GAPNet: uma nova abordagem genética para o problema de planejamento em inteligência artificial

Cássio S. Carvalho[1], **Marcos A. Castilho**[1], **Luis A. Künzle**[1], **Fabiano Silva**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19081 – 81531-980 – Curitiba – PR – Brasil

{cassio,marcos,kunzle,fabiano}@inf.ufpr.br

Abstract. *The proposed approach aims at solving a planning problem in Artificial Inteligence using genetic algorithms. The planning problem is translated into a Petri net, which permits identifying all the mutex over actions in an automatic manner. An action classification is adopted considering the involved conflicts, from which is possible to generate a chromossomic codification where each gene position is determinated according to a measure of it's influence across the net.*

Resumo. *A abordagem proposta tem como objetivo solucionar um problema de planejamento em inteligência artificial através de algoritmos genéticos. O problema de planejamento é traduzido em uma rede de Petri, a qual permite identificar, de forma direta sobre sua estrutura, todos os conflitos (*mutex*) entre as ações. Uma classificação dessas ações é feita considerando-se os conflitos envolvidos, o que gera uma codificação cromossômica onde a posição de cada gene está vinculada à importância do mesmo com relação à rede em análise.*

1. Introdução

A solução para um problema de planejamento clássico em inteligência artificial é encontrar uma seqüência de ações que, ao serem executadas em determinada ordem, permite transitar de um estado inicial para um objetivo. O programa que permite resolver essa classe de problemas é dito planejador [Weld 1999].

Diversas técnicas de representação foram consideradas nos últimos anos com relativo sucesso no que se refere ao planejamento clássico. Podemos citar: busca heurística no espaço de estados [Hoffmann and Nebel 2001] [Bonet and Geffner 1998], satisfatibilidade em CNF [Kautz and Selman 1992], redes de Petri (RdP) [Silva et al. 2000], satisfação de restrições, programação inteira, dentre outras.

A computação evolutiva [Mitchell 1998] é uma abordagem pouco explorada na área de planejamento. No contexto dessa área pode-se citar o uso de um de seus principais paradigmas, os algoritmos genéticos [Mitchell 1998]. Um dos poucos trabalhos encontrados que considera o uso de algoritmos genéticos para planejamento é o de Castilho e colegas [Castilho et al. 2004], embora encontremos trabalhos baseados nesta teoria para métodos SAT [Lardeux et al. 2006].

O trabalho descrito em [Castilho et al. 2004] apresentou bons resultados em domínios difíceis como *gripper* e mundo de blocos, onde pode-se observar o potencial da aplicação dos algoritmos genéticos. Contudo o método ali utilizado baseou-se em técnicas

não-clássicas da teoria dos AG's, em particular, os cromossomos não são binários e têm tamanho variável.

Neste trabalho é proposto o algoritmo *GAPNet*, que se baseia na estrutura de redes de Petri, tal como proposta em [Silva et al. 2000], onde se mostra que resolver o problema de planejamento clássico é equivalente a se resolver o problema de alcançabilidade de submarcação na rede de Petri acíclica construída de maneira similar ao grafo de planos.

A diferença principal entre a estrutura do grafo de planos e a rede de Petri está no fato de que os conflitos (mutex) são meta-informação no grafo de planos, enquanto que na rede de Petri eles fazem parte da estrutura. Outra diferença significativa é que o grafo de planos é uma estrutura estática, enquanto que na na rede de Petri existe uma dinâmica associada.

O algoritmo proposto explora o fato de que resolver o problema de alcançabilidade de submarcação na rede de Petri é simplesmente escolher os conflitos corretos que devem ser considerados para se encontrar uma solução correta. Assim, a idéia principal é construir os cromossomos como sendo os conflitos contidos na rede e deixar que o algoritmo genético resolva o problema de forma evolutiva. Para isto, os conflitos devem ser classificados com base no conceito de "influência" na rede.

