Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação

Luis Henrique Oliveira Rios

Algoritmos de Inteligência Artificial para o Jogo OpenTTD

Belo Horizonte
2008 / 2º semestre

Universidade Federal de Minas Gerais

Instituto de Ciências Exatas
Departamento de Ciências da Computação
Curso de Bacharelado em Ciência da Computação

Algoritmos de Inteligência Artificial para o Jogo OpenTTD

por

Luis Henrique Oliveira Rios

Monografia de Projeto Orientado em Computação II

Apresentado como requisito da disciplina de Projeto Orientado em Computação II do Curso de Bacharelado em Ciência da Computação da UFMG

Prof. Dr. *Luiz Chaimowicz*
Orientador

Belo Horizonte
2008 / 2º semestre

RESUMO

O OpenTTD é um jogo de simulação de construção e gerência de rotas de transportes. Nele os jogadores e os NPCs controlam empresas de transporte que concorrem entre si. O objetivo do jogo é tornar-se o magnata dos transportes.

O comportamento das empresas controladas pelo computador não é satisfatório. A principal causa desse problema é a realização de ações que não condizem com as ações de um jogador humano, ou seja, o comportamento criado não é inteligente. Devido ao grande número de opções de transportes disponíveis e da diversidade de algoritmos necessários para controlá-los, optou-se por focar os esforços no aprimoramento de duas atividades: construção das estações ferroviárias e colocação dos trilhos. Essas atividades representam bem o conjunto de desafios que deve ser superado pela inteligência artificial do jogo.

Inicialmente, realizou-se um estudo com intuito de compreender melhor o seu funcionamento e quais requisitos os algoritmos deveriam satisfazer. Em seguida, a análise de publicações na área de inteligência artificial apresentou algumas opções viáveis ao contexto do OpenTTD.

Finalmente, as técnicas selecionadas foram implementadas no jogo para realização de testes e da avaliação do desempenho. As novas técnicas superaram os algoritmos originais criando um comportamento semelhante ao de jogadores humanos. Além disso, os algoritmos de inteligência artificial propostos passaram a aproveitar recursos do jogo antes não utilizados plenamente o que contribuiu ainda mais para o aumento da qualidade das ações geradas para os NPCs.

Palavras-chave: Inteligência Artificial. Jogos de Simulação de Construção e Gerência.

ABSTRACT

OpenTDD is a transport route construction and management simulator. In the game, players and NPCs manage transport companies that compete against each other. The main goal is to become the transport tycoon.

The computer controlled companies behavior is not satisfactory. The main reason for this problem is that most actions taken by the artificial intelligence algorithms wouldn't be taken by a human player. So, the achieved behavior cannot be considered smart. The large number of transport types available together with the variety of algorithms necessary to control them narrowed this project focus to the improvement of two activities: railroad station construction and railroad tracks deployment. These two activities represent challenges that have to be overcome by game's artificial intelligence.

Initially, the game was studied in order to understand its operation and which requirements should the artificial intelligence algorithms satisfy. Then, a study of related work in artificial intelligence has provided some techniques and algorithms suitable for OpenTTD.

Finally, the selected techniques were implemented in the game to permit some tests and performance evaluation. The new techniques overcame original algorithms creating a behavior very similar to human player's behavior. Furthermore, artificial intelligence algorithms proposed started to use game features not yet fully considered. This also contributes to the increase the quality of NPCs' actions.

Keywords: Artificial Intelligence. Construction and management simulation games.

LISTA DE FIGURAS

LISTA DE SIGLAS

IA	Inteligência Artificial
NPC	Non-Player Character
OpenTTD	Open Transport Tycoon Deluxe
TTD	Transport Tycoon Deluxe

SUMÁRIO

1 INTRODUÇÃO

1.1 Visão geral do assunto

Os jogos de simulação consagraram-se como um dos estilos mais diversificados. Dentre as várias modalidades existentes, encontra-se a simulação voltada para construção e gerência. Nesse estilo de jogo, a tarefa dos jogadores é construir, expandir e administrar comunidades, instituições, empresas ou impérios usando recursos limitados. Alguns exemplos clássicos são: *Capitalism*, *Caesar*, *SimCity* e *Transport Tycoon*. As ferramentas oferecidas, normalmente, possuem duas finalidades básicas: construção e administração. A economia é, também, um importante elemento presente nesses jogos por estar relacionada com os recursos disponíveis aos jogadores.

Os recentes avanços, tanto de software quanto de hardware, permitiram aos desenvolvedores de jogos de simulação - voltados para construção e gerência - um aumento substancial no número de elementos e detalhes disponíveis, atingindo, conseqüentemente, um alto grau de imersão e realismo. Nesse contexto, o estudo e pesquisa de algoritmos de inteligência artificial têm ganhado notabilidade. O aumento da complexidade dos jogos torna necessário o desenvolvimento e adaptação de técnicas que sejam capazes de lidar com um grande número de variáveis. Além disso, sendo um dos principais componentes dos jogos, ela contribui para a qualidade do jogo e deve possuir um nível de sofisticação semelhante aos outros elementos que o compõem. Os próprios jogadores já esperam níveis de refinamento maior da inteligência artificial tendo como base o nível dos outros elementos do jogo.

Uma das aplicações da inteligência artificial nesses jogos é o controle dos NPCs (*Non-Player Characters*). Os NPCs podem ser entendidos como personagens ou entidades do jogo que não são controlados pelo jogador e sim pelo computador. Inicialmente, os NPCs eram usados para substituir um jogador humano na forma de um oponente. No entanto, com a evolução dos jogos, os NPCs também passaram a ser colaboradores ou mesmo existirem para enriquecer o ambiente e aumentar a jogabilidade.

Os algoritmos de inteligência artificial são responsáveis pelas decisões e ações dos NPCs. Alguns requisitos devem ser atendidos por esses algoritmos. O primeiro deles diz respeito ao comportamento do NPC. Suas características devem ser semelhantes as de um jogador humano. Portanto, são desejáveis, principalmente **(**Baillie-de Byl, 2004): astúcia,

imprevisibilidade, capacidade de adaptar-se ao estado corrente do jogo e capacidade de refletir sobre suas ações. Existe, também, outro requisito que deve ser atendido por um algoritmo de inteligência artificial a ser incorporado em um jogo de simulação.

