença entre as distâncias. É necessário forçar a estimativa a considerar as ações do caso resgatado, como mostrado em Tonidandel & Rillo (2001a).

Isso é feito da seguinte forma: primeiro executa-se a *FF-heurística* entre o estado inicial e o *Wi*, ou seja, chama-se a função ***estimativa_inicial(Wi)*** da Figura 5. Depois, utilizando-se as marcações feitas pela primeira estimativa, marcam-se todos os predicados de *Wi* como *falso* e todos os predicados de *Wf* como *verdadeiro*. Aplica-se novamente a *FF-heurística* a partir do estado inicial D_0 para *Df*, só que as marcas de *verdadeiro* e *falso*, tanto da primeira estimativa quanto as modificadas para *Wi* e *Wf*, farão com que a estimativa seja um resultado próximo da distância entre *Wf* e *Df*. O algoritmo do cálculo da estimativa da distância entre *Wf* e *Df* é dado pelo algoritmo ***estimativa_final(Wi,Wf,Df)***, apresentado na Figura 6. O cálculo da *ADG* é a soma das duas estimativas, entre o estado inicial D_0 e *Wi*, e entre *Wf* e *Df*. Tonidandel & Rillo (2001a) mostraram empiricamente que a similaridade *ADG* é mais precisa que outras similaridades normalmente utilizadas nos sistemas de PBC.

A precisão da regra de similaridade não é suficiente para tornar o sistema de PBC eficiente, pois faz-se também necessário reduzir o espaço de busca do caso similar. Para tanto, o sistema FAR-OFF utiliza o método conhecido como *Footprint-based Retrieval (FbR)* (Smyth & Mckenna, 1999), que permite encontrar, na maioria das vezes, o caso mais similar analisando apenas um subconjunto de casos da base de casos.

```
estimativa_inicial(G)

limpe todos Gi
para i:= 1 ... max faça
  Gi := {g ∈ G |level(g) = i};
fimpara
h:=0;
para i:= max ... 1 do
  para g ∈ Gi False no instante i faça
    selecione α_{level=i-1}; g ∈ add(α);
    h:=h+1;
para cada d_{level ≠ 0} ∈ pre(α), d False
no instante i-1 faça
      G_{level(d)} := G_{level(d)} ∪ {d};
    fimpara
    para cada d ∈ add(α) faça
      marque d True no instante i-1 e i;
    fimpara
  fimpara
fimpara
retorne h;
```

Figura 5 – Algoritmo da similaridade *ADG* que determina a solução relaxada a partir de um grafo relaxado, em que *G* é o estado objetivo. Este algoritmo foi extraído de Hoffmann & Nebel (2001).

```
estimativa_final (Wi,Wf,Df)

G:=Df;
para cada d ∈ Wi do
   marque d False em todos os níveis;
fimpara
para cada d ∈ Wf do
   marque d True no level(d) e level(d)-1
fimpara
h':=estimativa_inicial(G);
retorne h';
```

Figura 6 – O algoritmo *ADG* que extrai a estimativa da distância entre *Wf* e *Df* considerando as marcas deixadas pela execução da função estimativa_inicial(*Wi*). Este algoritmo foi extraído de Tonidandel & Rillo (2001a).

Este método de resgate de casos baseia-se na competência de cada caso. Competência é o espectro de problemas que um caso pode solucionar. Entretanto, como não há estimativas da competência para todos os problemas de um domínio, utiliza-se a suposição de representatividade (***Representativeness assumption***) (Smyth & Mckenna, 1999), que considera os casos armazenados como uma estimativa dos problemas do domínio. Conforme Smyth & Mckenna (1999), é possível definir, para uma base de casos *C*:

Definição 9: *CoverageSet(c) = {c' ∈ C: Solves(c,c')}*

Definição 10: *ReachabilitySet(c) = {c' ∈ C: Solves(c',c)}*

em que Solves(c,c') significa que o caso c, após adaptado, pode solucionar o caso c'.