As próximas seções estão organizadas da seguinte forma: a seção 2 define o problema de planejamento clássico em inteligência artificial; a seção 3 apresenta o formalismo de redes de Petri e as noções que são importantes neste trabalho; a seção 4 descreve a rede de Petri do *Petriplan*; a seção 5 apresenta o planejador *GAPNet*; a seção 6 apresenta os experimentos realizados.

2. Planejamento clássico

Formalmente, pode-se definir um problema de planejamento clássico como sendo a tripla $P =< O, I, G >$. Onde I é o estado inicial, G é o estado objetivo e O é o conjunto de ações existentes para formar um plano. A forma com que essas informações são descritas vem a ser um ponto muito importante. Sendo assim, justamente por sua simplicidade, a representação STRIPS se tornou a base do planejamento em inteligência artificial.

Em STRIPS, enquanto um estado é modelado como um conjunto de literais instanciados, uma ação é definida por três informações: uma lista (*pre*) de pré condições para que a ação seja executada; uma lista (*pos*) de características a serem adicionadas após a execução da ação; e uma lista (*del*) de características a serem removidas após a execução da ação. Formalmente, pode-se definir uma ação como sendo a tripla: $A = (pre, pos, del)$. Define-se que uma ação está habilitada para ser executada se e somente se as suas pré condições estiverem presentes na declaração do estado atual. Posteriormente à execução de uma determinada ação, os literais do estado atual devem ser atualizados de acordo com as remoções e adições listadas na descrição da mesma.

3. Redes de Petri

Uma RdP [Murata 1989] é uma quádrupla $N = (P, T, Pre, Pos)$, onde $P = \{p_1, p_2, \ldots, p_n\}$ é um conjunto finito de lugares, $T = \{t_1, t_2, \ldots, t_m\}$ é um conjunto finito de transições, $Pre : P \times T \rightarrow \mathbb{N}$ é a função de incidência de entrada e $Pos : P \times T \rightarrow \mathbb{N}$ é a função de incidência de saída. Uma rede de Petri com uma dada marcação inicial é denotada por (N, M_0), onde $M_0 : P \rightarrow \mathbb{N}$ é a marcação inicial.

Na representação vetorial (ou matricial) de uma rede de Petri, Pre e Pos são matrizes de n linhas (os lugares) e m colunas (as transições) e seus elementos pertencem a $\mathbb{N}$. O vetor $Pre(.,t)$ denota os arcos de entrada da transição t e seus pesos. O vetor $Pos(.,t)$ denota os arcos de saída da transição t e seus pesos.

A dinâmica de uma rede de Petri é dada pelo *disparo* das transições habilitadas, cuja ocorrência corresponde à mudança de estado do sistema modelado pela rede. Uma transição t de uma rede de Petri N está habilitada por uma marcação M se e somente se $M \geq Pre(.,t)$. Esta condição de habilitação, expressa através de uma desigualdade entre dois vetores, é equivalente a $\forall p \in P,\ M(p) \geq Pre(p,t)$.

Somente transições habilitadas podem ser disparadas. Se M é uma marcação de N que habilita uma transição t e M' é a marcação alcançada pelo disparo de t a partir de M, então $M' = M + Pos(.,t) - Pre(.,t)$. Note que o disparo de uma transição t a partir de uma marcação M alcança uma marcação M': $M \xrightarrow{t} M'$.

É possível generalizar esta fórmula para calcular uma nova marcação após o disparo de uma seqüência s de transições. Seja a matriz $C = Pos - Pre$, chamada *matriz de incidência* da rede de Petri e um vetor $\overline{s}$, chamado de *vetor característico* de uma seqüência de disparo s ($\overline{s} : T \rightarrow \mathbb{N}$, tal que $\overline{s}(t)$ é o número de vezes que a transição t aparece na seqüência s). O número de transições em T define a dimensão do vetor $\overline{s}$. Então, disparando a seqüência s de transições a partir de M, uma nova marcação M_g é calculada pela *equação fundamental* de N:

$$M_g = M + C.\overline{s}. \tag{1}$$

A equação fundamental pode ser usada para determinar um vetor $\overline{s}$ para uma dada rede N e duas marcações M and M_g. Uma solução que satisfaz este problema deve ser um vetor inteiro não negativo e ela é somente uma condição necessária para que M_g seja alcançável a partir de M. Esta condição torna-se necessária e suficiente para redes de Petri *acíclicas*, uma subclasse de redes de Petri que não possuem circuitos dirigidos [Murata 1989].