Apesar do aumento de recursos computacionais disponíveis, o tempo de resposta ainda é crítico. Os jogos, muitas vezes, são interativos, o que exige um tempo de resposta pequeno. São esses requisitos que tornam necessária a adaptação das abordagens clássicas e desenvolvimento de novos algoritmos de inteligência artificial que serão usados nos jogos de simulação.

1.2 Justificativa e motivação

Nesse contexto, surge a necessidade de estudar, desenvolver e avaliar algoritmos de inteligência artificial para o controle de NPCs em jogos que simulam construção e gerência. Em particular, esse trabalho propõe o estudo e avaliação de algoritmos de inteligência artificial que serão utilizados para criação e aprimoramento dos NPCs do jogo *OpenTTD*.

O *OpenTTD* (OpenTTD, 2008) é um clone, *open-source*, desenvolvido através da engenharia reversa do jogo original *Transport Tycoon Deluxe* de 1994. Utiliza os gráficos originais do jogo que são exibidos através de uma *engine* isométrica capaz de simular as três dimensões do espaço. O mapa é discreto, dividido em células que são as menores unidades nas quais pode-se construir elementos do jogo. Além de possuir todas as funcionalidades do jogo original, inúmeras melhorias e aperfeiçoamentos foram realizados aumentando significativamente os recursos e a jogabilidade.

O jogo é um simulador de construção e gerência de rotas de transportes. Existem dois modos de funcionamento: com um único jogador e com vários deles. No último caso, alguns jogadores podem ser controlados pelo computador. Outra importante característica é ambiente dinâmico do jogo - a simulação não é interrompida para aguardar uma ação dos participantes como nos jogos de turno.

No *OpenTTD*, o jogador é proprietário de uma empresa de transporte que concorre com outras transportadoras. Seu objetivo é tornar-se o magnata dos transportes, ou seja, fazer com que a sua companhia seja a melhor. Vários critérios são usados nessa avaliação: número de cargas transportadas, número de veículos utilizados, renda mínima e máxima das rotas, dívidas contraídas e dinheiro em caixa. Para obter uma boa avaliação nesses quesitos, é necessário que o jogador construa rotas de transportes lucrativas interligando indústrias e

cidades. As indústrias possuem diversos tipos de produção e demanda e as rotas devem ser construídas levando em consideração essas características. Existem quatro tipos básicos de transporte: ferroviário, rodoviário, aéreo e marítimo. Normalmente, o mais utilizado é o ferroviário por possuir a capacidade de transportar muita carga por grandes distâncias de maneira rápida.

Figura 1. A imagem mostra os quatro meios de transporte disponíveis no jogo.

O ponto central do *OpenTTD* é que o jogador construirá todos os elementos que envolvem a rota. Por exemplo, em uma rota ferroviária o primeiro passo é escolher o que será conectado através dela. Pode-se ligar cidades a cidades, indústrias a cidades e indústrias a indústrias. A escolha pode considerar diversos elementos: taxa da produção das cargas, dificuldade para construção das vias, distâncias entre ponto de origem e destino, quantidade já transportada por outras empresas e preço pago de acordo com a carga. Segue-se, então, a criação das estações ferroviárias. Para tal, muitas vezes, é necessário alterar o relevo para que se torne compatível com a construção. Essas terraplanagens, no entanto, possuem um custo

que pode se tornar muito elevado dependo do tipo de modificações. Além disso, é necessário que a estação esteja dentro de uma área denominada alcance da indústria ou cidade.

A colocação das vias também faz parte da tarefa de construir uma rota. No caso das ferrovias a colocação de vias é complexa. Deve-se ter o cuidado de evitar colisões entre os trens, congestionamentos e *deadlocks*. Existem mecanismos para que mais de um trem circule pelo mesmo trilho: sinais e trilhos múltiplos. Outros recursos que também podem ser usados para a construção das vias são os túneis, as pontes e as terraplanagens. Por fim, deve-se escolher a locomotiva e os vagões e programá-la para ir a determinadas estações e, então, carregar ou descarregar. A construção de rotas com os outros tipos de transportes envolve fases semelhantes com pequenas modificações. A construção das ferroviárias, porém, é a mais complexa.

Figura 2. A figura mostra como diversos trens podem compartilhar a mesma linha férrea através do uso de bifurcações e semáforos.

Os NPCs no jogo controlam empresas e, assim como os jogadores humanos, devem construir e gerenciar as rotas de transporte. Atualmente, existem dois tipos de NPCs no jogo

cada um controlado por um conjunto diferente de algoritmos de inteligência artificial. O primeiro foi criado para o *Transport Tycoon Deluxe*. Ele utiliza todos os meios de transportes, mas possui graves problemas para construção de vias, principalmente, de ferrovias. Freqüentemente, constrói trilhos redundantes, com custo elevado e traçado que desfavorece a aceleração das locomotivas. Além disso, diferentemente dos jogadores comuns, ele realiza terraplanagens sem custo. Isso acontece para não levar o NPC a falência já que ele usufrui desse recurso de maneira aleatória e indiscriminada. O segundo tipo de NPC é controlado por uma inteligência artificial criada por desenvolvedores do *OpenTTD*. Grandes melhorias foram realizadas, principalmente com relação a colocação de vias. No entanto, ele só trabalha com transportes rodoviários e não utiliza terraplanagens, recurso que é capaz de eliminar a inviabilidade da construção em alguns terrenos e melhorar significativamente o traçado das vias.

Figura 3. A imagem mostra alguns trilhos criados pela IA original do jogo cujo traçado prejudica a aceleração das locomotivas.

