Universidade do Porto

Faculdade de Engenharia

FEUP

Otimização na organização de um jantar

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo **B2_1**:

António Jorge Aguiar do Vale – up201404572 – up201404572@fe.up.pt
Pedro Daniel Oliveira Pacheco – up201406316 – up201406316@fe.up.pt
Telmo João Vales Ferreira Barros – up201405840 – up201405840@fe.up.pt

21 de Maio de 2017

Objetivo

O objetivo deste trabalho é distribuir pessoas num jantar por mesas para que estas se sintam mais confortáveis. Para isso iremos usar 3 critérios principais, sendo o primeiro a afinidade por idades, ou seja, pessoas com idades próximas devem ficar sentadas na mesma mesa. Depois iremos verificar a afinidade por profissão, desta forma pessoas com profissões semelhantes ou que tenham áreas em comum têm maior probabilidade de ficar juntas. Em último lugar temos a afinidade por hobbies, onde vemos a quantidade de hobbies em comum nas pessoas, e quanto maior for esse número, maior é a probabilidade de estas ficarem na mesma mesa. Sendo assim, iremos construir uma função de avaliação que respeite estes critérios e obtenha uma solução ótima para o nosso problema.

Especificação

Representação de estados

Exemplo: (por motivos de simplificação serão ignoradas as características das pessoas inscritas no jantar).

Pessoas inscritas:	Mesas disponíveis
{A,B,C,D,E} (grupo)	Mesa 1 (2 a 4 lugares)
{F,G,H} (grupo)	Mesa 2 (2 a 4 lugares)
I	Mesa 3 (3 a 5 lugares)
J	Mesa 4 (3 a 5 lugares)

Representação escolhida:

Para cada população um gene representa o número da mesa. O primeiro gene corresponde à mesa da primeira pessoa, o segundo gene à mesa da segunda pessoa e o esquema mantêm-se até todas as pessoas estarem representadas.

Pessoas	A	B	C	D	E	F	G	H	I	J
Representação	10	11	00	11	01	01	11	00	10	10
Mesa	3	4	1	4	2	2	4	1	3	3

Neste exemplo a distribuição pelas mesas seria a seguinte:

Mesa	Pessoas na mesa
Mesa 1	C, H
Mesa 2	E, F
Mesa 3	A, I, J
Mesa 4	B, D, G

De todas as formas de representação pensadas pelo nosso grupo esta foi definitivamente a mais adequada pois mantém o tamanho do cromossoma constante e unicamente dependente do número de pessoas e de mesas. Contudo existem certas restrições que podem facilmente ser violadas por meio desta representação e que por isso surgem na função de avaliação como elemento penalizador da solução. As restrições referidas são as seguintes:

1) Pessoas do mesmo grupo estarem sentadas na mesma mesa (restrição opcional se um grupo for maior que lotação máxima da maior mesa da sala)
2) O número de pessoas na mesa estar entre os limites mínimo e máximo de lotação da mesa (restrição obrigatória)
3) Todas as pessoas estarem sentadas (restrição obrigatória)

Função de *crossover*/mutação

A função de *crossover* utilizada será uniforme, isto é, os descendentes vão ter a contribuição de cada um dos pais com probabilidade de 0,5. O ponto de *crossover* é escolhido aleatoriamente

Relativamente à mutação usaremos o valor de 0,01 de probabilidade de mutação por cada bit e não por gene.

Função de avaliação

Como descrito no enunciado, *"pretende-se que as pessoas se sintam confortáveis com a companhia que vão ter durante o jantar, pelo que devem ser distribuídas de forma a terem alguma afinidade (etária, profissional, hobística, etc)*". A afinidade entre as pessoas será então o fator crucial de toda a função de avaliação.

Além da afinidade, será também importante penalizar os indivíduos que não respeitem as restrições enumeradas no tópico anterior.

As funções de avaliação abaixo são diferentes das apresentadas no relatório intercalar pois após análise mais cuidada dos resultados obtidos chegamos à conclusão que as mesmas levavam a avaliações incorretas. Dentro dos erros destacamos essencialmente a transformação das funções de penalização em quadráticas e cúbicas porque era recorrente a aceitação de soluções ótimas com um número elevado de mesas penalizadas. Para a avaliação da sala na totalidade estava a ser realizada a divisão da avaliação total de todas as mesas pelo número de mesas, sentimos por isso necessidade de atribuir maior peso às mesas com mais pessoas e menor às com menos pessoas pelo que agora essa divisão tem em conta o número de pessoas sentadas na mesa.

