Sistema de Planejamento Baseado em Casos

Flavio Tonidandel [1] and Márcio Rillo [1,2]

[1] Universidade de São Paulo - Escola Politécnica
Av. Luciano Gualberto 158, trav3
05508-900 - São Paulo - SP - Brasil
[2] Faculdade de Engenharia Industrial
Av. Humberto de A. Castelo Branco, 3972
09850-901 - São Bernardo do Campo - SP - Brasil
e-mails: flavio@lac.usp.br ; rillo@lsi.usp.br
phone: 11 3818 5530

Resumo. Este artigo apresenta, em linhas gerais, o desenvolvimento e a implementação de um sistema de planejamento baseado em casos. Este sistema tem como característica a incorporação de técnicas de manutenção de base de casos e uma nova regra de similaridade, chamada de ADG, que garante precisão na etapa de resgate. Todas as técnicas incorporadas vem no sentido de desenvolver um sistema de planejamento baseado em casos que seja, na maioria das situações, mais eficiente que qualquer sistema de planejamento generativo existente.

Abstract. This paper presents an overview of the development and implementation of a case-based planning system. This system has some incorporated techniques in case-base maintenance and a new similarity rule, called ADG, that guarantees an accurate retrieval phase. All these techniques are used to develop a case-based planning system that, in most situations, is more efficient than any other generative planer.

1. INTRODUÇÃO

Planejamento de ações (*Artificial Intelligence Planning Systems*) tem sido uma das partes centrais da Inteligência Artificial. A idéia que norteia um sistema de planejamento é a de desenvolver um solucionador geral de problemas baseado em um conjunto de ações e em características do mundo no instante inicial do planejamento [1,28]. O primeiro sistema de planejamento foi o STRIPS [1], que utilizava ações para inserir ou remover alguns literais de um conjunto que continha as verdades do estado do mundo.

Entretanto o STRIPS apresentava suas limitações. Algumas dessas limitações foram solucionadas pelo sistema NOAH [1], que introduziu o planejamento por ordem parcial e a detecção de interações entre objetivos. Após isto, diversos outros sistemas aprimoraram as técnicas introduzidas pelo STRIPS e pelo NOAH, como o sistema NONLIN [1] e o sistema SIPE [1].

Embora os avanços fossem significativos à época, os sistemas de planejamento eram direcionados para aplicação em robótica, e, portanto, não lhe faltaram críticas por não serem reativos e considerarem domínios estáticos, não comuns aos domínios encontrados por robôs.

Isto provocou a primeira mudança de paradigma da área. Ao invés de se concentrar no sistema de planejamento como um todo, as pesquisas se voltaram no desenvolvimento de sistemas reativos e na abordagem de cada componente de um sistema de planejamento individualmente.

Este novo paradigma permitiu, entre outros avanços: a introdução de novas descrições de ações (e.g. ADL) , o desenvolvimento e aprimoramento de novas técnicas e teorias (e.g. Teoria Causal), abstração e decomposição de planos, técnicas de planejamento reverso e planejamento baseado em casos. Excelentes bibliografias sobre planejamento e suas técnicas podem ser encontradas em [1,28].

No entanto, mesmo com o avanço de diversas teorias e técnicas, que possibilitaram o melhor entendimento das características fundamentais de planejamento ao longo dos últimos anos [6,8,9,18], não foi possível obter planejadores que tivessem performance computacional eficiente. Isto somente foi possível com o desenvolvimento do planejamento por busca heurística, introduzido pelo sistema Graphplan [3], e posteriormente aprimorado pelos sistemas HSP [4] e FF [10]. Este último, inclusive, foi o mais rápido planejador na competição do AIPS'2000 [2].

O paradigma de planejamento, então, mudou completamente, passando a se concentrar em como utilizar as diversas técnicas desenvolvidas durante anos na área para fornecer maior eficiência aos sistemas baseados em busca heurística.