Com as duas definições anteriores, o método *FbR* propõe construir um conjunto de casos, chamados de casos-*footprint,* que representam uma base de casos menor mas com a mesma competência da original. Cada caso-*footprint c* tem um conjunto de casos chamado de Conjunto-Relativo (Smyth & Mckenna, 1999) que são os casos pertencentes a *CoverageSet(c)* ∪ *ReachabilitySet(c)*. Deste modo, a união entre os casos-*footprint* e os Conjuntos-Relativos formam a base de casos original. A estrutura da base de casos é apresentada na Figura 7.

Para determinar os casos-*footprint*, segundo Smyth & Mckenna (1999), é necessário usar o algoritmo chamado CNN (*Condensed Nearest Neighborhood*) e uma métrica chamada de Competência-Relativa (*RelativeCoverage*). A métrica ordena os casos conforme sua competência relativa e o algoritmo CNN faz, passo a passo, a escolha dos casos-*footprint* e determina o Conjunto-Relativo de casos de cada um deles.

Uma vez determinados os casos-*footprint* e seus respectivos Conjuntos-Relativos, o método de resgate *FbR* pode ser facilmente executado para encontrar casos similares, o que é feito por dois processos distintos. Primeiro, o caso-*footprint* mais similar é encontrado, e funciona como uma espécie de caso-chave, depois é procurado o caso mais similar entre os casos do Conjunto-Relativo do caso-*footprint* escolhido.

O sistema FAR-OFF utiliza-se desse método *FbR* para diminuir o espaço de busca do processo de resgate. Para tanto, emprega a regra de similaridade *ADG* para calcular a competência dos casos, como na definição a seguir:

Definição 11: *Solves(c,c') se, e somente se, o valor ADG do caso c para outro caso c' for* $< \rho$

A definição anterior mostra que um caso *c* soluciona um outro caso *c*' se, e somente se, a regra de similaridade *ADG* do caso *c* para solucionar um problema descrito no caso *c'* for menor que um certo limite ρ.

Deste modo, o sistema FAR-OFF, utilizando-se da regra de similaridade *ADG* e do método de redução do espaço *FbR*, possui um sistema de resgate de casos rápido e preciso (Tonidandel & Rillo, 2001a).

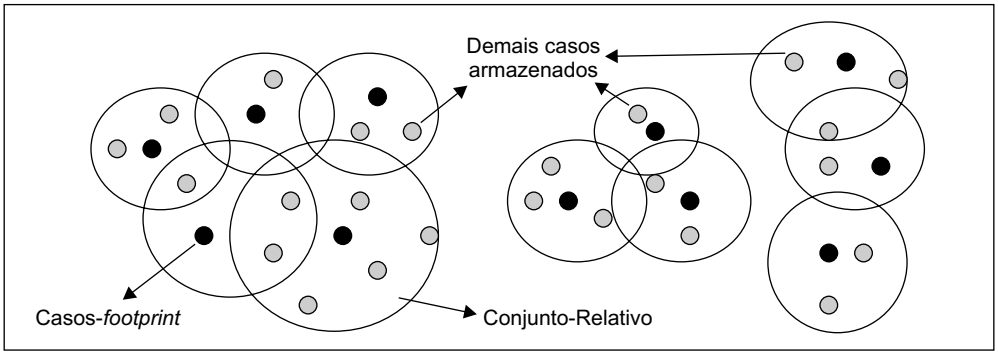

Figura 7 – Base de casos estruturada em casos-*footprint* e Conjuntos-Relativos.

4.2 Adaptação

Após a fase de resgate de casos, a fase de adaptação fica responsável por modificar o caso resgatado de modo a encontrar a solução do novo problema. No entanto, em muitos sistemas de PBC a etapa de adaptação se caracteriza por um processo que modifica o caso por completo, ou seja, adiciona ou remove ações, ou mesmo troca ações de lugar. Este tipo de processo necessita de muito tempo de processamento, tornando-o ineficiente.