Relativamente a uma mesa:

$$afinidadeIdade = 1 - \frac{numIntervalosDiferentes}{numTotalPessoas} \times 100$$

$$afinidadeProfissão = 1 - \frac{numÁreasDistintas}{numTotalPessoas} \times 100$$

$$afinidadeHobbies = \sum_{i=0}^{numHobbiesMesa} \left(\frac{\frac{numPessoasGostamHobby}{numTotalPessoas}}{numHobbiesTotais} \right) \times 100$$

$$fAvaliação = afinidadeIdade \times 0{,}3 + afinidadeProfissão \times 0{,}35 + afinidadeHobbies \times 0{,}35$$

A explicitação das profissões, áreas correspondentes e hobbies encontra-se explicitada nos Anexos I e II para melhor compreensão das funções de avaliação apresentadas.

$$PenalizaçãoGrupos = \sum_{i=1}^{totalGruposMesa} \left(membrosGrupo[i].total - \frac{membrosGrupo[i].naMesa}{membrosGrupo[i].total} \times totalGruposMesa \right)$$

$$PenalizaçãoMesa(minMesa > numPessoa) = Penalização(minMesa \leq numPessoas \leq maxMesa) + (minMesa - numPessoas)^2$$

$$PenalizaçãoMesa(maxMesa < numPessoa) = Penalização(minMesa \leq numPessoas \leq maxMesa) + (numPessoas - maxMesa)^3$$

A função de penalização acima referida aplica-se a cada mesa e procura eliminar soluções que separem os grupos ou que o número de pessoas numa mesa esteja fora do domínio da lotação da mesma.

A função de avaliação de todas as mesas resultará da soma da função de avaliação para cada mesa com um peso relativo ao número de pessoas dessa mesa.

$$Funçãodeadaptação = \frac{\sum_{i=1}^{nMesas}[(fAvaliação(Mesa[i]) - PenalizaçãoMesa) \times numPessoasMesa]}{numTotalPessoas} + 100 - PenalizaçãoGrupos$$

Critérios de paragem

Tendo em conta o facto de a nossa solução ainda não se encontrar implementada e funcional, consideramos ligeiramente precoce a definição de um critério de paragem pois dependerá bastante do comportamento da nossa solução e da eficácia da função de avaliação.

Contudo, um critério de paragem poderá passar pela definição de um valor **n** dependente da quantidade da amostra a testar. O problema de otimização pararia quando surgissem **n** gerações consecutivas que apresentassem um valor máximo constante.

Ficheiros input

Os ficheiros necessários para a execução do programa são um ficheiro com as mesas existentes na sala e as pessoas inscritas no jantar. O esquema dos ficheiros é o apresentado de seguida:

Ficheiro das mesas:	Ficheiro das pessoas:
Número de mesas do mesmo tipo	Número de pessoas no grupo (1 significa sem grupo)
Lotação mínima	Idade
Lotação máxima	Profissão
…	Número de Hobbies
	Hobby (* número de hobbies)
	…

Após a execução do programa são gerados dois ficheiros *output:* um log.csv que pode ser utilizado para analisar a evolução das melhores soluções, um ficheiro com o nome definido por argumento com a descrição da solução, pessoas e distribuição pelas mesas.

Desenvolvimento

No desenvolvimento deste trabalho optamos pelo uso da linguagem Java que nos permite uma melhor distinção dos conceitos e separação de camadas. Ponderamos uma solução em Prolog mas visto que para a implementação do algoritmo genético não iríamos usufruir da maior ferramenta que a linguagem nos disponibiliza, o *backtracking*, optamos por algo mais orientado a objetos e que nos permita separação de conceitos como Pessoa, Grupo etc. e fácil acesso a atributos como a lotação de uma mesa.

Não recorremos a nenhuma ferramenta/API, não sentimos necessidade de as utilizar uma vez que os algoritmos foram criados por nós e também fomos capazes de os implementar sem problemas usando as estruturas de dados já oferecidas pelo Java. O IDE utilizado para implementar a solução, validar e testar o código foi o Eclipse. O projeto foi testado com sucesso em Windows e sistemas baseados em Unix. Esta portabilidade é também oferecida pela linguagem escolhida.

