Uma Abordagem de Planejamento de Recursos para Jogos do Tipo RTS Baseada em Planejamento de Ordem Parcial

Augusto A. B. Branquinho e Carlos R. Lopes *

Faculdade de Computação - Universidade Federal de Uberlândia (UFU)
Av. João Naves de Avila, 2121. CEP 38.408-100 - Uberlândia - MG - Brasil.
augusto_b@pos.facom.ufu.br
crlopes@facom.ufu.br

Abstract. In general, a real-time strategy game is characterized by two phases. First, it is necessary to produce (gather) resources. The next phase is related to battles taking into account the resources that were produced. The resource production phase is a key factor to succeed. In this paper the authors propose a mechanism for resource production based on artificial intelligence planning using means-end analysis and scheduling. A partial order planning algorithm was developed to satisfy the resource goals and priorities were assigned to its actions in order to achieve better schedulings. This is an important feature because a player must make a decision under time constraints. Results show that our system presents a better performance compared to related approaches.

Key words: Games; Real-time; Planning; Resources.

1 Introdução

Os jogos de estratégia em tempo real (ou jogos RTS, um acrônimo para *Real-time Strategy*) é uma categoria muito apreciada de jogos de entretenimento. *Warcraft* e *Starcraft* são exemplos de jogos muito conhecidos que se enquadram nesta categoria. Normalmente são gêneros caracterizados por serem de guerra e o objetivo é alcançar um domínio militar ou territorial sobre o adversário. O papel do adversário pode ser desempenhado por outro jogador ou pelo computador. As decisões são destinadas para produção de recursos e táticas de batalha, com o objetivo de superar os oponentes. Para isso, é preciso um conjunto de ações, como coleta de recursos, construção de edifícios, treinamento de unidades, desenvolvimento de tecnologia e combate com o exército inimigo. Para todas as ações, é necessário decidir o melhor momento e a ordem que serão executadas. Além disso, o mapa inicial é desconhecido e com o decorrer do jogo pode sofrer alterações.

* Os autores gostariam de agradecer o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) e ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Uberlândia (Facom - UFU).

Normalmente, estes jogos envolvem um período inicial em que o jogador deve rapidamente construir sua economia através da produção de recursos, seguida pela campanha militar em que esses recursos são explorados para as ações ofensivas e defensivas. Assim, vencer a corrida na produção de recursos é muitas vezes um fator chave para o sucesso.

Os recursos dos jogos RTS incluem todas as matérias-primas, construções, civilização e unidades militares. Para construção de armamentos e edifícios, além de necessitar de matéria-prima, é preciso determinar a ordem e a quantidade que serão produzidos. Para essa tarefa Chan [1] utiliza de análise de meios-fim (*Means-end Analysis* - MEA) e escalonamento. A técnica MEA consiste em uma estratégia de busca para solução de problemas e foi introduzido em *General Problem Solver* (GPS) [5] e usado no planejador STRIPS [3].

A partir destas considerações, este trabalho pretende melhorar os resultados de [1] obtendo um menor *makespan* (tempo total do plano gerado). Ênfase é destinada ao processo de produção de recursos, sendo proposto um planejamento de ordem parcial juntamente de um processo que define prioridade para as ações serem executadas. A prioridade em executar ações permite que durante o escalonamento os recursos produzidos sejam aproveitados para execução das próximas ações. Este plano parcial é usado como entrada para o algoritmo de escalonamento de Chan [1].

Em síntese, o objetivo do trabalho é desenvolver um mecanismo de seleção de ações que consiga satisfazer rapidamente a uma meta de produção de recursos com o menor tempo possível. Este trabalho pode ser útil, por exemplo, como uma opção de interface para jogadores humanos, em que o jogador necessita especificar somente as metas que deseja atingir. Além disto, constitui-se numa área interessante de planejamento pois apresenta uma série de aspectos desafiadores tais como ações caracterizadas por tempo de duração e efeitos numéricos, concorrência de ações e sua execução num ambiente altamente dinâmico.