1.3 Objetivo

O presente trabalho tem como objetivo aprimorar a inteligência artificial do *OpenTTD* responsável por controlar os NPCs. Deseja-se aproximar as ações produzidas por esses algoritmos das ações realizadas pelos jogadores humanos. Além disso, as ferramentas oferecidas pelo jogo devem ser bem utilizadas pelos NPCs, ou seja, usadas de maneira a maximizar os benefícios com o menor custo possível. Desse modo, os NPCs apresentarão um comportamento mais lógico e condizente com a realidade do jogo.

Devido ao grande número de tipos transportes disponíveis no jogo e da diversidade de particularidades relacionadas com cada um deles, pretende-se focar os esforços no transporte ferroviário. O grande número de detalhes associados a esse transporte o torna o mais genérico no que se refere à construção. Assim, os algoritmos aplicados à construção de rotas ferroviárias podem, também, ser estendidos a outros meios de transporte do jogo.

O foco desse trabalho será, pois, a elaboração e adaptação de algoritmos para colocação de trilhos e construção de estações considerando as possibilidades fornecidas pelo jogo, como terraplanagens. Ambos problemas representam bem o conjunto de desafios que deve ser superado pelos algoritmos de inteligência artificial. A construção de estações é semelhante à criação de aeroportos, pois exige grandes áreas com terreno plano. Já a colocação de trilhos, é semelhante a criação de outras vias como rodovias e canais para navios.

Deseja-se a colocação de trilhos duplos, ou seja, um ao lado do outro para criação de fluxos de trens em sentidos opostos (como mostrado na Figura 2). Um outro requisito é a presença de características semelhantes com as dos trilhos colocados por jogadores humanos: inexistência de trilhos redundantes, grandes alterações de altitude e muitas curvas. Quanto a colocação das estações, o algoritmo deverá considerar a possibilidade de fazer terraplanagens. Essas terraplanagens deverão levar em conta o custo da alteração do relevo e os benefícios trazidos por ela. Por exemplo, próximo a uma cidade pode ser viável realizar um nivelamento um pouco mais caro para aumentar a área da cidade contemplada pela cobertura da estação.

2 REFERENCIAL TEÓRICO

Para realizar a construção e adaptação de tais algoritmos de inteligência artificial e integrá-los ao jogo *OpenTTD* foi necessário realizar alguns estudos que envolveram aspectos do jogo além de possíveis algoritmos. A seguir, serão apresentados tais estudos e os resultados mais relevantes de cada um deles.

2.1 Detalhamento do OpenTTD

Como mencionado anteriormente, o ponto central desse jogo é a criação e gerência de rotas de transportes. Para realizar a construção das vias, o jogador dispõe de algumas ferramentas que variam de acordo com o tipo de transporte. No entanto, pode-se dizer que as ferrovias, devido a sua complexidade, possuem o conjunto mais completo. De certa forma, as ferramentas dos outros meios de transportes estão relacionados com alguma delas. São elas:

- **Colocação de trilhos**: possibilita a construção das seis direções de trilhos presentes no jogo;
- **Túneis**: permite que o jogador construa um túnel. Somente duas direções são permitidas (eixo X e eixo Y);
- **Pontes**: as pontes podem ser usadas para transpor alguns obstáculos como trilhos de adversários, rios e acidentes geográficos. Assim como os túneis, somente podem ser colocadas em duas direções;
- **Alteração do relevo**: possibilita que o jogador realize modificações no relevo para torná-lo adequado aos seus propósitos. Por exemplo, para que seja possível construir um túnel a célula do mapa deve possuir um relevo especifico. Se o local no qual deseja-se construir um túnel não possui essa característica, a ferramenta em questão pode torná-lo adequado.
- **Semáforos**: são usados para compartilhar a mesma linha férrea com várias locomotivas. Existem quatro tipos: comum, de entrada, de saída e o *combo*. Cada um deles pode ser colocado de três diferentes maneiras: sentido horário, sentido anti-horário e nos dois sentidos. O semáforo de entrada fica verde se existe um ou mais semáforos de saída também verdes ao longo do caminho. O de saída funciona

como o semáforo normal, mas aciona o semáforo de entrada. O *combo* é um semáforo de entrada e saída simultaneamente. A Figura 2 mostra a utilização de semáforos simples, de entrada e de saída.

As empresas controladas pelo computador também utilizam dessas ferramentas para interagir com o ambiente do jogo. As ações são feitas através da função *"DoCommand"*. Todas ações realizáveis pelos jogadores utilizam internamente essa função. Mais detalhes a seu respeito serão apresentados na seção 3.2.1.

Para construção das estações o jogador dispõe basicamente de duas ações. A primeira delas é a terraplanagem. Além dela, existe a ação de criar a estação propriamente dita. O ponto central da criação das estações ferroviárias é a exigência de que o terreno esteja nivelado e próximo a indústria ou cidade a ser contemplada pela estação. O posicionamento das estações pode facilitar ou dificultar a colocação dos trilhos e também deve ser considerado.

2.2 Algoritmos de Busca

O problema da colocação de trilhos pode ser modelado como um problema de busca em grafos. As características que o caminho calculado pelo algoritmo deve possuir podem ser expressas através de uma função de utilidade. Devido ao tamanho dos mapas, optou-se por adotar técnicas que façam o uso de heurísticas. A seguir, serão discutidos os algoritmos encontrados na literatura relacionados com esse trabalho.

2.2.1 O Algoritmo A*

O A* (Russell; Norvig, 2003), (Hart; Nilsson; Raphael, 1968) é um algoritmo de busca heurístico do tipo *best-first* (ou seja, tenta expandir primeiro os nós mais próximos do nó destino usando a métrica definida) para grafos. No algoritmo em questão, a métrica é denotada por $f(n)$, onde n é um nó do grafo. Essa função é a soma de duas outras denotadas por $g(n)$ e $h(n)$, ou seja, $f(n) = g(n) + h(n)$. A primeira parcela da soma representa o custo da atingir n partindo da origem. A parcela $h(n)$ é a heurística. Ela representa o custo esperado para atingir o nó de destino partindo de n.

