Controlo Inteligente de Tráfego para Redes de AGVs

Pedro M. Domingos
Secção de Sistemas, Departamento de Engenharia Mecânica, IST
Projecto de CAD/CAM-INESC

Paulo R. Oliveira
CAPS/LRPI

RESUMO

Este artigo descreve a concepção e o funcionamento do sistema CITRA - Controlo Inteligente de Tráfego para redes de AGVs. Dada a planta duma fábrica, a localização inicial de um certo número de AGVs ("Automatic Guided Vehicles") e a localização final pretendida de cada um, o CITRA utiliza técnicas de inteligência artificial para encontrar um percurso para cada AGV tão curto quanto possível e que evite "colisões" com os outros. A necessidade de recorrer a métodos heurísticos deriva da explosão combinatória que se obtém quando se procura resolver o problema gerando sistematicamente todas as soluções possíveis.

ABSTRACT

This paper describes CITRA, an intelligent traffic control system for AGV networks. Given the map of a factory, the inicial location of a given number of AGVs (Automatic Guided Vehicles) and the desired final location of each of them, CITRA employs artificial intelligence techniques to find a path for each AGV that is as short as possible and avoids collision with other AGVs. The need to resort to heuristic methods derives from the combinatorial explosion that occurs when one tries to solve the problem by systematically generating all possible solutions.

1 Introdução

O estudo das fábricas flexíveis, ou FMSs (Flexible Manufacturing Systems), é hoje uma área activa de investigação. (Uma boa fonte de informação e de resultados recentes é a revista IEEE Transactions on Robotics and Automation). As FMSs são em geral compostas por um determinado número de estações de trabalho automatizadas, cada uma das quais executa uma tarefa ou um conjunto de tarefas necessária(s) à obtenção do(s) produto(s) final(ais). Neste contexto um problema importante é o do transporte de materiais, peças e produtos intermédios entre as várias estações. Uma solução é a utilização de AGVs, veículos automáticos cujo percurso pode ser reprogramado sempre que isso se justifique (p. ex.: devido a uma variação dos volumes de produção relativos dos vários produtos em fabrico, levando a uma reatribuição das estações; devido a modificações no processo produtivo e/ou no produto final, etc.). Por razões óbvias de custo e de eficiência interessa que o percurso de cada AGV seja o mais curto possível. Além disso verifica-se que, devido à forma como as vias de comunicação são realizadas na prática (p. ex. calhas ou carris), não é possível ter mais que um AGV em cada troço num dado instante. É então necessário encontrar, dadas as estações de partida e de chegada, um conjunto de percursos simultâneos para os AGVs envolvidos que satisfaça estes dois critérios.

O artigo é composto pela definição do problema em termos de entradas/saídas e de representação interna, que é tratada na próxima secção, pela exposição do método de resolução utilizado, que ocupa a seguinte, e por algumas considerações acerca da implementação realizada.

2 Definição do Problema

Para caracterizar completamente o problema é necessário definir como é representada a fábrica, como são representados os AGVs e os condicionalismos sobre o seu movimento, e a forma em que a solução é apresentada.

2.1 Representação da fábrica

A planta da fábrica é especificada por meio de uma das listas das "ruas" existentes, sendo cada "rua" definida pelas coordenadas (x_1, y_1) e (x_2, y_2) dos dois pontos P_1 e P_2 que une. As ruas admitem-se horizontais ou verticais no plano XY, isto é, ou $x_1 = x_2$ ou $y_1 = y_2$, podendo então cada rua ser representada por uma lista do tipo $(h\ y\ x_1\ x_2)$ ou $(v\ x\ y_1\ y_2)$ conforme for horizontal ou vertical respectivamente. As coordenadas são necessariamente inteiras. Por definição um cruzamento é um ponto P que é extremidade de 2, 3 ou 4 ruas simultaneamente, não devendo portanto haver ruas que se cortem a meio. (Por exemplo não devem existir simultaneamente na mesma planta as ruas $(h\ 0\ {-1}\ 1)$ e $(v\ 0\ {-1}\ 1)$; essa configuração será representada pelas quatro ruas $(h\ 0\ {-1}\ 0)$, $(h\ 0\ 0\ 1)$, $(v\ 0\ {-1}\ 0)$ e $(v\ 0\ 0\ 1)$.) Para que o sistema seja eficiente não deve haver cruzamentos só de duas ruas ambas horizontais ou ambas verticais. Admite-se que todas as ruas podem ser percorridas em ambos os sentidos.