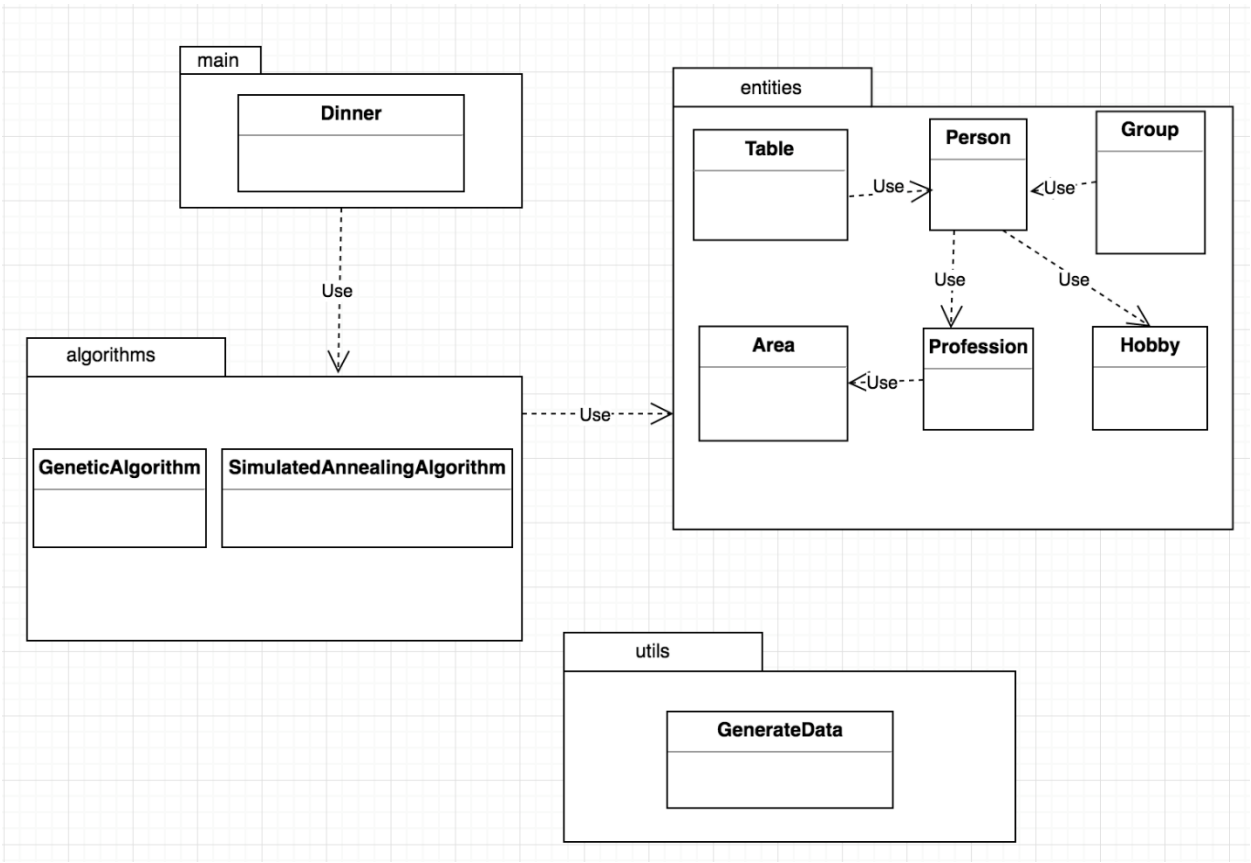
Para analisar os dados obtidos após a execução do programa e realizar a sua conversão para gráficos utilizamos o Microsoft Excel.

Estrutura da Aplicação

A aplicação encontra-se dividida em quatro módulos (*packages*):

- algorithms – contêm a implementação dos algoritmos genético e arrefecimento simulado.
- entities – contêm todas as classes necessárias para a representação do nosso problema em concreto: pessoa, grupo, mesa, profissão e sua área, hobby.
- main – contém a classe Dinner com o método main a partir do qual todo o projeto arranca.
- utils – contém a classe GenerateData usada para gerar aleatoriamente ficheiros de *input* com mesas e pessoas/grupos.

No diagrama abaixo apresentado é possível verificar a forma como todos os módulos e classes se relacionam.



Detalhes da implementação

Representação das cadeias de bits

As cadeias de bits usadas na representação dos indivíduos no algoritmo genético foram implementadas usando a estrutura de dados BitSet oferecida pelo Java. Esta estrutura possui já métodos como *flip(int index)* que alterna o valor atual de um bit na posição *index*, desta forma conseguimos poupar o tempo de implementarmos nós próprios uma estrutura com métodos globais de manipulação de bits.