Existem dois casos particulares do A* que devem ser destacados. Quando *h(n)=0*, para todo nó *n* do grafo, o algoritmo comporta-se como o algoritmo de Dijkstra. No entanto, se *g(n)=0*, para todo nó *n* do grafo, ele comporta-se como o algoritmo *best-first* guloso, ou seja, fica puramente heurístico.

Uma importante propriedade desse algoritmo está relacionada com a heurística escolhida. Se a heurística utilizada no A* é admissível, ou seja, nunca superestima o custo para atingir o nó destino, o algoritmo sempre encontra a solução ótima, caso ela exista.

O funcionamento desse algoritmo é bastante simples. Duas listas são utilizadas durante o processamento: *open list* e *closed list*. A primeira armazena os nós visitados que ainda não foram expandidos. A cada iteração, o nó com o menor valor de *f* é retirado dessa lista, expandido e colocado na *closed list*. Durante a expansão, os filhos desse nó que não estão na *closed list* são colocados na *open list*. O valor de *g* dos filhos será o custo de seu pai mais o custo para atingi-lo partindo do seu pai. Se o filho já está na *open list* e o valor de *g* calculado é menor que o valor atualmente associado a esse nó, atualiza-se o valor de *g* do nó. A *closed list* é utilizada para que os nós não sejam expandidos duas vezes. O algoritmo termina quando o nó destino é retirado da *open list* ou esta lista encontra-se vazia. Nesse caso não existe uma solução.

Um dos problemas do A* é que, no pior caso, o número de nós expandidos pode crescer exponencialmente com o tamanho da solução. Isso pode acarretar um gasto de tempo e espaço muito grande.

2.2.2 O Algoritmo LPA*

O LPA* (LifeLong planning A*) (Koenig; Likhachev; Furcy, 2004) é uma generalização do algoritmo A*. Ele é um algoritmo de busca incremental e heurístico. A proposta desses algoritmos é diminuir o custo para recalcular a solução no caso de alterações no grafo. Ou seja, caso haja uma modificação no grafo o algoritmo tenta encontrar a solução ótima mais rápido do que recalcular completamente a busca. Para isso, utilizam-se informações das execuções anteriores. O LPA*, portanto, deve ser usado em ambientes dinâmicos.

O algoritmo aplica-se a problemas de grafos finitos onde é possível determinar os predecessores e sucessores de um nó. Assim como no A*, para garantir a solução ótima, deve-se usar uma heurística admissível.

Seu funcionamento, mais complexo que do A*, utiliza uma fila de prioridade cuja chave envolve o custo para chegar ao nó partindo da origem, um *lookahead* desse custo e a heurística (distância esperada para chegar ao destino). A cada iteração, o algoritmo retira um nó dessa fila e o atualiza ou expande de acordo com os valores associados a ele. Os nós presentes na fila são sempre os nós inconsistentes, ou seja, aqueles cujos valores do *lookahead* diferem do custo para atingi-lo partindo da origem.

Quando ocorrem alterações nos pesos das arestas, é necessário atualizar cada nó afetado por essa mudança. O algoritmo exige apenas que os nós diretamente afetados sejam informados. Os demais são calculados por ele. Durante o processo, no entanto, nem todos os nós afetados serão reavaliados. A heurística é usada para minimizar esse número.

Um dos problemas do LPA* é o armazenamento dos valores calculados entre as execuções. Como o número de nós pode ser grande, a quantidade de memória para armazená-los pode estar indisponível. Outra dificuldade está relacionada com a informação necessária para conhecer os predecessores e sucessores de um nó durante a busca.

3 METODOLOGIA

3.1 Tipo de pesquisa

O objetivo desse trabalho é o aprimoramento dos algoritmos de inteligência artificial do *OpenTTD* responsáveis por controlar os NPCs. Deseja-se que o comportamento criado por esses algoritmos passe a demonstrar inteligência, ou seja, gerar ações semelhantes às de um jogador humano. Os focos foram a colocação de trilhos e a construção de estações ferroviárias. Para tal, foi usada uma pesquisa aplicada com procedimentos experimentais. Quanto aos objetivos, portanto, esse trabalho tem um caráter exploratório.

3.2 Procedimentos metodológicos

O presente trabalho foi realizado no período de julho a novembro de 2008. A primeira etapa envolveu um estudo bibliográfico para consolidar uma base conceitual e conhecer as publicações científicas relacionadas com os problemas abordados. A etapa seguinte foi analisar, detalhadamente, o *OpenTTD*.

Através delas, concluiu-se quais são os melhores algoritmos para realizar colocação dos trilhos e construção das estações ferroviárias. A metodologia usada para resolução desses problemas será descrita detalhadamente nas próximas subseções.

Por fim, essas soluções foram implementadas no jogo, testadas e avaliadas. Os resultados da avaliação serão apresentados na seção seguinte.

3.2.1 Implementação

Como mencionado na seção 2.1, as ações do jogo são realizadas através da função cuja declaração é *"CommandCost DoCommand(TileIndex tile, uint32 p1, uint32 p2, uint32 flags, uint32 procc);"*. O valor de retorno representa o custo da ação desejada. Existe um valor

especial que representa a falha do comando. Os parâmetros *tile*, *p1* e *p2* são usados para passar informações à função referentes a ação. Dessa forma, seus valores e interpretação dependem da ação que será executada.

O parâmetro *flags* é usado para controlar a execução dessa ação. Os valores mais importantes são *DC_EXEC* e *DC_QUERY_COST*. O primeiro é usado quando a ação deve ser executada. Já com o segundo deseja-se apenas conhecer o custo da ação ou se sua execução é legal de acordo com as regras do jogo.

Entretanto, existem situações onde o uso da *flag DC_QUERY_COST* retorna sucesso para uma dada ação, mas sua execução irá falhar. Isso acontece para ações complexas, ou seja, compostas de várias ações mais simples. Quando a *flag* em questão é passada à função, nenhuma ação é de fato executada, ou seja, o comando testa apenas se pode executá-las. O problema ocorre quando a execução de uma ação interfere com a execução de uma outra. Como ela não foi executada, a função *"DoCommand"* não consegue perceber os impedimentos gerados por essa interferência.