A relação de alcançabilidade entre marcações pelo disparo de uma transição pode ser estendida, por transitividade, à alcançabilidade pelo disparo de uma seqüência de transições. Portanto, em uma rede de Petri N, considera-se que a marcação M_g é alcançável a partir da marcação M se e somente se existe uma seqüência de transições s tal que: $M \xrightarrow{s} M_g$. O conjunto de alcançabilidade de uma rede de Petri marcada (N, M_0) é o conjunto $\mathcal{R}(N, M_0)$ tal que $(M \in \mathcal{R}(N, M_0)) \Leftrightarrow (\exists s M_0 \xrightarrow{s} M)$.

Denomina-se *problema de alcançabilidade* o problema de responder se existe uma seqüência de disparo s capaz de alcançar uma dada marcação $M_g \in \mathcal{R}(N, M_0)$ a partir de M_0. Um *problema de alcançabilidade de sub-marcação* consiste em responder se existe uma seqüência de disparo s capaz de alcançar um subconjunto de lugares $M_s \subset M_g$, onde $M_g \in \mathcal{R}(N, M_0)$. O problema de alcançabilidade é decidível [Murata 1989], mas não há ainda um algoritmo eficiente para isto.

4. Representação em redes de Petri

Em 2000, Silva [Silva et al. 2000] propôs um planejador chamado *Petriplan* o qual resolve um problema de alcançabilidade em Redes de Petri utilizando programação inteira.

Como a rede de Petri considerada modela um problema de planejamento em inteligência artificial, a solução para o problema de alcançabilidade representa uma solução para o problema de planejamento em questão. Para tornar isso viável foi implementado um procedimento que faz a tradução de um grafo de planos para uma rede de Petri.

Posteriormente, as novas versões do *Petriplan* constróem uma rede de Petri diretamente a partir dos arquivos de descrição em PDDL [Ghallab et al. 1998], sem a necessidade de passar pela representação do grafo de planos.

As características da rede gerada são semelhantes às de um grafo de planos, porém, o formalismo matemático no qual as redes de Petri se baseiam passa a ser um diferencial na análise dinâmica de um problema de planejamento. Conforme visto na seção 3, a transição entre estados (marcações) de um sistema pode ser facilmente executada a partir de cálculos matriciais, conforme a equação 1.

Como as características da rede se repetem a cada 4 camadas, será feita uma análise levando-se em conta somente as 4 primeiras (L_0, L_1, L_2, L_3). L_0 é uma camada de lugares e representa as proposições existentes em um estado, ao passo que a L_2 também é de lugares porém é constituída por várias instâncias das mesmas proposições de L_0. Por exemplo: imagine que em L_0 exista p, e que existam três ações que necessitam de p para serem executadas. Sendo assim, em L_2 haverá três réplicas de p. L_1, que é de transições, é responsável por fazer as cópias necessárias de cada proposição existente em L_0, gerando-as em L_2. Por último L_3, que também é de transições, é a camada das ações disponíveis para um determinado estado.

A dificuldade para se encontrar um plano nessa rede é a existência de conflitos entre ações de uma mesma camada. Enquanto no grafo de planos os mesmos são representados por ligações entre as respectivas ações conflitantes (metainformação), já na rede de Petri gerada, um conflito faz parte da estrutura da rede e é representado por um lugar p com as seguintes características:

- $M(p) = 1$;
- $\forall t \in T, Pre(p, t) = 0$
- Existem exatamente duas transições $t \in T$ onde $Pos(p, t) = 1$. Para todas as outras transições $t \in T$, $Pos(p, t) = 0$.