É exatamente por causa deste paradigma que este artigo se concentra no uso de técnicas de raciocínio baseado em casos [13], de modo que possa prover eficiência aos sistemas de planejamento por busca heurística na maioria das situações e domínios. Sistemas de Planejamento que utilizam o raciocínio baseado em casos para encontrar suas soluções são chamados de sistemas de planejamento baseado em casos (CBP).

Este artigo apresenta, em linhas gerais, as diversas etapas e resultados do projeto, extensão de [25], que visa implementar um sistema de planejamento baseado em casos eficiente. Na seção 2 é apresentada a Lógica de Transações, usada para formalizar os componentes do sistema de planejamento baseado em casos. Este, por sua vez, é especificado na seção 3, que ainda apresenta os avanços teóricos e os resultados obtidos no projeto. A seção 4 conclui o artigo e trata dos trabalhos futuros e dos aperfeiçoamentos que estão realizados.

2. A LÓGICA DE TRANSAÇÕES EM PLANEJAMENTO

A Lógica de Transações (TR) [5] é uma lógica que especifica formalmente atualizações de base de dados, caminhos de execução e procedimentos de transições seqüenciais. A TR é uma extensão da lógica de primeira ordem com a introdução de um novo operador chamado conjunção serial (⊗). Este operador representa uma transação, onde $\alpha \otimes \beta$ significa: "primeiro execute α, e então execute β".

Para descrever uma transação é considerada a seguinte notação: $P,D_0,...,D_n \models \phi$, onde P é chamado de base de transações, que é um conjunto de fórmulas em TR, assim como é ϕ. Cada D_i é um conjunto de fórmulas de primeira ordem chamadas de literais. 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, chamando 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, $P,D_k \models qry(nochão)$.

Devido à todas estas características formais, a semântica da TR se torna equivalente à semântica dos componentes de um sistema de planejamento [24], demonstrando que a TR é uma lógica apropriada para formalização de ações, estados e planos em planejamento. Diversos outros trabalhos já haviam utilizado TR em planejamento [17, 23,26].

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

Definição 1 (Estado) : *O estado D é um conjunto finito de predicados de primeira ordem e é representado em TR como uma base de dados. Cada* $d \in D$ *é denominado de literal.*

Os estados podem ser modificados por ações, as quais podem ser definidas como:

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

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

onde

- $pre(\alpha)$ *é uma fórmula TR que é composta por predicados qry(_) que representam as precondições da ação* α.
- $delete(\alpha)$ *é uma lista de remoção que representa todos os literais que não serão verdade após a execução da ação* α, *i.e., que serão removidos da base de dados.*
- $add(\alpha)$ *é uma lista de adição que representa todos os literais que serão verdade após a execução da ação* α, *i.e., que serão inseridos na base de dados.*

O resultado da execução de uma ação é um estado atualizado *D'* gerado após a remoção e 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, $pre(\alpha) \subseteq D$.. Com as definições de ações e estados é possível, seguindo [25], definir planos com o uso da Lógica de Transações:

Definição 3: (Plano) *Um plano* $\delta = \alpha_1 \otimes ... \otimes \alpha_n$ *é uma fórmula TR, onde* $\alpha_i \in \mathbf{A}$; $1 \leq i \leq n$.

Um plano pode ser vazio, i.e., sem ação em sua composição, e ser 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. Em substituição ao plano retornado pelo planejador, será utilizado uma instância *plan*:

Definição 4: (Instância *plan*) : *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:*

$$\textbf{\textit{Obj}}: plan \otimes Df$$

onde Df é um grupo de consultas que representam as características do estado final desejado.

Desta forma, dado um objetivo, o planejador deve encontrar um plano que irá substituir 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 $Df \subseteq D_k$.

3. PLANEJAMENTO BASEADO EM CASOS

Ao apresentar um objetivo com a estrutura descrita na definição 5, o planejador irá procurar um plano a partir do nada (‘*from scratch’*), ou seja, um plano que leva do estado inicial D_0 ao estado final D_k, onde $Df \subseteq D_k$. Este planejamento é chamado de planejamento generativo.