Para tornar o processo de adaptação mais eficiente, o sistema FAR-OFF não mexe na estrutura do caso resgatado, mas, sim, torna-o, em sua forma completa, parte da solução do problema. Para isto, o sistema precisa encontrar um plano entre o estado inicial D_0 e o *Wi* do caso e outro entre *Wf* e *Df*. Na maioria dos sistemas de PBC, a etapa de adaptação usa um planejador generativo, pois se não houver nenhum caso que possa ser resgatado para solucionar um determinado problema, o planejador generativo, na etapa de adaptação, poderá encontrar a solução a partir do nada (*from scratch*). O sistema FAR-OFF utiliza um sistema de planejamento generativo baseado no sistema FF (Hoffmann & Nebel, 2001). Este sistema generativo, embora mais simples, possui algumas heurísticas e métodos usados no sistema FF original.

Depois de ser resgatado um caso η, é passada a seguinte estrutura de objetivo para a etapa de adaptação:

$$\text{Obj: } plan \otimes \eta \otimes plan \otimes Df$$

As instâncias *plan* presentes na estrutura anterior indicam o local onde o planejador generativo da etapa de adaptação deverá encontrar um plano. Observe que a medida de similaridade *ADG* estima a quantidade de ações existentes em cada uma das instâncias *plan*, que são, respectivamente, as distâncias entre D_0 e *Wi* e entre *Wf* e *Df*.

Cada uma das instâncias *plan* será substituída por um plano gerado pelo sistema de planejamento generativo FF implementado e inserido na fase de adaptação. Isto é feito de forma simples; basta executar o sistema FF para que encontre um plano do estado inicial e satisfaça o objetivo definido por *Wi*. Depois, executam-se as ações do plano encontrado e as ações do caso encontrando um novo estado. Esse novo estado será, para o sistema FF, um novo estado inicial que encontrará a solução até o objetivo final.

A solução, portanto, é a concatenação dos planos encontrados e o caso resgatado.

5. Testes

Com o sistema definido, faz-se necessária a realização de testes que permitam verificar o comportamento, a performance e os resultados da aplicação do sistema de planejamento baseado em casos em domínios reais.

O exemplo tratado a seguir é do domínio de logística, mais especificamente da parte que trata do controle e distribuição de mercadorias (Figura 8).

Neste exemplo são considerados caminhões que circulam apenas dentro da cidade e aviões, que fazem o transporte entre cidades. Um problema de planejamento em logística se dá pela determinação das localidades de cada mercadoria e o local para onde elas devem ser transportadas.

Os testes realizados visam verificar o comportamento do sistema de planejamento baseado em casos e o sistema de planejamento generativo. Para melhor visualização dos resultados, o sistema FAR-OFF foi configurado para resolver os problemas sempre por intermédio de um caso resgatado da memória; isto quer dizer que o sistema não gerará nenhuma solução a partir do nada (*from scratch*). Em contrapartida, o sistema generativo será também verificado na solução dos problemas sem o uso de casos resgatados.

O uso de um planejador baseado em casos se justifica pelo fato de que os sistemas de planejamento baseado em casos são mais estáveis, isto é, podem solucionar um espectro maior de problemas que os sistemas generativos (Tonidandel & Rillo, 2002).

