TRABALHO PRÁTICO Nº 2

Resolução de Problema de Decisão/Otimização usando Programação em Lógica com Restrições

Descrição

Objetivo: O objetivo deste trabalho é a construção de um programa em Programação em Lógica com Restrições para a resolução de um dos problemas de otimização ou decisão combinatória sugeridos neste enunciado. Adicionalmente, deverá ser elaborado um artigo descrevendo o trabalho realizado e os resultados obtidos.

Sistema de Desenvolvimento: O sistema de desenvolvimento recomendado é o SICStus Prolog, que inclui um módulo de resolução de restrições sobre domínios finitos: clp(FD).

Condições de Realização

Constituição dos Grupos: Os grupos serão os mesmos do primeiro trabalho a menos que haja algum fator impeditivo. Nesse caso, os estudantes deverão entrar em contacto com o docente das aulas teórico-práticas.

Datas Importantes:

A partir de 22/11/2013	Escolha do enunciado no *Moodle* e posterior envio de *email* para hlc@fe.up.pt com a constituição do grupo (caso seja o mesmo do TP1 basta indicar o número do grupo) e com a escolha concretizada no *Moodle*. O assunto do *email* deverá ser da forma **PLOG 2013 TP2 Turma #TURMA**
12/12/2013, até às 12h00	(Opcional) Submissão, via *Moodle*, de versão preliminar do artigo (em formato Word ou PDF), com vista à obtenção de feedback do docente das aulas teórico-práticas. Esta submissão é opcional e não tem efeito na avaliação, tendo como objetivo alertar os grupos para eventuais falhas no artigo que possam comprometer a sua avaliação final. O nome do ficheiro deve ser da forma `PLOG_TP2_Pre_G#Grupo_T#Trabalho.DOC`. (Exemplo: `PLOG_TP2_Pre_G1_T1.DOC`)
15/12/2013	Entrega, via *Moodle*, da versão final do artigo (formato PDF) e do código fonte desenvolvido. Submeter um único ficheiro ZIP com nome da forma `PLOG_TP2_FINAL_G#Grupo_T#Trabalho.ZIP`. (Exemplo: `PLOG_TP2_FINAL_G1_T1.ZIP`)
16-20/12/2013	Demonstrações dos trabalhos nas aulas teórico-práticas.

Valorização da Avaliação: Ver ficha da Unidade Curricular no SIGARRA.

Esclarecimentos: Esclarecimentos sobre o enunciado e a realização do trabalho devem ser obtidos, preferencialmente, junto dos docentes no decorrer das aulas teórico-práticas.

TRABALHO PRÁTICO Nº 2

Artigo

Cada grupo deve elaborar e entregar um artigo e realizar uma demonstração da aplicação desenvolvida. O artigo deverá ser escrito em Português ou Inglês no formato LNCS (Lecture Notes in Computer Science) da Springer. Para tal, a Springer disponibiliza no seu sítio *templates* em Word e LaTeX (http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0), que deverão ser utilizados para a escrita do artigo. O artigo deverá ter entre 6 a 8 páginas (excluindo anexos). Sugere-se que contenha as seguintes partes:

Título: que transpareça o trabalho realizado (não tem que ser apenas o tema do trabalho).

Autores e Afiliações: identificação do trabalho e do grupo. Nas afiliações colocar a turma e o grupo segundo o seguinte exemplo: FEUP-PLOG, Turma 3MIEIC9, Grupo 123.

Resumo/Abstract: Deve contextualizar e resumir o trabalho, salientando o objetivo, o método utilizado e fazendo referência aos principais resultados e à principal conclusão que esses resultados permitem obter.

1. Introdução/Introduction: Descrição dos objetivos e motivação do trabalho, descrição do problema, (idealmente, referência a outros trabalhos na mesma área + descrição muito sucinta da aproximação utilizada, salientado as principais diferenças em relação aos outros trabalhos), e estrutura do resto do artigo.