O planejador generativo sempre inicia seu plano a partir do nada, mesmo para problemas que já tenham sido solucionados anteriormente. Uma forma de evitar que o planejador execute uma busca que já tenha sido executada em algum planejamento anterior é através do uso de técnicas de raciocínio baseado em casos.

O raciocínio baseado em casos (CBR) caracteriza-se por armazenar soluções de diversos problemas em forma de casos para posteriormente escolher um que possa ajudar em alguma nova solução [13]. O CBR possui três etapas distintas: Resgate (*Retriever*), Adaptação (*Adaptation*) e Armazenamento (*Storing*). Estas três etapas são realizadas sequencialmente, onde primeiramente é resgatado, da memória de casos, o caso mais similar ao novo problema a ser solucionado. Depois o caso é modificado pela etapa de adaptação, que tenta adequá-lo à solução do novo problema. O novo caso é então armazenado na memória para uso futuro.

O definição de um sistema de planejamento seguindo estas etapas foi introduzido pelo sistema CHEF [7], e passou a ser conhecido como planejamento baseado em casos (CBP). Após isto, outros sistemas de CBP foram surgindo. Como exemplo apareceu o sistema Prodigy/Analogy [27] que ao invés de armazenar planos como casos, armazenava o processo de busca realizado para encontrar o plano.

Os sistemas de CBP foram se aprimorando, como, por exemplo, o sistema Caper [11] que utilizava-se de processamento paralelo e o sistema MRL [12], que é o único sistema de CBP totalmente formalizado em lógica.

No entanto, embora os resultados apresentados pelos sistemas de CBP fossem sensivelmente melhores e o processamento mais rápido que os sistemas generativos, na

análise de pior caso, ambos sistemas são *NP-hard* [16]. Ou seja, no campo teórico, não há vantagem no uso de CBP em detrimento aos planejadores generativos

Na verdade, sem a inclusão de novas técnicas, os sistemas de CBP, atualmente, não apresentam vantagem nem no campo prático, pois os planejadores generativos melhoraram muito sua eficiência com a introdução de planejadores por busca heurística, como o HSP [4] e o FF [10].

Entretanto, com diversos avanços na área de CBR, como redução do espaço de busca de casos, manutenção da base de casos, entre outros, o uso de CBR em planejamento passa a ser novamente um atrativo para superar os resultados obtidos pelos sistemas generativos e fornecer eficiência aos planejadores por busca heurística.

3.1. Etapa de armazenamento de casos

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:*

$$\eta = Wi \otimes \alpha_1 \otimes .... \otimes \alpha_n \otimes Wf,$$

onde:

- $\alpha_i \subseteq A$; $1 \leq i \leq n$, *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 póscondição do caso.*

Intuitivamente, Wi é um conjunto de consultas para os literais que serão removidos pelo plano e que devem, portanto, estar no estado inicial de modo que permita a execução do plano. Este conjunto é idêntico ao conjunto gerado pelo processo *foot-printing* do sistema Prodigy/Analogy [27]. Já o Wf é o conjunto de consultas que representam os literais que serão inseridos e que estarão, necessariamente, presentes no estado final após a execução do plano.

Cada caso pode ser resgatado, adaptado e utilizado como uma solução para um novo problema. Sendo assim, é possível definir a competência de um caso, que nada mais é do que o espectro de problemas que este caso pode, dentro de um certo limite, solucionar após a adaptação [19]. O limite é imposto pelo esforço que o caso, se resgatado, exigirá do processo de adaptação.

Deste modo, tanto a competência quanto o esforço da adaptação são elementos fundamentais para a manutenção da base de casos. A eficiência de um sistema de CBR está diretamente ligada à qualidade dos casos armazenados e, por isso, a manutenção da base de casos se tornou algo tão importante em CBR que possui um ramo específico de pesquisa, denominado Manutenção de Base de Casos (CBM) [15].

