WPCCG'2016

Anais do I Workshop de Pesquisas em Computação dos Campos Gerais

28 e 29 de setembro de 2016

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Prefácio

Este volume contém os artigos apresentados no WPCCG 2016: I Workshop de Pesquisa em Computação dos Campos Gerais, ocorrido nos dias 28 e 29 setembro de 2016, na UTFPR (Universidade Tecnológica Federal do Paraná), em Ponta Grossa, Paraná, Brasil.

Ao todo, o Workshop teve 20 submissões. Cada submissão foi revisada por, ao menos, 2 revisores das áreas da Computação. Dos artigos submetidos, 17 foram aceitos para apresentação e publicação no WPCCG 2016.

O WPCCG 2016 é um evento recém-criado para divulgar pesquisas em desenvolvimento ou concluídas de alunos e docentes de Instituições de Ensino Superior. Organizado por docentes do DAINF (Departamento Acadêmico de Informática) do Câmpus Ponta Grossa da UTFPR, o evento está em sua primeira edição.

O comitê de programa teve participação de docentes da UTFPR (Câmpus Ponta Grossa), bem como docentes de outros Câmpus da UTFPR e ainda de outras Instituições, como PUC-PR, CEFET-RJ e UERJ.

O gerenciamento de toda conferência foi feito por meio do sistema EasyChair.

18 de outubro de 2016
Ponta Grossa, Paraná, Brasil

Gleifer Vaz Alves
Sheila Morais de Almeida
André Pinz Borges

Comitê do Programa

André Pinz Borges	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Augusto Foronda	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Carlos Eduardo Pantoja	Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)
Dênis Silva	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Santa Helena
Diana Sasaki	Universidade Federal do Rio de Janeiro (PESC/COPPE)
Erikson Morais	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Gleifer Alves	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Heitor Murilo Gomes	Pontifícia Universidade Católica do Paraná (PUCPR)
Jean Paul Barddal	Pontifícia Universidade Católica do Paraná (PUCPR)
Lourival Góis	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Richard Ribeiro	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Richardson Ribeiro	Universidade Tecnológica Federal do Paraná (UTFPR) - Campus Pato Branco
Sarah Sakamoto	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Saulo Queiroz	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Simone Nasser Matos	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa e Instituto Tecnológico de Aeronáutica (ITA)
Sheila Almeida	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa
Vinícius Andrade	Universidade Tecnológica Federal do Paraná (UTFPR) - Câmpus Ponta Grossa

Sumário

Author Index

Índice de Palavras-chave

Previsão da quantidade de classes em Classificação Hierárquica Multirrótulo

Thissiany Beatriz Almeida
Universidade Tecnológica Federal do Paraná
Ponta Grossa – PR - Brasil
thissianyalmeida@alunos.utfpr.edu.br

Helyane Bronoski Borges
Universidade Tecnológica Federal do Paraná
Ponta Grossa – PR - Brasil
helyane@utfpr.edu.br

RESUMO
Muitos dos problemas de classificação descritos na literatura de Aprendizagem de Máquina dizem respeito à classificação de dados em que cada exemplo é associado a uma classe pertencente a um conjunto finito de classes, todas em um mesmo nível. No entanto, vários problemas de classificação, são de natureza hierárquica, em que classes podem ser subclasses ou superclasses de outras classes. Em muitos problemas hierárquicos, um ou mais exemplos podem ser associados a mais de uma classe simultaneamente. Esses problemas são conhecidos como problemas de classificação hierárquica multirrótulo. Nesse trabalho, foi utilizada a técnica ML-kNN para a predição de problemas multirrótulos, visando determinar o número de classes que podem ser atribuídas a um exemplo. Através dos experimentos e análise estatística pode-se mostrar que as adaptações realizadas na técnica ML-kNN trouxeram contribuições significativas com relação as medidas de precisão e revocação.

Palavras-chave
Classificação Hierárquica Multirrótulo, ML-kNN, Aprendizagem de Máquina.

ABSTRACT
Many Machine's Learning classification problems are described in the literature associating the classification of data with a class belonging to a finite set of classes, all at a same level. However, many classification problems are by hierarchical nature, in which classes may be subclasses or superclasses of other classes. With many hierarchical problems, one or more examples may be associated with more than one class simultaneously. Those problems are known as multi-label hierarchical classification problems. In this paper, the ML-KNN techniques used to address the prediction of multi-label problems, aiming to determine the number of classes that may be assigned to an example. Through the experiments and statistical analysis can be shown that the adaptations accomplished in the technique ML-kNN brought significant contributions to relationship the measures of precision and revocation.

Keywords
Hierarchical Multi-label Classification, ML-kNN, Learning Machine.

WPCCG'16, Setembro28, 2016, Ponta Grossa, Paraná, Brasil.

1. INTRODUÇÃO
O processo de classificação de dados na Aprendizagem de Máquina (AM) tem como objetivo atribuir uma classe para um novo exemplo a partir de suas características (atributos). Os problemas de classificação podem ser divididos em dois grandes grupos: Classificação Plana e Classificação Hierárquica, sendo diferenciados pelo relacionamento de dependência entre as classes [19]. A tarefa de classificação em AM pode também ser categorizada conforme a quantidade de classes a serem estimadas para um determinado exemplo. Sendo assim, esta categorização pode ser aplicada em problemas tradicionais (unirrótulo) ou problemas multirrótulos.

A motivação inicial para as pesquisas na área de classificação multirrótulo surgiu com a dificuldade causada por ambiguidades em problemas de categorização de textos [17].

A classificação hierárquica multirrótulo é considerada uma área de pesquisa relativamente nova[3][7][10], o que proporciona o interesse de pesquisadores de diferentes áreas. Esse tipo de classificação pode ser utilizado para categorização de textos [1][2], predição de proteínas em dados de bioinformática [3][4], classificação de gêneros musicais e imagens [5], entre outros.

Tem-se nesse trabalho como objeto de estudo os Problemas de Classificação Hierárquica Multirrótulo, onde as classes possuem relação hierárquica umas com as outras, sendo este relacionamento representado em formato de grafos acíclicos direcionados (DAG, do inglês *Directed Acyclic Graph*). Leva-se em consideração a organização hierárquica com o objetivo de aumentar a capacidade preditiva.

Neste trabalho é utilizado o algoritmo ML-kNN [21] para a determinação do número de classes a ser atribuído a um exemplo de teste. Os experimentos foram realizados utilizando dez bases de dados da área genômica funcional, *Gene Ontology*, sendo estas estruturas em DAG. Para a avaliação foram utilizadas variações no algoritmo ML-kNN para os valores de *k* e *threshold*, onde *k* recebe os valores 3, 5 e 7, enquanto o *threshold* varia entre 0.5, 0.7 e 0.8.

2. CONCEITOS FUNDAMENTAIS
2.1 Classificação Hierárquica Multirrótulo
Problemas de classificação hierárquica têm por objetivo a classificação de cada novo dado de entrada em um dos nós folhas fornecendo um conhecimento mais específico e útil [6]. Pode ocorrer, no entanto, do classificador não apresentar uma confiabilidade desejada na classificação em uma das classes do nível mais profundo, sendo mais seguro realizar a classificação nos níveis mais elevados.

A classificação hierárquica multirrótulo tem surgido como uma nova categoria de problemas de classificação, com características tanto dos problemas de classificação multirrótulo, quanto de problemas de classificação hierárquica. Problemas pertencentes a esta nova categoria são denominados de problemas de classificação

hierárquica multirrótulo (HMC, do inglês *Hierarchical Multilabel Classification*).
Em um problema de classificação hierárquica multirrótulo, um exemplo pode pertencer a múltiplas classes ao mesmo tempo e essas classes são organizadas de maneira hierárquica. A hierarquia pode ser representada em formato de árvore ou de um Grafo Acíclico Direcionado (DAG). Dessa forma, um exemplo pertencente a uma classe, automaticamente pertence a todas as suas superclasses [7].
A principal diferença entre a estrutura em árvore e a estrutura DAG é que, na estrutura em árvore, cada nó, exceto o nó-raiz tem somente um nó-pai, enquanto que no DAG cada nó, exceto o nó-raiz, pode ter um ou mais nós – pai [7].
Vários métodos podem ser utilizados no tratamento de tarefas de classificação hierárquica multirrótulo. Na literatura, há vários trabalhos propondo e analisando abordagens e métodos para tratamentos de problemas hierárquicos multirrótulo (HMC) [3][7][10][19], contudo, não há um consenso sobre qual algoritmo utilizar para o tratamento de problemas hierárquicos multirrótulo.
Pode-se dizer também que problemas de classificação hierárquica multirrótulo são mais complexos que os demais problemas de classificação, uma vez que as classes envolvidas no problema, além de estarem estruturados em uma hierarquia, os exemplos podem pertencer a mais de uma classe ao mesmo tempo [7].

2.2 Precisão Hierárquica e Revocação Hierárquica

No trabalho de Kiritchenko et al. (2004), foram propostas duas medidas de avaliação baseadas nas medidas convencionais de precisão e revocação, levando em consideração os relacionamentos hierárquicos entre as classes. Essas medidas foram chamadas de precisão e revocação hierárquicas e levam em consideração classificações nos nós internos e nós-folha.
Cada exemplo pertence não apenas à sua classe, mas também a todos os ancestrais dessa classe na estrutura hierárquica. Dessa maneira, dado um exemplo qualquer (x_i, Y_i), com x pertencente ao conjunto X de exemplos, Y_i o conjunto de classes preditas para o exemplo x_i, e Y_i' o conjunto de classes verdadeiras do exemplo x_i, os conjuntos Y_i e Y_i' podem ser entendidos para conterem suas correspondentes classes ancestrais da seguinte maneira: $\hat{Y} = \bigcup_{y_i \in Y_i} Ancestrais(y_k)$ e $\hat{Y} = \bigcup_{y_i \in Y_i'} Ancestrais(y_i)$.
A precisão e revocação hierárquica (Prec e Rev) são calculadas utilizando as equações abaixo, respectivamente.

$$Prec = \frac{\sum_i |\hat{Y}_i \cap \hat{Y}_i'|}{\sum_i |\hat{Y}_i|}$$

$$Rev = \frac{\sum_i |\hat{Y}_i \cap \hat{Y}_i'|}{\sum_i |\hat{Y}_i'|}$$

Essas medidas contam o número de classes preditas corretamente, juntamente com o número de classes ancestrais dessas classes preditas corretamente, assumindo que exemplos também pertencem aos ancestrais de suas classes corretas [22].

3. METODOLOGIA

3.1 Aplicação das técnicas

Para a realização dos experimentos foram escolhidas 10 bases de dados de funções de proteínas, sendo que estas já estavam normalizadas conforme o trabalho de Borges [19]. Com relação ao algoritmo escolheu-se o classificador ML-kNN [21] que é um dos algoritmos mais tradicionais utilizados em problemas de classificação multirrótulos. Este algoritmo é disponibilizado na ferramenta Mulan [20], sendo que esta trabalha em conjunto com as classes Java do Weka [18], um ambiente conhecido e utilizado pela comunidade de aprendizado de máquina e mineração de dados [18].
O ML-kNN é uma adaptação do algoritmo kNN [12] para o problema multirrótulo. Esse algoritmo determina o conjunto de rótulos do exemplo a ser classificado, baseado na probabilidade máxima a posteriori calculada a partir da frequência de cada rótulo entre os k vizinhos mais próximos, comparado com a frequência em todos os exemplos por meio do cálculo de distância.
O *framework* Mulan utiliza dois formatos de arquivo como entrada, sendo estes o ARFF (*Attribute-Relation File Format*) e o XML (*eXtensible Markup Language*). No arquivo ARFF é possível definir o tipo de dados que estão sendo carregados, e então fornecer seus próprios dados. No arquivo, foi definido cada coluna e o que cada coluna contém, fornecemos cada linha de dados em um formato delimitado por vírgulas. No Mulan cada rótulo vira um atributo classe do tipo {0,1}, onde 1 representa que aquele exemplo é pertencente a classe e 0 a ausência daquela classe. O arquivo XML é responsável por representar a hierarquia/dependência entre os rótulos/classes da base a ser analisada. Este arquivo se faz necessário durante as etapas de classificação e avaliação.
Ao ser analisado o arquivo XML exigido pelo framework Mulan foi verificado que não era possível representar problemas hierárquicos multirrótulo em formato de Grafo Acíclico Direcionado, somente exemplos que tenham a hierarquia estruturada em formato do tipo árvore, que são estruturas mais simples, onde um nó filho tem apenas um nó pai.
As bases escolhidas para este trabalho possuem sua hierarquia estruturada em formato do tipo DAG, sendo esta estrutura mais complexa quando comparada ao do tipo árvore, pois um nó filho pode ter múltiplos pais. Devido a esse fato foi necessário criar uma nova forma de representar a hierarquia dos rótulos substituindo assim, a necessidade de fornecer o arquivo do tipo XML como entrada, e permitindo a utilização de qualquer estrutura hierárquica do tipo DAG.

Tabela 1. Características das bases de dados GO

Bases	Quant Amostras	Quant. Atributos	Quant. Classes	Quant. Max. Níveis
Cellcycle	3751	77	4125	13
Church	3749	27	4125	13
Derisi	3719	63	4119	13
Eisen	2418	79	3573	13
Expr	3773	551	4131	13
Gasch1	3758	173	4125	13
Gasch2	3773	52	4131	13
Pheno	1586	69	3127	13
Seq	3900	478	4133	13
Spo	3697	80	4119	13

O programa desenvolvido lê o arquivo que contém a definição da estrutura hierárquica entre as classes armazenando-a em um grafo. Após isso são criados dois conjuntos de dados, um para a base de treinamento e o outro para a base de teste, e por fim instanciado o classificador passando como parâmetro o valor de k-vizinhos. Lembrando que o valor de k-vizinhos pode influenciar durante o processo de classificação. A construção do modelo de classificação dá-se ao treinar o classificador com a base de dados de treinamento. Para cada instância da base de teste é calculada a distância euclidiana com todas as instâncias da base de treinamento o que

resulta em um vetor. Este vetor varia de 1 até o número de classes e para cada classe é feita uma comparação com o valor de corte *(threshold)*. Caso o valor de semelhança seja igual ou maior ao valor de corte então atribuímos a classe para aquela instância. Após a classificação das classes geradas pela semelhança com os k-vizinhos, o algoritmo faz uma varredura nas classes hierarquicamente ancestrais das preditas e também seta as mesmas como classes pertencentes a instância analisada.

Foi adotada a técnica de Spyromitros [11] antes utilizada apenas para o algoritmo BRkNN. Onde caso nenhuma das classes tenha sido predita para um exemplo, será predita a classe que apresentar o maior valor de semelhança com o exemplo, com o objetivo de viabilizar o cálculo da medida de precisão.

3.2 Avaliação e análise dos resultados

Considerando as peculiaridades inerentes aos problemas de classificação hierárquica multirrótulo, medidas específicas para estes tipos de problemas devem ser utilizadas para avaliar os modelos de classificação gerados para solucioná-los. Tais medidas requerem algumas considerações adicionais, além dos aspectos normalmente considerados na avaliação de modelos convencionais de classificação.

Após as adaptações do algoritmo ML-kNN foram realizados alguns experimentos para comparação do desempenho sendo utilizadas como base as medidas de precisão hierárquica e revocação hierárquica [22], sendo que essas medidas levam em consideração classificações nos nós internos e nós-folha. Para isso, foram escolhidos valores para a variável de corte (*threshold*) de 0.5, 0.7 e 0.8. Considerando o valor da variável de corte constante e variando-se o valor do k-vizinhos obteve-se os resultados como são mostrados nas Tabelas 1, 2 e 3, conforme os valores de *threshold* de 0.5, 0.7 e 0.8, respectivamente.