As demais seções estão estruturadas da seguinte forma: a seção 2 apresenta o problema que vai ser tratado e uma breve descrição do planejamento de Chan [1]. A seção 3 descreve a arquitetura de planejamento. A Seção 4 apresenta um procedimento que determina as ações mais importantes dos planos criados na Seção 3. Na seção 5 são discutidos os resultados. Considerações finais a respeito do trabalho desenvolvido são feitas na seção 6.

2 Caracterização do Problema

O problema de planejamento em jogos do tipo RTS consiste em encontrar uma sequência de ações que satisfaça um conjunto de metas dentro de um limite de tempo. A restrição de tempo é devido à necessidade de se executar ações durante o decorrer do jogo, sendo que normalmente o ambiente não é totalmente conhecido. Para este planejamento os componentes chave são os recursos e ações.

Os recursos a serem empregados enquadram-se em uma das seguintes categorias: *Require*, *Borrow*, *Consume* e *Produce*. *Require* é um recurso necessário para execução de uma ação, não sendo consumido nem ocupado. *Borrow* é um

recurso necessário que não é consumido, porém, ocupado até que seja finalizada a ação. *Consume* é algo necessário, sendo consumido no início da ação. *Produce* é o resultado, sendo "adicionado" ao ambiente no término da ação.

As ações de produção de recursos possuem um tempo de duração e um conjunto de pré-condições. As pré-condições referem-se a recursos necessários para a execução da ação, os quais podem ser associados aos seguintes tipos: *Require*, *Borrow* e *Consume*. Já os efeitos de uma ação especificam os recursos produzidos. A representação das ações e recursos usados no contexto deste trabalho é exibida na Figura 1. Tais ações encontram-se presentes no jogo Warcraft II.

```
resource townhall
resource peasant
resource barracks
resource supply
resource footman
resource gold
resource wood

action build-townhall :duration 1530
  :borrow 1 peasant :consume 1200 gold 800 wood
  :produce 1 townhall
action build-peasant :duration 225
  :borrow 1 townhall :consume 400 gold 1 supply
  :produce 1 peasant
action build-barracks :duration 1240
  :borrow 1 peasant :consume 700 gold 450 wood
  :produce 1 barracks
action build-supply :duration 620
  :borrow 1 peasant :consume 500 gold 250 wood
  :produce 4 supply
action build-footman :duration 200
  :borrow 1 barracks :consume 600 gold 1 supply
  :produce 1 footman
action collect-gold :duration 510
  :require 1 townhall :borrow 1 peasant
  :produce 100 gold
action collect-wood :duration 1570
  :require 1 townhall :borrow 1 peasant
  :produce 100 wood
```

Figura 1. Especificação das ações e recursos [2].

Observou-se que o planejamento especificado em [1] não encontra a melhor solução quando um recurso produzido não é aproveitado para a execução das ações restantes. O quanto antes for possível escalonar uma ação *build-peasant* para satisfazer uma meta, melhor pode ser o plano, já que isso permite que o *peasant* criado possa ser usado nas próximas ações. Nas Figuras 2 e 3 são apresentados alguns planos sequenciais e escalonados demonstrando este problema.

Na parte superior das Figuras 2 e 3 são apresentados os tempos de duração dos planos, sendo o início no instante 0. Os valores subsequentes são referentes ao tempo de início e término das ações. São definidos 4 planos (a, b, c, d), tendo como estado inicial (*1 Townhall, 1 Peasant, 2 Supply, 600 Gold*) e meta (*1 Townhall, 3 Peasant*).