2. Descrição do Problema/Problem Description: Descrever sucintamente o problema de otimização ou decisão em análise.

3. Ficheiros de Dados/Data Files: Descrever os problemas a resolver e o conteúdo dos respetivos ficheiros de dados. Devem ser construídos pelo menos três problemas distintos em ficheiros de texto separados. (Idealmente: utilização de *datasets* usados noutros trabalhos de outros autores).

4. Variáveis de Decisão/Decision Variables: Descrever as variáveis de decisão e os seus domínios.

5. Restrições/Constraints: Descrever as restrições rígidas e flexíveis do problema e a sua implementação utilizando o SICStus Prolog.

6. Função de Avaliação/Evaluation Function. Descrever, quando for o caso, a forma de avaliar a solução obtida e a sua implementação utilizando o SICStus Prolog.

7. Estratégia de Pesquisa/Search Strategy: Descrever a estratégia de etiquetagem ("*labeling*") implementada, nomeadamente no que diz respeito à ordenação de variáveis e valores. Deve também ser descrita a forma de implementação desta estratégia utilizando a linguagem.

8. Visualização da Solução/Solution Presentation. Explicar os predicados para visualizar a solução em modo de texto (obrigatório) ou modo gráfico (opcional).

9. Resultados/Results: Demonstrar exemplos de aplicação em casos práticos com diferentes complexidades e analisar os resultados obtidos. Devem ser utilizadas formas convenientes para apresentação dos resultados (tabelas e/ou gráficos). (Idealmente: comparação de resultados com outros autores/trabalhos.)

10. Conclusões e Perspetivas de Desenvolvimento/Conclusions and Future Work: Que conclusões retiram deste projeto? O que mostram os resultados obtidos? Quais as vantagens e limitações da solução proposta? (Idealmente: conclusão sobre a comparação com os resultados obtidos por outros autores/trabalhos.) Como poderiam melhorar o trabalho desenvolvido?

Bibliografia: Livros, artigos, páginas Web, usados para desenvolver o trabalho, apresentados segundo o formato sugerido no *template*.

Anexos: Código fonte, ficheiros de dados e resultados, e outros elementos úteis que não sejam essenciais ao relatório (não são contabilizados para o limite de 6 a 8 páginas).

Problemas de Otimização/Decisão Sugeridos

Puzzles (2D/3D)

1. Hanjie - www.puzzlemix.com/Hanjie
2. Hashi - http://www.conceptispuzzles.com/index.aspx?uri=puzzle/hashi
3. Hoo-Doo - http://www.jaapsch.net/puzzles/hoodoo.htm
4. Izzi - http://www.jaapsch.net/puzzles/izzi.htm
5. KenKen - http://www.kenken.com/game
6. One Two and Three - http://logicmastersindia.com/lmitests/dl.asp?view=1&attachmentid=277
7. Rubik's Maze - http://www.jaapsch.net/puzzles/rubmaze.htm
8. Snake - http://logicmastersindia.com/lmitests/dl.asp?attachmentid=379&view=1
9. The Great Pyramid Pocket Puzzle - http://www.jaapsch.net/puzzles/pyramid.htm
10. Water Fun - http://logicmastersindia.com/lmitests/dl.asp?view=1&attachmentid=277

Problemas de Otimização

11. Escalonamento de Dispositivos Consumidores de Energia Elétrica
12. Gestão de Servidores *Cloud*
13. Calendarização de Competição Desportiva
14. Geração de Horários Escolares Universitários
15. Aterragem de Aviões

Descrição Resumida dos Problemas

Puzzles 2D/3D

Consultar os sítios para informações de problemas. A abordagem deve permitir lidar com tamanhos diferentes de tabuleiros e números diferentes de peças. É valorizada a geração dinâmica de problemas, e.g. gerar aleatoriamente o problema a ser resolvido. Deve ser possível visualizar a solução em modo de texto, de uma forma que facilite a sua validação.