O desafio principal em CBM é o de como diminuir o tamanho da base de casos sem no entanto perder em qualidade. Uma forma é através dos métodos de preservação da competência. Entre eles se destacam as políticas de remoção de casos [22] e de seleção de casos [19, 29].

Outro fator importante que também afeta a eficiência é a capacidade de armazenamento da memória de casos, ou base de casos. Em CBP, os casos variam de tamanho, e podem facilmente ocasionar o enchimento da base de casos. Os métodos de CBM existentes não se preocupam com o espaço na memória de casos, pois consideram que os casos possuem o mesmo tamanho. Quando uma base de casos está cheia, i.e., quando não há mais espaço para

armazenar nenhum caso, alguns casos precisam ser removidos. A remoção de casos é necessária para que a base de casos possa ter sua qualidade melhorada com a inclusão de um novo caso que por ventura possa aumentar tanto sua performance quanto sua competência.
Entretanto, a única política de remoção de casos baseada em competência é o método *baseado-em-tipo* [22]. Este método não apresenta resultados tão satisfatórios quanto os métodos de seleção de casos, como o *máximo-benefício* [29] e o RC-CNN [19].

Os métodos de seleção de casos, por sua vez, não são suficientemente eficientes pois, ao invés de escolher quais casos remover, estes escolhem quais casos deixar e, quando se trata de liberação de espaço na memória, normalmente o número de casos a remover é bem menor do que os que ficarão na base de casos.

Por causa disto, foi desenvolvida uma teoria de remoção de casos que apresenta resultados tão bons quanto as políticas de seleção de casos [19, 29] e com eficiência maior quando se trata de liberação e espaço. A teoria, chamada de *mínimo-prejuízo* [26], ainda garante um limite mínimo para o valor da competência resultante.

O algoritmo de remoção do *mínimo-prejuízo* é um algoritmo guloso (‘*greedy*’) que a cada passo, escolhe o caso que cause menos prejuízo à competência geral da base de casos. O valor do prejuízo é exatamente o número de problemas que o caso soluciona e que nenhum outro caso na base de casos pode solucionar. O algoritmo vai removendo os casos até que o espaço liberado seja suficiente para o armazenamento de um novo caso.

Este método de remoção garante um limite mínimo para a competência. Este limite mínimo também é alcançado pelo método de seleção de casos *máximo-benefício* [29]. Os dois limites são diferentes, sendo o limite mínimo do método *mínimo-prejuízo* maior na extremidade onde há muitos casos restantes e poucos removidos. Esta situação ocorre na liberação de espaço, o que nos leva a concluir que o método proposto é, teoricamente, melhor que o método do *máximo-benefício*.

No entanto, alguns testes foram necessários para verificar que, além do resultado teórico, os resultados práticos também comprovavam a superioridade do método proposto. Isto pode ser verificado no resultado apresentado na figura 1. Esta figura mostra o desempenho médio dos diversos métodos de remoção e seleção de casos aplicados a base de casos. Com o resultado apresentado, é possível concluir que o método proposto apresenta resultados tão bons quanto os métodos de seleção de casos, sendo realizado com maior rapidez quando o número de casos a remover é inferior a 70% do total de casos, o que acontece normalmente quando se trata de liberação de espaço na memória.

Com o método proposto é possível evitar que o enchimento da memória seja um empecilho para o aumento da performance e da competência da base de casos.

3.2. Resgate de casos

Além da manutenção da base de casos, outro fator que impede o aumento da eficiência de um CBP é a etapa de resgate de casos. Percorrer muitos casos ou utilizar métrica de similaridade imprecisa podem tornar o sistema ineficiente. Diversos métodos foram propostos no intuito de reduzir o espaço de busca, fazendo com que somente um sub-conjunto de casos sejam analisados. Para isto, um ótimo método é o proposto por Smyth e McKenna [20], que baseado na competência de cada caso, permite determinar o sub-conjunto de casos onde se encontra o caso mais similar.