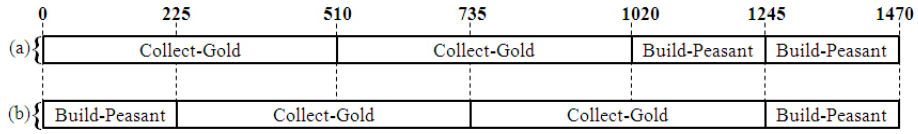

Figura 2. Dois planos sequenciais considerando o estado inicial (*1 Townhall, 1 Peasant, 2 Supply, 600 Gold*) e a meta (*1 Townhall, 3 Peasant*). Na parte superior são exibidos os tempos no decorrer dos planos. (a) Plano sequencial gerado pelo planejamento sequencial de [1]. (b) Plano sequencial gerado pelos métodos de planejamento de ordem parcial e definição de prioridade propostos neste trabalho.

Para gerar o plano *a* da Figura 2 foi usado o planejador sequencial de Chan [1]. Tal plano sequencial é usado como entrada para o escalonamento deste mesmo autor, tendo como resultado o plano *c* da Figura 3, com *makespan* igual a 1245. É fácil observar que um melhor plano pode ser encontrado se uma ação *build-peasant* for executada primeiro, deste modo seu recurso *peasant* pode ser usado para o escalonamento das próximas ações. O problema que ocorre no plano *c* é que as ações *collect-gold* estão no início, sendo escalonadas antes de *build-peasant*.

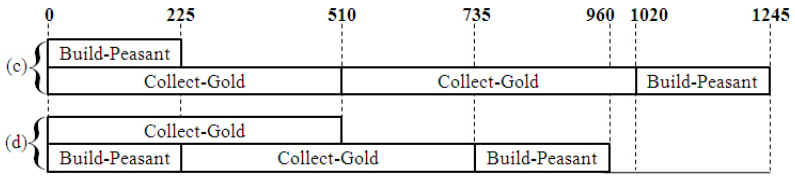

Figura 3. Dois planos escalonados, ou seja, com ações em paralelo. Na parte superior são exibidos os tempos no decorrer dos planos. (c) Escanolamento do plano sequencial *a* da Figura 2 utilizando o algoritmo de escalonamento proposto por [1]. (d) Escanolamento do plano sequencial *b* da Figura 2 utilizando o algoritmo de escalonamento proposto por [1].

O plano *b* da Figura 2 apresenta brevemente a proposta deste trabalho, de priorizar ações mais importantes que possam disponibilizar recursos para o escalonamento das próximas ações. Para criar este plano sequencial é usado um planejamento de ordem parcial e um algoritmo de prioridade, de modo que seja possível priorizar as ações mais importantes e disponibilizar uma sequência de ações para o algoritmo de escalonamento de [1].

O plano *b* quando escalonado pelo algoritmo de escalonamento de [1] tem como resultado o plano *d* com *makespan* igual a 960. Com isso, o plano *d* apresenta um melhor *makespan* que *c*. De modo geral os planos apresentam melhores resultados ao se priorizar as ações. A seguir, na seção 3 é descrito com detalhes o planejamento proposto neste trabalho.

3 Arquitetura de Planejamento

Para gerar um plano, onde a ordem de todas as ações são específicadas completamente, inicialmente é proposto um planejamento de ordem parcial. O planejamento de ordem parcial especifica todas as ações que devem ser executadas, mas não determina uma ordem específica para a execução das mesmas. As ações são escolhidas conforme a meta em relação ao estado atual. Segundo [6], o planejamento de ordem parcial consiste em "qualquer algoritmo de planejamento que possa inserir duas ações em um plano sem especificar qual delas deve ser executada primeiro".

Como descrito na seção 2 e citado por [1] [2], o planejamento para jogos do tipo RTS possui algumas propriedades específicas. Apesar da possibilidade de utilizar outros planejadores, um simples planejamento usando MEA é suficiente para essa tarefa. Sendo assim, foi desenvolvido um algoritmo de planejamento de ordem parcial chamado *MeaPop*. Devido a restrição de espaço deste artigo, apenas os algoritmos mais importantes são apresentados com pseudocódigos.