Problemas de otimização

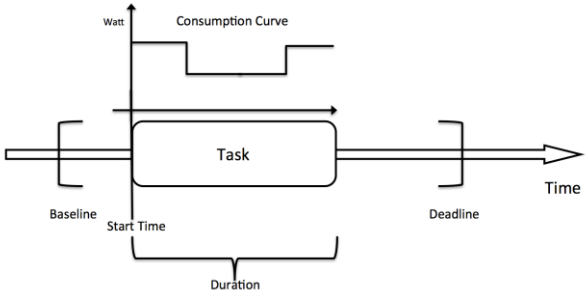
Seguem-se as descrições dos problemas. As abordagens devem permitir problemas com diferentes dimensões. São valorizadas experiências com dimensões elevadas. Deve ser possível visualizar a solução em modo de texto, de uma forma que facilite a sua validação.

11. Escalonamento de Dispositivos Consumidores de Energia Elétrica

As chamadas *smart grids* tornam potencialmente lucrativo o escalonamento inteligente dos dispositivos que consomem energia elétrica. Alguns desses dispositivos, dada a sua natureza, não são escalonáveis, como por exemplo candeeiros ou televisões. Outros dispositivos podem ser escalonados para uma qualquer altura dentro de uma janela temporal, como por exemplo máquinas de lavar, fornos, ou termoacumuladores.

Os dados deste problema são os seguintes:

- Tarefas: são atividades consumidoras de energia, definidas por uma *baseline* (data de início mais cedo possível), *deadline* (data mais tarde para conclusão da tarefa), duração, início e consumo energético (ver figura).



Fonte: Jaime Azevedo, Effective Scheduling of Energy Consumption in Smart Grids, MSc Thesis, FEUP, 2013.

  Dependendo do dispositivo, o consumo energético ao longo da execução da tarefa pode ser constante ou variável, sendo conhecido de antemão.

- Preço da energia: assume-se que o preço da energia ao longo do dia é conhecido, variando consoante a hora do dia.
- Energia disponível: existe uma potência contratada fixa, sendo que a energia disponível em cada hora do dia é a remanescente após retirado o consumo previsto da operação dos dispositivos não escalonáveis.

Com base nestes dados, pretende-se obter um escalonamento das tarefas escalonáveis, tendo em conta que:

- Todas as tarefas devem ser escalonadas numa janela temporal de 24h;
- Todas as tarefas devem ser executadas dentro da sua janela permitida, entre o *baseline* e o *deadline*;
- Em nenhum momento o consumo de energia pode ultrapassar a energia disponível.

Como critérios de otimização, predem-se explorar as seguintes situações:

- Obter o escalonamento com o menor custo a pagar pela energia ao longo do dia;
- Obter um escalonamento uniforme em termos de consumo energético, que permita avaliar a hipótese de baixar a potência contratada.

Defina o problema como um problema de satisfação de restrições, e resolva-o com PLR, de modo a que seja possível resolver problemas desta classe com diferentes parâmetros, isto é, deverá ser possível resolver problemas com mais ou menos tarefas escalonáveis, com restrições parametrizáveis relativas aos custos da energia, etc.

TRABALHO PRÁTICO Nº 2

12. Gestão de Servidores *Cloud*

Uma empresa que fornece serviços *cloud* possui vários servidores, destinados a funções diferentes, tais como distribuição de processamento, autenticação, armazenamento e execução de *scripts*. Existe apenas um servidor principal de distribuição (doravante designado por SPD), cujo objetivo é decidir para que servidor deve enviar os pedidos que lhe chegam. Esta decisão depende do tipo de pedidos que chegam à *cloud* e também da carga que já exista nos outros servidores. Se chegarem mais pedidos enquanto o SPD ainda está a processar um pedido, estes entram numa fila. O tamanho da fila de pedidos pendentes no SDP é configurável, mas tem custos extra. É possível enviar alguns dos pedidos desta fila para um servidor secundário de distribuição (SSD). Porém, o SSD possui menos recursos e induz atrasos significativos na comunicação, pelo que a delegação deve ser evitada sempre que possível. Contudo, um pedido é automaticamente direcionado para o SSD quando a fila do SPD está cheia.