Outro problema é a determinação de quais casos são similares ao problema proposto. Qualquer método de redução de base de casos, como qualquer resgate de casos, pode não apresentar resultados satisfatórios se a regra de similaridade não tiver precisão. Entende-se por similar o caso que é mais facilmente adaptado para solucionar um novo problema.

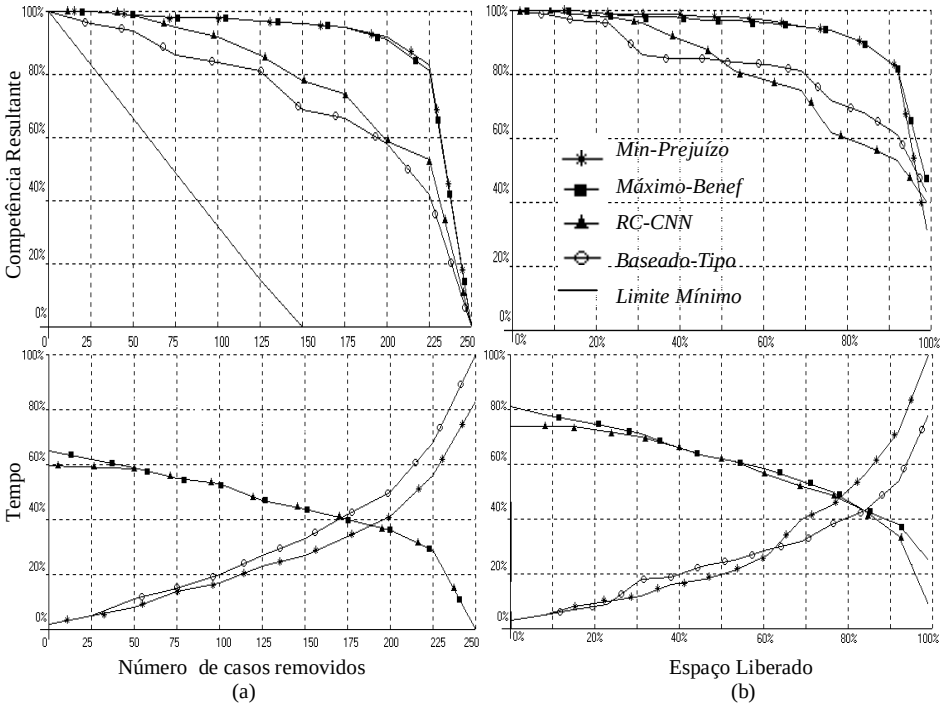

Fig. 1. Média de resultados da competência e do tempo após a remoção de determinado número de casos (a) e após a liberação de uma certa porcentagem de espaço de memória (b) em bases de casos com 250 casos. O tempo é apresentado em porcentagem relativa ao maior tempo gasto durante o experimento.

Diante disso, diversos métodos foram propostos com regras de similaridade baseado no esforço da etapa de adaptação [14, 21]. No entanto, todos estes métodos são dependentes do domínio e incorporam características específicas do domínio ao qual foram propostos.

Foi exatamente devido ao fato das regras de similaridade baseadas em esforço de adaptação existentes não poderem ser aplicadas à sistemas independentes do domínio, que foi proposta uma regra de similaridade precisa, baseada na estimativa da distância entre estados para ser usada nos mais diversos sistemas de CBP.

Esta regra de similaridade, denominada Similaridade Guiada pela Distância-Ação (ADG), surgiu da relação existente 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 o sistemas de planejamento por busca heurística se basearam. Estes calculam uma heurística baseada na distância entre estados para guiar a busca no espaço de estados. Esta distância entre estados é medida pela estimativa do número de ações existentes entre os dois estados.

Com a mesma heurística usada pelo sistema FF [10], 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 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.