Nessa rede há dois tipos de *mutex* (conflitos): *mutex* de proposição (literal) e *mutex* de ação. Uma vez considerados os *mutex* de ação, não é necessário preocupar-se com os de proposição, pois esses são derivados a partir dos primeiros [Blum and Furst 1995].

Conhecidas as características da rede de Petri gerada, é interessante perceber onde está a vantagem de se obter uma estrutura como esta. Suponha que uma nova versão da RdP seja adaptada para modelar problemas de planejamento não clássico, incorporando, por exemplo, o uso de recursos e também características temporais. Nesse momento, um sistema planejador o qual resolve apenas problemas clássicos poderá ser facilmente reutilizado ou adaptado para solucionar problemas não clássicos, visto que continuará tendo que tratar um problema de alcançabilidade em redes de Petri. O principal motivo de propor um sistema planejador baseado nesta classe é justamente se beneficiar com a característica mencionada.

5. *GAPNet*

Ao analisar o planejador *AGPlan* baseado em algoritmos genéticos [Castilho et al. 2004], nota-se o potencial do paradigma considerado. Tendo como referência este trabalho, o planejador *GAPNet* (*Genetic Algorithm for Petri Nets*) propõe solucionar um problema de alcançabilidade em redes de Petri incluindo características importantes e não consideradas no trabalho citado.

O *AGPlan* usa o grafo de planos [Blum and Furst 1995] para gerar sua população inicial, impedindo uma codificação binária de seus cromossomos e determinando que os mesmos sejam de tamanhos variados. O *GAPNet*, por outro lado, utiliza uma codificação binária vinculada a um conjunto de transições conflitantes da rede e realiza uma classificação das ações envolvidas, caracterizando-as pela influência de cada uma perante a obtenção dos objetivos. Adicionalmente, essa codificação possibilita o uso de cromossomos de tamanhos fixos.

O *GAPNet* parte de um problema modelado por uma rede de Petri conforme vista na seção 4 e aplica uma técnica evolucionária para tratar os conflitos existentes na mesma. Esse tratamento permite resolver um problema de alcançabilidade na rede em análise, o que significa solucionar o problema de planejamento por ela representado. Os genes do cromossomo estão associados à execução (1) ou não (0) de uma transição conflitante. A disposição dessas transições no cromossomo é definida como segue.

Dada uma instância de rede de Petri gerada pelo *Petriplan*, onde um conflito entre transições é representado por $p = (t, t')$, obtém-se um conjunto de conflitos $M = \{(t_1, t_2) \in M \mid \exists p = (t, t')\}$. Neste conjunto são incluídos apenas os conflitos presentes nas camadas de ações (L_3, conforme seção 4).

A importância de uma transição t está associada ao número de outras transições cuja execução depende de t. Uma classificação é feita construindo um grafo G não dirigido onde: cada nodo representa uma transição distinta do conjunto M; cada aresta representa um conflito entre duas transições. Cada componente conexa em G representa uma camada de transições da rede e a importância de uma transição t perante o conjunto M é encontrada a partir de G e de um parâmetro $prof$, conforme descrito no algoritmo 1.

Algoritmo 1 Cálculo de importância de transições

1: A partir de G, selecione a componente conexa G' na qual t está presente;
2: Retorne o número de transições distintas alcançáveis a partir de t, percorrendo no máximo $prof$ arestas;

Calcula-se a importância de cada transição t, desde que t não represente uma ação de manutenção. Calculadas essas importâncias, classifica-se as transições considerando suas importâncias e a camada a qual cada uma pertence. Para isso organiza-se as mesmas de forma que as primeiras transições sejam as presentes nas camadas iniciais e, as últimas, as presentes nas camadas finais. Dentro de cada camada, ordena-se as transições em ordem crescente do valor de importância. Por fim, mapeia-se a ordem obtida de transições para cada gene do cromossomo.

Essa classificação diferencia as transições de acordo com a dificuldade de satisfazer o conjunto de conflitos. Dessa forma espera-se que seja mais vantajoso resolver primeiro os conflitos que envolvem transições de alta prioridade. Quanto às transições

relacionadas à ações de manutenção, é mantida uma cópia das mesmas e de seus respectivos conflitos. Essas informações serão utilizadas pelo algoritmo de avaliação de soluções candidatas. Como a codificação considera as transições da camada L_2, que representam ações do domínio analisado, as mesmas serão referenciadas pelo termo "ação".