O subsistema de servidores deve ser dimensionado de forma a reduzir tanto o tempo de resposta como o custo. Existem três fatores principais a considerar: memória em disco, memória RAM e processamento. Dispõe-se de 20.000€ para dimensionar todo o sistema. Tem que haver pelo menos o dobro de servidores de execução de *scripts* em relação aos de autenticação.

Os dados referentes à configuração base dos servidores, tempos de execução de tarefas, preços de *hardware* e requisitos dos diferentes tipos de pedidos encontram-se nas tabelas seguintes.

Configuração base	Espaço	RAM	CPU
Servidor de autenticação	120	100	200
Servidor armazenamento	4000	85	75
Servidor execução *scripts*	70	400	1000

Tarefa	Tempo (ms)
Decisão de encaminhamento do SPD	100
Decisão de encaminhamento do SSD	250
Delegação do SPD para o SSD	50
Pedido ao servidor de autenticação	400
Pedido ao servidor de armazenamento	850
Pedido ao servidor de execução de *scripts*	100

Hardware	Preço (€)
Cada "slot" na fila de pedidos do SPD	50
Cada "slot" na fila de pedidos do SSD	30
Servidor de autenticação (config. base)	1000
Servidor armazenamento (config. base)	1800
Servidor execução *scripts* (config. base)	2800
250 unidades de espaço adicionais	100
400 unidades de RAM adicionais	250
300 unidades de CPU adicionais	420

Tipo de pedido	Requisitos (unidades)		
	Espaço	RAM	CPU
Autenticação	50	80	75
Armazenamento	120	40	30
Execução de *scripts*	30	150	220

O sistema deve ser dimensionado para cem milhões de pedidos por dia (100.000.000) onde 25% dos pedidos são de autenticação, 60% de execução de *scripts* e 15% de armazenamento. Todos os pedidos passam pelo SPD numa primeira instância, podendo ser delegados para o SSD, com os custos referidos acima. Como deve ser planeado o sistema de forma a minimizar o tempo de resposta do sistema e os custos de implementação? Defina o problema como um problema de satisfação de restrições e resolva-o com PLR, de modo a que seja possível resolver problemas desta classe com diferentes parâmetros. Isto é, deverá ser possível resolver problemas para diferentes quantidades de pedidos, com restrições parametrizáveis relativas aos tempos de execução dos pedidos, custos, etc.

13. Calendarização de Competição Desportiva

Numa competição desportiva do tipo *round robin*, tal como o campeonato nacional de futebol, cada equipa joga contra cada uma das restantes equipas uma vez em cada uma das voltas da competição. Tipicamente, numa competição com duas voltas, o calendário da segunda volta é idêntico ao da primeira, mantendo-se a ordem dos jogos e variando apenas o local de realização (isto é, uma equipa que jogou fora na primeira volta joga em casa na segunda, e vice-versa).

Cada equipa está sedeada numa determinada cidade, devendo ser evitada a realização de mais do que um jogo na mesma cidade numa dada jornada. Cada equipa deve alternar ao máximo os jogos em casa e fora (ou seja, se numa dada jornada, uma equipa jogou em casa, na seguinte deverá jogar fora).

Poderá ainda considerar o efeito das deslocações quando a competição se realiza em territórios vastos (como seja o caso dos Estados Unidos da América), tentando minimizar as deslocações de uma equipa, e.g. juntando os jogos a realizar fora numa determinada região ou cidade em jornadas consecutivas (note-se que este objetivo pode colidir com o de alternância casa/fora).

Interessa também definir grupos de clubes cujos jogos simultâneos em casa devem ser balanceados (por exemplo, em cada jornada dois dos quatro maiores clubes devem jogar em casa, e os restantes dois fora).