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*. Esta heurística constrói um grafo de possibilidades de ações e estados a partir do estado inicial, ignorando a lista de remoção [10]. Depois, para calcular a distância, determina em direção reversa, i.e., do objetivo para o estado inicial, quais ações satisfazem os objetivos em cada ramo do grafo, onde 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 objetivos a serem

cumpridos estiverem no estado inicial, encontrando a estimativa da distância pelo número de ações usadas para satisfazer os objetivos ao longo do processo [10].

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 era possível aplicar diretamente a *FF-heurística* para *Wf* e para *Df* e determinar a diferença entre as distâncias. O que é necessário fazer é forçar que a estimativa considere as ações do caso resgatado.

Isso é feito da seguinte forma. Primeiro executa-se a *FF-heurística* entre o estado inicial e o *Wi*. Depois, utilizando-se das marcações feitas pela primeira estimativa, marca-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 para *Df*, só que as marcas de *verdadeiro* e *falso*, tanto do primeira estimativa, quando as modificadas para *Wi* e *Wf*, farão com que a estimativa seja um resultado muito próximo da distância entre *Wf* e *Df*.

O cálculo da ADG é a soma das duas estimativas, entre o estado inicial e *Wi*, e entre *Wf* e *Df*. Para verificar a precisão da ADG, foi realizado aproximadamente 300 testes com duas base de casos com 100 casos, uma para o domínio do mundo de blocos e outra para o domínio de logística, ambos usados na competição do AIPS'2000 [2].

Cada teste escolhia aleatoriamente dois casos, o estado inicial D_0 e o estado final *Df*. Cada caso era utilizado para solução do problema que era encontrar um plano entre D_0 e *Df*. O caso que requeria menor tempo de adaptação era considerado o mais similar. Depois, foi aplicado nos dois casos a métrica de similaridade ADG, a métrica de similaridade utilizada pelo sistema MRL [12] e a do sistema Prodigy/Analogy [27]. A porcentagem de acerto de cada métrica é mostrado na figura 2, que mostra a precisão encontrada pela métrica ADG.

Tal precisão se faz em função de um tempo maior de processamento que outras métricas. Mas este aumento de tempo não significa uma diminuição na eficiência, pelo contrário, resulta em um ganho de eficiência, pois a precisão da resposta diminui o tempo de adaptação do caso resgatado.

3.3. Adaptação

Após o resgate de um caso similar, deve-se modificar o caso resgatado para que este possa ser a solução do novo problema. Em muitos sistemas de CBP 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 um processo ineficiente.

Uma maneira de tornar o processo de adaptação eficiente é não mexer na estrutura do caso resgatado e torná-lo, em sua forma completa, parte da solução do problema. Para isto, faz-se necessário encontrar um plano entre o estado inicial e o *Wi* do caso e outro entre *Wf* e *Df*.

Na maioria dos sistemas de CBP, a etapa de adaptação usa um planejador generativo, pois se não houver nenhum caso que possa ser resgatado para solucionar um devido problema, o planejador generativo, na etapa de adaptação, poderá encontrar a solução a partir do nada (‘*from scratch’*).

Ao se resgatar um caso η, é então passado a seguinte estrutura de objetivo para a etapa de adaptação:

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

As instâncias *plan* presentes na estrutura acima indicam o local aonde 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*.

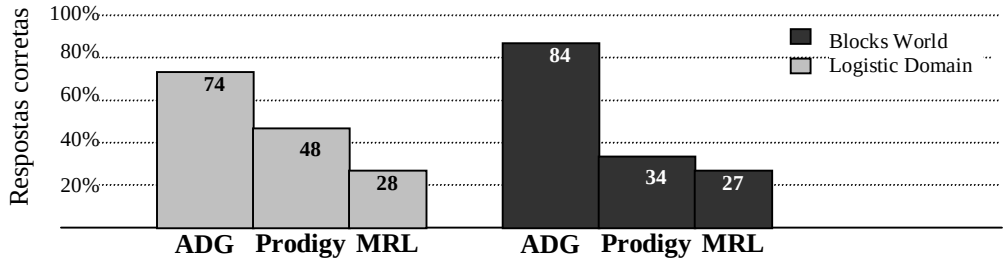

Fig. 2. Porcentagem de acerto das diversas métricas de similaridade

Um outro problema encontrado durante o planejamento é decidir se resgatar um caso é melhor e mais eficiente do que planejar a partir do nada. Uma heurística que tem apresentado bons resultados é comparar o valor da ADG dos casos com o valor estimado da distância entre D_0 e *Df* diretamente. No entanto esta parte ainda precisa ser melhorada por um detalhamento maior, pois deve-se ainda considerar que há o tempo de busca realizado na base de casos.

