Inteligência Artificial

Relatório Final

Otimização da Distribuição de Público num Espectáculo

Estudantes:
Francisco Tuna Andrade
Luís Manuel Câmara Spínola
Nuno Duarte Ribeiro Silva Fonseca Oliveira

May 20, 2018

Objetivos

O objetivo deste trabalho é aplicar algoritmos de otimização estudados nas aulas (como arrefecimento simulado e algoritmos genéticos) na distribuição de lugares de uma sala de espetáculos. Esta distribuição deve, por um lado, procurar minimizar a distância das trocas a fazer relativamente a uma distribuição inicial de lugares e, por outro, maximizar a coesão de grupos de pessoas inicialmente definidos, isto é, deve-se procurar que pessoas de um mesmo grupo fiquem em lugares contíguos.

Especificação

No nosso projeto iremos fazer algumas assunções acerca do enunciado, nomeadamente que a sala é sempre retangular, que uma pessoa pode pertencer a, no máximo, um grupo e que o número de espetadores a considerar é menor ou igual ao número de lugares disponíveis.

Para a resolução deste problema de otimização serão usados algoritmos genéticos e arrefecimento simulado, dando o nosso programa três opções acerca do processo que quer utilizar chegar à solução:

- Arrefecimento Simulado
- Algoritmos Genéticos
- Arrefecimento Simulado + Algoritmos Genéticos

Neste relatório, considerem-se m e n como as dimensões da sala de espetáculos.

Estados

No nosso trabalho, nós definimos um estado como uma distribuição de pessoas pelos lugares da sala, assumindo que a cada pessoa foi atribuído exatamente um lugar.

Um estado é considerado adjacente a outro se poder ser obtido através de uma permutação de dois lugares. Um destes lugares pode estar vazio, pelo que nesse caso a passagem para um estado adjacente corresponde a uma pessoa mudar de lugar.

A **função de vizinhança**, que permite achar o próximo estado adjacente, e que é importante no caso do algoritmo do Arrefecimento Simulado, é calculada gerando 2 dois pares de números aleatórios que correspondem a dois pares de posições no tabuleiro, que serão as posições das duas pessoas a permutar na passagem de um estado para o outro.

Função de Avaliação

A função de avaliação será a mesma no caso dos Algoritmos Genéticos e no caso do Arrefecimento Simulado. A função de avaliação é então:

$$(\sum(|G|-1)) - \frac{(\sum_{spectator \quad s} distance_s(initialDistribution, finalDistribution)^2)}{\sqrt{m \times n} \times (m^2+n^2)} \tag{1}$$

No primeiro somatório da expressão G corresponde a um conjunto de pessoas do mesmo grupo situados em lugares contíguos. Ou seja, o primeiro somatório corresponde à soma do valor absoluto dos conjuntos de espetadores do mesmo grupo sentados em lugares contíguos, ao qual é subtraído o valor de 1. Este primeiro somatório será designado no decorrer do relatório como *groupsPoints*.

Por outro lado, o lado direito da subtração principal da expressão corresponde à soma dos quadrados das distâncias entre a posição inicial de um espetador e a sua posição final a dividir por $\sqrt{(m \times n)} \times (m^2+n^2)$. Este lado direito da expressão será designado no decorrer do relatório por *sumSquareDistances*

Representação dos cromossomas

Note-se que o modo de representação dos cromossomas foi alterado relativamente ao relatório intercalar.

Na aplicação dos algoritmos genéticos, cada cromossoma irá corresponder a uma distribuição de espetadores pela sala.

Um cromossoma será assim constituído por um vetor onde cada elemento desse vetor é um par com dois elementos, sendo que que cada um desses elementos representa uma posição dum tabuleiro. Um cromossoma representa assim uma permutação entre a distribuição inicial de espetadores na sala e outra distribuição sendo o vetor referido acima o conjunto de troca de posições necessárias para chegar de uma distribuição a outra.

A **função de cruzamento** entre os cromossomas A e B corresponde a juntar a parte inicial de um vetor A com a parte final de outro vetor B para gerar um descendente, assim como juntar a parte inicial de um vetor B com a parte final de outro vetor A. O ponto de corte será gerado aleatoriamente e a probabilidade de cruzamento será 10%.

A **função de mutação** significa trocar o primeiro elemento de um pair no vetor de cromossomas com o segundo elemento do pair seguinte no mesmo vetor de cromossomas. A probabilidade de mutação é 0.0001

Aplicação dos Algoritmos

Como já foi referido anteriormente serão usados 3 algoritmos diferentes no nosso projeto.

O primeiro, **Arrefecimento Simulado** consiste em ir gerando estados adjacentes, usando a função de vizinhança já descrita acima até que a temperatura chegue a 0. A probabilidade de transitar para um estado seguinte dependerá da função de avaliação do estado atual e do seguinte, de acordo com a fórmula estudada nas aulas.