O último parâmetro da função (*procc*) representa a ação que será executada. Existem diversas ações disponíveis como: *CMD_BUILD_RAILROAD_TRACK*, *CMD_TERRAFORM_LAND*, *CMD_BUILD_RAIL_VEHICLE* e *CMD_LEVEL_LAND*.

A ação *CMD_LEVEL_LAND* apresenta o problema descrito anteriormente. Ela é composta por diversas execuções de *CMD_TERRAFORM_LAND*. Como cada execução pode interferir na execução das demais e a função só é capaz de perceber isso durante a execução, algumas vezes a função *"DoCommand"* com a *flag DC_QUERY_COST* retorna sucesso equivocadamente para essa ação.

O principal causa da interferência entre as execuções da ação *CMD_TERRAFORM_LAND* é a complexidade do sistema de terraplanagens. No *OpenTTD* pode-se alterar a altura dos quatro vértices de uma célula do mapa. As alterações em um desses vétices podem causar alterações nos outros vértices e sempre causa mudanças no formato das células vizinhas, já que estas compartilham o vértice. Em alguns casos, uma simples mudança de altura em um vértice propaga alterações por diversas células além das vizinhas.

A inteligência artificial utiliza a função *"DoCommand"* para agir no ambiente do jogo. Já para conhecer o ambiente do jogo, deve-se usar as estruturas de dados internas. A mais importante delas é a representação do mapa que é um arranjo. Cada posição desse arranjo possui diversos atributos que descrevem a respectiva célula do mapa. Existem funções que convertem uma coordenada para um índice no arranjo e vice-versa. Outras estruturas

também relevantes são as listas de indústrias, cidades, estações e veículos. É importante ressaltar que todas essas informações também estão disponíveis para os jogadores humanos. Dessa forma, a inteligência artificial não tem nenhum benefício adicional por acessá-las diretamente.

3.2.2 Colocação dos Trilhos

O primeiro passo para solucionar o problema da colocação de trilhos foi modelá-lo como um problema de busca em grafos. Cada nó desse grafo corresponde a uma peça de trilho dupla. Essas peças foram criadas com o intuito de facilitar a representação. Elas são construídas a partir da união dos segmentos de trilhos simples fornecidos pelo jogo. As peças possuem direção e sentido não podendo, pois, ser encaixadas de qualquer modo. Os semáforos são associados às peças, garantindo que a união delas formará duas pistas em sentidos opostos sinalizadas corretamente. Os túneis e pontes são considerados peças duplas especiais de tamanho variável.

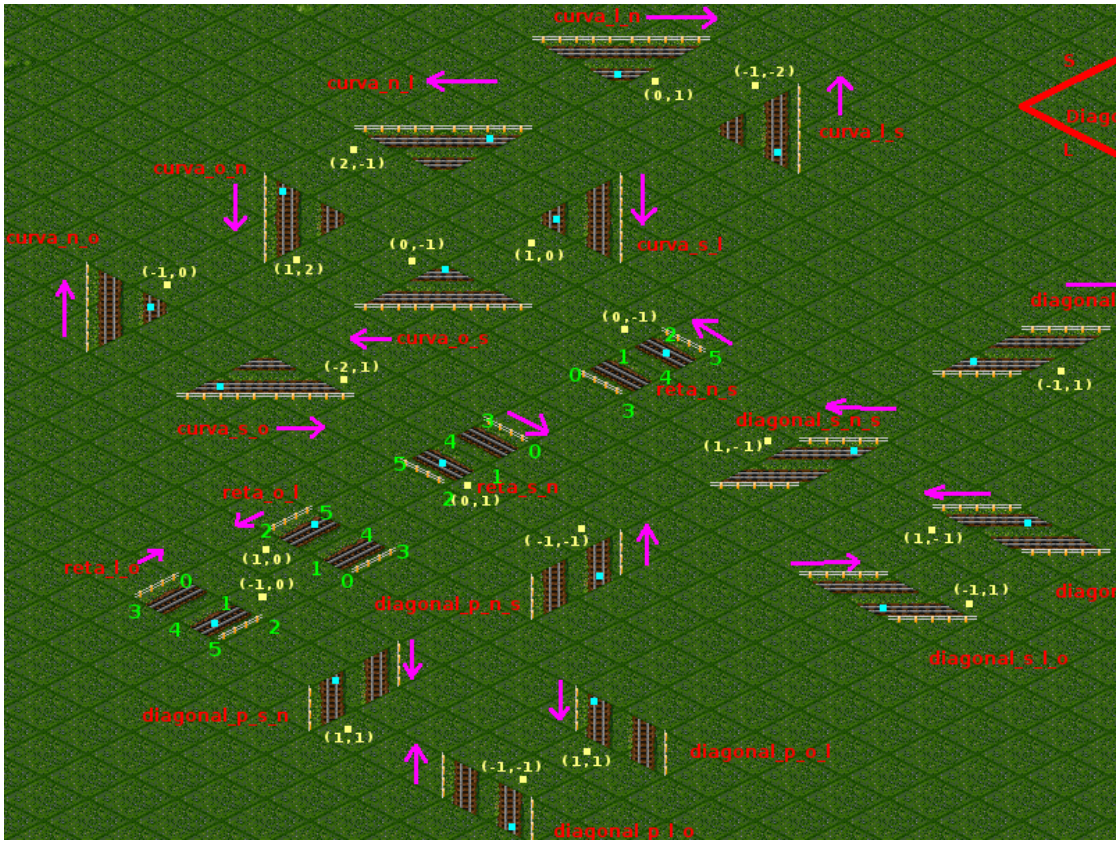

Figura 4. A imagem mostra as peças duplas usadas para criação dos trilhos duplos. As setas cor de rosa indicam o sentido. Os retângulos amarelos são pontos de referencia onde a próxima peça deverá ser encaixada (usando os retângulos azuis). Os números verdes caracterizam os vértices das peças paralelas aos eixos das abscissas e ordenadas.