Trabalho desenvolvido no laboratório da Secção de Sistemas e Controlo do Departamento de Engenharia Electrotécnica do IST.

Original Recebido em Novembro de 1990

2.2 Movimento dos AGVs

O ponto de partida de cada AGV é especificado pelas suas coordenadas x e y, e identicamente para o ponto de chegada. Deverá ser um extremo de qualquer rua, incluindo cruzamentos. Um AGV pode então ser representado por uma lista do tipo $(x_1\ y_1\ x_2\ y_2)$, em que (x_1, y_1) é o ponto de partida e (x_2, y_2) o de chegada. Os pontos de partida e de chegada devem ser diferentes entre si. Admite-se que todos oa AGVs se movem à mesma velocidade, tomando-se como unidade o tempo que um AGV leva a percorrer uma rua de comprimento unitário. O tempo total gasto por um AGV num percurso é então a soma dos comprimentos das ruas que percorreu com o número de instantes em que esteve parado. O sistema procura minimizar duas quantidades: a média para todos os AGVs do tempo gasto no percurso e o máximo dos tempos individuais gastos, que é o tempo que a operação global leva a completar-se.

Admite-se que se dá uma colisão sempre que dois ou mais AGVs se encontram na mesma rua, ou seja: uma rua só pode ser percorrida por um AGV de cada vez, em qualquer sentido. Note-se que podem estar dois AGVs parados no mesmo cruzamento, desde que ocupem ruas diferentes. A situação de "troca de ruas", em que num dado instante o AGV A está na rua 1 e o AGV B está na rua 2, e o no instante seguinte A está em 2 e B está em 1, é também considerada uma colisão. Só são consideradas válidas soluções em que não haja colisões de qualquer dos tipos.

2.3 Forma da solução

O sistema CITRA procura sempre encontrar uma solução, mesmo que não seja a óptima. Acusará erro se o problema for impossível. A solução é calculada *off-line*, sem limites ao tempo de cálculo necessário, e apresentada graficamente, através de animação desde o início até ao fim das várias operações. O CITRA fornece igualmente como saída utilizável por processos posteriores uma lista ordenada dos movimentos que cada AGV tem de efectuar em cada instante para ir do seu ponto de partida até ao ponto de chegada. Estes movimentos podem ser:

- **u:** Para cima.
- **d:** Para baixo.
- **l:** Para a esquerda.
- **r:** Para a direita.
- **w:** Espera.

Por exemplo, para percorrer a rua (h 0 0 4) seguida da rua (v 4 0 -3) a sequência de movimentos é **rrrrddd**.

O sistema garante que nenhum dos AGVs tem alguma vez que recuar, isto é as listas de movimentos nunca conterão as sequências **u(w)d**, **d(w)u**, **r(w)l** e **l(w)r**, onde **(w)** representa zero ou mais **w**. Apresenta também no final os valores do tempo médio e máximo correspondentes à solução adoptada.

3 Resolução do Problema

O primeiro passo da solução do problema consiste em encontrar para cada AGV o caminho óptimo desde o ponto de partida até ao ponto de chegada, independentemente dos outros AGVs. Desde que exista pelo menos um caminho, ele é de certeza encontrado, e se existir mais que um caminho possível (que é obviamente o caso normal) o sistema garante que o caminho encontrado é de facto o óptimo, ou seja, o de menor comprimento[1]. Em seguida comparam-se os percursos gerados para todos os AGVs, de forma a detectar se há ou não colisões entre AGVs, de acordo com a definição de colisão dada acima. Se não houver qualquer colisão, está imediatamente encontrada a solução do problema, e o CITRA devolve-a. Se houver uma ou mais colisões, o sistema vai então tentar resolvê-las perturbando os caminhos dos AGVs envolvidos nelas; ver-se-á em seguida em mais pormenor como isto é feito. Esta perturbação pode criar novas colisões, e o processo repete-se até ser encontrado um conjunto de caminhos livre de colisões. Este é então devolvido pelo CITRA. A solução encontrada não é necessariamente óptima, no sentido de minimizar os tempos máximo e/ou médio; no entanto, na prática tenderá a sê-lo, porque em cada conjunto de perturbações os caminhos individuais encontrados são novamente os óptimos, sujeitos às restrições introduzidas para resolver as colisões.

3.1 O algoritmo A*

Para tornar mais fácil compreender como o algoritmo A* é utilizado pelo CITRA, é conveniente começar por rever rapidamente o seu funcionamento. O A* foi inicialmente proposto em [1], e mais tarde melhorado em [2]; a referência aqui utilizada [3] incorpora esta correcção.