Logistic Domain

```
:types (truck,airplane - vehicle)
       (package,vehicle - physobj)
       (airport,location - place)
       (city,place,physobj - object)

 :predicates
      in-city(place,city)
      at(physobj,place)
      in(package,vehicle)


:actions

LOAD-TRUCK(package,truck,place)
:precondition => at(truck,place)∧at(package,place)
:delete list => at(package,place)
:insert list => in(package,truck)

UNLOAD-TRUCK(package,truck,place)
:precondition => at(truck,place)∧in(package,truck)
:delete list => in(package,truck)
:insert list => at(package,place)

LOAD-AIRPLANE (package,airplane,place)
:precondition at(package,place)∧at(airplane,place)
:delete list => at(package,place)
:insert list => in(package,airplane)

UNLOAD-AIRPLANE(package,airplane,place)
:precondition => at(truck,place)∧in(package,airplane)
:delete list => in(package,airplane)
:insert list => at(package,place)

DRIVE-TRUCK (truck, place_from, place_to,city)
:precondition => at(truck, place_from)∧
      in-city(place_from,city)∧in-city(place_to,city)
:delete list => at(truck, place_from)
:insert list => at(truck, place_to)

FLY-AIRPLANE (airplane, airport_from, airport_to)
  :precondition => at(airplane, airport_from)
  :delete list => at(airplane, airport_from)
  :insert list => at(airplane, airport_to)
```

problem logistics-4-0

```
:domain logistics

:objects
 apn1 - airplane
 apt1 - airport
 apt2 - airport
 pos1 - location
 pos2 - location
 cit1 - city
 cit2 - city
 tru1 - truck
 tru2 - truck
 obj11 - package
 obj12 - package
 obj13 - package
 obj21 - package
 obj22 - package
 obj23 - package


:initial state
at(apn1,apt2)
at(tru1,pos1)
at(obj11,pos1)
at(obj12, pos1)
at(obj13,pos1)
at(tru2,pos2)
at(obj21,pos2)
at(obj22,pos2)
at(obj23, pos2)
in-city(pos1,cit1)
in-city(apt1,cit1)
in-city(pos2,cit2)
in-city(apt2,cit2)


:goal
at(obj11,apt1)
at(obj23,pos1)
at(obj13,apt1)
at(obj21,pos1)
```

Figura 8 – Descrição do domínio de logística e o problema 4-0 usado nos testes.

Outro fator a favor dos sistemas de PBC é que, em situações complexas, o sistema de PBC pode solucionar um problema em tempo menor do que se fosse usado um sistema generativo. Isto pode ser verificado em um estudo preliminar em Tonidandel & Rillo (2001b).

Para realização dos testes, serão utilizados o exemplo citado no domínio de logística e os problemas utilizados pelos sistemas de planejamento na competição do AIPS'2000 (Bacchus, 2000).

Esses problemas variam em complexidade. Por exemplo, o problema 4-0, apresentado na Figura 8, é o mais simples e possui 2 caminhões, 1 avião, 2 cidades, cada uma com 1 localidade e 1 aeroporto, e 6 pacotes para serem transportados. Já o problema 12-0 possui 4 caminhões, 1 avião e 4 cidades, cada uma com 1 aeroporto e 12 pacotes para serem transportados.

Cada um desses problemas possui uma base de casos composta por casos gerados aleatoriamente. Determinou-se para esses testes que cada base de casos teria 800 casos previamente realizados com 25% de casos-*footprint*. Para obter essa base de casos foi usado um semeador de casos variando-se o valor de ρ (Definição 11) para cada tipo de problema.

Os resultados apresentados (Tabela 1 e Figura 9) são promissores, pois as soluções apresentadas por ambos os sistemas, generativo e baseado em casos, solucionam os diversos problemas em tempo aceitável. A vantagem obtida com o planejador baseado em casos é que este é um sistema mais estável e dificilmente deixará de encontrar uma solução para um certo problema, e em tempo aceitável, como mostrado no artigo de Tonidandel & Rillo (2002) para o mundo de blocos.

Analisando os resultados, percebe-se que o sistema generativo soluciona os problemas em tempo menor que o sistema baseado em casos. Entretanto, isso se deve a pelo menos três fatores. Primeiro, o sistema FAR-OFF está configurado para solucionar problemas apenas por intermédio de casos resgatados, o que o impossibilita de decidir *a priori* se deveria utilizar um caso armazenado ou se partiria direto para o planejamento generativo. Segundo, o sistema generativo baseado no sistema FF apresenta seus melhores resultados no domínio de logística, o que não acontece em outros domínios, como, por exemplo, o Mundo de Blocos. Terceiro, as bases de casos são compostas por casos gerados aleatoriamente, o que impede que as bases de casos sejam compostas por problemas habituais já realizados e repetidos.