Tabela 2. Resultados para os experimentos com *threshold* =0.5

***Threshold* = 0.5**						
Bases	**K=3**		**K=5**		**K=7**	
	Prec.	**Rev.**	**Prec.**	**Rev.**	**Prec.**	**Rev.**
Cellcycle	0,736	0,297	0,759	0,289	0,747	0,297
Church	0,769	0,248	0,772	0,247	0,735	0,263
Derisi	0,757	0,257	0,753	0,262	0,761	0,261
Eisen	0,743	0,298	0,741	0,301	0,744	0,296
Expr	0,771	0,293	0,768	0,299	0,756	0,305
Gasch1	0,763	0,291	0,761	0,297	0,761	0,299
Gasch2	0,747	0,287	0,762	0,278	0,771	0,274
Pheno	0,765	0,244	0,733	0,255	0,743	0,248
Seq	0,771	0,286	0,776	0,282	0,780	0,283
Spo	0,741	0,280	0,736	0,285	0,745	0,280

Tabela 3. Resultados para os experimentos com *threshold* =0.7

***Threshold* = 0.7**						
Bases	**K = 3**		**K = 5**		**K = 7**	
	Prec.	**Rev.**	**Prec.**	**Rev.**	**Prec.**	**Rev.**
Cellcycle	0,899	0,201	0,899	0,209	0,910	0,201
Church	0,897	0,188	0,895	0,189	0,897	0,188
Derisi	0,895	0,191	0,892	0,192	0,893	0,194
Eisen	0,855	0,218	0,871	0,214	0,882	0,208
Expr	0,884	0,229	0,887	0,228	0,897	0,221
Gasch1	0,881	0,219	0,881	0,229	0,893	0,219
Gasch2	0,899	0,199	0,895	0,207	0,900	0,204
Pheno	0,894	0,185	0,892	0,186	0,887	0,187
Seq	0,912	0,201	0,906	0,206	0,900	0,212
Spo	0,902	0,194	0,890	0,204	0,889	0,200

Tabela 4. Resultados para os experimentos com *threshold* =0.8

***Threshold* = 0.8**						
Bases	**K = 3**		**K = 5**		**K = 7**	
	Prec.	**Rev.**	**Prec.**	**Rev.**	**Prec.**	**Rev.**
Cellcycle	0,918	0,187	0,925	0,182	0,930	0,183
Church	0,946	0,154	0,947	0,154	0,946	0,154
Derisi	0,943	0,156	0,943	0,155	0,936	0,163
Eisen	0,923	0,166	0,906	0,186	0,910	0,186
Expr	0,924	0,193	0,928	0,195	0,933	0,196
Gasch1	0,920	0,187	0,921	0,195	0,926	0,192
Gasch2	0,919	0,183	0,927	0,181	0,933	0,179
Pheno	0,926	0,160	0,937	0,151	0,930	0,157
Seq	0,928	0,181	0,918	0,194	0,910	0,193
Spo	0,940	0,158	0,926	0,174	0,928	0,171

Para comparação das 9 variações de experimentos realizados adotou-se a utilização do teste estatístico de Wilcoxon. Para realizar esse teste estatístico foram escolhidos os valores obtidos na medida de precisão.

No teste estatístico foi definida que a hipótese nula assume que a diferença de desempenho entre algoritmos não é significativa. Com nível de confiança $\alpha = 0.05$ a hipótese nula não pode ser rejeitada se $-1.96 \leq z \leq 1.96$.

A seguir, os resultados obtidos pelo Teste de Wilcoxon levando em consideração a variação do valor de k. Nas Tabelas 5, 6 e 7 são encontrados os valores tabulados do teste para k assumindo os valores 3, 5 e 7.

Tabela 5. Resultados do teste com Threshold = 0.5

***Threshold* = 0.5**	
K = 3 com K = 5	Z-Score = 0.153
K = 3 com K = 7	Z-Score = 0.051
K = 5 com K = 7	Z-Score = 0.459

Tabela 6. Resultados do teste com Threshold = 0.7

***Threshold* = 0.7**	
K = 3 com K = 5	Z-Score = 0.948
K = 3 com K = 7	Z-Score = 0.714
K = 5 com K = 7	Z-Score = 1.631

Tabela 7. Resultados do teste com Threshold = 0.8

***Threshold* = 0.8**	
K = 3 com K = 5	Z-Score = 0.153
K = 3 com K = 7	Z-Score = 0.296
K = 5 com K = 7	Z-Score = 0.051

Através dos valores de z encontrados nas Tabelas 5, 6 e 7 pode-se concluir que nenhum valor permite que a hipótese nula seja rejeitada, ou seja, não há melhora no desempenho do algoritmo realizando a variação dos valores de k.

Nas tabelas 8, 9 e 10 são encontrados os valores tabulados do teste para *threshold* assumindo os valores 0.5. 0.7 e 0.8.

Tabela 8. Resultados do teste com K = 3

K = 3	
Threshold = 0.5 com 0.7	Z-Score = -2.8031
Threshold = 0.5 com 0.8	Z-Score = -2.8031
Threshold = 0.8 com 0.7	Z-Score =-2.8031

Tabela 9. Resultados do teste com K = 5

K = 5	
Threshold = 0.5 com 0.7	Z-Score = -2.8030
Threshold = 0.5 com 0.8	Z-Score = -2.8030
Threshold = 0.8 com 0.7	Z-Score = -2.8030

T**abela 10. Resultados do teste com K = 7**

K = 7	
Threshold = 0.5 com 0.7	Z-Score = -2.8032
Threshold = 0.5 com 0.8	Z-Score = -2.8032
Threshold = 0.8 com 0.7	Z-Score = -2.8032

Ao contrário do resultado obtido na variação dos valores de *k*, é possível perceber que neste teste, os valores obtidos de *z*, são superiores ao limite de -1.96, o que significa que a hipótese nula pode ser rejeitada e que há melhoras no desempenho do algoritmo.

4. CONCLUSÃO

Neste trabalho foram apresentados experimentos com as bases de dados estruturadas em formato de Grafo Acíclico Direcionado adaptando-se o algoritmo de classificação hierárquica multirrótulo, o ML-kNN. Dentre essas mudanças, pode-se citar a modificação do arquivo XML utilizado como arquivo de entrada pelo framework Mulan, onde encontra-se implementado o algoritmo ML-kNN. Pode-se ressaltar também o fato da utilização da técnica antes utilizada apenas no algoritmo BR-kNN, se na fase de predição não for atribuída nenhuma classe a instância, atribuísse a essa instância então a classe que possuir o maior valor de confiança.

Com base nos testes estatísticos, o algoritmo ML-kNN tem um desempenho superior levando-se em consideração a medida de precisão quando assume valor de threshold igual a 0.8, esse fato pode ser explicado na comparação entre o valor da confiança dos rótulos com o threshold, pois quanto maior o valor do threshold a tendência é que seja predito um número cada vez menor de classes, entrando na condição onde é atribuída apenas a classe com a maior confiança dentre os rótulos daquela instância.

5. REFERÊNCIAS

[1] Dumais, S. and Chen, H. *Hierarchical classification of web content*. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Athens, Greece, pp. 256-263, 2000.

[2] Sun, A. and Lim, E.-P. *Hierarchical text classification and evaluation*. In Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society, pp. 521-528, 2001.

[3] Costa, E. P., Lorena, A. C, Carvalho, A.P.L.F, &Freitas,A. A. (2007). A review of Performance Evaluation Measures for Hierarchical Classifiers. In Proceedings of the AAAI07 – Workshop on Evaluation Methods for Machine Learning II, P.1-6.

[4] Holden, N. and Freitas, A. *A Hierarchical classification of protein function with ensembles of rules and particle swarm optimisation*. Soft Comput. vol. 13, pp. 259-272, 2008.

[5] Barutcuoglu, Z. and DeCoro, C. *Hierarchical shape classification using Bayesian aggregation. In Proceedings of the IEEE International Conference on Shape Modeling and Applications*. Matsushima, Japan, pp. 44-44, 2006.

[6] Carvalho, A. C. P. F.; Freitas, A. A. ***Tutorial on Hierarchical Classification with Applications in Bioinformatics***.v.1.São Paulo: Idea Group, 2007.

[7] Cerri, R., Carvalho, A. C. P. L. F., e Costa, E. P(2008). Classificação hierárquica de proteínas utilizando técnicas de aprendizado de máquina. In *II Workshop on Computational Intelligence*, páginas 1-6, Salvador.

[8] Guyon, I. and Elisseeff, *A. An introduction to feature extraction. In Feature Extraction, Foundations and Applications*. Springer, pp. 1-24, 2006.

[9] Yang, Y. and Pedersen, J. O. *A comparative study on feature selection in text categorization*. In Proceedings of the Fourteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., pp. 412-420, 1997.

[10] Santos, A. M., Canuto, A. M. P. *Investigating the influence of repart in ensemble systems designed by boosting*. In: *IJCNN. Hong Kong*: IEEE, 2008. P. 2907-2914.

[11] Spyromitros, E.; Tsoumakas, G.; Vlahavas, I.*An empirical study of lazymultilabel classification algorithms.In:Hellenic conference on Artificial Intelligence*,p. 401–406, Berlin, Alemanha, 2009.

[12] Aha, D. W., Kibler, D. e Albert, M. K. (1991). *Instance-based learning algorithms. Machine Learning*, 6(1): 37-66.

[13] Quinlan, J.R. C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). 1. Ed. San Francisco, California: Morgan Kaufmann, 1993. Paperback.

[14] Schapire, R. E.; Singer, Y. Boostexter: A boosting-based system for text categorization. Machine Learning, v. 39, n. 2/3, p. 135-168, 2000.

[15] Hall, M.; Frank, E.; Holmes, G.;Pfahringer, B.; Reutemann,P.; Witten, I. H. 2009. The WEKA data mining software: an update. **SIGKDD Explor. News,** v. 11, p.10-18, 2009.

[16] Borges, Helyane Bronoski; Nievola, J. C. *Multi-Label Hierarchical Classification using a Competitive Neural Network for Protein Function Prediction.* In: 2012 International Joint Conference on Neural Networks (IJCNN 2012), 2012, Brisbane, Austrália. 2012 International Joint Conference on Neural Networks (IJCNN 2012). Piscataway, NJ: IEEE Press, 2012. v. 1. p. 1-8.

[17] Tsoumakas, G., Katakis, I., Vlahavas, I. (2010) "Mining Multi-label Data", Data Mining and Knowledge Discovery Handbook, O. Maimon, L. Rokach (Ed.), Springer, 2nd edition, 2010.

[18] Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern recognition 40(7), 2038–2048 (2007).

[19] Kiritchenko, S.; Matwin, S.; Famili, A. F. *Hierarchical text categorization as a tool of associating genes with gene ontology codes*. In: *Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics*, Pisa, Italia,2004.

Um Método para a Refatoração de Software Baseado em Frameworks de Domínio

Víctor P. A. Barros
Universidade Tecnológica Federal do Paraná
Av Monteiro Lobato, s/n, Km 04
Ponta Grossa, Paraná
victorambiel@outlook.com

Simone N. Matos
Universidade Tecnológica Federal do Paraná
Av Monteiro Lobato, s/n, Km 04
Ponta Grossa, Paraná
snasser@utfpr.edu.br

RESUMO

Este trabalho criou um método de refatoração usando como referência os métodos da literatura, capaz de ajudar os desenvolvedores na refatoração de aplicações construídas com os conceitos de *frameworks* de domínio. O método proposto é formado por três etapas principais: *Entender o sistema*, *Ordenar os módulos* e *Refatorar Módulos*. A diferença entre o método proposto e os da literatura é que prevê a aplicação de metapadrões, inversão de controle e uso de ferramenta de refatoração em suas etapas. O estudo de caso em que o método foi aplicado é o Framework de Formação de Preço de Venda (FrameMK), desenvolvido pelo Grupo de Pesquisa em Sistemas de Informação do Câmpus Ponta Grossa, que tem a finalidade de calcular o preço de venda de um produto ou serviço. Os resultados da aplicação do método no FrameMK foram: melhorou a complexidade do código, diminui a quantidade de *bad smells* (sintomas no código fonte que indicam problemas mais graves no software) e a duplicação de código, o código ficou mais reusável e flexível e houve um aumento na qualidade do software em relação a expectativas do seu ciclo de vida.

Palavras-chave

refatoração; frameworks; metodologias

ABSTRACT

This paper created a method refactoring with reference to the methods of literature, able to assist developers in refactoring applications built with domain frameworks concepts. The proposed method consists of three main steps: Understanding the system, Sort modules and Refactor modules. The difference between the proposed method and the literature is that it provides for metapatterns, inversion of control and use of refactoring tool in their steps. The case study in which the method was applied is the Framework of Sales Price Formation (FrameMK), developed by the Research Group Information Systems on Campus Ponta Grossa, which have the purpose of calculating the selling price of a product or service. The results of applying the method in FrameMK were: improved code complexity, reduces the quantity of bad smells (symptoms in the source code that indicate more serious problems in the software) and the duplication code, the code became more reusable and flexible and there was an increase in the quality of software in relation to expectations of its cycle life.

Keywords

refactoring; frameworks; methodologies

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

1. INTRODUÇÃO

A refatoração de software foi apresentada por Fowler [10] como um conjunto de técnicas que facilita modificar a estrutura interna do software, sem alterar o seu comportamento externo.

O processo de refatoração pode ser melhorado com a adoção de métodos que guiam o processo na identificação das partes do código que devem ser refatoradas e indicam qual a melhor técnica a ser utilizada em determinado tipo de problema.

Alguns métodos de refatoração já foram publicados, como o de Mens e Tourwé [11] que possui uma sequência de seis passos a serem seguidos para se aplicar a refatoração e não é focado em um tipo de software específico. Outro trabalho é o de Rapeli [12], que está focado na refatoração de sistemas em Java com Padrões de Projeto. Estes métodos não exploram a refatoração de aplicações construídas com o conceito de *frameworks* de domínio, as quais contém características que envolvem aplicação de metapadrões, padrões de projeto, inversão de controle, entre outros.

Um *framework* é uma estrutura que tem como o objetivo prover uma funcionalidade genérica, que serve de apoio para a construção de uma outra aplicação [9]. Eles podem ser classificados como de infra-estrutura, *middleware* e domínio (ou aplicação). Os *frameworks* de infra-estrutura simplificam o desenvolvimento de sistemas de infra-estrutura portáveis e eficientes. Os *frameworks middleware* integram aplicações e componentes distribuídos, escondendo o baixo nível de comunicação entre os componentes distribuídos. Os *frameworks* de domínio têm um foco no desenvolvimento de aplicação em domínios específicos tais como: agricultura, formação de preço de venda, entre outros.

Este trabalho criou um método de refatoração que pode ser aplicado na estrutura de software construído com os conceitos de um *framework* de domínio, utilizando como base

métodos já publicados, buscando assim obter um melhor resultado na refatoração do *framework*. Para isto, buscou-se quais os conceitos utilizados para a construção de um *framework*, assim foi possível identificar quais pontos devem ser considerados em sua refatoração, introduzindo os conceitos de metapadões, inversão de controle, padrões de projeto e técnicas de refatoração. Também foi proposto automatizar e facilitar a aplicação da refatoração utilizando ferramentas que auxiliam na análise de código.

O uso do método proposto foi aplicado no *framework* de domínio na Formação de Preço de Venda (FrameMK) [4] que está em desenvolvimento por acadêmicos da computação da Universidade Tecnológica Federal do Paraná, câmpus Ponta Grossa. Este *framework* tem como principal objetivo oferecer um ambiente em que o usuário possa calcular o preço de venda de um produto ou serviço. Foi escolhido este *framework* porque o código da aplicação foi desenvolvido por várias pessoas e com isto necessitava de alterações em sua estrutura interna.

2. METODOLOGIA

O método proposto (Figura 1) é dividido em três etapas principais, sendo que para cada uma foi criado um fluxograma explicitando o seu funcionamento.