O A* é geralmente utilizado em inteligência artificial para resolver problemas de travessia com custo mínimo de "OR-graphs" – grafos em que os nós representam os diferentes pontos de um espaço de estados, e os ramos partindo de um nó representam os diferentes operadores que é possivel aplicar nesse ponto; há que escolher um só operador dentre as várias alternativas – daí o "OR".

O A* faz busca "melhor primeiro", isto é , em cada passo expande, dentre os nós do grafo já atingidos mas ainda não expandidos, o nó que parece mais prometedor. O objectivo é chegar à solução o mais depressa possível, sem ter que fazer a expansão exaustiva de todos os nós do grafo. Para isso o A* necessita de saber o custo de chegar a um nó, e de ter uma estimativa do custo de chegar ao objectivo a partir do nó. Prova-se [3] que se esta medida nunca sobrestimar o verdadeiro custo, o algoritmo encontra necessariamente a solução de custo mínimo. Por outro lado, se a medida subestimar demasiado o custo isso levará a perda de esforço em expansões inúteis.

1 Note-se que um AGV isolado nunca tem que parar para esperar que os outros passem.

Na totalidade o A* requer 6 parâmetros: nó inicial, função que calcula o custo de atingir um nó a partir doutro, função que calcula uma estimativa do custo futuro, predicado que detecta se o objectivo foi atingido, função que gera os sucessores de um nó, e um parâmetro que indica se se pretende apenas encontrar a solução, ou também interessa saber o caminho seguido para lá chegar.

3.2 Procura de caminhos

A procura de um caminho de um ponto inicialpara um ponto final através de uma determinada planta é um problema típico de busca em espaços de estados, em que o espaço de estados é o conjunto de todos os "pontos válidos" da planta. Em primeira análise estes são todos os pontos da planta (que são em número finito porque as coordenadas são inteiras e limitadas). Na prática é mais eficiente considerar que são apenas os cruzamentos e os pontos de onde os AGVs podem partir e onde podem chegar, ou seja: pertence ao espaço de estados um ponto que seja extremo de uma ou mais ruas. Isto é assim porque em pontos no meio de ruas só há uma alternativa útil, que é andar para a frente, e portanto não há qualquer decisão a tomar. Por conveniência, os operadores que fazem a transição de um ponto do espaço de estados para outro consideram-se como sendo as próprias ruas. Uma vez que não é permitido recuar, e só há ruas verticais e horizontais, em cada ponto há no máximo três alternativas, e portanto o factor de ramificação é convenientemente baixo.

Dado um ponto de partida, o conjunto de todos os percursos possíveis a partir dele pode ser representado por um grafo em que cada nó representa um ponto do espaço de estados, e cada ramo partindo de um ponto representa a aplicação de um operador válido nesse ponto. No caso do CITRA este grafo é a própria planta, ou mais precisamente a parte dela que for atingível a partir do ponto inicial. O A* é obviamente aplicável a esta situação. Os parâmetros utilizados são:

- Nó inicial: Ponto de partida do AGV
- Custo de atingir um nó a partir doutro: Comprimento da rua que liga directamente os dois pontos correspondentes. (Se essa rua não existir, a passagem de um nó para o outro nem sequer é considerada)
- Estimativa do custo futuro: Distância que se percorreria se houvesse ruas ligando o nó directamente ao objectivo, uma horizontal para cobrir a diferença de coordenadas X, e uma vertical para cobrir a diferença de coordenadas Y. Esta estimativa garante a optimalidade da solução.
- Detecção do objectivo: O objectivo foi alcançado se o ponto atingido for o ponto de chegada do AGV.
- Sucessores de um nó: Pontos que estão directamente ligados a ele por ruas, exceptuando o ponto em que o AGV esteve imediatamente antes.
- O nó-objectivo final é conhecido (é o ponto de chegada do AGV), e o que se pretende encontrar é o caminho para lá chegar.

3.3 Resolução de colisões

Achados os percursos individuais é no entanto necessário garantir ainda que eles não implicam colisões entre AGVs. Esta é de longe a parte mais complexa, pois envolve interacção entre os diferentes AGVs e portanto um conjunto de possibilidades muito maior. Encontrar um algoritmo de optimalidade garantida que não passe por uma busca exaustiva é uma tarefa extremamente difícil, senão impossível. O método adoptado é o seguinte. Considere-se um espaço de estados em cada elemento é um conjunto de caminhos para todos os AGVs, ou seja, um "candidato a solução". Podemos então aplicar o algoritmo A* à resolução de colisões da forma que se descreve a seguir.