O algoritmo A*, então, é utilizado para solucionar esse problema de busca em grafos. As peças iniciais são adicionadas a *open list* de acordo com a direção e sentido da estação de origem. A cada iteração, o algoritmo retira uma peça dessa lista e verifica se é possível construir cada uma das peças que podem sucedê-la. Caso seja possível, essas são adicionadas na *open list*. A função de utilidade é usada para tornar os trilhos construídos mais semelhantes aos trilhos criados pelos jogadores humanos. Ela penaliza o caminho aumentando o custo sempre que uma curva é feita. Com isso, o algoritmo evita as curvas. Além disso, existem penalizações para pontes e túneis que são elementos mais caros. As pontes são colocadas sempre que existe um obstáculo a frente. O tamanho é determinado de modo a vencer esses obstáculos, respeitando um limite máximo. Os túneis são colocados sempre que o relevo é

favorável a sua construção. Eles também têm um limite de tamanho. A heurística usada no A* é a distância de *Manhattan*.

Ao final da execução do algoritmo A*, se uma solução foi encontrada, começa o processo de alteração do relevo. As alterações são feitas nos trechos de trilhos paralelos aos eixos das abscissas e das ordenadas (na Figura 4 pode-se ver as peças que constituem esses trechos: *reta_n_s, reta_s_n, reta_l_o* e *reta_o_l*). A razão disso é que somente essas peças podem ser colocadas em locais com alteração de altura. Para determinar a altura das peças são usados seus vértices centrais. Na Figura 4 eles são representados pelos números 1 e 4.

O processo de nivelamento do terreno segmenta esses trilhos de maneira a encontrar trechos com pouca variação de altura (no máximo três níveis). Caso esse trecho seja uma subida ou descida, ou seja, as altitudes são crescentes ou decrescentes, apenas alterações locais são feitas em cada peça. Essas mudanças locais tentam fazer com que as linhas de vértices que formam a peça fiquem na mesma altura (na Figura 4 essas linhas são 0, 1 e 2 ; 3 , 4 e 5). O outro caso é quando o trecho não é crescente e nem decrescente. Calcula-se, então, a altura comum ao maior número de vértices (considerando os vértices de todas as peças do trecho) e a altera-se a altura dos vértices desse trecho de forma a atingir essa elevação. É possível limitar a quantidade máxima de recursos que será gasta com alterações de relevo e o custo máximo de uma alteração individual, ou seja, alteração em um vértice de uma peça.

Durante esse processo, o posicionamento de algumas peças, calculado anteriormente, pode ser invalidado. Para resolver isso, executa-se novamente o algoritmo A*. Nessa situação, o algoritmo LPA* evitaria recalcular completamente a rota. No entanto, como mencionado na seção 2.1, o sistema de terraplanagens do jogo é complexo. Assim, uma simples alteração em um vértice pode invalidar várias peças colocadas em células próximas tornando difícil a detecção dos nós afetados. O espaço exigido pelas estruturas também representou um obstáculo à sua utilização.

Na segunda execução do A*, a função de utilidade é alterada. Ela passa a priorizar o caminho calculado na primeira execução. Isso é feito diminuindo o custo associado às peças que o formam. Além disso, a função de utilidade passa a penalizar as alterações de altitude. O objetivo dessa alteração é fazer com que o terreno alterado seja usado para colocação dos trilhos o máximo possível e as novas possibilidades para colocação de túneis (criadas após as alterações) sejam exploradas. Somente pequenos trechos da rota onde surgiram as incompatibilidades deverão ser desviados.

Terminada a execução do segundo A*, chega hora da construção dos trilhos, semáforos, pontes e túneis usando as ações fornecidas pelo jogo. Nesse passo, o algoritmo lê a estrutura que representa o caminho calculado e constrói as respectivas peças, túneis e pontes.

3.2.3 Construção das Estações

A construção das estações necessita de um terreno plano próximo o suficiente da indústria ou cidade almejada. Como essa necessidade também existe em outros meios de transporte (principalmente o aéreo no momento da construção dos aeroportos), criou-se funções genéricas para realização dessa atividade. São elas *"AlocaTerrenoCidade" e "AlocaTerrenoIndústria"*.

Tais funções foram criadas de maneira a oferecer o máximo de flexibilidade possível para que pudessem ser aproveitas em diversos contextos. Um dos parâmetros é uma função de avaliação. A avaliação deve retornar uma nota de 0 a 1 avaliando o terreno passado a ela. A nota 0 significa que o terreno não deve ser alocado. Esse mecanismo permite uma flexibilidade muito grande através do uso de diversos critérios.

O funcionamento dessas funções consiste em verificar quais terrenos sem construção satisfazem a condição imposta pela função de avaliação e escolher aquele com o melhor custo benefício. Dessa forma, o algoritmo tenta maximizar a função de utilidade seguinte: $f(c,a) = p(1 - c) + (1 - p)a$. O termo p é um constante de 0 a 1 e representa o peso do custo $(1 - c)$ em relação a avaliação a fornecida pela função de avaliação. O custo está associado as terraplanagens que devem ser realizadas para que o terreno fique plano. O custo c é representado por um número entre 0 e 1. Faz-se isso simplesmente dividindo cada um dos custos dos terrenos válidos pelo maior custo encontrado. As funções permitem, também, limitar os recursos gastos na atividade de alteração de relevo através de um parâmetro.

Como mencionado na seção 3.2.1, a função de nivelamento do terreno pode falhar na hora da execução. Para resolver esse problema, modificou-se o procedimento do jogo responsável pelo nivelamento para que cobrasse do jogador apenas se houvesse sucesso. Desse modo, escolhe-se o terreno com melhor custo benefício até que a terraplanagem associada a ele seja realizada corretamente.

A construção de estações ferroviárias, portanto, utiliza as funções descritas anteriormente para alocar o terreno necessário. A função de avaliação é utiliza para garantir

que a estação alcançará a indústria ou cidade em questão. Também é usada para posicionar a estação dentro do terreno. O posicionamento envolve duas questões: priorizar a direção de saída que favoreça a colocação de trilhos e posicionar a estação dentro do terreno de modo a aproximá-la da indústria ou cidade em questão.