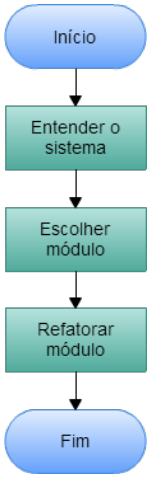

Figura 1: Processo Geral do Método Proposto

A primeira etapa *Entender o sistema* tem a finalidade de permitir ao desenvolvedor um entendimento geral de como o sistema funciona. A segunda etapa, *Escolher módulos*, tem como objetivo classificar os módulos que compõem o sistema para que estes possam ser refatorados na etapa posterior. Por fim, a última etapa *Refatorar módulo* realiza o processo refatoração propriamente dito.

Dentre as etapas do método, a primeira está baseada no método de Rapeli [12], que contém a etapa de *Entender o sistema* e o passo de *Gerar diagramas de classe*. As outras etapas foram determinadas por este trabalho.

2.1 Etapa 1 - Entender o Sistema

Esta etapa consiste em três passos que são essenciais para a realização da refatoração do sistema. O primeiro passo é o *Utilizar o sistema*, onde seu objetivo é que o desenvolvedor tenha o primeiro contato com o sistema que será refatorado, verificando como ele funciona e interage.

O segundo passo, *Verificar módulos*, tem como objetivo identificar quais são os módulos do sistema. No método proposto foi considerado um módulo como sendo um subsistema ou pacote. Muitos sistemas são divididos em subsistemas, como por exemplo, um *Enterprise Resource Planning* (ERP), que possui módulos que se interagem entre si, tais como: Fiscal, Financeiro, entre outros. Separar o sistema em módulos facilita o entendimento sobre seu funcionamento. Muitas vezes a separação por módulos não é explícita, sendo assim, pode-se considerar um módulo um conjunto de pacotes que tem como objetivo fornecer uma funcionalidade específica, como por exemplo, um pacote de Regra de Negócio que contempla todas as classes de regras lógicas do sistema.

O terceiro passo é o *Gerar diagrama de classe dos módulos*. O objetivo é criar o diagrama de classe dos módulos que foram identificados no passo anterior, podendo assim, ter uma representação visual e mais clara de como as classes do sistema funcionam e se relacionam entre si.

2.2 Etapa 2 - Escolher Módulo

Esta etapa do processo de refatoração tem como foco ordenar de forma crescente os módulos pela quantidade de *bad smells*, para que o processo de refatoração se inicie do menor para o maior módulo.

Para se obter a quantidade de *bad smells* de cada módulo é utilizada uma ferramenta de análise de código. O presente trabalho utilizou a ferramenta *SonarQube* [6], pois os pesquisadores já tinham um conhecimento prévio sobre seu funcionamento, porém há outras ferramentas disponíveis que podem ser utilizadas, como a *FindBugs* [3], *Checkstyle* [1] e *CodePro AnalytiX* [2].

A ordenação dos módulos fica a critério do desenvolvedor utilizar o melhor algoritmo de ordenação que se encaixa em seu contexto, desde que o algoritmo utilizado cumpra com a finalidade dessa etapa, ordenando os módulos do menor para o maior, em relação a quantidade de *bad smells* de cada módulo.

A escolha de começar pelo módulo que contém a menor quantidade de *bad smells* pode facilitar a refatoração para desenvolvedores menos experientes, pois o número de refatorações que deverão ser feitas tendem a ser menores.

2.3 Etapa 3 - Refatorar Módulo

A última etapa do método proposto é *Refatorar Módulo* e é o momento em que de fato o código é refatorado.

Esta etapa inicia com a iteração de i começando em 1 até m, sendo que m representa a quantidade total de módulos. Dentro da iteração verifica-se se o desenvolvedor deseja refatorar o módulo i, estando os módulos ordenados em ordem crescente da quantidade de *bad smells*, conforme a etapa 2.

Se o desenvolvedor desejar refatorar o módulo, então é utilizada a ferramenta de análise de código novamente, porém agora o foco é realizar a refatoração do módulo propriamente.

Após a análise do projeto a partir da ferramenta, é verificado os *bad smells* existentes, e tem-se uma condição de verificação. Se há *bad smells* no sistema, o próximo passo é aplicar as técnicas de refatoração, essas focadas em *frameworks*.

Tais técnicas são: as técnicas de refatoração apresentadas

por Fowler [10], aplicação de padrões de projeto, aplicação de metapadrões e aplicação de inversão de controle (IC). A escolha de qual técnica deve ser realizada inicialmente fica a critério do desenvolvedor a medida que ele analisa o código e consegue identificar a aplicação de alguma técnica.

As técnicas de refatoração de Fowler [10] e aplicações de padrões já são conhecidos na literatura. O processo de refatoração usando metapadrões e IoC pode ser encontrado no trabalho publicado por Barros [8].

3. RESULTADOS

Para a refatoração do FrameMK foram modificadas ao todo 30 (trinta) classes dentro do módulo *app*, 23 (vinte e três) classes do módulo *Persistence* e *BussinnesRule* e 5 (cinco) classes do módulo *service*.

Foram aplicadas 295 (duzentos e noventa e cinco) refatorações sugeridas pela ferramenta *SonarQube* [6], sendo removidos 61% dos *bad smells* presentes dentro do módulo *app*.

A partir da ferramenta *SonarQube* [6] também foi possível medir o *SQALE* [7] do módulo *app*, tendo como parâmetros o quão objetivo, preciso, de fácil reprodução e automatizado é o código, sendo que antes da refatoração a nota do módulo *app* no *SQALE* [7] era B e após a refatoração ela subiu para a nota A. A Tabela 1 ilustra como eram algumas características do FrameMK antes e depois da refatoração.

Tabela 1: Estatísticas da refatoração do FrameMK

Informações	Antes	Depois
Quantidade de classes	37	39
Quantidade de bad smells	477	182
Complexidade	531	236
Duplicações	39,2%	35,8%
Linhas de código	2797	2921
Quantidade de funções	144	151
SQALE rating	B	A

Utilizando a técnica de Fowler [10] foi possível aplicar uma refatoração dentro do módulo *app*, que gerou a redução da duplicação de código de 3 (três) classes que continham códigos idênticos.

Foi utilizado também o metapadrão *Unification*, sendo necessário a criação de uma nova classe, assim foi possível unificar 3 (três) métodos utilizados por 3 (três) classes diferentes.

Durante o processo de refatoração foram identificados pontos de maior facilidade para a aplicação do método e pontos de maior dificuldade.

Na primeira etapa *Entender o sistema*, o *framework* de domínio FrameMK já era conhecido pelo autor, assim tinha-se um conhecimento prévio de como interagir com o *framework*, porém em alguns casos o desenvolvedor que irá aplicar a refatoração utilizando o método proposto pode não ter um conhecimento prévio sobre o *framework* que irá refatorar, o que pode levar um pouco mais de tempo para se executar tal etapa. A maior dificuldade da primeira etapa foi realizar a engenharia reversa do código para gerar o diagrama de classes, pois o FrameMK contém 344 classes, e que muitas delas não possuem dentro de suas estruturas os atributos de outras classes as quais estão relacionadas.

A segunda etapa em que os módulos são ordenados também não houve dificuldade, pois foi implementado o algoritmo *Insertion Sort* para ordenar os módulos. Sendo assim, foi necessário apenas analisar o módulo a partir da ferramenta *SonarQube* [6] e utilizar os dados obtidos sobre a quantidade de *bad smells* como entrada no algoritmo de ordenação. Este processo poderia ser integrado com a ferramenta, visto que os dados são obtidos por meio dela. Isto representa uma restrição que deve ser resolvida se o método proposto for automatizado.

A terceira etapa em que o sistema foi refatorado, houve maiores dificuldades em relação as outras, pois era necessário um bom conhecimento sobre as técnicas de refatoração, padrões de projeto, inversão de controle e metapadrões para identificar partes de códigos no *framework* que era necessário aplicar tais conceitos.

4. TRABALHOS RELACIONADOS

Em comparação com os outros trabalhos, o método proposto tenta unir o que há de melhor entre o que Rapeli [12] e Mens e Tourwé [11] podem oferecer no contexto de *frameworks*. Assim como Rapeli [12], o método proposto também oferece uma forma melhor de se entender como o sistema funciona, contendo a etapa de gerar diagrama de classes. Também oferece a aplicação de padrões de projeto de forma específica, como um meio de se refatorar o *framework*.

No método proposto também é possível identificar os tipos de refatoração a serem aplicados, assim como o de Mens e Touwé [11].

A Tabela 2 apresenta a comparação entre o método de Rapeli [12], Mens e Tourwé [11] e o método proposto neste trabalho. Diferente dos métodos de Rapeli [12] e Mens e Tourwé [11], o novo método apresenta a utilização de ferramentas automatizadas, que auxiliam na refatoração do *framework*, identificando mais facilmente pontos do código que necessitam de atenção. Também apresenta a refatoração a partir da utilização de Inversão de Controle e Metapadrões, que são dois conceitos utilizados na criação de um *framework*, logo devem ser levados em consideração na refatoração.

Tabela 2: Comparação entre os métodos de Rapeli [12], Mens e Tourwé [11] e Método Proposto

Características	Rapeli	Mens e Tourwé	Método Proposto
Compreender a funcionalidade do sistema	**X**	**X**	**X**
Gerar diagrama de classe	**X**		**X**
Aplicar padrões de projeto de forma específica	**X**		**X**
Testar sistema refatorado	**X**	**X**	
Refatorar para qualquer linguagem		**X**	
Analisar código para refatorar	**X**	**X**	**X**
Identificar tipo de refatoração a serem aplicadas		**X**	**X**
Utilizar ferramentas automatizadas para análise do código			**X**
Aplicar Inversão de Controle			**X**
Aplicar Metapadrões			**X**

O método proposto não contempla etapas que devem ser realizadas para a fase de teste, somente explicita que a validação deve ser realizada, mas não demonstra como. Devido

a isso, na Tabela 2, a característica *Testar sistema refatorado* foi deixada em branco.

A diferença do método proposto em relação aos métodos da literatura são: uso de ferramentas de análise de código, utilização do guia para a aplicação de metapadrões e inversão de controle. Em relação ao método de Mens e Tourwé [11] e Rapeli [12] possui as seguintes semelhanças: compreender o sistema e analisar código. Comparando com Mens e Touwé [11] o método possui formas de identificar os locais do código que a refatoração será aplicada. Considerando o método de Rapeli [12], a igualdade está nas tarefas de aplicar padrões de projeto e criar o diagrama de classe.

5. CONCLUSÃO

Este trabalho apresentou um novo método para a refatoração, porém diferente dos métodos já publicados, pois foca no contexto de *frameworks* de domínio em que se utilizou as características tais como: padrões de projeto, metapadrões, inversão de controle e ferramentas automatizadas para análise de código, além de utilizar também técnicas de refatoração.

O método proposto é composto por 3 (três) etapas principais: *Entender o sistema*, *Ordenar módulos* e *Refatorar módulo*, sendo que cada etapa contém seus passos a serem seguidos.

Para a validação, o método proposto foi aplicado em um estudo de caso, o *framework* de domínio FrameMK, que é desenvolvido e mantido pelo Grupo de Pesquisa em Sistemas de Informação (GPSI) [5]. A aplicação contemplou todas as etapas e passos propostos, desde o entendimento do sistema, seguindo para a escolha dos módulos até chegar na etapa de refatoração, onde foi aplicada as características identificadas como importantes para um *framework* de domínio.

Com as refatorações efetuadas no FrameMK conseguiu-se diminuir significativamente a quantidade de *bad smells* que estavam presentes no código fonte, conseguindo também aumentar a flexibilidade, diminuir a complexidade, reduzir a duplicação de código, além de fornecer também um código fonte mais legível, possibilitando com que o framework possa ser expandido com maior facilidade.

5.1 Trabalhos Futuros

Os trabalhos futuros que podem ser realizados a partir desta pesquisa são: A identificação de novas características na construção de *frameworks* de domínio que devem ser levados em consideração no processo de refatoração. A aplicação de métricas para avaliar quantitativamente a contribuição do método na refatoração de *frameworks*. Aplicar o método proposto em outros estudos de caso ou módulos. Automatizar o processo de detecção de metapadrões em código fonte. Refinar o modelo de classes do FrameMK de modo a facilitar a engenharia reversa. Definir as etapas que devem ser realizadas na fase de validação do *framework*.

6. REFERÊNCIAS

[1] Checkstyle 6.7. http://checkstyle.sourceforge.net/. Accessado: 05/09/2016.

[2] Codepro analytix. https://marketplace.eclipse.org/content/codepro-analytix. Accessado: 05/09/2016.

[3] Findbugs 3.0.1. http://findbugs.sourceforge.net/. Accessado: 05/09/2016.

[4] Framemk. http://gpes.pg.utfpr.edu.br/framemk/. Accessado: 05/09/2016.

[5] Grupo de pesquisa em sistemas de informação (gpsi). http://gpes.pg.utfpr.edu.br/gpes/. Accessado: 05/09/2016.

[6] Sonarqube 5.1.1. http://www.sonarqube.org/. Accessado: 05/09/2016.

[7] Sqale. http://www.sqale.org/wp-content/uploads/2011/05/SQALE-Method-EN-V0-09.pdf. Accessado: 05/09/2016.

[8] V. P. A. Barros. Um método para refatoração de software baseado em frameworks de domínio, 2015.

[9] M. Fayad, D. C. Schmidt, and R. E. Johnson. *Building application frameworks: object-oriented foundations of framework design*. 1999.

[10] M. Fowler. *Refactoring: improving the design of existing code*. Pearson Education India, 2009.

[11] T. Mens and T. Tourwé. A survey of software refactoring. *IEEE Transactions on software engineering*, 30(2):126–139, 2004.

[12] L. R. C. Rapeli. *Refatoração de sistemas Java utilizando padrões de projeto: um estudo de caso*. PhD thesis, Dissertação (Mestrado em Ciência da Computação), UFSCar–São Carlos, 2005.

Desenvolvimento de Interface Gráfica para Gerenciamento de um Smart Parking

Felipe Felix Ducheiko
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato, s/n - km 04
84016-210 - Ponta Grossa - PR - Brasil
felipeducheiko@alunos.utfpr.edu.br

Gleifer Vaz Alves
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato, s/n - km 04
84016-210 - Ponta Grossa - PR - Brasil
gleifer@utfpr.edu.br

RESUMO

O projeto MAPS (*MultiAgent Parking System*) é idealizado com objetivo de desenvolver um Sistema Multiagente para alocação de vagas em um estacionamento inteligente, onde um agente *manager* é responsável pela alocação de vagas aos agentes *drivers*. O artigo aqui apresentado faz parte do projeto MAPS e tem como objetivo implementar uma interface gráfica, a qual poderá auxiliar o controle e gestão de um estacionamento inteligente.

Palavras-chave

Sistema Multiagente; Cidade Inteligente; Estacionamento Inteligente; *Framework* JaCaMo; Interface Gráfica.

ABSTRACT

The MAPS project (MultiAgent Parking System) is designed with the objective of developing a Multi-Agent System for allocation of parking spots in a smart parking. Where an agent manager is responsible for assigning the parking spot to the agent drivers. Here we aim to build a graphic interface in order to help the management of a smart parking and it is part of the MAPS project.

Keywords

Multi-Agent system; Smart city; Smart Parking; JaCaMo framework; graphic interface.

1. INTRODUÇÃO

Frequentemente novas tecnologias são desenvolvidas ou aprimoradas a fim de solucionar problemas enfrentados pela população. Cidade Inteligente (*Smart City*) refere-se ao uso de novas tecnologias na tentativa de solucionar os problemas das cidades a fim de melhorar a qualidade de vida de seus habitantes.