O nó inicial é (em primeira aproximação) o conjunto dos caminhos óptimos gerados isoladamente para os vários AGVs. Pode-se definir uma medida, designada por "quantidade de conflito", que é calculada da forma que se segue. Consideremos por exemplo uma rua cuja ocupação desde o princípio até ao fim da "solução" foi (0 0 1 1 0 0 2 2 2 2 3 3 0 0 1 1). A contribuição dessa rua para a quantidade de conflito é $2+2+2+2+3+3=14$, ou seja, a soma para todos os instantes do número de AGVs que a ocupam instante após instante, excepto se forem apenas 0 ou 1 (uma vez nesse caso não há colisão na rua). A quantidade de conflito é a soma deste resultado para todas as ruas, que pode também ser entendida como a soma para todos os AGVs do tempo passado em colisão com outros AGVs. Um nó é um nó de sucesso (objectivo atingido) se para ele a quantidade de conflito for nula. Em cada passo do algoritmo interessa apenas expandir o nó mais prometedor, que é de todos os nós ainda não expandidos aquele para o qual a quantidade de conflito é menor. A quantidade de conflito é uma medida da distância do nó à solução, e desempenha portanto o papel de estimativa do custo futuro. Como interessa apenas chegar o mais rapidamente possível a uma solução, e a sequência de tentativas feita para lá chegar é irrelevante, o custo passado deve ser sempre nulo.

Os sucessores de um conjunto de caminhos são os conjuntos de caminhos que se obtém tentando resolver as colisões desse caminho de todas as formas possíveis. Simplificadamente isto significa o seguinte. Uma colisão ocorre quando dois ou mais AGVs tentam ocupar uma dada rua durante intervalos de tempo que se sobrepõem total ou parcialmente. Se houver N AGVs envolvidos numa colisão podemos considerar que há N maneiras de a resolver, consistindo cada uma delas em **reservar** a rua para cada um dos N AGVs durante o tempo em que ele a pretende ocupar, obrigando portanto os restantes a procurar alternativas, uma das quais é de esperar até que a rua fique livre. (Não interessa desviar todos os AGVs, já que a utilização do troço em conflito faz parte de um caminho já óptimo, logo as alternativas são necessariamente de custo maior ou igual.) Para resolver (ou pelo menos tentar resolver) todas as M colisões de um dado conjunto de caminhos é necessário fazer M

reservas, uma para cada colisão. Como uma dada solução de uma colisão pode ser combinada com todas as soluções de todas as outras, se a colisão i envolver N_i AGVs o número de sucessores do nó será $\Pi_{i=1,..,M} N_i$. O factor de ramificação depende portanto do número de colisões que se verificam, e do número de AGVs envolvido em cada uma delas.

Um sucessor particular obtém-se fazendo um conjunto particular de M reservas, e gerando um novo conjunto de caminhos que tome em conta essas reservas, e que nessa situação seja ainda o conjunto óptimo, em termos de minimizar os tempos médio e total gastos. Isto pode ser feito pelo método já utilizado para encontrar a primeira tentativa, mas generalizado para:

- Incluir na geração de sucessores a verificação, para cada rua possível, da existência ou não de reservas feitas para outros AGVs e que se sobreponham ao período de tempo durante o qual o AGV a pretende utilizar.
- Incluir nos sucessores possíveis de um cruzamento não só as ruas que saem desse ponto, como a espera na rua através da qual o AGV chegou a esse cruzamento, até que surja um intervalo livre na rua seguinte, de duração igual ou superior ao tempo necessário a percorrê-la. Estas esperas serão geradas para cada uma das ruas seguintes possíveis de serem ocupadas.
- Incluir o tempo gasto em esperas no cálculo do custo passado, de forma a que os caminhos envolvendo esperas sejam tratados no A* de forma inteiramente idêntica à dos restantes, levando à escolha do caminho de duração mínima qualquer que seja a combinação de esperas e desvios envolvida nele.