Na verdade, os problemas em um sistema real tendem a se repetir. Se isso acontecer, o planejador baseado em casos encontrará a solução muito mais rapidamente do que o planejador generativo. Isto ocorre porque haverá pouca ou nenhuma adaptação de casos, e o tempo de planejamento será próximo ao tempo de resgate, o que pelos testes são tempos menores que o do planejamento generativo.

A grande vantagem do sistema FAR-OFF sobre os sistemas generativos, no entanto, é sua estabilidade. Isso ocorre por que o sistema FAR-OFF tem diversas formas de encontrar uma solução, tantas quantas forem o número de casos similares resgatados. Como mostrado no artigo de Tonidandel & Rillo (2002), essa característica possibilita que o sistema FAR-OFF resolva 100% dos problemas do Mundo de Blocos, enquanto o sistema generativo resolve apenas 71% dos mesmos.

Entretanto, a maior desvantagem do sistema FAR-OFF é que os planos obtidos como solução são ligeiramente maiores que os planos obtidos pelo sistema generativo. Isto pode ser causado, primeiramente, pela base de casos que foi formada aleatoriamente e que, portanto, não é uma boa representação do real uso de planejamento no domínio de logística. Segundo, porque não há ainda nenhum tipo de verificação de casos que possam ser emendados para solução de novos problemas.

Portanto, o sistema FAR-OFF pode e deve ser melhorado com inclusões de novas técnicas que permitam emendar e abstrair casos para formar uma solução mais próxima da solução ótima.

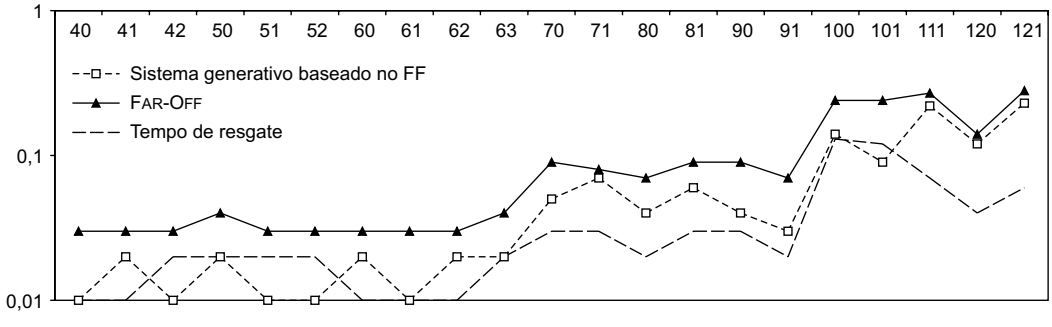

Figura 9 – Gráfico do resultado dos problemas da Tabela 1.
O eixo Y está em escala logarítmica e representa o tempo em segundos.

Tabela 1 – Os resultados dos problemas de logística. São apresentados o tempo e o número de ações das soluções dos sistemas generativo e FAR-OFF, o tempo gasto para resgate dos casos similares, o número de casos-*footprint* que existe em cada base de casos utilizada e o valor de ρ usado para determinar esses casos-*footprint*.