A concepção de Cidade Inteligente surgiu durante a última década com a fusão de várias ideias, tendo o intuito de melhorar a eficiência e a competitividade das cidades, criando novas maneiras para solucionar problemas. A essência do conceito é integrar as tecnologias que até agora têm sido desenvolvidas separadamente umas das outras, mas que tem ligações claras em seu funcionamento e podem ser desenvolvidas de forma integrada [1].

Um dos principais problemas enfrentados atualmente pelas cidades é o de mobilidade urbana. Estima-se que em Nova York cerca de 40% do tráfego é gerado por carros a procura de vagas de estacionamento, este fato ocasiona um agravamento dos congestionamentos e por conseguinte a emissão de poluentes [6].

Quando se percebe que a grande demanda de vagas de estacionamentos não está sendo satisfeita, normalmente provem a noção de que a solução é um aumento quantitativo do número de vagas. Porém, nem sempre essa é a solução mais sensata, pois a utilização das mesmas vagas de modo mais inteligente pode solucionar ou amenizar o problema. Um *Smart Parking* (Estacionamento Inteligente) faz uso de dispositivos de *hardware*, capazes de detectar o nível de ocupação dos estacionamentos, e de *softwares* integrados para gerir a atribuição desses espaços de maneira a otimizar sua utilização [7].

Várias abordagens de pesquisa em *Smart Parking* estão sendo realizadas para solucionar problemas de mobilidade. Estas abordagens podem ser categorizadas conforme as tecnologias empregadas em sua implementação, algumas das categorias são: *Automated Parking* (foco em mecanismos computadorizados), *E-Parking* (utiliza Internet ou SMS) e *Agent Based Guiding System* (ABGS), esta última é caracterizada pela utilização de Sistemas Multiagentes na implementação de *Smart Parkings* [8].

Sistemas Multiagentes (SMAs) são definidos como sistemas compostos de vários elementos computacionais que realizam interações, sendo tais elementos conhecidos como agentes. Esses sistemas possuem duas características importantes: primeiramente são, ao menos em certa medida, capazes de ações autônomas e em segundo lugar têm a capacidade de interagir uns com os outros de maneira análoga às interações sociais humanas. Além disso, os agentes estão envolvidos em um ambiente onde eles podem ter organização, comunicação, entre outros aspectos [9].

Com o propósito de aplicar métodos e técnicas de SMA, na criação de uma solução para alocação de vagas e gerenciamento de um *Smart Parking*, foi concebido o projeto MAPS (*MultiAgent Parking System*). No desenvolvimento do MAPS é usado o *framework* JaCaMo, o qual possui três componentes principais: *Jason* (programação de agen-

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

tes autônomos), *Cartago* (programação de artefatos do ambiente) e *Moise* (programação da organização do SMA) [4].

Na versão inicial do MAPS era usado apenas o console padrão do JaCaMo, o qual oferece uma interface de texto com informações básicas a respeito dos agentes. Porém, a interface de usuário deve ser clara, intuitiva e de fácil compreensão para a uma efetiva utilização do sistema, visto que usuários têm um grande potencial para fazer interpretações inesperadas de elementos da interface e para executar o seu trabalho de forma diferente do aguardado [5]. A partir disso, notou-se a necessidade de criar uma interface gráfica mais intuitiva e com informações adicionais sobre o *Smart Parking*.

Especificamente, o artigo aqui apresentado tem como objetivo principal apresentar o desenvolvimento de uma interface gráfica para o projeto MAPS, esta interface é voltada para auxiliar o controle e gestão do *Smart Parking*. A implementação deste novo recurso visa também futuras expansões do projeto. Como, por exemplo, a aplicação de diferentes algoritmos para alocação de vagas, onde o uso da interface poderá facilitar a visualização (e comparação) dos resultados gerados pelos algoritmos implementados.

2. PROJETO MAPS

O projeto MAPS é desenvolvido no GPAS (Grupo de Pesquisa em Agentes de Software - UTFPR - PG) com a meta principal de elaborar soluções para Estacionamentos Inteligentes. Para implementar o *Smart Parking* foram criados dois tipos de agentes (implementados em *Jason*): os agentes *drivers* que interagem e utilizam o Sistema Multiagente e o agente *manager* que é responsável por informar, alocar e gerenciar as vagas do estacionamento [2].

O sistema também é composto por dois artefatos (implementados no *Cartago*): o artefato *control* que é responsável pelo gerenciamento dos motoristas na fila e o artefato *gate* que é responsável pela cancela. A alocação de vagas é realizada conforme o grau de confiança (*degree of trust*, ou apenas *trust*) de cada motorista, visto que o grau de confiança e a reputação são temas centrais para a interação efetiva em um SMA aberto em que os agentes entram e saem do sistema [3]. O grau de confiança para cada agente *driver* é incrementado a cada momento que o *driver* usa o estacionamento. Logo, aquele *driver* que utiliza o estacionamento com maior frequência, terá um maior grau.

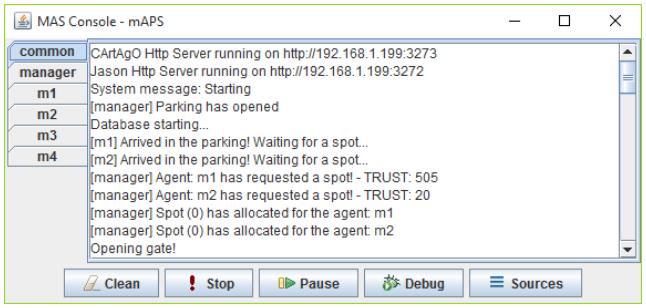

Figura 1: Console do *framework* JaCaMo

Atualmente o MAPS conta apenas com um console, disponibilizado pelo próprio *framework* JaCaMo, que pode ser visualizado na Figura 1. Esse console tem como objetivo exibir a execução dos diferentes agentes do sistema, mas são apenas informações das ações dos agentes, por exemplo, qual agente conseguiu uma determinada vaga. Todavia, não há informações a respeito da quantidade de vagas já ocupadas no estacionamento, por exemplo. Assim, identificou-se a necessidade de desenvolver uma interface gráfica que atende-se tais requisitos. A partir desta nova interface acredita-se que será possível realizar não apenas o gerenciamento do *Smart Parking*, mas também a visualização de simulações do SMA.

3. IMPLEMENTAÇÃO DA INTERFACE

Para o desenvolvimento de uma interface é necessário levar em consideração a usabilidade, que consiste na utilização de métodos que têm o objetivo de facilitar o uso de um sistema. O desenvolvimento desta interface gráfica se deu pautado nos seguintes requisitos de usabilidade: (i) facilidade de aprendizagem: o sistema deve ser facilmente assimilado pelo usuário; (ii) eficiência: o sistema deve ser eficiente para que o usuário possa atingir uma boa produtividade; (iii) facilidade de memorização: o sistema deve ser facilmente memorizado, para que depois de algum tempo sem o utilizar, o usuário se recorde como usá-lo; (iv) segurança: o sistema deve prever erros, evitar que os usuários os cometam e, se o cometerem, permitir fácil recuperação ao estado anterior; e (v) satisfação: o sistema deve ser usado de uma forma agradável, para que os usuários fiquem satisfeitos com a sua utilização [5].

A linguagem Java foi escolhida para implementar a interface gráfica, devido a facilidade de realizar sua comunicação com o *framework* JaCaMo. A interface gráfica desenvolvida também é facilmente extensível, devido às futuras alterações nas versões do sistema.

Na tela inicial da interface gráfica, que pode ser visualizada na Figura 2, o usuário já possui informações sobre o nível de ocupação do *smart parking* e sobre a situação da fila de espera. Referente ao nível de ocupação do *smart parking* o usuário pode verificar o número de motoristas na fila e o tempo médio de espera, quanto a situação da fila de espera o usuário pode verificar a quantidade total de vagas e o número de vagas livres e em uso.

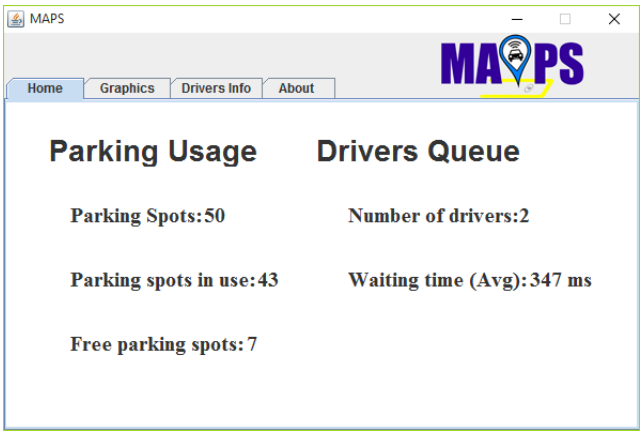

Figura 2: Aba Home

Na aba *graphics*, que pode ser visualizada na Figura 3, o usuário possui informações sobre o nível de ocupação do *smart parking*, através de um gráfico de barra (na Figura 3 mais de 3/4 (três quartos) das vagas estão ocupadas) e sobre a situação da fila de espera, através de um gráfico de linha

que relaciona o horário e quantidade de motoristas na fila (na Figura 3 o valor máximo de *drivers* na fila de espera foi de 5). Estas informações são apresentadas através de gráficos simples para que sejam facilmente verificadas.

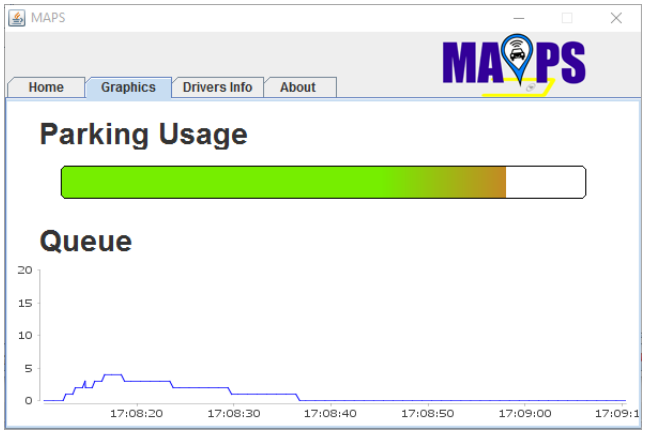

Figura 3: Aba Graphics

Na Figura 4 tem-se a aba *drivers info* que traz os valores máximo, mínimo e médio do tempo de permanência *(Parking time)* e também do grau de confiança *(Trust Degree)* dos motoristas que utilizam o *smart parking*. Essas informações são úteis para melhorias futuras, por exemplo se o valor médio do *trust* estiver alto pode ser um indício de que novos motoristas teriam dificuldade de conseguir vagas. Logo, essas informações podem ajudar o agente *Manager* a calibrar os valores para assegurar o bom funcionamento do SMA.

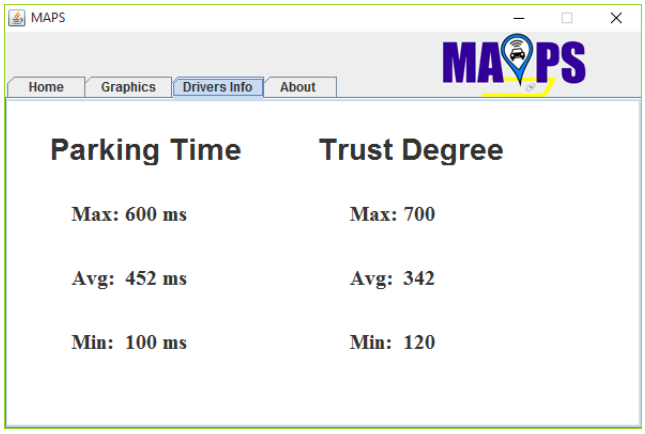

Figura 4: Aba Drivers Info

Para incorporar a interface desenvolvida em Java ao sistema MAPS foi criado o artefato *GUInterface*, utilizando o *Cartago*. Este artefato é responsável pela criação da interface gráfica, nele se encontram todos as operações responsáveis pela manipulação da mesma.

4. CONCLUSÃO

Este trabalho apresenta a implementação de uma interface gráfica voltada ao controle e gestão do MAPS. Isto é uma evolução para o projeto, pois além de ajudar na administração do *Smart Parking* pode auxiliar na visualização e comparação de resultados gerados por simulações, o que é útil para testar novas extensões do projeto, como novos algoritmos para distribuição de vagas e gerenciamento da fila de espera dos agentes *drivers*. Com a interface gráfica é possível perceber facilmente se os *drivers* estão permanecendo pouco tempo no estacionamento, ou seja, se há grande rotatividade de vagas. Se isso ocorre, o especialista (responsável pela administração do *Smart Parking*) pode tentar alternativas, como reorganizar as vagas do estacionamento de forma a buscar um melhor uso dos recursos, no caso as vagas. Outra contribuição é a possibilidade de futuramente adaptar esta interface para a versão embarcada do projeto MAPS, assim conseguindo um acompanhamento do funcionamento do *Smart Parking* em diferentes plataformas.

O emprego do *framework* JaCaMo facilitou a implantação dos recursos aqui propostos, visto que o *framework* faz uso de artefatos em Cartago. Por sua vez o código usado em Cartago é basicamente a linguagem Java, isso facilitou a conexão da interface (implementada em Java) com o projeto MAPS.

Em relação aos trabalhos futuros é possível destacar os seguintes: (i) Avaliação de uso da interface; (ii) Criação de interface para dispositivos móveis. Até o momento a interface (aqui apresentada) não foi avaliada usando técnicas de usabilidade de interface. Logo, é necessário realizar a devida avaliação de uso, para aplicar eventuais alterações e ajustes. Além disso, uma das extensões do projeto MAPS diz respeito a criação de aplicativos para dispositivos móveis. E, portanto, será necessário criar uma interface para tal finalidade.

5. REFERÊNCIAS

[1] M. Batty, K. Axhausen, G. Fosca, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali. Smart Cities of the Future. 2012.

[2] L. F. S. d. Castro, G. V. Alves, and A. P. Borges. Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking. 10º Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações. Maceió - AL. mai. 2016. 2016.

[3] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. 2006.

[4] JACAMO. The JaCaMo approach. Disponível em <http://jacamo.sourceforge. net/?page_id=40>, 2011.

[5] N. Jakob. *Usability Engineering*. 1993.

[6] A. Koster, F. Koch, and A. L. Bazzan. Incentivising Crowdsourced Parking Solutions. 2014.

[7] D. D. Nocera, C. D. Napoli, and S. Rossi. A Social-Aware Smart Parking Application. 2014.

[8] G. Revathi and V. R. S. Dhulipala. Smart Parking Systems and Sensors: A Survey. 2012.

[9] M. Wooldridge. *An Introduction to MultiAgent Systems*. J. Wiley, New York, 2nd edition, 2009.

Implementação e verificação formal de estratégias para desvio de obstáculos de veículos autônomos modelados como agentes racionais

Lucas Emanuel Ramos Fernandes
Universidade Tecnológica Federal do Paraná
Ponta Grossa, Brasil
lucfer@alunos.utfpr.edu.br

Vinicius Custodio
Universidade Tecnológica Federal do Paraná
Ponta Grossa, Brasil
viniciuscustodio@alunos.utfpr.edu.br

Gleifer Vaz Alves
Universidade Tecnológica Federal do Paraná
Ponta Grossa, Brasil
gleifer@utfpr.edu.br

Resumo

Em um futuro não tão distante, veículos autônomos serão uma realidade no tráfego urbano. Empresas como Google, Tesla e Uber estão realizando pesquisas para o desenvolvimento de um software capaz de tomar decisões próprias sem intervenção humana. Uma possível abordagem para a criação dessa autonomia necessária para controlar o veículo é por meio da implementação de agentes racionais. Neste cenário, o agente deve ter a capacidade de guiar um veículo e decidir rotas a serem seguidas. Esta tarefa aparentemente simples se torna complicada quando se trata do desvio de obstáculos. Sendo, necessário implementar um agente capaz de identificar quais decisões dever ser tomadas nestas situações. Na área de agentes existem diversas técnicas utilizadas para o desvio de obstáculos. Porém, todo software é susceptível à erros, então é necessário validar as decisões tomados pelo agente, e isto pode ser realizado através do uso da verificação formal. Portanto, este trabalho propõe uma pesquisa sobre a implementação de um agente racional e a verificação formal da tomada de decisão, onde tais agentes representam veículos autônomos em cenários específicos para desvio de obstáculos.