Feita esta generalização verifica-se que o conjunto de caminhos inicial é apenas o caso particular que se obtém quando não há quaisquer reservas feitas. O sistema pode então ser simplificado considerando antes como nó inicial da resolução de conflitos o conjunto de caminhos nulos, em que portanto também não são feitas quaisquer reservas, e obtendo-se automaticamente a solução inicial como sucessor único desse nó. O funcionamento do sistema toma então a forma de uma chamada ao algoritmo A* para resolução de colisões, em que na função de geração de sucessores o mesmo algoritmo é recursivamente chamado, com parâmetros diferentes, para procura de caminhos. As considerações acima contemplam apenas as colisões por ocupação simultânea de uma rua. Para evitar também as colisões por "troca de ruas" é necessário na detecção de sucesso verificar também a existência ou não desse tipo de colisões, e além de gerar reservas gerar também proibições, da forma que se segue.

Cada "colisão por troca" pode ser resolvida de duas formas: proibindo o AGV A de passar para a rua 2 no instante seguinte, ou proibindo o AGV B de passar para a rua 1 nesse instante. Havendo N colisões destas haverá então 2^N formas de as resolver simultaneamente. Cada uma destas alternativas tem então que ser combinada em cada passo do algoritmo com todas as maneiras já vistas de resolver as colisões do outro tipo, para gerar os sucessores de um dado conjunto de caminhos. Na geração de sucessores ao nível da procura de caminhos a existência de proibições deve agora ser também levada em conta. Se o AGV sobre o qual recai a proibição optar por esperar na rua em que está poderá então haver uma colisão por ocupação nessa rua. Esta será resolvida desviando dela um dos AGVs, resolvendo consequentemente a anterior colisão por troca.

Resumindo, os parâmetros passados ao A* para resolução de colisões são:

- Nó inicial: Conjunto de caminhos nulos (isto é, listas de movimentos vazias).
- Custo de atingir um nó a partir doutro: Zero, uma vez que não estamos interessados na sequência utilizada, como já se viu.
- Estimativa de custo futuro: Quantidade de conflito.
- Detecção do objectivo: Quantidade de conflito nula.
- Sucessores de um nó: Conjuntos de caminhos obtidos procurando resolver as colisões nesse nó de todas as formas possíveis introduzindo restrições aos percursos (no nó inicial as restrições são nulas).
- Pretende-se apenas conhecer o nó-objectivo atingido, independentemente do caminho seguido para lá chegar.

Obtida finalmente uma solução, ela é convertida para a forma de uma sequência para cada AGV de operadores **u, d, l, r** e **w**, tal como já descrito, e apresentada graficamente como simulação animada.

4 Implementação

O sistema encontra-se implementado na linguagem LISP, por ser esta mais adequada que as linguagens convencionais à programação de algoritmos de busca heurística. Em particular foi utilizado o dialecto muLISP-86 [4], destinado a microcomputadores tipo IBM PC e compatíveis, o que garante uma grande portabilidade e facilidade de utilização do sistema.

O sistema CITRA apresenta-se ao utilizador como uma função LISP, que pode ser utilizada no ambiente muLISP como qualquer outra função, passando-lhe os parâmetros adequados (planta da fábrica e AGVs na forma descrita, e diversas opções de funcionamento) e recebendo os resultados (lista de movimentos para cada AGV). Para o utilizar é portanto vantajoso, embora não imprescindível, conhecer o muLISP. Para a visualização da solução é necessário um PC com placa gráfica CGA ou superior; independentemente disso, no entanto, o sistema devolve sempre a lista-solução como valor da função.

Informações detalhadas sobre o sistema e a sua utilização podem ser encontradas no relatório técnico e manual do utilizador [5].

5 Conclusão

O sistema CITRA foi já testado num conjunto representativo de exemplos, tendo-se obtido resultados excelentes, isto é, óptimos ou sem melhoramentos aparentes a fazer. O tempo de cálculo necessário varia naturalmente com a complexidade do problema, mas nos

casos testados e num PC/AT é tipicamente da ordem dos minutos.

Referências

[1] Hart, P.E., Nilsson, N.J. e Raphael, B., (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions on SSC*, 4.

[2] Hart, P.E., Nilsson, N.J. e Raphael, B., (1972). Correction to 'A Formal Basis for the Heuristic Determination of Minimum Cost Paths', *ACM SIGART Newsletter*, **37**.

[3] Rich, Elaine, (1983). *Artificial Intelligence*, McGraw-Hill.

[4] Rich, Albert D. , (1986). *Microsoft* LISP–*Artificial Intelligence for the MS-DOS Operating System*, Soft Warehouse Inc..

[5] Domingos, Pedro M.D., (1988). *CITRA – Controlo Inteligente de Tráfego para Redes de AGVs: Relatório Técnico e Manual do Utilizador*, Instituto Superior Técnico.