A população inicial é gerada definindo-se quantas e quais ações serão executadas. Para tal, define-se o menor número possível de ações executáveis como sendo o número de componentes conexas do grafo G e o maior número possível de ações executáveis como sendo o tamanho do cromossomo. Gera-se aleatoriamente um número n entre o limite inferior e o superior. Por último, marca-se n ações executáveis no cromossomo, selecionadas aleatoriamente. As seções 5.1 e 5.2 apresentam, respectivamente, a função de aptidão e os operadores genéticos propostos. A seção 5.3 descreve o processo evolutivo.

5.1. Função de aptidão

A função de aptidão é dividida em duas sub-funções: verificação da consistência de um cromossomo e dificuldade para obtenção de objetivos. A verificação da consistência analisa se os valores associados aos genes de um cromossomo satisfazem a relação de conflitos do conjunto M. Considerando que o conjunto M possui m conflitos, esta função retorna um valor entre zero e m, indicando quantos conflitos foram satisfeitos. A dificuldade para obtenção de objetivos identifica barreiras para alcançar o estado final. Este procedimento utiliza as equações matriciais do formalismo de redes de Petri e disparará as ações marcadas no cromossomo. Após, uma busca nos lugares dessa rede permite identificar falhas no disparo de transições. O valor de retorno dessa função está entre zero e o número de lugares existentes na rede.

Algoritmo 2 Uso da rede de Petri na avaliação de uma solução candidata

```
1: Marque a rede N de acordo com sua marcação (estado) inicial; {Não esquecer que, assim como descrito na seção 4, todos os
   mutex apresentam também marcação inicial igual a um.}
2: Crie uma lista LC contendo uma identificação para todas as camadas de padrão L1 e L3, organizadas na ordem na qual aparecem
   na rede.
3: Ma = ações de manutenção; {O vetor Ma contém as ações de manutenção que são conflitantes}
4: Crie uma lista vazia usedactions;
5: for all C | C ∈ LC do
6:     if (C é do padrão L1) then
7:         for all t | t ∈ C do
8:             if (t é executável) then
9:                 marque t como "1" no vetor característico;
10:            else
11:                marque t como "0" no vetor característico;
12:    else
13:        for all t | t ∈ C do
14:            if t ∈ S then
15:                marque a executabilidade de t de acordo com o valor encontrado no cromossomo S;
16:                caso tenha sido marcada como executável, adicione t em usedactions;
17:            else
18:                if (t ∋ Ma) then
19:                    if (t é executável) then
20:                        marque t como "1" no vetor característico;
21:                    else
22:                        marque t como "0" no vetor característico;
23:        for all t | t ∈ Ma do
24:            if ((t é executável) and (não conflitante(t,usedactions))) then
25:                marque t como "1" no vetor característico;
26:                insira t em usedactions;
27:            else
28:                marque t como "0" no vetor característico;
29:    Atualize a marcação da rede de Petri utilizando o vetor característico;
```

Evoluir o conjunto de soluções candidatas significa maximizar a consistência do cromossomo e minimizar a dificuldade de obtenção de objetivos. Uma solução obrigatoriamente deverá conter um valor de dificuldade igual a zero, o que só é possível com consistência máxima. Uma vez que o cálculo de consistência do cromossomo é simples, será detalhado somente o cálculo do valor da dificuldade de obtenção de objetivos. Para esse cálculo, estão disponíveis: uma relação de quais ações devem e não devem ser executadas, obtida a partir da valoração do cromossomo; uma instância da rede de Petri a qual modela o problema de planejamento em questão; uma cópia das ações de manutenção conflitantes, com seus respectivos conflitos.