Palavras-chaves

Veículos Autônomos; Agentes Inteligentes; Verificação Formal

Abstract

Autonomous vehicles will soon be a reality in urban traffic. Companies such as Google, Tesla, and Uber are conducting research to develop and spearhead a software capable of making on-demand need-based decisions without human interaction. One possible approach at creating the needed autonomy is through the implementation of rational agents with the ability to guide a vehicle and decide the routes to be followed. This seemingly simple task becomes far more complicated when dealing with preventative collision measures. These unforeseen events require the implementation of an agent capable of identifying what decision should be made in these unique and immediate situations. Rational multi-system agents have many varying techniques ready for object avoidance adoption; however, all software is prone to errors and therefore these techniques of decision making need to be validated as the "best course of action" in the given situation. Thus, this paper proposes a research on the implementation of a rational agent and a formal verification technique to oversee the decision making process, where such agents represent autonomous vehicles in specific scenarios for obstacle avoidance.

Keywords

Autonomous Vehicles; Intelligent Agents; Formal Verification

1. INTRODUÇÃO

O processo de urbanização presenciado no últimos anos tem alterado o cenário urbano mundial. Mais da metade da população hoje vive dentro das cidades, e a previsão para as próximas décadas é que o número de habitantes cresça ainda mais [18]. O aumento de automóveis em circulação é proporcional ao crescimento populacional nas zonas urbanas, e problemas como congestionamento são intensificados. Logo, é necessária a criação de novas estratégias para a administração do tráfego urbano. Uma das soluções propostas e em destaque nos últimos anos é o veículo autônomo, sendo considerada por muitos o próximo grande avanço da indústria automobilística [22].

Um veículo autônomo é um carro capaz de transitar no perímetro urbano, ou em rodovias, por meio de decisões tomadas pelo software. Os veículos autônomos trazem consigo uma grande mudança em toda a dinâmica da indústria e na forma como os carros são utilizados. Wallace *et al* [22] descrevem em seu trabalho o cenário a seguir, possível através da utilização de carros autônomos:

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

> Imagine que você esteja saindo do trabalho às 6:25 da noite. Você ainda possui diversas tarefas a realizar antes de dormir, e um percurso de 25 minutos de carro até a sua casa. Com seu celular, você requisita um carro que, alguns minutos depois estaciona a sua frente sem motorista. Ao entrar no veículo, você diz "Casa" para indicar seu destino. No percurso, você faz suas atividades restantes, quando chega em casa pode usar o tempo para relaxar e curtir a família. Quando você sai do carro ele se move para o próximo cliente.

O exemplo anterior é um dos benefícios da adoção desta tecnologia, que é baseado em um sistema de partilha de automóveis. Com isso, além de trazer maior comodidade, será possível diminuir a quantidade de veículos em circulação e aumentar a disponibilidade de vagas em estacionamentos. Durante sua vida útil, um carro só está ativo 5% desse tempo. Por meio compartilhamento de carros autônomos este índice pode subir para 75% [8]. Consequentemente, a utilização de menos carros acarretará na diminuição de emissão de dióxido de carbono (CO2).

Dentre as outras vantagens dos veículos autônomos, temos: (i) O potencial de diminuir o congestionamento nas cidades por meios de predição de tráfego e análise padrões no trânsito [12]; (ii) Jornadas mais rápidas e movimentação entre veículos com o uso comboios sincronizados de automóveis trocando dados [8]; (iii) Possível redução do número de acidentes de trânsito com um software como motorista, uma vez que o fator humano é a causa de 90% dos acidentes no trânsito [19].

Atualmente, existem 33 empresas desenvolvendo projetos para carros autônomos [5], e as referências na área são: Uber, Google e Tesla. O Uber prepara-se para realizar testes com carros autônomos [6], aonde será possível requisitar o carro utilizando o aplicativo, que o buscará e o levará ao destino. A Google possui diversos carros autônomos em fase de teste, aproximadamente 58 carros circulam por cidades dos Estados Unidos [15]. E a Tesla já possui um modelo semi-autônomo em comercialização, o Tesla Model S [20].

Uma das capacidades esperadas em um veículo autônomo é a habilidade de navegação autônomo, ou seja, não é necessário um motorista. A navegação dessa categoria de veículos requer que o automóvel esteja preparado para todas as situações possíveis dentro do trânsito. Além de controlar funções vitais como acelerar, frear e identificar sinais de trânsito, o automóvel estar apto a detectar e desviar de obstáculos em seu trajeto para evitar colisões [16]. Os obstáculos podem ser de diferentes formas e tamanhos, e serem estáticos ou dinâmicos. Pessoas atravessando a rua, animais na rodovia e veículos quebrados no meio do trânsito são possíveis cenários onde o automóvel deve ser capaz de desviar para garantir a segurança de seus passageiros.

O desvio de obstáculos pelo veículo está relacionado com a tomada de decisão do software controlador do mesmo. É necessário que o sistema seja capaz de interpretar o ambiente onde está e decidir qual ação é a melhor opção a ser executada. Logo, é imprescindível a correta compreensão dos dados. Além disso, o software deve ser de qualidade para garantir a segurança do veículo autônomo e seus passageiros.

Para entender como implementar e verificar as decisões por um veículo autônomo, é necessário compreender a tecnologia utilizada para o desenvolvimento do software controlador deste sistema. Uma das abordagens para o desenvolvimento do software controlador de veículos autônomos é por meio de agentes. Um agente é uma entidade computacional inserida em um ambiente capaz de tomar decisões por si próprio [23]. Para verificar as decisões realizadas por um agente, é necessário saber por que o agente decidiu executar determinada ação [13]. Então, a utilização de um agente capaz de explicar os motivos que o levou a tomar suas decisões é necessário, e esse é o propósito dos agentes racionais [24]. O desenvolvimento de agentes racionais pode ser realizado com Jason [4] ou Gwendolen [10] (derivada do Jason), ambas linguagens têm sua implementação baseada em Java.

Com base nos conceitos abordados anteriormente, este trabalho tem o intuito de apresentar propostas de pesquisas para implementar e verificar as decisões tomadas por um veículo autônomo, modelado através de um agente racional.

Na seção 2 são explorados os conceitos envolvendo a implementação da navegação do automóvel e as técnicas que podem vir a ser utilizadas. A seção 3 descreve a importância da verificação do agente controlador do veículo autônomo e de qual forma é possível garantir a corretude das ações realizadas pelo mesmo. Por fim, as considerações finais descrevem as conclusões deste trabalho.

2. ELABORAÇÃO E IMPLEMENTAÇÃO DE PLANOS DE DECISÕES PARA DESVIO DE OBSTÁCULOS

A navegação consiste em encontrar um caminho para atingir um determinado objetivo. O caminho deve ser livre de colisões [11], porém isso não é uma tarefa simples, diversos fatores influenciam na navegação. Um aspecto importante que se busca na navegação é que ela seja capaz de desviar de obstáculos. Quando se trata em navegação de veículos isso se torna ainda mais importante e também um fator de risco. Pois, assegurar que veículos encontra um caminho é livre de colisões é garantir a segurança dos ocupantes.

As técnicas do Obstáculos de Velocidade Recíproca [2] e o das Trajetórias Elípticas [9] são duas técnicas de desvio de obstáculos que são planejadas para serem utilizadas em conjunto com agentes e sistemas multiagentes.

O método das Trajetórias Elípticas tem como princípio as leis newtonianas, que através de interações de atração e repulsão o agente consegue desviar de obstáculos. O método Obstáculos de Velocidade Recíproca garante o desvio de forma reativa, ou seja, quando um obstáculo surge, existe um plano a ser executado para se realizar o desvio. Este método é uma extensão do Obstáculos de Velocidade, o qual garante que é possível evitar uma colisão por meio da análise de todas as possíveis velocidades no qual o agente pode colidir.

Para a aplicação das técnicas de desvio obstáculos é preciso realizar uma análise e compreensão dos possíveis cenários e obstáculos. Um cenário inicial considerado será utilizando somente obstáculos estáticos, desconsiderando velocidades e curvas, ou seja o veiculo somente se locomove em linha reta. Considerando veículos autônomos, os obstáculos podem ser pedestres, animais, carros e diversos outros, sendo indispensáveis a incorporação deste conhecimento ao agente, para que este tenha a capacidade de reconhecê-los para então tomar a decisão adequada. Um agente com tais capacidades, possivelmente será um agente racional, aonde esse possui

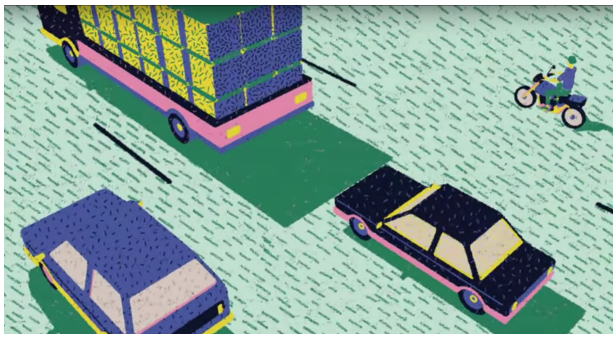

Figura 1: Cenário onde carro autônomo possui um SUV à esquerda, uma motocicleta à direita e um caminhão carregado à sua frente [17].

todos os conhecimentos para evitar obstáculos e e acidente ao mesmo tempo, sendo que uma decisão errada por gerar graves consequências.

Com a definição dos possíveis cenários e tomadas de decisões será possível a implementação do agente no Jason [4] incorporando as tomadas de decisões aos agentes. No Cartago [1] por sua vez será possível o implementação de alguns cenários, escolhendo obstáculos e o ambiente de inserção do agente que foi implementado com o Jason.

Considerando as decisões tomadas será necessário a realização da verificação de desta, para assegurar que são seguras e não sucetíveis a falhas.

3. VERIFICAÇÃO FORMAL DA TOMADA DE DECISÃO DE AGENTES RACIONAIS

Para entender a importância de verificar as decisões tomadas pelo sistema controlador do veículo autônomo, considere o seguinte cenário e a Figura 1, proposto por Lin *et al* [17]:

> Um caminhão carregado de madeiras está se movendo na frente de um carro autônomo. Nas faixas ao lado encontram-se uma motocicleta e uma SUV (*sport utility vehicle*, ou carro utilitário esportivo). Uma das torras se desprende do carregamento do caminhão e irá colidir com o veículo. No entanto, não há tempo para frear antes da batida. Agora o automóvel tem três opções: desviar em direção a SUV ou para a moto, ou enfrentar o impacto.

A situação descrita anteriormente, e evidenciada na Figura 1, é uma versão moderna e real do famoso Dilema do Bonde (*Trolley Problem*) proposto por Phillipa Foot [14]. Independente da escolha feita em cenários como o anterior, deve-se garantir que o veículo autônomo irá se comportar conforme foi implementado. Como o veículo irá atuar no trânsito e pode causar danos à terceiros e aos seus próprios passageiros, é imprescindível que o software por trás dessa tecnologia funcione corretamente. Então, o automóvel não deve executar uma ação própria de um cenário crítico (e.g., eminência de colisão com outro veículo) em um cenário comum (e.g., desalerar e fazer uma leve curva à direita), ou vice-versa.

Por meio da verificação formal, é possível validar as escolhas feitas pelo agente controlador do veículo autônomo e suas especificações. A verificação formal é utilizada para garantir a corretude do algoritmo utilizado em um sistema com relação as suas propriedades. Esta metodologia é utilizada em sistemas de segurança crítica, onde é imprescindível o desempenho correto do software [7]. O *model checking* é uma das técnicas de verificação formal, que por meio da análise do conjunto finito de estados de um software valida os requisitos do mesmo [7]. Contudo, não é possível determinar com exatidão quais decisões serão tomadas por um agente antes de sua execução. Visser *et al.* [21] utilizando-se do *model checking program* (uma extensão do *model checking*) desenvolveram uma ferramenta denominada J*ava PathFinder*, que permite realizar o model checking durante a execução do software. Como base no *Java PathFinder*, Bordini *et al.* [3] criaram o *framework Agent JPF*, uma versão específica para agentes racionais.

Assim, a análise e verificação do processo de tomada de decisão de um agente poderá ser realizada por meio do uso do *model checking* e do *model checking program*. E, com o auxílio da ferramenta *Agent JPF*, será possível verificar as estratégias e planos do agente, impondo cenários críticos e cotidianos, com o intuito de validar as especificações do mesmo.

A verificação formal será realizada após a conclusão do desenvolvimento do agente. O agente será exposto às situações para qual foi programado e terá suas tomadas de decisões analisadas.

4. CONSIDERAÇÕES FINAIS

Este trabalho mostra a relevância e importância dos trabalhos para desvio de obstáculos e a verificação das decisões. As tomadas de decisões de um carro são extremamente importantes e com os avanços na área de veículos autônomos tornam-se ainda mais relevantes. Um carro precisa estar preparado para desviar e reagir a todos os obstáculos que o afetam, além de não poder sofrer com defeitos de software. Sendo assim é necessário garantir a segurança das decisões tomadas por meio de métodos formais (*model checking*).

Para trabalhos futuros, temos o desenvolvimento das técnicas de desvio de obstáculos e a verificação formal do mesmo. Inicialmente serão considerados cenários, agentes e planos de decisões simples. Gradualmente será adicionado novas funcionalidades e capacidades aos agentes, e os cenários se tornarão mais complexos. De forma que a complexidade dos planos de decisões aumentará. E, durante todas as etapas, será realizada a verificação formal dos planos de decisões dos agentes.

5. REFERÊNCIAS

[1] A. O. Alessandro Ricci, Mirko Viroli. Cartago: Framework for prototyping artifact-based environments in mas. *Environments for Multi-Agent Systems*, 4389:67–86, 2006.

[2] V. D. Berg, M. C. Lin, and D. Manocha. Reciprocral velocity obstacles for real-time multi-agent navigation. *IEEE Conference Robotics and Automation*, pages 1928–1935, 2008.

[3] R. H. Bordini, L. A. Dennis, B. Farwer, and M. Fisher. Automated verification of multi-agent programs. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, ASE '08, pages 69–78, Washington, DC, USA, 2008. IEEE Computer Society.

[4] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[5] CBINSIGHTS. 33 corporations working on autonomous vehicles, 2016.

[6] M. Chafkin. Uber's first self-driving fleet arrives in pittsburgh this month, 2016.

[7] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

[8] M. Cliffe. Driverless cars – the route to more than smart cities, 2016.

[9] B. Dafflon, J. M. Vilca, F. Gechter, and L. Adouane. Adaptive autonomous navigation using reactive multi-agent system for control law merging. In S. Koziel, L. Leifsson, M. Lees, V. V. Krzhizhanovskaya, J. Dongarra, and P. M. A. Sloot, editors, *Proceedings of the International Conference on Computational Science, Computational Science at the Gates of Nature, Reykjavík, Iceland, 1-3 June, 2015, 2014*, volume 51 of *Procedia Computer Science*, pages 423–432. Elsevier, 2015.

[10] L. A. Dennis and B. Farwer. *Gwendolen: A BDI Language for Verifiable Agents*. University of Aberdeen.

[11] F. Ducatelle, G. A. D. Caro, A. Förster, M. Bonani, M. Dorigo, S. Magnenat, F. Mondada, R. O'Grady, C. Pinciroli, P. Rétornaz, V. Trianni, and L. M. Gambardella. Cooperative navigation in robotic swarms. *Swarm Intelligence*, 8(1):1–33, 2014.