Algoritmo de planejamento de ordem parcial.

```
var
  R := Vazio; {Lista de recursos.}
  S := Estado inicial de recursos;
procedure MeaPop (G)
  {Meta G que deve ser satisfeita.}
  var
    satisfazG: Boolean;
    s_i, r_i, g_i : Integer;
  begin
    satisfazG := true;
    for all recurso R_i do
      s_i := quantidade de R_i em S;
      r_i := quantidade de R_i em R;
      g_i := quantidade de R_i em G;
      AlocarRecurso(s_i, g_i, R_i);
      succ(R, G);
      if ((s_i + r_i) < g_i) then
        satisfazG:= false;
        SatisfazMeta(s_i, r_i, g_i, R_i, G);
      else if (s_i < g_i) then
        satisfazG := false;
      end if
    end for
    if satisfazG then
      succ(S, G);
    end if
  end.
```

O algoritmo *MeaPop* apresenta o planejamento de ordem parcial. O parâmetro deste procedimento corresponde ao estado meta G que deve ser satisfeito a partir do estado inicial S. A variável R é uma lista que armazena um conjunto de recursos que ao serem criados pelas ações podem ser usados para satisfazer novas pré-condições. Este processo de utilizar os recursos ao serem criados permite satisfazer algumas metas sem a necessidade de criar novas ações.

As variáveis s_i, r_i e g_i, representam a quantidade de recursos R_i para S, R e G, respectivamente. Este procedimento verifica cada recurso R_i não satisfeito ($(s\_i + r\_i) < g\_i$) e constrói um estado intermediário usando o algoritmo *SatisfazMeta*. O objetivo deste estado é de diminuir a diferença entre S e G. Se todos os recursos de G forem satisfeitos por S, é estabelecida uma relação entre essa meta e o estado inicial pelo método $succ(S, G)$.

O algoritmo *AlocarRecurso* é responsável por alocar qualquer recurso do estado inicial que possa ser consumido e que satisfaz parcial ou totalmente uma meta de G. Por exemplo, se o estado inicial S contém *300 Gold* e como uma meta de recursos de G seja *500 Gold*, os recursos de S são destinados para essa meta. Deste modo nenhum recurso *consume* é usado por metas distintas.

Procedimento que satisfaz a meta G definindo novas ações no plano.

```
procedure SatisfazMeta (s_i, r_i, g_i, R_i, G)
  var
    alfa, k : Integer;
    A_i : Acao;
    A : Estado de Recursos;
  begin
    A_i  := acao que produz R_i;
    alfa := unidades de R_i produzidas por A_i;
    k    := ceil((g_i - s_i - r_i) / alfa);
    if R_i = recurso do tipo ‘‘consume’’ then
      A := CriarMeta(k, A_i);
      succ(A, G);
      MeaPop(A);
      if ((alfa * k + s_i + r_i - g_i) > 0) then
        R.adicionar(A, alfa * k + s_i + r_i - g_i);
      end if
    else
      for i := 1 to k do
        A := CriarMeta(1, A_i);
        succ(A, G);
        MeaPop(A);
        R.adicionar(A, alfa);
      end for
    end if
  end.
```

O procedimento *succ(R, G)* é responsável por relacionar qualquer estado que produz recurso e esteja em R com as metas de G. Da mesma forma que *Alocar-Recurso*, se o recurso de R é consumível ele será destinado apenas a respectiva meta. Com isso é possível usar os recursos sem a necessidade de novas ações. Na lista R são armazenadas referências de cada ação, de modo que os recursos produzidos no decorrer do planejamento possam ser usados para satisfazer novas metas. Basicamente, este método cria um "*link causal*" entre a meta e a ação necessária para satisfazê-la.