As transições consideradas na codificação do cromossomo são aquelas que, além de pertencerem à camada L_3, possuem pelo menos um conflito e não são de manutenção. Sendo assim, existe um grupo de ações que nunca aparecem no cromomossomo. Esse grupo é formado pelas transições: pertencentes à camada L_1; pertencentes à camada L_3 e que são livres de conflitos; pertencentes à camada L_3 que possuem conflitos porém estão associadas à ações de manutenção.

O algoritmo 2 descreve como uma rede de Petri N é usada na avaliação de uma solução candidata S. A partir da marcação M' obtida pela execução deste algoritmo faz-se: $M_r = M' - M_f$, onde M_f é a marcação objetivo. Caso nenhum lugar de M_r apresente marcação negativa, um plano foi encontrado. A função de aptidão, portanto, apresenta três variações: 1) número de lugares satisfeitos na rede de Petri; 2) relação entre conflitos satisfeitos e lugares não satisfeitos na rede; 3) número de lugares satisfeitos na rede considerando uma ponderação de acordo com as camadas da mesma.

5.2. Operadores Genéticos

Além dos operadores clássicos de mutação e cruzamento são propostos dois novos operadores genéticos. O primeiro é o de mutação baseado em conflitos: para cada indivíduo selecionado aleatoriamente seleciona-se também de forma aleatória um conflito do conjunto M, analisando e modificando as ações envolvidas da seguinte forma: caso as duas ações estejam habilitadas para execução, desabilite aleatoriamente uma delas; caso nenhuma delas esteja habilitada, habilite aleatoriamente uma delas; caso uma esteja habilita e outra não, troque a execução de uma pela outra.

O segundo operador proposto é o de reorganização de ações. Analisando as características da rede de Petri considerada, verifica-se a existência de diversas transições que representam uma mesma ação. Visto que a codificação utilizada neste planejador abstrai tal detalhe, é possível executar a ação correta em um ponto inadequado do plano (uso de uma transição errada). Portanto, esse operador modifica a posição de execução de uma ação na rede, procurando outra transição que represente a mesma ação porém em outra camada. Este método prioriza a escolha de transições presentes nas camadas iniciais da rede, conforme descrito no algoritmo 3.

5.3. Seleção Natural

A seleção por torneio elege os indivíduos que serão submetidos ao operador de cruzamento. Para selecionar k indivíduos é necessário executar k torneios de tamanho t cada um, conforme mostra o algoritmo 4. A atualização da população consiste em preservar os melhores indivíduos da geração atual e inserir os melhores indivíduos gerados pelos

Algoritmo 3 Reorganização de ações no cromossomo

1: Salve em C todas as transições marcadas como "1" no cromossomo original;
2: Zere todo o cromossomo original;
3: Ordene as transições de C de acordo com seu aparececimento na rede de Petri. Transições da primeira camada, da segunda e ainda por diante;
4: **for all** $t \mid t \in C$ **do**
5: identifique em cada camada da rede uma transição correspondente a t, ou seja, aquela que representa a mesma ação que t;
6: selecione a primeira transição que não esteja em conflito com as já marcadas no cromossomo original;
7: marque esta transição como "1" no cromossomo original;
8: Identifique as camadas da rede onde nenhuma transição está marcada no cromossomo original;
9: Marque aleatoriamente no cromossomo original algumas transições pertencentes às camadas identificadas pelo passo anterior.

Problema	***GAPNet***	***AGPlan***	***Graphplan***	***Blackbox***	***FFPlan***
prob-blocos-sussman	0.12	0.02	0.02	0	0.01
prob-blocos-04	8.91	0.2	0.17	0.01	0.08
prob-blocos-05	35.65	0.3	0.42	0.01	0.14
prob-blocos-06	12.31	0.12	0.17	0.01	0.05
prob-blocos-07	60.25	0.12	0.36	0.01	0.09
prob-blocos-08	206.54	1.5	0.8	0.02	0.16
prob-blocos-09	589.2	1.8	1.5	0.03	0.26
prob-blocos-10	11035	1.7	2.84	0.04	0.41
prob-blocos-11	38082	2.0	4.75	0.06	0.65
prob-blocos-12	114351	1.9	8.0	0.08	0.95
prob-logistics-02	0.04	0	0	0	0.01
prob-logistics-03	0.63	0	0.01	0	0.02
prob-logistics-04	27.45	0.01	0.06	0	0.19
prob-logistics-05	0.12	0	0.02	0	0.04
prob-logistics-06	38.37	0.03	0.1	0	0.13
prob-logistics-07	-	0.09	0.2	0	0.98
prob-logistics-08	-	0.8	0.33	0	1.61
prob-mystery-00	0.12	0.14	0.04	0	-
prob-mystery-01	24.09	0.18	-	0.01	0.19
prob-mystery-02	-	2.8	-	1.0	-