[12] D. J. Fagnant and K. M. Kockelman. Bpreparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Eno Center for Transportation*, 2, 2013.

[13] M. Fisher, L. Dennis, and M. Webster. Verifying autonomous systems. *Commun. ACM*, 56(9):84–93, Sept. 2013.

[14] P. Foot. The problem of abortion and the doctrine of double effect. *Oxford Review*, 5:5–15, 1967.

[15] Google. Self-driving cars reports, 2016.

[16] J. Levinson. Towards fully autonomous driving: System and algorithms. *Intelligent Vehicles Symposium*, 4:164–128, 2011.

[17] P. Lin, Y. Du, A. Adkins, K. O'Shea, J. Wang, and C. Misirlioglu. The ethical dilemma of self-driving cars, 2015.

[18] U. Nations. *Department of Economic and Social Affairs: Population Division. World Urbanization Prospects: The 2014 Revision*. KPMG Center for Automotive Research, 2015.

[19] M. Peden, R. Scurfield, D. Sleet, D. Mohan, A. A. Hyder, E. Jarawan, and C. Mathers. *World report on road traffic injury prevention. World Health Organization*. 2004.

[20] Tesla. Tesla models, 2016.

[21] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda. Model checking programs. *Automated Software Engg.*, 10(2):203–232, Apr. 2003.

[22] R. Wallace and G. Silberg. *Self-Driving cars: The next revolution*. KPMG Center for Automotive Research, 2012.

[23] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley Sons, 2 edition, 2009.

[24] M. Wooldridge and A. Rao. *Foundations of Rational Agency*. Kluwer Academic Publishers, 1999.

Técnicas de Classificação em Problemas Relacionados a Doenças Cardíacas

Douglas Guisi
UTFPR PB
Av. do Conhecimento
Pato Branco – Pr
guisiagudos@gmail.com

Jonatas Loureiro de Almeida Jr.
UTFPR PB
Av. do Conhecimento
Pato Branco – Pr
jonatas.adiemla@gmail.com

André Pinz Borges
UTFPR PG
Av Monteiro Lobato, s/n, Km 04
Ponta Grossa – Pr
apborges@utfpr.edu.br

Marcelo Teixeira
UTFPR PB
Av. do Conhecimento
Pato Branco – Pr
marceloteixeira@utfpr.edu.br

Richardson Ribeiro
UTFPR PB
Av. do Conhecimento
Pato Branco – Pr
richardsonr@utfpr.edu.br

Gabriel Gomes de Sousa,
Hudson dos Santos Lapa
UTFPR PB
sousa@alunos.utfpr.edu.br
hudsonslpa@gmail.com

RESUMO

Este artigo apresenta a aplicação de técnicas de classificação em problemas relacionados a doenças cardíacas. As doenças cardiovasculares são os distúrbios ligados ao coração e aos vasos sanguíneos, com uma vasta gama de síndromes clínicas, sendo a aterosclerose a mais frequente em diagnóstico. Neste artigo, realizou-se um estudo com base nos repositórios de dados destas doenças, a fim de encontrar os parâmetros presentes em exames laboratoriais e prontuários médicos que melhor se conectam para a descoberta de pacientes com possíveis riscos de doenças cardiovasculares. Os resultados atingidos por esta pesquisa mostram que a existência de diabetes, hábitos e concentrações de substâncias no corpo, aumentam a possibilidade de contração de novas doenças relacionadas.

Palavras Chaves

Mineração de Dados; Classificação; Doenças Cardíacas.

ABSTRACT

This article presents the application of classification techniques in problems related to heart-disease. Cardiovascular diseases are disorders related to heart and blood vessels, with a wide range of clinical syndromes, the most common diagnosis of atherosclerosis. In this article, we carried out a study based on data repositories of these diseases to find the parameters present in laboratory tests and medical records to better connect to the discovery of patients with possible risk of cardiovascular disease. The results obtained in this study show that the existence of diabetes, habits and concentrations of substances in the body, increase the possibility of further contraction of diseases.

Keywords

Data Mining; Classification, Heart Diseases.

1. INTRODUÇÃO

As doenças cardiovasculares possuem consideráveis síndromes clínicas, porém as relacionadas à aterosclerose (acúmulo de gordura nos vasos sanguíneos) são as mais frequentes em diagnósticos [6] [29]. Segundo o Ministério da Saúde do Brasil, as doenças cardiovasculares são as responsáveis por aproximadamente 30% dos óbitos, colocando o Brasil entre os 10 países mais afetados do mundo [24].

Os fatores que contribuem para o surgimento destas síndromes podem ser genéticos, mas os principais dependem dos hábitos do paciente, como sedentarismo e tabagismo [6]. A diabete, doença não relacionada ao sistema circulatório também é objeto de estudo como um fator das doenças cardiovasculares. Pacientes com diabetes *mellitus* tipo 2, por exemplo, tem um risco de duas a quatro vezes maior de vir a desenvolver doenças coronárias [19]. Esta doença evolui sem apresentar sintomas notórios ou conhecimento do paciente, fatores que dificultam o diagnóstico. Estima-se que metade das pessoas portadoras não tem conhecimento de estarem com a diabetes *millitus* tipo 2 [22].

Atividades clínicas, como consultas, exames laboratoriais, prescrições médicas, diagnósticos, vacinações, entre outras, são realizadas diariamente por diferentes profissionais da área da saúde. Os dados dessas atividades quando agrupados, emergem um conjunto de dados geralmente na ordem de milhares. Uma possibilidade para obter o conhecimento implícito inerente a um relacionamento específico desses dados é aplicar as etapas do processo de descoberta de conhecimento (*Knowledge Discovery in Databases* - KDD), explorando principalmente os métodos de Mineração de Dados [9][10][28]. Segundo [13], as principais aplicações da Mineração de Dados na saúde estão em efetividade de tratamentos médicos, gerenciamento de sistemas, e detecção de uso indevido e/ou fraudes de recursos destinados à saúde.

Na efetividade de tratamentos médicos, as aplicações de Mineração de Dados podem gerar informações que auxiliem o profissional da área da saúde nas tomadas de decisões como prescrições, exames e encaminhamentos. Ou ainda por meio da análise dos dados, pode-se chegar a conclusões sobre causas de uma doença, sintomas e tratamentos.

Neste artigo integramos o WEKA[31], um software com uma coleção de algoritmos de aprendizagem de máquina para executar tarefas de mineração de dados, à bases de dados *online* de pacientes, obtidos a partir de repositórios de dados (*benchmarks*) relacionados às doenças cardíacas, para a detecção de diabetes e doenças cardiovasculares.

Foram testados algoritmos de classificação usando diferentes métodos, como árvores de decisão, redes neurais artificiais, regressão e redes bayesianas, na intenção de encontrar os melhores modelos para os conjuntos de dados utilizados.

WPCCG'16, Setembro 28, 2016, Ponta Grossa, Paraná, Brasil.

2. SISTEMAS DE APRENDIZAGEM COM MINERAÇÃO DE DADOS

Sistemas de aprendizagem baseados em mineração de dados são geralmente formados por técnicas e procedimentos, que se baseiam na aplicação de algoritmos sobre grupos de dados para a extração de algum conhecimento implícito, que esteja inerente a um relacionamento específico desses dados [28].

Os algoritmos utilizados em sistemas de aprendizagem são normalmente baseados em diferentes áreas do conhecimento (*e.g.*, matemática e estatística), e trabalham por meio de agrupamentos, classificação ou associação, possibilitando a criação de regras sobre os dados. O conhecimento é gerado a partir da interpretação dessas regras [28].

Neste artigo é utilizado o paradigma de classificação. Classificação é o processo de encontrar um conjunto de modelos (funções) que descrevem e distinguem classes ou conceitos, com o propósito de utilizar o modelo para predizer a classe de objetos que ainda não foram classificadas [21].

Um método bastante conhecido são as árvores de decisão [25], aplicado na inferência indutiva. Uma árvore de decisão usa a estratégia dividir para conquistar para resolver um problema de decisão. Nas árvores de decisão, um problema complexo é dividido em problemas mais simples. Nesses subproblemas é aplicada recursivamente a mesma estratégia. As soluções dos subproblemas podem ser combinadas na forma de uma árvore, para produzir a solução de um problema complexo [21]. Essa é a ideia básica por trás de algoritmos baseados em árvores de decisão. Em domínios da área médica, árvores de decisão foram usadas para a análise de custo-efetividade de vacinas [8], problemas de fratura ortopédica [30], tomadas de decisões clínicas [5] [2] [3], tratamento de feridas crónicas [17], etc.

Outras técnicas para classificação foram inspiradas em paradigmas probabilísticos e também foram testadas com dados da área médica. Por exemplo, o *Naive Bayes* é um classificador probabilístico baseado na hipótese no qual as variáveis fornecidas são independentes [18]. Suas primeiras aplicações na medicina surgiram na década de 90, sendo elas na predição do prognóstico de pacientes e seleção de tratamentos. Um exemplo de sua utilização está na classificação de radiografias de tórax para o diagnóstico de pneumonia infantil [27].

Outra técnica são as Redes Neurais Artificiais, inspiradas na estrutura dos neurônios humanos [4]. A principal função delas é a predição de eventos. Uma das dificuldades com esse método, dependendo do problema, é sua complexidade e compreensão das predições realizadas. Vários estudos e aplicações das redes Neurais Artificias em problemas na área médica são encontrados, por exemplo, em [1] [7] [11] [12] [15].

Há ainda diversas outras aplicações que se beneficiaram das técnicas de classificação, como o desenvolvimento de antidepressivos [16], doença da artéria coronária [20], análise de fatores genéticos para a predisposição ao câncer cervical [26], e processamento de sinais EEG [14]. Esses são apenas alguns dentre os vários trabalhados dedicados ao uso de sistemas de aprendizagem na área da saúde.

3. METODOLOGIA

Para gerar a descoberta do conhecimento foi utilizado o Weka. Esse software oferece recursos para a execução de tarefas relacionadas ao pré-processamento de dados como, por exemplo, a seleção e a transformação de atributos (e.g., remoção de dados errôneos, criação de escalas para dos dados). O Algoritmo 1 demonstra um exemplo de como utilizar uma técnica de mineração de dados com a linguagem Java.

```
Instances data = new Instances(new BufferedReader(new
FileReader(
"C:\\weather.nominal.arff")));
data.setClassIndex(data.numAttributes()-1); //Definir o
índice dos dados
String[] options = new String[1];
options[0] = "-U"; //Árvore sem podas
J48 tree = new J48();//Nova instância da árvore
tree.setOptions(options); //Configurar o classificador
tree.buildClassifier(data); //Construir o classificador
```

Algoritmo 1. Exemplo em Java para invocar uma técnica de mineração de dados.

Uma prática utilizada na mineração de dados da saúde é o uso de repositórios de dados de acesso geral e/ou restrito. Existem vários repositórios *on-line* gratuitos e irrestritos, que contém milhares de bases de dados reais, as quais podem ser utilizadas pelos pesquisadores em seus algoritmos e técnicas de mineração de dados. Os dados clínicos estão ligados intrinsecamente a regras éticas e de sigilo, de modo que não é possível encontrá-los com facilidade e muito menos utilizá-los para análise em pesquisas com Mineração de Dados. Logo, o uso de dados de repositórios permite aos pesquisadores experimentar algoritmos na busca por informações.

Este trabalho utilizou as seguintes bases: Heart-statlog[1], Diabetes[2] e Arrhytmia[3], com um objetivo mais específico: elencar os atributos de cada base de dados relacionada a uma doença cardiovascular, que podem ocasionar a ocorrência de uma segunda doença do gênero. A motivação deste trabalho parte de uma pesquisa, desenvolvida por [23], que revelou que doenças cardiovasculares como infarto, hipertensão e angina, possuem relação com a diabetes. Segundo [29] os principais riscos para a ocorrência de doença cardiovascular são: elevado LDL (conhecido como o colesterol ruim), baixo HDL (conhecido como o colesterol bom), alta pressão sanguínea, elevada glicose no sangue, sedentarismo, obesidade, consumo de tabaco. Dessa forma, é possível identificar relações entre a existência de diabetes, hábitos e concentrações de substâncias no corpo e a possibilidade de contração de novas doenças relacionadas.

Após a verificação das correlações existentes entre diferentes atributos e o risco de contrair determinada doença, foram realizados testes computacionais utilizando técnicas de mineração de dados a fim de mostrar resultados com as correlações existentes. A abordagem utilizada para a mineração dos dados foi a classificação. Essa funcionalidade tem por objetivo segmentar um conjunto de dados em subconjuntos específicos. Os dados pertencentes a um subconjunto gerado devem possuir características em comum. Dessa forma, ao final da execução de uma tarefa de classificação, serão criados *n* subconjuntos de dados, cada qual agrupando dados com características específicas. Uma vez aplicada, essa funcionalidade pode atuar sobre novos conjuntos, prevendo em qual subconjunto eles se enquadram.

Para a realização dos testes de classificação foram utilizadas as bases de dados de detecção de diabetes e de doenças cardiovasculares disponíveis nos repositórios públicos indicados. Da base diabetes foi usado 9 atributos, dentre eles: número de vezes que a mulher engravidou, pressão sanguínea (em *mm/Hg*), grossura da camada de pele do tríceps (em *mm*), IMC e idade. Nas bases relacionadas a doença cardiovascular foi utilizado 13 atributos, dentre eles: idade, sexo, escala numérica de dor no peito

[1] http://tunedit.org/repo/UCI/heart-statlog.arff

[2] http://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/diabetes.arff

[3] http://tunedit.org/repo/UCI/arrhythmia.arff

(0-10), nível de açúcar no sangue (em *mg/dl*), taxa máxima de batimentos cardíacos alcançada no exame, entre outros.

O objetivo da aplicação de Mineração de Dados foi detectar a presença ou ausência de diabetes, dada a informação prévia de doença cardíaca. Para a criação da nova base de dados realizou-se uma junção das duas anteriores, eliminando apenas atributos repetidos em ambas: sexo e pressão sanguínea. Portanto, a nova base de dados foi composta por 20 atributos, 1490 instâncias (registros) e duas classes que indicam (positivo) ou não (negativo) a presença de diabetes. Após isso, foram aplicadas técnicas de classificação.

4. RESULTADOS EXPERIMENTAIS

Os resultados experimentais foram obtidos com a ferramenta WEKA e são apresentados na Tabela 1. Para dividir o conjunto de dados em treinamento e teste, foi empregado o método de validação cruzada *k-fold* [28] como forma de avaliar a capacidade de generalização das técnicas de aprendizagem a partir dos conjuntos de dados de treinamento e teste. No procedimento de validação cruzada *k-fold*, o conjunto de dados é dividido aleatoriamente em *k* subconjuntos. Destes, *k*-1 são utilizados para o treinamento e um é utilizado para teste. Esse processo é repetido *k* vezes até que todos os subconjuntos de dados sejam utilizados no conjunto de teste. Dessa forma, diferentes classificadores são obtidos, e a precisão das classificações dos conjuntos de treinamento e de teste pode ser avaliada. Todos os testes foram realizados utilizando validação cruzada com 10-*fold*.

Os resultados da Tabela 1 mostram que as técnicas de classificação por Regressão e *Naive Bayes* obtiveram os melhores resultados, detectando a presença de diabetes tipo 2 com um percentual de acerto médio de 65,7%, sendo que a técnica de regressão foi a que melhor identificou a diabetes nos pacientes. Estas técnicas buscam encontrar funções, geralmente de primeira ordem, capazes de mapear as instâncias do conjunto de dados em valores reais. Exemplos da utilização dessas técnicas se encontram na estimativa da probabilidade de um paciente vir a possuir a doença, dado o resultado de um conjunto de diagnósticos de exames [20] e [14].

Table 1. Percentual de acertos dos algoritmos.