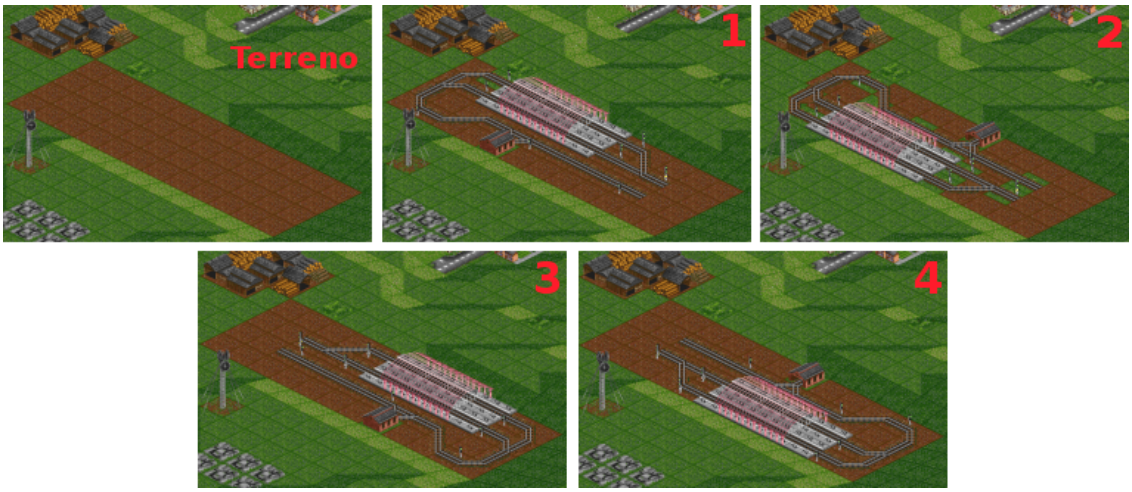

Figura 5. A figura mostra quatro maneiras diferentes de construir a estação dentro de um terreno. A indústria alvo encontra-se no canto superior esquerdo. As opções 3 e 4 não a alcançam e são, portanto, eliminadas.

Por fim, estação é construída no terreno alocado e os trilhos são colocados de modo que o conjunto seja compatível com o mecanismo de colocação de trilhos descrito anteriormente. Para aumentar a generalidade desse algoritmo, o tamanho da estação e número de pista são parâmetros.

4 RESULTADOS E DISCUSSÃO

Os resultados desse trabalho são algoritmos capazes de realizar ações semelhantes às realizadas pelos jogadores humanos.

Um desses algoritmos foi implementado através das funções *"AlocaTerrenoCidade" e "AlocaTerrenoIndústria"*. Ele é capaz de escolher um terreno com as dimensões desejadas maximizando o custo benefício. O benefício, por exemplo, seria ampliar a área de cobertura em uma cidade para um aeroporto ou estação. Já o custo estaria associado às alterações no relevo necessárias para que o terreno possa comportar o aeroporto ou estação.

O jogador humano, ao longo do jogo, realiza várias ações considerando esse custo benefício. Quando está no começo do jogo, evita grandes despesas com construções já que seus recursos ainda são escassos. No entanto, quando o jogo avança e as cidades crescem ele pode investir mais recursos na construção de um aeroporto, por exemplo, bem mais próximo à cidade. Tal aeroporto terá grande oferta e demanda de passageiros e correio conseguindo, assim, gerar receita suficiente para cobrir os custos do investimento inicial e produzir lucro.

Os algoritmos propostos oferecem flexibilidade suficiente para gerar esses comportamentos. Pode-se usar a função de avaliação para medir o benefício da aproximação do terreno da cidade ou estação em questão. O peso do custo medirá a preocupação com o custo ou benefício. A figura a seguir mostra o resultado do uso da função *"AlocaTerrenoCidade"* com diferentes pesos para o custo.

Figura 6. A figura mostra dois terrenos alocados pela função *"AlocaTerrenoCidade"*. A diferença de posicionamento desses terrenos é causada pelo peso do custo diante do benefício propiciado. O benefício nesse caso é o número de casas da cidade alcançadas pelo aeroporto. O alcance está representado pelos quadros azuis. Quando o peso do custo é menor, a função gasta mais recursos, mas consegue um terreno bem próximo à cidade. No entanto, quando a relevância do custo é maior, o terreno alocado fica mais afastado da cidade e o gasto de recursos financeiros é menor.

O algoritmo para construção de estações ferroviárias utiliza as funções discutidas anteriormente. O comportamento gerado é, novamente, semelhante ao de um jogador humano. No momento da construção das estações ferroviárias, o jogador procura orientá-la de modo a facilitar a construção dos trilhos. Através da função de avaliação, o algoritmo proposto é capaz de gerar esse comportamento. A Figura 7 mostra o impacto dessa orientação na etapa de colocação dos trilhos. Além de gerar de gerar trilhos com menos curvas, a busca é bem mais rápida quando o posicionamento das estações é considerado. A principal motivo é que muitas curvas podem ser necessárias quando a estação está mal posicionada (suponha que o trilho esteja vindo em uma direção e a saída da estação está na direção oposta, serão necessárias algumas alterações na direção para chegar a ela). Como o algoritmo penaliza essas curvas o número de nós expandidos é muito grande até que elas sejam consideradas.