Tabela 1. Comparativo (em segundos) - mundo dos blocos, *logistics*, *mystery*

operadores genéticos. O critério de parada do processo evolutivo é encontrar um plano ou atingir um número máximo de gerações. O melhor indivíduo é monitorado em cada geração e, caso o mesmo atinja um determinado número de gerações sem evolução, novos indivíduos são inseridos na população.

Algoritmo 4 Seleção por torneio

1: Crie uma lista L, vazia;
2: **for** $S = 1$ to k **do**
3: Selecione aleatoriamente t indivíduos;
4: Adicione o indivíduo de melhor *aptidão* a lista L;
5: **return** L;

6. Experimentos

Para avaliar o desempenho do planejador *GAPNet* foram propostos problemas nos domínios mundo dos blocos, *logistics* e *mystery*. Os resultados foram analisados com relação aos seguintes aspectos: tempo de solução dos problemas; número de gerações processadas; percentual de execuções bem sucedidas; gráfico de evolução da população.

A tabela 1 apresenta um comparativo de desempenho entre o *GAPNet* e outros planejadores: *AGPlan* é o planejador baseado em algoritmos genéticos que usa como referência o grafo de planos [Castilho et al. 2004]; o *Graphplan* faz uma busca exaustiva no grafo de planos; o *Blackbox* é uma combinação entre *SatPlan* e o *Graphplan*;

Planejador	Problema	Nº de Execuções	Resolvidos	Percentual
gapnet	prob-blocos-sussman	52	52	100%
gapnet	prob-blocos-04	88	88	100%
gapnet	prob-blocos-05	88	87	98.9%
gapnet	prob-blocos-06	88	85	96.6%
gapnet	prob-blocos-07	70	66	94.3%
gapnet	prob-blocos-08	57	46	80.7%
gapnet	prob-blocos-09	47	30	63.8%
gapnet	prob-blocos-10	36	18	50%
gapnet	prob-blocos-11	15	4	26.7%
gapnet	prob-blocos-12	17	4	23.5%
gapnet	prob-logistics-02	86	86	100%
gapnet	prob-logistics-03	74	64	86.5%
gapnet	prob-logistics-04	56	30	53.4%
gapnet	prob-logistics-05	67	67	100%
gapnet	prob-logistics-06	30	28	93.3%
gapnet	prob-logistics-07	9	0	0.%
gapnet	prob-logistics-08	9	0	0.%
gapnet	prob-mystery-00	39	39	100%
gapnet	prob-mystery-01	40	39	97.5%
gapnet	prob-mystery-02	10	0	0.0%

Tabela 2. Para cada problema, a coluna "percentual" representa uma relação entre execuções bem sucedidas (resolvidos) e o total de execuções.

FFPlan é uma implementação simplificada do *FF*, sem a busca local. Com exceção do *Blackbox*, todos os demais planejadores são versões implementadas no ambiente IPE [Marynowski 2004].

A tabela 2 apresenta um percentual de sucesso ($100 * (resolvidos/execucoes)$) para cada problema considerando todas as parametrizações do planejador proposto, o que significa que para alguns parâmetros em específico o desempenho é superior. Um exemplo disso é quanto à função de aptidão, onde foram testadas as três alternativas descritas na seção 5.1. Ao considerar apenas as opções 2 e 3 da função de aptidão, o percentual de sucesso obtido para os últimos problemas fica acima do apresentado na tabela 2.