Caso a quantidade de recursos de S mais R não sejam suficientes para satisfazer G, novas ações são destinadas para satisfazer essa diferença. Para essa tarefa é usado o algoritmo *SatisfazMeta*. Este procedimento escolhe a ação A_i que produz o recurso R_i a ser satisfeito. Em seguida é realizado o cálculo de quantas ações são necessárias para satisfazer a respectiva meta.

A escolha de como as ações são criadas dentro do plano é realizada de acordo com o tipo do recurso R_i. Se R_i é *consume* será criado um elemento no plano representando um conjunto de k ações A_i. Caso contrário, serão criadas k ações A_i. Para realizar essa tarefa é usada a função *CriarMeta*, que determina as pré-condições necessárias para as k ações serem executadas, armazenando este resultado em A. Este valor A é tratado como sendo um estado que faz parte do plano e deve ser satisfeito.

Para cada elemento do plano criado é construída a relação deste elemento com a meta através da função *succ(A, G)*. Posteriormente é realizada uma chamada do procedimento *MeaPop* para satisfazer as pré-condições das k ações. Ao final do *MeaPop*, os recursos são adicionados na variável R. Se forem do tipo *consume* é considerada apenas a quantidade excedente, ou seja, a diferença entre a meta e os recursos criados ($alfa * k + s\_i + r\_i - g\_i$).

Um exemplo deste planejamento de ordem parcial é apresentado na Figura 4. Este plano é um "grafo", onde cada vértice corresponde a uma quantidade de ações e as arestas definem a relação de precedência, ou seja, uma ação pode ser executada somente após todas as precedências. Para facilitar a descrição deste "grafo", cada vértice é associado com uma representação, onde: *Start* é o estado inicial com *1 Townhall, 1 Peasant* e *900 Gold*; *Goal* é o estado meta com *1 Townhall* e *4 Peasant*; g_i são as ações, sendo, $1 \leq i \leq n$, para n igual ao número de vértices sem considerar *Start* e *Goal*.

Após este plano ser criado, é aplicada uma política para o estabelecimento de prioridades. A seguir, na seção 4, é apresentado este procedimento.

4 Definição de Prioridades para Ações

Uma vez gerado o plano de ordem parcial, adota-se um procedimento que estabelece prioridades para as ações que constituem o plano. Isso possibilita que durante o escalonamento os recursos sejam melhores aproveitados. A prioridade associada a uma ação é representada por um valor numérico. Quanto menor este valor maior deve ser a prioridade a ser dada para a execução da ação.

O procedimento para o cálculo das prioridades será descrito a seguir com base no plano especificado na Figura 4. O procedimento inicia pela identificação das ações que contribuem diretamente para a satisfação das metas, que são: G_1, G_4 e G_6.

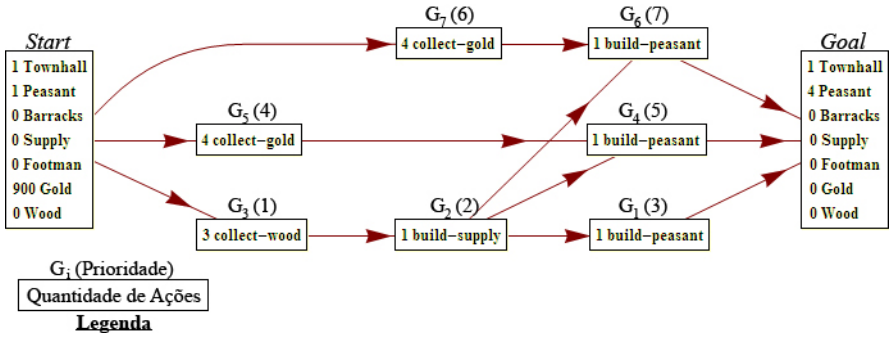

Figura 4. Plano de ordem parcial criado com o algoritmo *MeaPop*, a partir do estado inicial *Start* (*1 Townhall, 1 Peasant, 900 Gold*) e meta *Goal* (*1 Townhall, 4 Peasant*). Na parte superior de cada ação é associado uma representação. Entre parenteses temos a prioridade, quanto menor este valor, maior é a prioridade.