Variação da temperatura

A variação da temperatura na implementação do algoritmo de arrefecimento simulado foi implementada usando a seguinte expressão: t[i] = t[i-1] * alpha. Desta forma a variação da temperatura é maior nos estados iniciais e à medida que o tempo aumenta a sua variação é menor. Tanto o valor de temperatura inicial como o alpha podem ser configurados pelo utilizador.

Geração de ficheiros aleatórios

A classe *GenerateData* do *package utils* pode ser usada para gerar ficheiros *input* de mesas e pessoas aleatórios. Para as mesas são gerados 10 tipos de mesas diferentes, para cada tipo podem existir 2 a 10 mesas, cada tipo de mesa pode ter um mínimo entre 1 e 18, e um máximo entre 1 e 20. Para as pessoas são gerados 50 grupos, cada grupo terá entre 1 a 5 pessoas, a idade de cada pessoa será entre 1 a 100 anos, a profissão é uma da lista apresentada no Anexo I, cada pessoa terá ainda entre 0 a 5 hobbies da lista apresentada no Anexo II.

Experiências

Inicialmente implementamos as funções de avaliação e penalização referidas no relatório intercalar. No entanto como já descrito em cima apercebemo-nos que a avaliação individual de cada mesa não ia de encontro ao pedido pelo enunciado do problema. Por esse motivo foram realizadas muitas tentativas que visavam apenas refinar a avaliação individual de cada mesa. Não guardamos registo destas experiências mas consistiam na análise empírica dos valores obtidos e a sua comparação ao desejado/ requerido pelo enunciado. Através desta análise foi possível detetarmos, por exemplo, que a nossa função de avaliação da idade era fraca pois avaliava de forma bastante positiva o cenário: mesa com três pessoas de idades: 10, 40, 70.

O conjunto de experiências seguinte passou por analisar os resultados obtidos para o mesmo conjunto de dados pelos dois algoritmos implementados. Além da solução final procuramos também fazer uma comparação da sua evolução. Os ficheiros usados como exemplo encontram-se na pasta “testes”. Os resultados obtidos encontram-se na pasta “testes/output”. Foram gerados 3 ficheiros de mesas e 3 ficheiros de pessoas através da classe GenerateData já descrita neste relatório. Os ficheiros *output* têm a seguinte nomenclatura output_<nome_algoritmo><#ficheiroTables><#ficheiroPeople>.txt , onde <nome_algoritmo> pode ser “gen” ou “sim”, <#ficheiroTables> e <#ficheiroPeople> é o número do ficheiro respetivo usado de 1 a 3. Os parâmetros de execução foram:

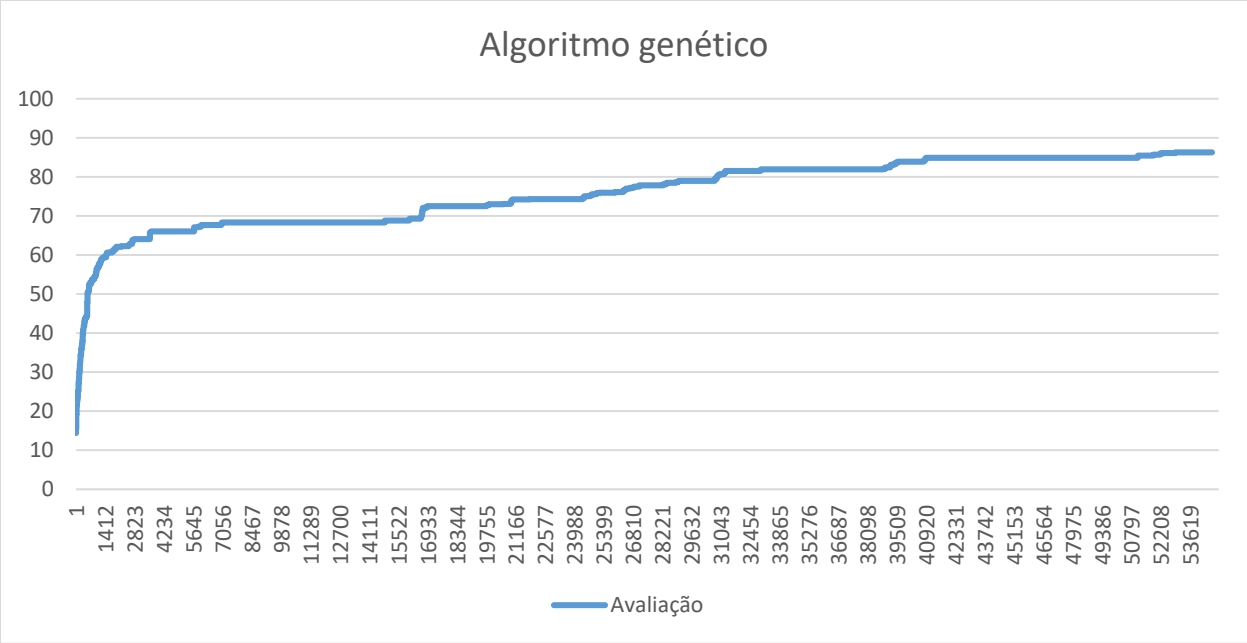
Algoritmo Genético:

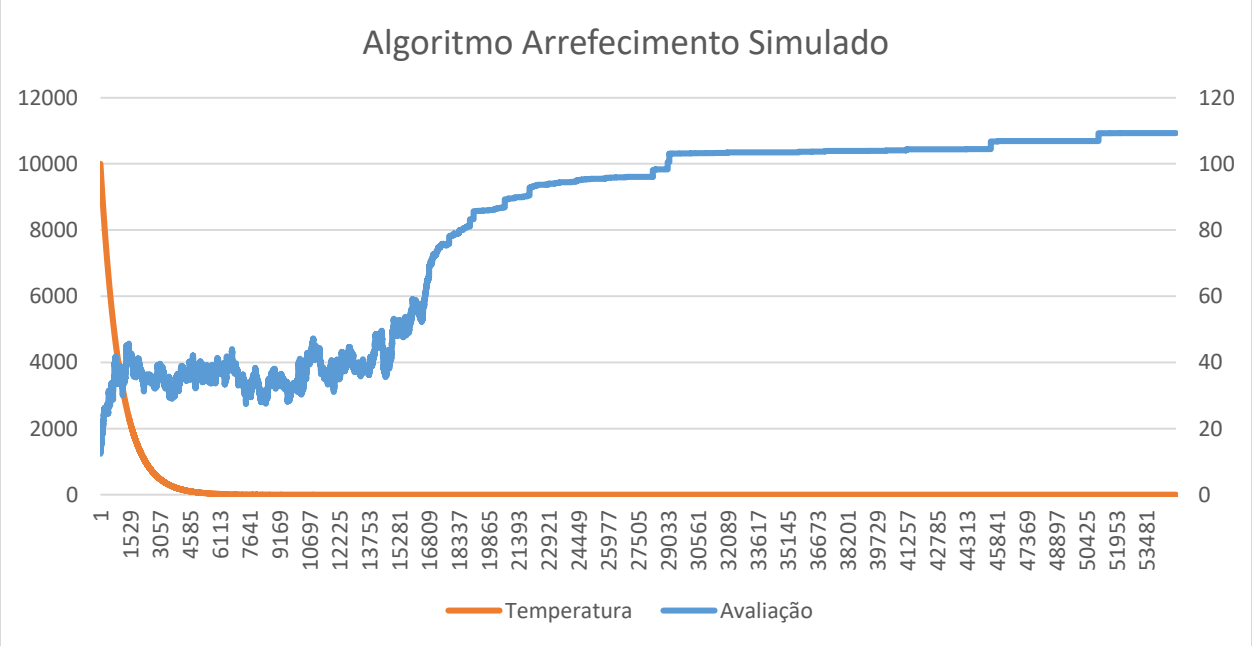
- Loops sem evolução: 500000
- Tamanho da população inicial: 30
- Seleção elitista: 5

Algoritmo Arrefecimento Simulado:

- Temperatura inicial: 100000
- Alpha: 0.99999

Os gráficos abaixo apresentados permitem comparar a evolução das soluções nos dois algoritmos. No entanto como os ficheiros log.csv gerados pelas execuções acima descritas ficam com um tamanho muito elevado não conseguimos gerar um gráfico com todos os valores. Por esse motivo os gráficos foram gerados com execuções menos pesadas dos algoritmos.