DOMÍNIO: LOGÍSTICA	Sistema generativo		FAR-OFF				
			Solução		Resgate de casos		
Problemas	Tempo	#Ações	Tempo	#Ações	Tempo	#Casos-*foot*	ρ
LogisticDomain-4-0	0,01	20	0,03	35	0,01		
LogisticDomain-4-1	0,02	19	0,03	26	0,01	334	10
LogisticDomain-4-2	0,01	15	0,03	17	0,02		
LogisticDomain-5-0	0,02	27	0,04	54	0,02		
LogisticDomain-5-1	0,01	17	0,03	27	0,02	326	10
LogisticDomain-5-2	0,01	8	0,03	10	0,02		
LogisticDomain-6-0	0,02	25	0,03	35	0,01		
LogisticDomain-6-1	0,01	14	0,03	34	0,01	208	12
LogisticDomain-6-2	0,02	26	0,03	39	0,01		
LogisticDomain-6-3	0,02	25	0,04	37	0,02		
LogisticDomain-7-0	0,05	38	0,09	92	0,03	320	26
LogisticDomain-7-1	0,07	45	0,08	72	0,03		
LogisticDomain-8-0	0,04	34	0,07	83	0,02	156	30
LogisticDomain-8-1	0,06	48	0,09	61	0,03		
LogisticDomain-9-0	0,04	39	0,09	80	0,03	171	30
LogisticDomain-9-1	0,03	31	0,07	38	0,02		
LogisticDomain-10-0	0,14	50	0,24	52	0,13	348	40
LogisticDomain-10-1	0,09	44	0,24	51	0,12		
LogisticDomain-11-1	0,22	67	0,27	70	0,07	236	45
LogisticDomain-12-0	0,12	46	0,14	49	0,04	190	45
LogisticDomain-12-1	0,23	75	0,28	90	0,06		

6. Considerações finais

O FAR-OFF é um sistema de planejamento baseado em casos que pode ser aplicado no controle inteligente de processos e da produção, em que o próprio sistema decide qual o plano que deve ser realizado para atingir certos objetivos. Neste artigo, alguns testes experimentais foram realizados no domínio de logística. Embora somente o domínio de logística não seja suficiente para medir a real capacidade do sistema em aplicações reais, os resultados podem ser considerados bons, porque o sistema FAR-OFF conseguiu solucionar todos os problemas em tempo aceitável, mesmo que os planos encontrados não sejam ótimos, pois possuem mais ações do que as realmente necessárias.

O sistema FAR-OFF apresentado neste artigo é um sistema versão STRIPS, isto é, os domínios possuem representação simples e não comportam efeitos condicionais como domínios descritos por ações do tipo ADL.

Entretanto, o sistema FAR-OFF pode ser facilmente expandido para considerar ações ADL e, conseqüentemente, trabalhar com domínios muito mais complexos.

7. Agradecimentos

Este trabalho tem o apoio da FAPESP sob o contrato nº 98/15835-9.

Referências Bibliográficas

ALLEN, J.; HENDLER, J.; TATE, A. A. *Readings in planning*. San Mateo, California: Morgan Kaufmann Publishers, 1990.

BACCHUS, F. *AIPS-2000 Planning Competition Results*. Available in: <http://www.cs.toronto.edu/aips2000/>.

BLUM, A.; FURST, M. Fast planning through planning graph analysis. *Artificial Intelligence*, n. 90, v. 1-2, p. 279-298, 1997.

BONET, B.; GEFFNER, H. Planning as heuristic search: new results. In: EUROPEAN CONFERENCE ON PLANNING – ECP'99. 1999, Durham. *Proceedings...* Durham, UK.

BONNER, A. J.; KIFER, M. *Transaction logic programming*. Toronto, Canadá: University of Toronto. 1995, Technical Report CSRI-323.

ERNST, G. W.; NEWELL, A. *GPS*: a case study in generality and problem solving. New York: Academic Press, 1969.

FIKES, R. E.; NILSSON, N. J. STRIPS: A new approach to theorem proving in problem solving. *Journal for Artificial Intelligence*, v. 2, p. 189-208, 1971.

HAMMOND, K. *Case-based planning*: viewing planning as a memory task. Academic Press, 1989.

HOFFMANN, J. Local search topology in planning benchmarks: an empirical analysis. In: IJCAI-01 INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2001, Seattle, Washington. *Proceedings...* Seattle, Washington: Morgan Kaufmann, 2001, p. 453-458.