Para G_1 são determinadas todas as relações de precedências, neste caso G_2. É dada menor prioridade para G_1, já que não é possível de ser executada antes de G_2. Este procedimento é realizado para todas as ações precedentes. Ao final temos que, G_3 possui prioridade sobre G_2 e G_2 sobre G_1. Para cada uma dessas ações é associado um valor de prioridade começando em 1. Desta forma, respectivamente, as prioridades são 1, 2 e 3, para G_3, G_2 e G_1.

A seguir, este método é executado para G_4. A diferença é que neste passo o valor de prioridade considera o procedimento anterior de cálculo e desta forma começa com prioridade 4. Ao final deste procedimento, temos como resultado os valores de prioridade especificados na Figura 4. É importante salientar que o cálculo de prioridades a partir de G_4 não altera as prioridades obtidas a partir de G_1, assim como o cálculo de prioridades a partir de G_6 não altera as prioridades obtidas a partir de G_1 e G_4.

Para definir o tempo exato do plano, é criado um plano sequencial ordenando as ações pela prioridade. Este plano sequencial é usado como entrada para o algoritmo de escalonamento especificado em [1].

5 Discussão dos Resultados

Para análise dos resultados foi usado um ambiente de testes que realiza o planejamento seguindo os conceitos apresentados na seção 3. Após a execução do *MeaPop* é aplicada a estrutura de prioridade. A seguir, é criado um plano se-

quencial e finalmente ocorre o escalonamento pelo mesmo algoritmo de escalonamento especificado em [1]. O tempo de CPU gasto em cada plano é calculado pela média dos tempos de 10 execuções para cada problema, sendo medido em milissegundos (ms). Para os testes, [1] e [2] utilizam o *Wargus*, que é um módulo do jogo *Warcraft II*.

Tabela 1. Resultado dos planejamentos de [1] e *MeaPop* para alguns problemas.

Start	**Goal**	**Chan**		**MeaPop**	
		makespan	**tempo (ms)**	**makespan**	**tempo (ms)**
1 Townhall 1 Peasant	10000 Gold	51000	0,744	51000	0,597
	5000 Wood	78500	0,403	78500	0,317
	5000 Gold 1000 Wood	41200	0,485	41200	0,384
	10 Footman	63960	1,433	63960	1,000
1 Townhall	9 Peasant	4345	0,383	4830	0,419
4 Peasant	10 Peasant	4855	0,455	5055	0,511
1700 Gold	11 Peasant	5365	0,541	5280	0,617

A Tabela 1 mostra alguns resultados obtidos. A coluna *Start* é o estado inicial e *Goal* é a meta que deve ser satisfeita. Em *Chan* temos os resultados do planejamento de [1] e *MeaPop* os resultados do planejamento descrito no trabalho. Tanto para *Chan*, quanto *MeaPop*, são apresentados o *makespan* e o tempo dos planos correspondente aos respectivos estado inicial e meta.

Todas as metas da Tabela 1, onde o estado inicial é *1 Townhall* e *1 Peasant*, apresentam iguais resultados de *makespan* para ambos, Chan e *MeaPop*. Isso ocorre pois nenhuma ação é escalonada, sendo basicamente um plano ótimo sequencial. Assim como Chan, o trabalho proposto demonstra boas soluções para os casos mais simples, além de ter encontrado mais rapidamente os planos.

Com o estado inicial *1 Townhall*, *4 Peasant* e *1700 Gold*, observamos que para as metas *9 Peasant* e *10 Peasant*, os resultados foram piores. Isso ocorre devido a forma com que as ações são escalonadas, não importando que seja dada prioridade em criar *peasant*. Entretanto, se o número de *peasant* aumentar, o *MeaPop* encontra melhores resultados que Chan, como pode ser observada para a meta *11 Peasant*.