Conclusões

Consideramos que o desenvolvimento deste projeto foi um sucesso, uma vez que conseguimos implementar os dois algoritmos que nos tínhamos comprometido desde início, o genético e o arrefecimento simulado. Este trabalho permitiu-nos melhorar um bocado as nossas capacidades com o java e ajudou-nos a adquirir novos conhecimentos acerca da inteligência artificial.

A forma como iríamos representar os indivíduos no algoritmo genético foi a parte mais complicada, uma vez que pensamos em várias hipóteses, mas todas elas tinham limitações. A função de avaliação também nos apresentou algumas dificuldades, uma vez que não sabíamos o peso que iríamos atribuir a cada um dos critérios. Teve por isso de ser alterada e refinada a partir da apresentada no relatório intercalar

Em suma, o trabalho colocou-nos vários desafios que foram uma mais valia para o aumento dos nossos conhecimentos nesta unidade curricular, bem como competências essenciais para o nosso futuro.

Futuramente, o melhoramento desta aplicação poderia passar pela implementação de mais algoritmos, pela adição de mais atributos às pessoas inscritas no jantar de forma a se aproximar ainda mais de uma situação da realidade ou pela transformação da apresentação da solução final numa forma mais apelativa visualmente com recurso a uma interface gráfica.

Recursos

Bibliografia

- Material e apontamentos associados das aulas teóricas de IART lecionadas pelo professor Eugénio Oliveira;
- Acompanhamento com o Monitor Ricardo Sequeira e o Doutor Henrique Cardoso nas aulas teórico-práticas de IART;
- https://www.burakkanber.com/blog/machine-learning-genetic-algorithms-part-1-javascript/, acedido a 20/03/2017;
- https://kunuk.wordpress.com/2010/09/27/genetic-algorithm-example-with-java/, acedido a 20/03/2017;
- http://jenetics.io/, acedido a 30/03/2017

Software

- Eclipse Java;
- Java versão 8;
- Microsoft Excel 2016.

Distribuição do trabalho

Todos os elementos do grupo contribuíram de forma igual para o resultado final do trabalho.

Manual de utilizador

O código fonte do projeto encontra-se na pasta *"src"*, todos os passos abaixo indicados devem ser executados a partir da pasta extraída do *zip* submetido.

Compilação

Para compilar o projeto pode ser usada qualquer ferramenta desde que a máquina tenha o Java instalado.

Para simplificar existe um script de compilação com o nome compile.sh na pasta submetida.

Execução

Para executar o projeto são precisos dois ficheiros *input*, com as mesas e com as pessoas. Os parâmetros necessários são os ficheiros *input*, o ficheiro *output* onde ficará guardada a melhor solução obtida, o algoritmo pretendido e os seus argumentos respetivos. Exemplo (código compilado na pasta bin):

Algoritmo Genético

java –cp bin main.Dinner <Input Table file> <Input People file> <Output file> genetic <Max Loops Wo Evolution> <Population Size> <N Elite Selection>

Algoritmo do Arrefecimento Simulado

java –cp bin main.Dinner <Input Table file> <Input People file> <Output file> genetic simAnnealing <Initial Temperature> <Alpha>

Gerar ficheiros para teste

Para gerar ficheiros *input* aleatórios para teste é preciso indicar o nome dos ficheiros desejados.

Exemplo:

java –cp bin utils.GenerateData <Input Table file> <Input People file>

Anexo I - Profissões e Áreas

Área	Profissão
Saúde	Médico
	Farmacêutico
	Nutricionista
	Psicólogo
	Fisioterapeuta
	Enfermeiro
Engenharia	Informático
	Químico
	Industrial e Gestão
	Aeronáutica
	Civil
Turismo	Guia de turismo
	Agente de viagens
	Bartender
	Recepcionista
Educação	Matemático
	Biólogo
	Físico
	Pedagógico
	Geógrafo
Comunicação	Relações Públicas
	Publicidade
	Jornalismo
	Locutor de Rádio
Design e Arte	Cinema
	Moda
	Danca
	Teatro
	Música
	Fotografia

Anexo II - *Hobbies*

Hobbies
Xadrez
Futebol
Ler
Jogos online
Ginásio
Exercício ao ar livre
Ciclismo
Detecção de aeronaves
Golfe
Passeios de barco
Ténis
Rádio Amador
Aerografia
Apreciador música
Ir as compras
Desenhos Animados
Aquário
Culinária
Artes
Dança do ventre
Conduzir
Natação
Basebol
Tiro com arco
Astrologia
Colecionador