O segundo, **Algoritmos Genéticos**, consiste em inicialmente gerar aleatoriamente 20 cromossomas diferentes e aplicar as funções de cruzamento e mutação já referidas anteriormente durante 50 gerações com uma política elitista de 5 e escolher o cromossoma com uma maior função de avaliação.

A terceira opção, a junção dos dois algoritmos mencionados anteriormente, consiste em aplicar o Arrefecimento Simulado 20 vezes para gerar os cromossomas iniciais e, de seguida usar os Algoritmos Genéticos para o obter o cromossoma final que irá corresponder à distribuição final.

Desenvolvimento

No desenvolvimento do nosso programa foi usada a linguagem *C++*, sendo que alguns elementos do grupo usaram o IDE *Eclipse*, enquanto outros usaram o *Visual Studio*.

Estrutura da Aplicação

Na nossa aplicação começa-se por gerar uma distribuição aleatória de espetadores pela sala utilizando alguns parâmetros escolhidos pelo utilizador, que são o número de linha e colunas da sala, o número de espetadores e o número máximo de espetadores.

Na imagem seguinte mostra-se o pedido destes parâmetros ao utilizador.

Figure 1: Generation of a Random Distribution

De seguida é mostrado ao utilizador a distribuição aleatória gerada e a divisão das pessoas pelos grupos, como se pode observar na imagem seguinte:

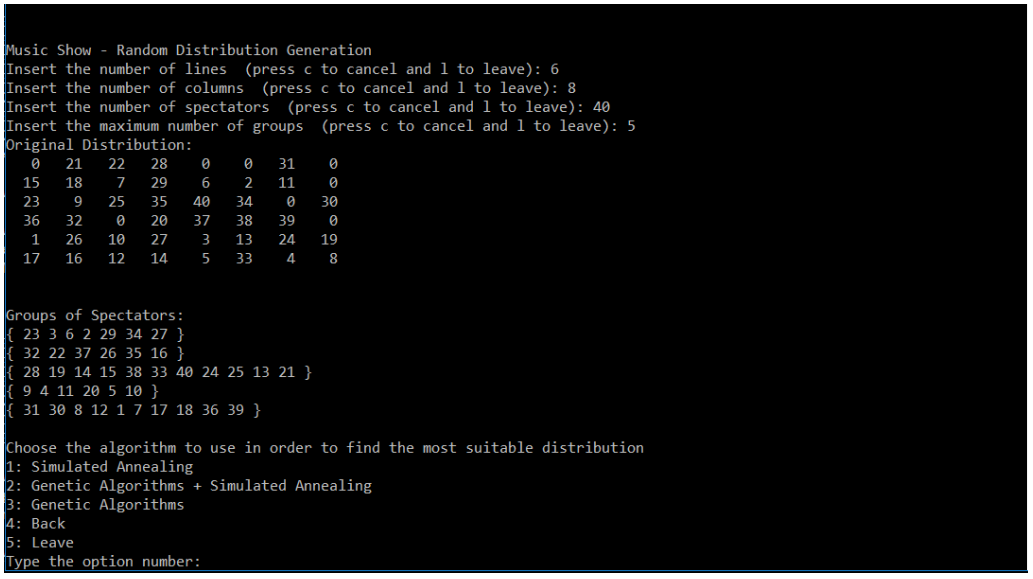

Figure 2: Choice of Algorithm

Por fim, é mostrado ao utilizador a distribuição gerada:

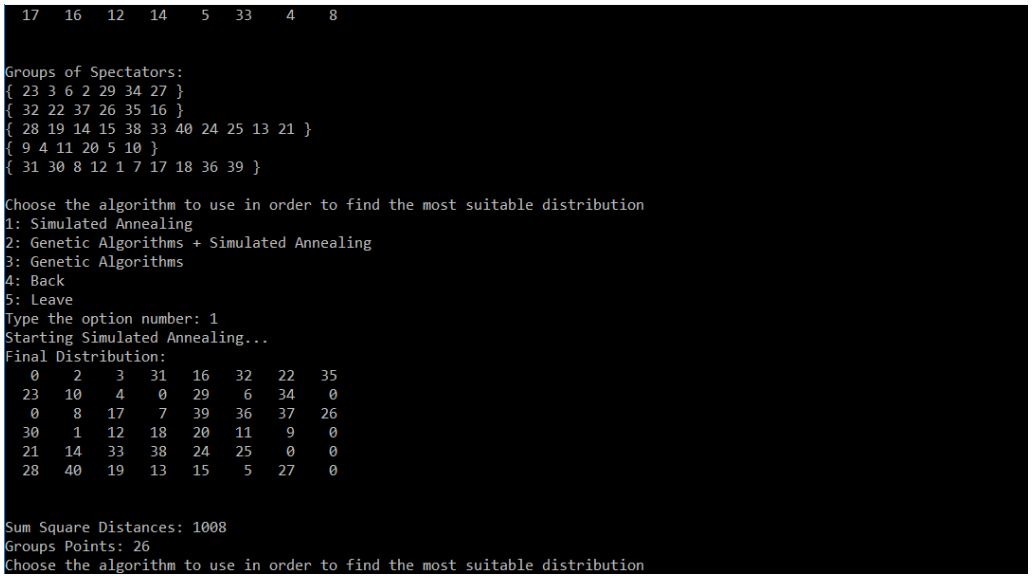

Figure 3: Final Distribution

Implementação

No nosso trabalho a classe principal é a classe *Room* que representa uma distribuição de pessoas pela sala, sendo esta classe que lida com os detalhes do algoritmo *Simulated Annealing*. Foi implementada ainda a classe *Chromosome* que representa um cromossoma que vai ser usado no Algoritmo Genético e aind a classe *GeneticAlgorithm* que implementa o algoritmo Genético.

As funções que lidam com a geração de distribuições aleatórias foram declaradas no ficheiro *Randomizer.h*, enquanto que as funções que implementam os menus a mostrar ao utilizador foram declaradas no ficheiro *Menus.h*. No ficheiro *Algorithms.h*, foram definidas as várias propriedades dos algoritmos como temperatura inicial no Arrefecimento Simulado ou número de gerações nos Algoritmos Genéticos.

Experiências

Nesta secção será analisada a performance dos 3 diferentes algoritmos para salas de tamanhos diferentes. Note-se que na tabela A.S. + A.G. refere-se à aplicação conjunta do Arrefecimento Simulado e dos Algoritmos Genéticos. As variáveis *groupsPoints* e *sumSquareDistances* são as mesmas que foram referidas acima na função de avaliação. Recorde-se que o nosso objetivo é maximizar o *groupsPoints* e minimizar o *sumSquareDistances*. Para todos os testes o número de espetadores é o máximo permitido pelas dimensões da sala e o número de grupos é 10.

Medidas da Distribuição	Arrefecimento Simulado	Algoritmos Genéticos	A.S. + A.G.
5×5 (*groupsPoints*)	12	4	14
5×5 (*sumSquareDistances*)	150	428	98
10×10 (*groupsPoints*)	73	15	74
10×10 (*sumSquareDistances*)	1476	6220	1506
15×15 (*groupsPoints*)	153	32	157
15×15 (*sumSquareDistances*)	10112	26498	7268
20×20 (*groupsPoints*)	259	52	264
20×20 (*sumSquareDistances*)	34732	83548	34008

Table 1: Experiências

Conclusões

Como se pode observar pelos dados recolhidos, a junção do Arrefecimento Simulado e dos Algoritmos Genéticos é o método que apresenta os melhores resultados, o que já seria expetável. O algoritmo Arrefecimento Simulado apresenta bons resultados, mas que nunca são tão bons como o A.S + A.G. Isto talvez se deverá ao facto de o Arrefecimento Simulado ser bastante bom a encontrar máximos locais mas não tanto a encontrar máximos globais de uma função. Quanto ao Algoritmos Genéticos sozinho, este apresenta resultados bastante insatisfatório comparado com os outros. Tal pode-se dever o facto de só terem sido usados 10 gerações. Como na aplicação dos Algoritmos Genéticos sozinhos são gerados cromossomas totalmente aleatórios, 10 gerações não são suficientes para produzir bons resultados. Por outro lado, na junção do Arrefecimento Simulado com os Algoritmos Genéticos, os Algoritmos Genéticos são aplicados a máximos locais da função, o que leva a que 10 gerações sejam suficientes para produzir resultados satisfatórios.

Através da análise do resultados obtidos nesta experiência, consideramo-nos satisfeitos com o nosso projeto, dado que o nosso principal algoritmo, o A.S. + A.G. foi capaz de produzir resultados bastante bons, que julgamos, poderiam ser aplicados numa situação real.

Melhoramentos

Em algoritmos de otmização existe sempre espaço para melhorar, pelo que se nos fosse incumbida a tarefa de desenvolver uma melhoria a este trabalho, focar-nos-íamos em diminuir o tempo de execução e em tentar apresentar melhores resultados, seja por mudanças na função de avaliação ou por mudanças na implementação dos nossos algoritmos.

Recursos

De seguida apresentamos algumas ferramentas que estamos a utilizar no nosso trabalho:

- GitHub `https://github.com/`
- VisualStudio `https://www.visualstudio.com/`
- Overleaf `https://www.overleaf.com/`

Elementos do Grupo

De seguida é mostrada a percentagem que cada estudante contribuiu para a realização do projeto

- Francisco Andrade 57/100
- Luís Spínola 33/100
- Nuno Oliveira 10/100

Apêndice

Manual do Utilizador

- Importar o código src para um IDE de C++, como por exemplo Eclipse
- Compilar o código
- Correr o programa
- Inserir os parâmetros da distribuição aleatória que se quer gerar
- Escolher o algoritmo que se quer utilizar