4 CONCLUSÃO E TRABALHOS FUTUROS

Este artigo apresentou um sistema de planejamento baseado em casos que, devido aos diversos avanços, tanto teóricos quanto práticos, apresentados, é tão ou mais eficiente que os atuais sistemas de planejamento baseados em busca heurística [4, 10].

As etapas do sistema de CBP proposto incorporam melhorias significativas, que permitem não só um aumento na velocidade de planejamento, mas também na qualidade das soluções encontradas. O primeira melhoria foi proposta na etapa de armazenamento, mais precisamente na manutenção da base de casos, que levou a uma política de remoção de casos que opera quando a memória está cheia. Esta política permite maximizar o valor da competência resultante tanto quanto as demais políticas existentes em tempo significativamente menor.

Outra melhoria significativa foi na etapa de resgate, onde a grande barreira para o aumento da eficiência do sistema era a precisão das regras de similaridade existentes para sistemas de CBP. Testes mostraram que a métrica de similaridade proposta, a ADG, apresenta resultados muito mais precisos que as demais métricas. Tudo isso, alinhado com a etapa de adaptação, que, ao invés de modificar os casos resgatados, apenas os completa para formar a solução desejada, nos mostra que é possível melhorar a eficiência dos sistemas de CBP.

No entanto, há ainda diversos aspectos a serem aperfeiçoados, tais como a definição de uma métrica apropriada de competência que englobe não apenas a quantidade de problemas que um caso pode solucionar, mas também o esforço para solução desses problemas. Testes preliminares mostraram que o sistema proposto é, na maioria das situações, mais rápido que os sistemas generativos. No entanto, outros testes ainda devem ser aplicados.

BIBLIOGRAFIA

1. Allen, J; Hendler, J; Tate, A. *Readings in Planning*. Morgan-Kaufmann Publishers, 1990.
2. Bacchus, F. *AIPS-2000 Competition Results*. Available in: http://www.cs.toronto.edu/aips2000/.
3. Blum, A.; Furst, M. *Fast Planning Through Planning Graph Analysis*. Artificial Intelligence, 90(1-2): p.279-298. 1997.
4. Bonet, B; Geffner, H. Planning as Heuristic Search: New Results. In: *Proceedings of European Conference on Planning – ECP'99*. Durham, UK. 1999.
5. Bonner, A.J., Kifer, M.: Transaction logic programming. *Technical Report, CSRI-323*, Department of Computer Science, University of Toronto. 1995.
6. Drabble, B. *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems* – AIPS'96. Edinburgh, AAAI Press, 1996.