Algoritmos	Acertos (%)
Classificação por Regressão	72,6
Naive Bayes	71,1
Rede Neural Perceptron Multicamadas	65,0
J48 (árvore de decisão)	61,9
Logística Bayesiana com Regressão	61,2

A base de dados utilizada foi numérica, isto é, todas as instâncias possuem em seus atributos domínios com valores inteiros ou reais, tornando propício a aplicação de técnicas de Regressão e Probabilísticas na classificação dos dados. Isto também explica a melhora na taxa de classificação obtida por estas técnicas em relação às demais, uma vez que as demais aceitam outros tipos de dados e, portanto, não são especificamente criados para trabalhar com números.

5. CONCLUSÕES E DISCUSSÕES

Apesar do processo de descoberta do conhecimento apresentar aplicabilidade à área da saúde, nota-se há falta de sistemas comerciais integrados. Existem diversas empresas que promovem serviços de prontuário eletrônico de pacientes, sistemas de suporte a decisão e controle de gestão, porém ainda não é possível fazer uma mineração de dados utilizando os dados clínicos de um país inteiro, por exemplo. Isso se deve ao isolamento entre esses sistemas, que ocorre por estratégias comerciais ou mesmo para respeitar as regras de sigilo dos dados. No entanto a ideia de utilizar escopos de dados abrangentes como deste trabalho pode representar a possibilidade de algumas descobertas importantes, como por exemplo, quais fatores climáticos influenciam na transmissão de determinadas doenças, entre outras informações de nível regional e nacional. Essas hipóteses serão objetos de estudos.

Neste trabalho foi realizado um estudo em repositórios de dados de doenças cardiovasculares, na intenção de encontrar quais os parâmetros presentes em exames laboratoriais ou prontuários médicos que melhor se relacionam na descoberta de pacientes com possíveis riscos de doenças cardiovasculares.

Nos repositórios usados para este trabalho, foram selecionados atributos na intenção de identificar a ocorrência de uma segunda doença do gênero. Outras frentes estão sendo investigadas, como por exemplo, usar outras bases que possam fazer relação a outras doenças cardiovasculares; estender a metodologia proposta a outros problemas da saúde como doenças cancerígenas e respiratórias. Há ainda questões de interação dos resultados do projeto com instituições, como por exemplo, firmar parcerias com instituições públicas e/ou privadas para o benefício de usar dados oriundos de sistemas de prontuários eletrônicos do Brasil. Algumas dessas hipóteses já estão sobre investigação para trabalhos futuros.

6. AGRADECIMENTOS

Está pesquisa é apoiada por Decit/SCTIE/MS, por intermédio do CNPq, com apoio da Fundação Araucária e da SESA-PR. Programa Pesquisa para o Sistema Único de Saúde: Gestão Compartilhada em Saúde - PPSUS.

7. REFERÊNCIAS

[1] Akay, M. and Welkowitz, W. (1993) “Acoustical detection of coronary occlusions using neural networks”. Journal of Biomedical Engineering, 15(6), pp. 469-73.

[2] Aleem, I. S, Jalal, H., Sheikh, A. A. (2009) "Clinical decision analysis: Incorporating the evidence with patient preferences". In: Patient Preference and Adherence. Vol. 3, pp. 21-24.

[3] Bae,J-M. (2014) “The clinical decision analysis using decision tree”. In: Epidemiology and Health. Vol. 36, pg 1-7.

[4] Bishop, C. M. (1995) "Neural Networks for Pattern Recognition". Oxford: Oxford University Press.

[5] Bonner, G. (2001) “Decision making for health care professionals: use of decision trees within the community mental health setting”. In: Journal of Advanced Nursing, vol. 35, pp. 349-356.

[6] Brasil (2006) “Prevenção clínica de doenças cardiovasculares, cerebrovasculares e renais”. Brasília: Ministério da Saúde.

[7] Costa, F. O. de, Motta, L. C. S. and Nogueira, J. L. T. (2010) “Uma abordagem baseada em Redes Neurais Artificiais para o auxílio ao diagnóstico de doenças meningocócicas”. In: Revista Brasileira de Computação Aplicada, v. 2, n. 1, pp. 79-88.

[8] De Soarez, P. C. (2009) “Use of decision analysis in the programs of vaccination against varicella”. São Paulo: “Faculdade de Medicina da Universidade de São Paulo”.

[9] Fayyad, U. M. (1996) "Data mining and knowledge discovery: making sense out of data". IEEE Expert: IEEE

Expert: Intelligent Systems and Their Applications. Vol. 11, Issue 5, pp. 20-25.
[10] Hann, J. and Kamber, M. (2006) "Data Mining: Concepts and Techniques". Second Edition. San Franciso, CA : Morgan Kaufmann.
[11] Iyer, A., Jeyalatha, S. and Sumbaly, R. (2015) “Diagnosis of Diabetes Using Classification Mining Techniques”. International Journal of Data Mining & Knowledge Management Process (IJDKP). Vol.5, No.1, pp. 1-14.
[12] Junga, S-K. and Kim, T-W. (2016) “New approach for the diagnosis of extractions with neural network machine learning”. American Journal of Orthodontics and Dentofacial Orthopedics, Vol 149, Issue 1, pp. 127-133.
[13] Koh, H. and Tan, G. (2005) “Data Mining Applications in Healthcare”. In: Journal of Healthcare Information, v. 19, pp.64-72.
[14] Kutlu, Y., Isler, Y., Kuntalp, D. and Kuntalp, M. (2006) “Detection of Spikes with Multiple Layer Perceptron Network Structures” In: Signal Processing and Communications Applications, 2006.
[15] Lahner, E., Intraligi, M., Buscema, M. (2008) “Artificial neural networks in the recognition of the presence of thyroid disease in patients with atrophic body gastritis”, World Journal of Gastroenterology. Vol. 14, pp. 563-8.
[16] Lesch, K. P. (2004) “Serotonergic gene expression and depression: implications for developing novel antidepressants”. In: Journal of Affective Disorders, vol. 62, p.57-76.
[17] Letourneau, S., Jensen, L. (2008) "Impact of a decision tree on chronic wound care". Journal Wound Ostomy Continence Nurs, vol. 25, pp. 240-247.
[18] Lewis, D. D. (2005) “Naive (Bayes) at forty: The independence assumption in information retrieval”. In: Lecture Notes in Computer Science, v. 1398, pp. 4-15.
[19] Lima, B. P., Morais, C. A., Contrera, D., Casale, G., Pereira, M., Gronner, M., Diogo, T., Torquarto, M., Oishi, J. and Leal, A. (2003) “Prevalence of diabetes mellitus and impaired glucose tolerance in the urban population aged 30-69 years in Ribeirão Preto (São Paulo), Brazil”. In: São Paulo Med. J., v.121, pp.224-230.
[20] Liping, A. and Lingyun, T. (2005) “A rough neural expert system for medical diagnosis, Services Systems and Services Management”, vol. 2, pp. 1130-1135.
[21] Mitchell, T. (1997) "Machine Learning". New York: McGraw-Hill.
[22] Moreira, T. (2013) “Narrativas de pessoas com diabetes atendidas na rede básica: determinantes da hospitalização”. Dissertação de Mestrado. UFB.
[23] Nesto, R. (2004) “Correlation between cardiovascular disease and diabetes mellitus: current concepts”. In: The American Journal of Medicine, v. 116, Issue 5.
[24] Portal, B. (2016) “Doenças cardiovasculares causam quase 30% das mortes no País”. http://www.brasil.gov.br/saude. Acesso em: 05 de junho de 2016.
[25] Quinlan, C. (1993) “C4.5: Programs for machine learning”. Morgan Kaufmann.
[26] Saraee, M. and Ritchings, T. (2004) “Medical data mining: case of cervical cancer screening”. In: METMBS 04, 21-24 June 2004.
[27] Souza, R. T. (2013) “Avaliação de classificadores na classificação de radiografias de tórax para o diagnóstico de pneumonia infantil”. Dissertação de Mestrado. Programa de Pós-graduação em Ciência da Computação (INF), Universidade Federal de Goiás, pp. 63.
[28] Witten, I. and Frank, E. (2005) “Data Mining: Practical machine learning tools and techniques”. In Morgan Kaufmann.
[29] Yusuf, S., Reddys, S., Ôunpuu, S. and Anand S. (2001) “Global burden of cardiovascular diseases - part I: general considerations, the epidemiologic transition, risk factors, and impact of urbanization”, In: Clinical Cardiology: New Frontiers, v. 104, pp. 2746–2753
[30] Zorman, M., Podgorelec, V., Kokol, P., Peterson and M., Lane, J. (2000) “Decision tree's induction strategies evaluated on a hard real world problem”. In: Proceedings of the 13th IEEE Symposium on Computer-Based Medical Systems CBMS'2000, pp. 19-24.
[31] Hall, M., Frank, E., Holmes, G. Pfahringer, B., Reutemann, P., Witten, I. H. (2009) “The WEKA Data Mining Software: An Update”. SIGKDD Explorations, Volume 11, Issue 1.

Detecção de Falhas em Painéis Fotovoltaicos Usando Imagens Infravermelhas de Baixa Resolução Geolocalizadas

Clécius J. Martinkoski
PPGCC - Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato, s/n - Km 04
Ponta Grossa, Paraná
clecius.martinkoski@gmail.com

Ionildo José Sanches
PPGCC - Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato, s/n - Km 04
Ponta Grossa, Paraná
ijsanches@gmail.com

RESUMO

A termografia infravermelha é um método não destrutivo e pode ser utilizada na avaliação de módulos fotovoltaicos com o objetivo de detectar falhas. Este artigo propõe uma metodologia para analisar a saúde de um parque fotovoltaico usando uma câmera infravermelha de baixa resolução e informações georeferenciadas, movendo o ponto de vista do observador e provendo uma relação de grande custo-benefício em compasso com as tendências de microgeração residencial.

Palavras-chave

Imagens infravermelhas, Processamento de imagens, Painéis fotovoltaicos, Energias renováveis.

ABSTRACT

The infrared thermography is a non-destructive method and can be used for the evaluation of photovoltaic modules in order to detect faults. This paper proposes a methodology to analyse of photovoltaic park health using a low resolution infrared camera and georeferenced information moving the observer point of view to provide a high cost-benefit relation in compass to residential microgeneration tendencies.

Keywords

Infrared Imaging, Image Processing, Photovoltaic, Renewable energies.

1. INTRODUÇÃO

Com a crescente demanda de energia elétrica e o agravamento dos impactos ambientais gerados pelos métodos tradicionais de geração, tem-se popularizado as ditas energias renováveis, dentre as quais se destacam a energia eólica e a energia solar fotovoltaica, sendo estas o presente e o futuro da geração renovável respectivamente [6].

A energia solar fotovoltaica é a terceira fonte de energia renovável mais importante, atrás da hidráulica e eólica. Ela desponta com uma crescente popularidade devido as recentes políticas de micro e mini geração, que permitem uma matriz energética distribuída com menores custos de transmissão. Entretanto, a mesma sofre limitações tecnológicas e orçamentárias para sua popularização em massa [3].

Um dispositivo de termografia infravermelho é usado para converter radiação infravermelha, emitida pela superfície de um corpo, em impulsos elétricos que podem ser visualizados através de uma imagem em escala de cinza ou colorida utilizando pseudo-cores.

Para efetuar a detecção dos pontos de falha de forma não intrusiva e com antecipação para viabilizar a manutenção preventiva, tem-se optado pelas imagens infravermelhas. Pontos quentes como nas figuras 1 e 2, que comumente estão associados a células defeituosas ou a regiões de sombreamento, podem vir a danificar a placa [1]. Este tipo de pesquisa vem sendo alvo de vários estudos durante os últimos anos [1].

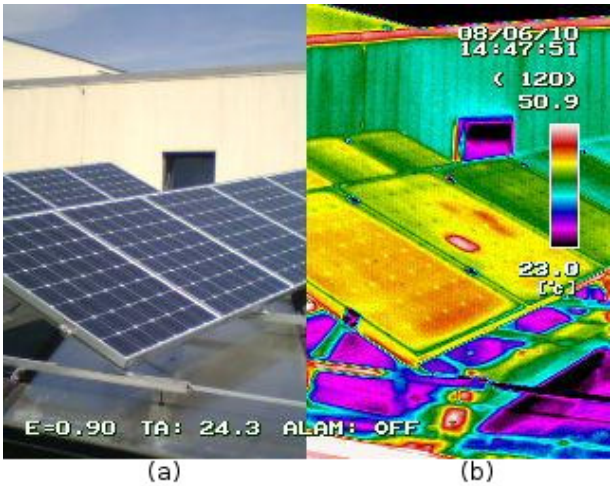

Figura 1: Imagens de painéis fotovoltaicos. (a) Imagem na faixa do espectro visível e (b) na faixa do espectro infravermelho [4].

Também é sabido que as tecnologias aplicadas ao uso de

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

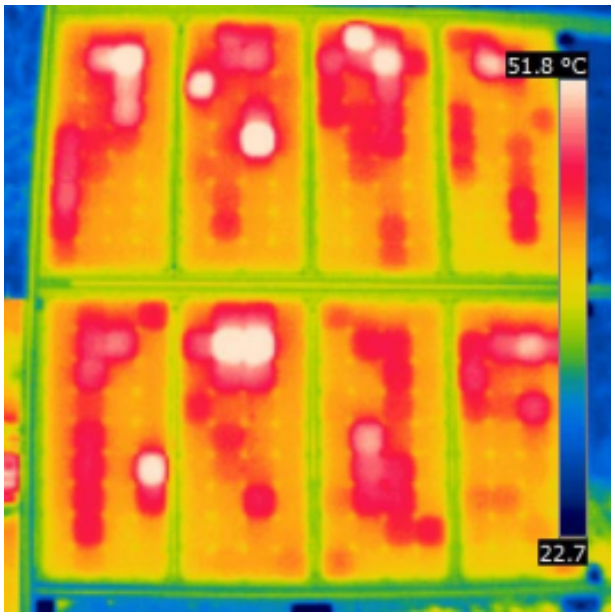

Figura 2: Imagem infravermelha de um painel fotovoltaico com falhas [5].

termografia infravermelha, são incompatíveis, sob a perspectiva de custos, com a visão da mini e micro geração, tornando proibitiva a disseminação dessas tecnologias nestes cenários [2].

A utilização da termografia infravermelha, objetivo desta pesquisa, contextualiza-se nas restrições orçamentarias delimitadas nos cenários de mini e micro geração. Para esta adequação faz-se necessário o uso de equipamentos mais limitados, como câmeras infravermelhas de baixa resolução e baixo volume, que possam inclusive serem acopladas a pequenos drones para realizar as inspeções.

O foco deste trabalho é compilar as tecnologias existentes na área de análise termográfica com infravermelho aplicadas a painéis fotovoltaicos, observando da perspectiva de equipamentos limitados e utilizar de tecnologia via *software* para maximizar a qualidade dos diagnósticos e a eficiência do conjunto.

2. METODOLOGIA DE PESQUISA

Para atingir o objetivo proposto pautar-se-á nos estudos recentes de análise termográfica de painéis fotovoltaicos, que têm apresentado um resultado bem satisfatório na eficiência de detecção de problemas [7]. O foco é aplicar este conhecimento compilado em imagens de baixa resolução, com deslocamento do observador em direção ao ponto quente, para obter um diagnóstico mais preciso sem a necessidade de um *hardware* de captura de imagens térmicas muito avançado.

Para viabilizar o mapeamento de grandes áreas, com uso de câmeras infravermelhas de baixa resolução, será deslocado espacialmente o observador, que permitirá uma aproximação em direção ao ponto quente para obtenção de novas imagens refinadas. Assim executando, recursivamente, até que se possa determinar qual o ponto onde a falha se encontra efetivamente.