A Figura 5 mostra a comparação dos planos gerados pelos planejamentos de Chan e *MeaPop*. O estado inicial é (*1 Townhall 1 Peasant*), com o objetivo de atingir as metas: *10000 Gold* e *40 Footman*. Para realizar essa tarefa, é usado durante o planejamento um valor incremental do recurso *Peasant*, o que permite em geral de decrementar o *makespan* devido a maior possibilidade de escalonar as ações. Observamos que o *MeaPop* apresentou melhores planos e tempo de processamento em relação a Chan. Além disso, é interessante observar que mesmo aumentando o número de *Peasant* o *makespan* do *MeaPop* difere cada vez mais dos planos em Chan.

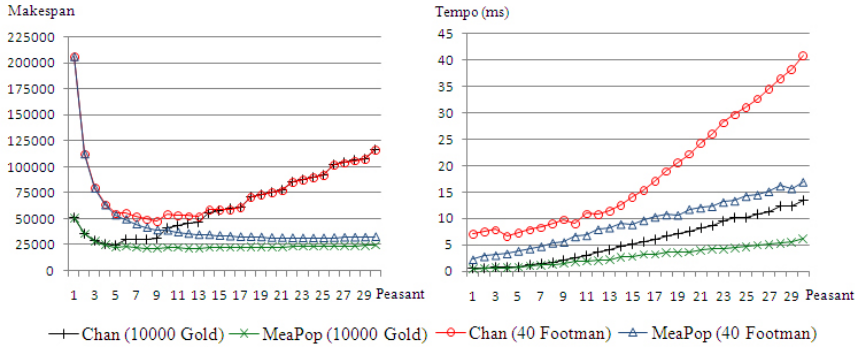

Figura 5. Comparação dos planejamentos de Chan e *MeaPop* considerando o estado inicial (*1 Townhall 1 Peasant*) e meta incremental do recurso *peasant*. A esquerda o gráfico com o *makespan*. A direita o gráfico do tempo em milisegundos.

6 Conclusão

Neste trabalho foi apresentado um algoritmo de planejamento de ordem parcial para geração de planos de recursos para jogos RTS. Sobre o plano parcial é aplicado um processo de prioridade que define a ordem em que as ações devem ser executadas.

Os resultados obtidos, de modo geral, foram melhores. Além de gerar planos com menor *makespan*, o tempo de processamento foi menor, mantendo uma das principais caracterésticas deste problema: a restrição de tempo.

Para trabalhos futuros, existe a proposta de integrar um mecanismo de aprendizagem ao sistema. Essa melhoria consiste no uso de algoritmos de busca e aprendizado em tempo real, tais como: LRTA* [4], SLA* e SLA*T [7]. O objetivo destes algoritmos é de substituir o procedimento de escalonamento de [1].

Referências

1. Chan, H., Fern, A., Ray, S., Wilson, N., Ventura, C.: Online Planning for Resource Production in Real-Time Strategy Games. ICAPS. 65–72 (2007).
2. Chan, H., Fern, A., Ray, S., Wilson, N., Ventura, C.: Extending Online Planning for Resource Production in Real-Time Strategy Games with Search. Workshop on Planning in Games, ICAPS (2007).
3. Fikes, R., Nilsson, N.J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence. 2, 189–208 (1971).
4. Korf, R.E.: Real-time heuristic search. Artificial Intelligence, Vol. 42:189–211 (1990).
5. Newell, A., Simon, H.A.: GPS, a program that simulates human thought. Computers & thought. MIT Press, Cambridge, MA, USA. 279–293 (1963).
6. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall (1995).
7. Shue, L., Li, S.: A Generalized Heuristic Learning Approach to Project Scheduling Problems with Resource Constraints. Journal of the Chinese Institute of Industrial Engineers (TSSCI, EI). Vol. 25, 3. 204–214 (2008).