O sistema, implementado recorrendo a PLR, deve permitir uma parametrização do número de equipas e de voltas. Deve ainda permitir a especificação de restrições de precedência entre jogos ou jornadas mínimas e máximas para a ocorrência de jogos entre determinadas equipas. Deverá ser possível resolver problemas para diferentes números de equipas, com restrições parametrizáveis relativas aos critérios a utilizar para efeitos de otimização, etc.

14. Geração de Horários Escolares Universitários

O problema de Geração de Horários de um curso universitário (como o MIEIC) consiste em definir o dia, hora e sala em que cada aula será lecionada.

Considera-se que estão já atribuídas disciplinas e turmas aos docentes – cada docente tem atribuído um conjunto de aulas (T, TP ou PL) lecionadas a uma turma ou conjunto de turmas (totalizando cerca de 10 horas de aulas). Cada ano letivo possui diversas disciplinas e várias turmas (4 a 6, no caso do MIEIC), sendo normal que nas aulas teóricas as turmas sejam agrupadas.

Cada docente tem um conjunto de restrições sobre o seu horário (cada 'slot temporal'), podendo as mesmas ser rígidas ou flexíveis. Restrições e preferências semelhantes existem relativamente às turmas e salas. Por exemplo, algumas turmas poderão ter preferência por um horário matinal, enquanto outras terão preferência por aulas da parte da tarde.

Cada sala tem um tipo (Normal, Informática, etc.) e uma dada capacidade (em número de alunos). Cada tipo de aula de cada disciplina necessita de um dado tipo de sala pré-determinado e cada turma tem um número de alunos definido.

Construa um programa em PLR para resolver o problema da geração de horários. Deve-se garantir que nenhum docente, turma ou sala tem mais de uma aula em simultâneo, minimizando o número de 'buracos' e aulas isoladas nos horários de docentes e turmas. Deve-se ainda tentar respeitar as preferências de horário de docentes e turmas. Deverá ser possível resolver problemas para diferentes números docentes, disciplinas, turmas e salas, com restrições parametrizáveis relativas às preferências, capacidades das salas, etc.

15. Aterragem de Aviões

Numa pista de aterragem de um aeroporto é necessário escalonar as aterragens dos aviões que vão chegar. Cada avião *i* tem uma hora preferencial p_i para aterragem (determinada por exemplo tendo em conta o consumo ótimo de combustível) e uma janela temporal [a_i, b_i] que limita as suas possibilidades de aterragem (tendo em conta o tempo mais cedo possível para o avião chegar e o limite de combustível).

Duas funções de custo estão associadas a uma aterragem numa hora diferente de p_i, tendo em conta o acréscimo de consumo de combustível:

- α_i determina o custo por unidade de tempo de uma aterragem antecipada;
- β_i determina o custo por unidade de tempo de uma aterragem retardada.

Após cada aterragem, existe um período δ_i durante o qual a pista não pode ser utilizada, dependente do tipo de avião que acabou de aterrar.

Cada avião cuja aterragem deve ser escalonada é pois representado por: *voo(i,*a_i*,*p_i*,*b_i*,*α_i*,*β_i*,*δ_i*)*. Por exemplo, o voo representado por *voo(1,30,70,90,2,3,4)* indica que o avião 1 consegue aterrar entre os tempos 30 e 90, mas prefere aterrar no tempo 70. Aterrar num tempo *t* entre 30 e 69 tem um custo de 2*(70-*t*), e aterrar num tempo *t* entre os tempos 71 e 90 tem um custo de 3*(*t*-70). Após aterrar, a pista não poderá ser utilizada nas 4 unidades de tempo subsequentes.

Pretende-se obter um escalonamento das aterragens previstas para um determinado período, tentando minimizar os custos associados.

Defina o problema como um problema de satisfação de restrições, e resolva-o com PLR, de modo a que seja possível resolver problemas desta classe com diferentes parâmetros, isto é, deverá ser possível resolver problemas com mais ou menos voos, com diferentes números de pistas, com restrições parametrizáveis relativas aos custos de aterragens fora dos tempos preferenciais, etc.