HOFFMANN, J.; NEBEL, B. The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research,* v. 14, p. 253-302, 2001.

KETTLER, B. P. *Case-based planning with a high-performance parallel memory*. 1995. Tese (Doutorado) – University of Maryland Park, Maryland.

LEAKE, D. et al. Case-based similarity assessment: estimating adaptability from experience. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 14., 1997. AAAI Press, 1997.

NEBEL, B.; KOEHLER, J. Plan reuse versus plan generation: a theoretical and empirical analysis. *Artificial Intelligence Special Issue on Planning and Scheduling*, n. 76, p. 427-454, 1995.

PEDNAULT, E. P. D. ADL: exploring the middle ground between STRIPS and the situation calculus. In: CONFERENCE ON PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING, 1., 1989, Toronto, Ontario. *Proceedings...* Toronto, Ontário, 1989. p. 324-332.

SANTOS, M.; RILLO, M. Approaching the plans are programs paradigm using transaction logic. In: EUROPEAN CONFERENCE ON PLANNING, 4., 1997. Springer-Verlag, 1997. p. 377-389.

SMYTH, B.; KEANE, M. Adaptation-guided retrieval: questioning the similarity assumption in reasoning. *Journal of Artificial Intelligence*, v. 2, n. 102, p. 249-293, 1998.

SMYTH, B.; MCKENNA, E. Footprint-based retrieval. In: INTERNATIONAL CONFERENCE IN CASE-BASED REASONING, 3., 1999. ALTHOUFF, K.; BERGMANN, R.; BRANTING, K. (Eds.). *Lecture Notes in Artificial Intelligence*, v. 1650, Springer-Verlag, 1999. p. 343-357.

TONIDANDEL, F.; RILLO, M. Case-based planning in transaction logic framework. In: WORKSHOP ON INTELLIGENT MANUFACTURING SYSTEMS, 1998. Elsevier Science, 1998. p. 281-286.

TONIDANDEL, F.; RILLO, M. O uso da lógica de transações em planejamento. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE – SBAI'99, 4., São Paulo, 1999. p. 239-244.

TONIDANDEL, F.; RILLO, M. Handling cases and the coverage in a limited quantity of memory for case-based planning systems. In: IBERAMIA/SBIA 2000. SICHMAN, J. S.; MONARD, M. C. (Eds.). *Lecture Notes in Artificial Intelligence*, v. 1952, Springer-Verlag, 2000. p. 23-32.

TONIDANDEL, F.; RILLO, M. An accurate adaptation-guided similarity metric for case-based planning. In: INTERNATIONAL CONFERENCE ON CASE-BASED REASONING (ICCBR-2001), 4., 2001. AHA, D.; WATSON, I. (Eds.). *Lecture Notes in Artificial Intelligence*, Springer-Verlag, v. 2080, p. 531-545, 2001a.

TONIDANDEL, F.; RILLO, M. An efficient case-based planning system: preliminary results. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE (SBAI-01), 5., 2001. *Anais*... Canela, 2001b.

TONIDANDEL, F.; RILLO, M. The FAR-OFF system: a heuristic search case-based planning. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE PLANNING AND SCHEDULING (AIPS-2002), 6., 2002. A ser publicado. Toulouse, 2002.

ACTION PLANNING FOR INTELLIGENT MANUFACTURING AUTOMATION

Abstract

This paper investigates the use of the FAR-OFF system in the Manufacturing Automation field. The FAR-OFF system has the feature of heuristic search based systems, which have been presenting excellent results over the last years in the planning area. However, instead of being a generative planning system, the FAR-OFF system is a case-based planner that can guarantee stability to solve problems in a reasonable amount of time. The results presented by its application in the logistic domain show that it is a promising system for intelligent automation.

Key words: intelligent action planning, case-based reasoning, case-based planning.