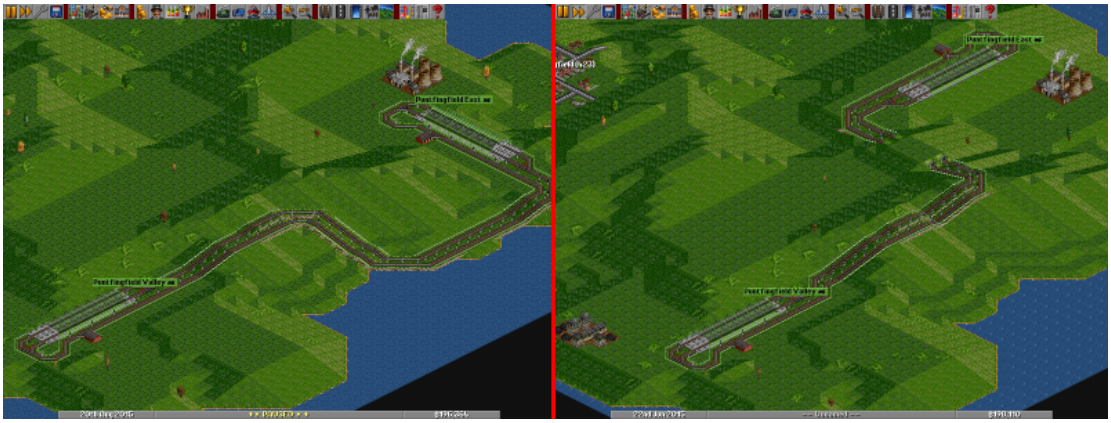

Figura 7. Na figura da esquerda o posicionamento das estações não foi tão cuidadoso quanto na direita. Observe que para esse pequeno exemplo o número de curvas diminui de três para duas.

Outro aspecto importante da metodologia para construção das estações adotada é sua total compatibilidade com o sistema de colocação de trilhos criado. Os trilhos colocados juntos da estação integram-se perfeitamente a malha ferroviária gerada. Também minimizam os congestionamentos e impedem os *deadlocks* e acidentes.

Figura 8. A imagem mostra duas estações de tamanho, número de pistas e orientação diferentes criadas usando o algoritmo proposto. Note que os trens sempre chegam por um lado e saem pelo outro. Dessa maneira, evita-se os *deadlocks* e minimiza-se os congestionamentos.

O algoritmo para colocação de trilhos é outro importante resultado desse trabalho. Os trilhos colocados são duplos e sinalizados permitindo criar um fluxo de trens em ambos sentidos. O traçado dos trilhos é feito com o cuidado de imitar comportamentos de jogadores como poucas alterações de altitude e poucas curvas.

Além disso, o algoritmo usa o recurso de terraplanagem de maneira inteligente para melhorar o traçado das ferrovias diminuindo o número de alterações de altitude. Essa é uma das características mais marcantes dos trilhos colocados pelos jogadores humanos.

Figura 9. A imagem superior mostra o mesmo trecho de trilhos da imagem inferior. A diferença é que na primeira não foram realizadas alterações no relevo. Percebe-se claramente a diminuição do número de mudanças de altitude ao longo do trecho.

O procedimento proposto, portanto, é superior a IA original do jogo (única que utiliza trilhos e terraplanagem). Superior, primeiramente, pelo fato de realizar racionalmente as alterações de relevo e pagar por elas. A duplicidade dos trilhos também o torna superior em relação a IA original do jogo. Ela permite que vários trens circulem com segurança e eficiente pelos trilhos colocados. O traçado também é melhor já que não faz curvas desnecessárias e prioriza as retas.

5 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho buscou aprimorar os algoritmos de inteligência artificial do jogo *OpenTDD* responsáveis pela realização da colocação de trilhos e construção de estações. O objetivo foi criar algoritmos capazes de gerar ações inteligentes, ou seja, semelhantes às ações dos jogadores humanos.

Inicialmente, realizou-se um estudo com intuito de compreender melhor o funcionamento do jogo e quais requisitos os algoritmos de inteligência artificial deveriam satisfazer. Em seguida, a análise de publicações na área de inteligência artificial apresentou algumas opções viáveis ao contexto do jogo. Finalmente, as técnicas selecionadas foram implementadas para realização de testes e da avaliação do desempenho.

Os resultados do desenvolvimento desse projeto estão de acordo com a proposta, pois os algoritmos para colocação de trilhos e construção de estações criados para o *OpenTTD* demonstram inteligência, ou seja, geram comportamentos com características semelhantes aos dos jogadores humanos. Além disso, recursos antes não usados racionalmente pela inteligência do jogo, contribuíram significativamente para elevar a qualidade das soluções geradas pelos algoritmos.

Futuramente, pretende-se aprofundar mais esse trabalho para que seja possível elaborar artigos a serem submetidos para conferências nacionais ou internacionais da área. Dentre as tarefas inicias, está a criação de métricas mais precisas para avaliação dos procedimentos propostos. Uma das idéias é comparar os trilhos e estações criados pelo algoritmo com trilhos e estações de jogadores iniciantes e avançados. A abordagem utilizada na construção desses trilhos duplos expressa através da abstração de peças duplas, permite que os algoritmos construídos sejam facilmente utilizados para outros meios de transporte como as rodovias. Essa extensão é uma das continuações desse trabalho. Existe, também, a possibilidade de aplicar conceitos da área de Pesquisa Operacional para aprimoramento desses algoritmos como, por exemplo, para melhorar a distribuição dos recursos no momento da escolha de trechos que serão nivelados. Além disso, pretende-se compartilhar os resultados desse trabalho com a comunidade mantenedora do *OpenTTD*.

BIBLIOGRAFIA CITADA

Baillie-de Byl, Penny. *Programming Believable Characters for Computer Games*: (Game Development Series). Rockland: Charles River Media, Inc., 2004.

Hart, Peter E.; Nilsson, Nils J.; Raphael, Bertram. A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. on SSC*. v. SSC-4, n. 2, p. 100-107, Jul. 1968.

Koenig, Sven; Likhachev, Maxim; Furcy, David. Lifelong planning A*, *Journal of Artificial Intelligence Research*. v. 155, n. 1-2, p. 93-146, 2004.

OpenTTD. *OpenTTD - About*. Disponível em: http://www.openttd.org/about.php. Acesso em: 19 de novembro de 2008.

Russell, Stuart J.; Norvig, Peter. *Artificial Intelligence*: A Modern Approach. Pearson Education, 2003.