Dentre os problemas tratados, no domínio mundo de blocos o maior deles é modelado por uma rede de Petri com 25472 lugares, 2477 transições e 10182 conflitos. No domínio *mystery* o maior contém 16474 lugares, 1104 transições e 8469 conflitos. No domínio *logistics* 3448 lugares, 777 transições e 1667 conflitos. Enquanto que o número de conflitos está relacionado ao tamanho do cromossomo obtido, as dimensões da rede de Petri caracterizam o custo de processamento envolvido no algoritmo 2.

7. Conclusões

Este trabalho apresentou uma abordagem evolutiva com a finalidade de resolver problemas de planejamento em inteligência artificial. O algoritmo proposto tem como princípio tratar um conjunto de transições conflitantes em uma rede de Petri, buscando encontrar uma correta seqüência de disparos na rede em análise.

A função de aptidão implementada realiza um caminhamento na rede de Petri em questão de acordo com a valoração encontrada para as ações conflitantes no cromossomo. Adicionalmente, também foram propostos dois novos operadores genéticos: operador de mutação baseado em conflitos e operador de reorganização de ações.

Propôs-se também um classificação de ações, criando-se uma noção de influência entre as mesmas de acordo com os conflitos envolvidos. Este processo permite diferenciar

as ações pelo grau de interferência que cada uma exerce sobre a rede de Petri, identificando quais delas são potencialmente mais importantes para a obtenção de objetivos.

Embora para muitos problemas uma solução tenha sido encontrada com poucas gerações, o tempo de processamento obtido pelo planejador foi bem elevado em relação aos demais analisados. Numa comparação da presente proposta com a do algoritmo *AG-Plan* [Castilho et al. 2004] observa-se que o número de gerações é, muitas vezes, equivalente. O estrangulamento existente no *GAPNet* são os sucessivos cálculos matriciais exigidos ao atualizar a marcação da rede de Petri durante a avaliação de cada solução candidata.

Por outro lado, o principal objetivo ao desenvolver este sistema planejador é poder usufruir das vantagens das redes geradas pelo *Petriplan*. Disponibilizar um algoritmo que permita resolver um problema de alcançabilidade nessas redes é importante, visto que o mesmo pode ser reutilizado ou adaptado para tratar problemas de planejamento não clássico, tais como aqueles envolvendo tempo e recursos.

Referências

Blum, A. L. and Furst, M. L. (1995). Fast planning through planning graph analysis. In *Proc. of the 14th IJCAI*, pages 1636–1642, Montreal, Canada.

Bonet, B. and Geffner, H. (1998). HSP: Planning as heuristic search. In *Entry at the AIPS-98 Planning Competition*, Pittsburgh.

Castilho, M. A., Künzle, L. A., Lecheta, E., Palodeto, V., and Silva, F. (2004). An investigation on genetic algorithms for generic strips planning. In *IBERAMIA*, pages 185–194.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL—the planning domain definition language.

Hoffmann, J. and Nebel, B. (2001). The ff planning system: Fast planning generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Kautz, H. and Selman, B. (1992). Planning as satisfiability. In *ECAI '92: Proceedings of the 10th European conference on Artificial intelligence*, pages 359–363, New York, NY, USA. John Wiley & Sons, Inc.

Lardeux, F., Saubion, F., and Hao, J.-K. (2006). Gasat: a genetic local search algorithm for the satisfiability problem. *Evol. Comput.*, 14(2):223–253.

Marynowski, J. E. (2004). Ambiente de planejamento ipê. Master's thesis, Universidade Federal do Paraná, Curitiba PR, Brasil.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Silva, F., Castilho, M., and Künzle, L. (2000). Petriplan: a new algorithm for plan generation (preliminary report). In *Lecture Notes in Artificial Intelligence*, volume 1952, pages 86–95. International Joint Conference IBERAMIA'2000 - SBIA'2000.

Weld, D. S. (1999). Recent advances in ai planning. *AI Magazine*, 20(2):93–123.