7. Hammond, K. *Case-Based Planning:Viewing Planning as a Memory Task*. Academic Press, 1989.
8. Hammond, K. *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems* – AIPS'94. Chicago, AAAI Press, 1994.
9. Hendler, J. *Proceedings of the First International Conference on Artificial Intelligence Planning Systems* – AIPS'92. Maryland, Morgan Kaufmann Publishers, 1992.
10. Hoffman, J. A Heuristic for Domain Independent Planning and its Use in an Enforced Hill-climbing Algorithm. In: *Proceedings of 12th International Symposia on Methodologies for Intelligent Systems*. North Carolina, USA. 2000.
11. Kettler, B. P. *Case-Based Planning with a High-Performance Parallel Memory*. Maryland, 1995. Tese (Doutorado), University of Maryland Park.
12. Koehler, J. Planning from Second Principles. *Artificial Intelligence*, 87. Elsevier. p.148-187, 1996.
13. Kolodner, J. *Case-Based Reasoning*. Morgan Kaufmann Publishers. 1993.
14. Leake, D., Kinley, A., Wilson, D. Case-Based Similarity Assessment: Estimating Adaptability from Experience. In: *Proceedings of 14th National Conference on Artificial Intelligence – AAAI'97*. AAAI Press, 1997.
15. Leake, D.B., Wilson, D.C. Categorizing Case-Base Maintenance: Dimensions and Directions. In: *Smyth, B., Cunningham, P. (Eds.): 4th European Workshop on Case-Based Reasoning EWCBR-98*. Lecture Notes in Artificial Intelligence, Vol 1488, p.196-207. Springer-Verlag, 1998.
16. 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.
17. Santos, M., Rillo, M. Approaching the *Plans are Programs* Paradigm using Transaction Logic. In: *Steel, S., Alami,R (Eds) Proceedings of 4th European Conference on Planning – ECP'97*. Lecture Notes in Artificial Intelligence, vol. 1348, p. 377-389. Springer-Verlag. 1997.
18. Simmons, R.; Veloso, M.; Smith, S. *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems* – AIPS'98. Pittsburgh, AAAI Press, 1998.
19. Smyth, B., McKenna, E. Building Compact Competent Case-Bases. In: *Althouff, K., Bergmann, R., Branting, K. (Eds.) Proceedings of the 3rd International Conference in Case-Based Reasoning. ICCBR'99*. Lecture Notes in Artificial Intelligence, Vol 1650, p.329-342. Springer-Verlag, 1999.
20. Smyth, B., McKenna, E. Footprint-Based Retrieval. In: Althouff, K., Bergmann, R., Branting, K. (Eds.) Proceedings of the 3rd International Conference in Case-Based Reasoning. ICCBR'99. Lecture Notes in Artificial Intelligence, Vol 1650, p.343-357. Springer-Verlag, 1999.
21. Smyth, B., Keane, M. Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. In: *Journal of Artificial Intelligence*, 102(2), p. 249-293. 1998.
22. Smyth, B., Keane, M. Remembering to Forget: A Competence-preserving Case-deletion Policy for Case-based Reasoning Systems. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence IJCAI'95*. p. 377-382. Morgan Kaufmann Publishers,1995.
23. Tonidandel, F, Rillo, M. Case-Based Planning in Transaction Logic Framework. In*: Proceedings of Workshop on Intelligent Manufacturing Systems (IMS'98)*. p. 281-286.Elsevier Science. 1998.
24. Tonidandel, F. ; Rillo, M. Equivalência semântica entre a Lógica de Transações e os Componentes de um Sistema de Planejamento. In: *Anais do 4º. Simpósio Brasileiro de Automação Inteligente – SBAI'99*. São Paulo. 1999.
25. Tonidandel, F. *Planejamento Baseado em Casos usando Lógica de Transações*. Dissertação de Mestrado. Escola Politécnica da USP. São Paulo, 1999.
26. Tonidandel, F, Rillo, M. Handling Cases and the Coverage in a Limited Quantity of Memory for Case-Based Planning Systems. In: *Sichman; Monard (Eds). Proceedings of IBERAMIA/SBIA 2000*. Lecture Notes in Artificial Intelligence, Vol 1952, p. 23-32. Springer-Verlag. 2000.
27. Veloso, M. *Planning and Learning by Analogical Reasoning*. Lecture Notes in Artificial Intelligence, Vol 886. Springer-Verlag, 1994.
28. Yang, Q. *Intelligent Planning: A Decomposition and Abstraction Based Approach*. Springer-Verlag, 1997.
29. Zhu J., Yang Q. Remembering to Add: Competence-preserving Case-Addition Policies for Case-Base Maintenance. In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence IJCAI'99*. Morgan Kaufmann Publishers, 1999.