O deslocamento do observador será baseado na imagem, para determinar se há ou não uma anomalia, no ponto geográfico onde foi capturado e a direção em que a imagem foi obtida, podendo-se assim determinar para onde o observador deve se deslocar para refinamento da imagem obtida. O processo será repetido até que a informação da imagem seja suficientemente precisa para acusar o ponto em que o ponto quente realmente se encontra.

O cenário simulado será de um conjunto de placas distribuídas em pontos georreferenciados. As imagens serão obtidas de algum destes pontos e baseada na sua localização, direção e informações contidas nas mesmas. O algoritmo fará a análise e decidirá o deslocamento do ponto de aquisição das imagens, visando refinar a detecção do ponto quente e determinar em qual ponto dentro da matriz de placas que ele se encontra.

Espera-se obter um índice de detecção de falhas grande em uma área extensa mesmo com imagens de baixa resolução, demonstrando que a aproximação física do ponto de observação é uma alternativa viável a câmeras térmicas de alta resolução.

Os testes serão executados com imagens reais obtidas de painéis fotovoltaicos e serão enviadas ao algoritmo com sua determinada localização geoespacial. As imagens serão tanto de painéis normais quanto de painéis com pontos quentes. Analisando o seu funcionamento, será possível determinar quão eficiente e ágil é o método para detectar pontos quentes a uma distância relativamente grande.

3. CONCLUSÃO

O uso de termografia infravermelha apesar de bem aplicado em cenários específicos ainda apresenta resistência a popularização. Desta forma, a utilização de câmeras de baixa resolução, aliada ao deslocamento inteligente do observador para otimizar a detecção em grandes áreas, poderá ser uma alternativa viável para aplicação em micro e mini geração, viabilizando aplicações como detecção em um conjunto de casas que podem dividir o ônus financeiro e se beneficiar de um monitoramento ostensivo e eficiente dos painéis fotovoltaicos instalados.

A simulação computacional do cenário através da aquisição de imagens, suas respectivas coordenadas e coleta de informações de saída, bem como instrução de deslocamento ou informação de falha, possibilita uma visão desacoplada do algoritmo e maximiza as possibilidades de aplicações práticas e a integração com outros estudos.

Como uma intenção de pesquisa, este artigo já denota o rumo da pesquisa a curto prazo, tendo por objetivo corroborar as hipóteses e determinar a viabilidade e aplicabilidade da técnica supracitada.

Sob uma ótica de longo prazo as possibilidades de trabalhos futuros perpassam por análises práticas da técnica sobre parques já existentes, comparativos sobre o uso desta técnica com relação as já existentes, refinamento do processo de detecção e aproximação através de variadas metaheurísticas.

4. REFERÊNCIAS

[1] G. Acciani, G. B. Simione, and S. Vergura. Thermographic analysis of photovoltaic panels. In *International Conference on Renewable Energies and Power Quality*. European Association for the Development of Renewable Energies, Environment and Power Quality, March 2010.

[2] P. Battalwar, J. Gokhale, and U. Bansod. Infrared thermography and ir camera. *International Journal of Research In Science and Engineering*, 1(3):9–14, May 2015.

[3] M. V. X. Dias. Geração distribuída no brasil: Oportunidades e barreiras. Master's thesis, Universidade Federal de Itajubá, 2005.

[4] C. Energética. Casa energética - termografia fotovoltaico. Disponível em <http://www.casaenergetica.it/servizi/analisi_termografica/termografia_fotovoltaico.html>. Accessado em 11-09-2016.

[5] S. B. Garcia, I. Zanesco, A. Moehlecke, and F. S. Febras. Análise por termografia de módulos fotovoltaicos com células solares com base n e diferentes malhas de metalização posterior. In *IV Congresso Brasileiro de Energia Solar e V Conferencia Latino-Americana da ISES*, September 2012.

[6] J. M. Pearce. Photovoltaics — a path to sustainable futures. *Elsevier, Future*, 34(7):663–674, September 2002.

[7] G. S. Spagnolo, P. D. Vecchio, G. Makary, D. Papalillo, and A. Martocchia. A review of ir thermography applied to pv systems. In *International Conference on Environment and Electrical Engineering*, May 2012.

Aplicação do Desenvolvimento Baseado em Domínio (DDD) na Criação de uma Ferramenta para Geração Automática de E-Commerce B2B E B2C

André Krzyk Taras
Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná
Ponta Grossa - Paraná
andrektaras@gmail.com

Bruno Vichinheski
Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná
Ponta Grossa - Paraná
brunovichinheski@gmail.com

Simone Nasser Matos
Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná
Ponta Grossa - Paraná
snasser@utfpr.edu.br

RESUMO

O comércio eletrônico (*e-commerce*) é utilizado por milhões de pessoas e pode ser dividido em categorias tais como: empresa para empresa (B2B), consumidores (B2C), governos (G2G), dentre outras. Existem diversas plataformas para criação de *e-commerce*, porém atendem apenas uma categoria. Este trabalho aplicou o *Domain-Drive Design* (DDD) para a criação dos módulos fundamentais que compõem uma plataforma *e-commerce* nas categorias B2B (*Business To Business*) e B2C (*Business To Customers*). O padrão de desenvolvimento DDD foi escolhido pela: necessidade de compreensão do domínio, alta manutenabilidade e capacidade de solucionar problemas intrínsecos nos sistemas de *e-commerce* atuais.

Palavras-Chave

B2B. B2C. DDD.

ABSTRACT

E-commerce is used by millions of people and can be divided by categories like Business To Business (B2B), Business To Customers (B2C) or Government To Government (G2G), among others. There are platforms used to create e-commerce, however it's only destined to one category. This paper applied Domain-Drive Design (DDD) to create fundamental modules that compose an e-commerce platform in B2C and B2B categories. The DDD pattern was chose to understand the complexity problem, maintainability and ability to solve specific problems in the current e-commerce.

Keywords

B2B. B2C. DDD.

1. INTRODUCÃO

Uma plataforma e-commerce é responsável por realizar transações de compra e venda de produtos e serviços por meio da Internet. Essa plataforma possibilita a interação entre compradores e vendedores de diferentes distâncias [1].

O sistema e-commerce possui um alto crescimento a cada ano; no Brasil o faturamento desse comércio eletrônico chegou a R$48,5 bilhões em 2015 e para 2016 a estimativa é que alcance R$56,8 bilhões [3].

A evolução da tecnologia e do comércio eletrônico, fez com que muitas empresas desenvolvessem plataformas e-commerce para seus clientes aumentarem a competitividade de venda de seus produtos. Tais plataformas possuem alto nível de complexidade, porém, muitas são desenvolvidas sem grande planejamento, impedindo o seu desenvolvimento, deixando os seus clientes com um sistema engessado e que não acompanha as tendências de mercado [2].

A utilização de plataformas e-commerce propicia a necessidade de um produto com baixo custo de manutenção, de modo que possa acompanhar as constantes novidades das tecnologias empregadas no seu desenvolvimento de forma eficaz e com isso proporcione um preço compatível.

Diversas metodologias de desenvolvimento podem ser utilizadas quando se deseja construir um sistema de forma padronizada e planejada. O modelo de desenvolvimento proposto por Erick Evans [4], Domain-Drive Design (DDD), foi o escolhido para o desenvolvimento deste trabalho pois possui como objetivo construir um software de fácil manutenção e é indicado para projetos que contemplam domínios complexos assim como o de e-commerce.

Evans [4] afirma que utilizar DDD é um desafio, pois requer um grande esforço da equipe de desenvolvimento e um conhecimento aprofundado dos objetivos deste software, porém ressalta vantagens de possuir um sistema de fácil manutenção e a capacidade de não se tornar obsoleto quando novas tecnologias surgirem devido a sua capacidade de adaptação.

Este trabalho aplicou o DDD na criação de uma plataforma capaz de gerar e-commerce B2B e B2C, utilizando o framework Symfony e algumas bibliotecas para a camada de infra-estrutura, proporcionando flexibilidade à aplicação.

2. APLICAÇÃO DO DDD

Para a criação da plataforma a solução foi aplicar o desenvolvimento baseado em domínio (DDD) que proporciona gerar um modelo adaptável e flexível [4].

O modelo de desenvolvimento DDD é composto pelas melhores práticas de desenvolvimento e os padrões de projeto mais

utilizados, sua abordagem de desenvolvimento divide-se em quatro etapas [4]:

- Trabalhar o modelo do domínio;
- Construir o modelo baseado em domínio;
- Refatorar o modelo;
- *Design* estratégico;

Ao adotar a estratégia de desenvolvimento baseado em domínio, a implementação irá se espelhar no modelo, portanto, a maioria dos esforços será em seu aperfeiçoamento.

2.1 Trabalhar o modelo do domínio

Nesta etapa do desenvolvimento foi realizado um estudo com o propósito de analisar todos os módulos e funcionalidades presentes nas categorias B2C e B2B, assim como mostra a Tabela 1. Após este levantamento foram definidos dois módulos para implementação, os módulos escolhidos foram "Cadastro e Login de Usuários" e "Catálogo de Produtos".

Tabela 1. Módulos e funcionalidades das Categorias

Módulo	Funcionalidade	Categoria
Cadastro e *Login* de Usuários	▪ Incluir Usuário ▪ Definir Tipo de Usuário ▪ Realizar Login	▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C
Catálogo de Produtos	▪ Incluir Produto ▪ Listar Produtos	▪ B2B e B2C ▪ B2B e B2C
Relacionamento entre Cliente e Empresa	▪ Contrato ▪ Atendimento ao Cliente ▪ Suporte ao Cliente	▪ B2B ▪ B2B e B2C ▪ B2B e B2C
Transação e Entrega	▪ Adicionar Produto ▪ Excluir Produto ▪ Alterar Quantidade do Produto ▪ Calcular o Frete ▪ Calcular o Valor do Produto ▪ Verificar Formas de Pagamento ▪ Verificar Cupom Desconto ▪ Confirmar Pagamento ▪ Finalizar Compra ▪ Cancelar Compra ▪ Verificar Carrinho ▪ Verificar *Login* ▪ Verificar Endereço para Entrega	▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C ▪ B2B e B2C

Esses módulos foram escolhidos e implementados com o objetivo de mostrar que ambos possuem uma funcionalidade que é exatamente igual (Cadastro e Login de Usuários) e outra que apresenta uma pequena variação (Catálogo de Produtos), pois na categoria B2B pode-se aplicar regras para mostrar os produtos aos usuários, por exemplo, se o usuário tiver efetuado o login no website, este possui alguns benefícios ao visualizar os produtos podendo exibir por: preços, quantidade de itens disponíveis, entre outros detalhes [6]. Na plataforma B2C a relação de produtos mostrados aos usuários é a mesma tanto para um que tenha efetuado o login quanto para outro que apenas está navegando no website.

2.2 Construir o modelo baseado em domínio

Para a construção do modelo de e-commerce foi necessário conhecer as principais funcionalidades das categorias B2B e B2C e isolá-las de forma a obter um modelo comum para ambas. Com isso o modelo e-commerce foi dividido em camadas conforme descreve o DDD: interface, aplicação, domínio e infra-estrutura. A camada de infra-estrutura é a primeira relatada devido a utilização de um framework para organização e controle de toda a estrutura do projeto.

2.2.1 Camada de infra-estrutura

Para o início da implementação optou-se pela escolha de um framework que pudesse atender os requisitos desta camada. Foram realizadas pesquisas e testes com os frameworks mais conhecidos para aplicações WEB (Zend, CakePHP, Symfony) e optou-se pela utilização do Symfony 3 [5], pois foi o framework com a melhor documentação e rescursos disponíveis dentre os testados.

Na camada de infraestrutura o framework Symfony objetiva realizar a organização do trabalho, bem como a interação e persistência dos dados entre as camadas. Para o processo de instalação do framework Symfony foi necessário fazer o download do código-fonte [5]. Depois disso, foi realizado o download do Composer (ferramenta que gerencia dependências em PHP e permite a declaração de bibliotecas no projeto, bem como a instalação e atualização das mesmas).

2.2.2 Camada de interface

A camada de interface do sistema e-commerce interage diretamente com o usuário através de formulários, listagens de produtos, entre outros. O framework Symfony possui um diretório padrão para todas as interfaces. As interfaces para os módulos de cadastro e login de usuários e catálogo de produtos foram implementadas em HTML e TWIG, este último compila templates em código PHP otimizado, isto faz com que os códigos se tornem mais rápidos, seguros e flexíveis.

2.2.3 Camada de aplicação

Uma importante ferramenta disponível no framework Symfony são os Controllers, responsáveis por receber requisições HTTP, criar e retornar uma resposta HTTP.

Uma pasta Controller foi criada para reunir os controladores necessários na aplicação, a função de um controlador é renderizar o conteúdo de uma página ou até mesmo redirecionar para uma página de erro caso a página não exista. Os controladores dos módulos de Cadastro e Login de Usuários e Catálogo de Produtos têm como objetivo gerenciar uma requisição específica de um usuário, como por exemplo, a confirmação de um formulário de cadastro ou a visualização da página de um produto.

2.2.4 Camada de domínio

Para modelagem do DDD devem ser seguidos sete blocos de construção que fundamentam e isolam o domínio do restante da aplicação: entidades, objetos de valores, serviços, módulos, agregados, fábricas e repositórios. Para o desenvolvimento dos módulos de Cadastro e Login de Usuários e Catálogo de Produtos foram utilizados apenas quatro blocos de construção: entidades, serviços, módulos e repositórios, não foram encontradas aplicações para os demais blocos nessa implementação.

O desenvolvimento da camada do domínio iniciou-se com a compreensão dos blocos de construção e a divisão correta dos elementos de uma aplicação e-commerce. Em uma pasta chamada

Entity são armazenadas as entidades do domínio que foram identificadas nos módulos implementados, cada classe contém uma assinatura (@Entity), que é identificada pelo Doctrine (biblioteca padrão do Symfony que gerencia o banco de dados) como uma tabela do banco de dados.

2.3 Refatorar o modelo

Durante o processo de refatoração nesse trabalho, não foram aplicadas técnicas de refatoração. O objetivo da refatoração foi obter um melhoramento dos modelos e aplicar padrões de projeto. A Tabela 2 mostra os módulos trabalhados (aqueles que foram identificados na primeira etapa do DDD), o respectivo número de refatorações e uma breve descrição sobre o que foi refatorado.

Tabela 2. Módulos Refatorados

Módulo	Refatorações	Descrição
Cadastro e *Login* de Usuários	2	▪ Exclusão de classes redundantes e tornando-as atributos ▪ Melhoramento do relacionamento entre as classes
Catálogo de Produtos	3	▪ Incluso a classe *Repositorio* para realizar consultas ▪ Incluso as classes de busca ▪ Inclusão de métodos diferenciados na categoria B2B
Relacionamento entre Cliente e Empresa	1	▪ Melhoramento da classe *Compra* para suprir ambas as categorias de *e-commerce*
Transação e Entrega	5	▪ Inclusão da classe *ItemCarrinho* ▪ Inclusão da classe *Rota*; mudança dos atributos da classe *Transportadora* ▪ Inclusão da classe *Compra* e *Venda* ▪ Generalização das classes *Transportadora* e *Filial* como classe *PessoaJuridica* ▪ Inclusão das classes: *ProdutosCompradosLoja, Movimentação e Saldo*

2.4 Design Estratégico

As fases inicias do DDD são importantes para o desenvolvimento do produto final, porém a parte mais relevante desta estratégia é quando o sistema torna-se complexo, por isso se deve dividir o sistema em contextos delimitados. Para isto, utiliza-se técnicas para a manipulação e compreensão de grandes modelos [4].

Como esta etapa possui um foco no final do desenvolvimento e também quando existe algum tipo de integração com outros sistemas, ela não foi utilizada nesta pesquisa.

3. RESULTADOS

A página inicial desenvolvida para a ferramenta apresenta as opções de e-commerce disponíveis pela aplicação, ou seja, as categorias B2C e B2B, assim como mostra a Figura 1.

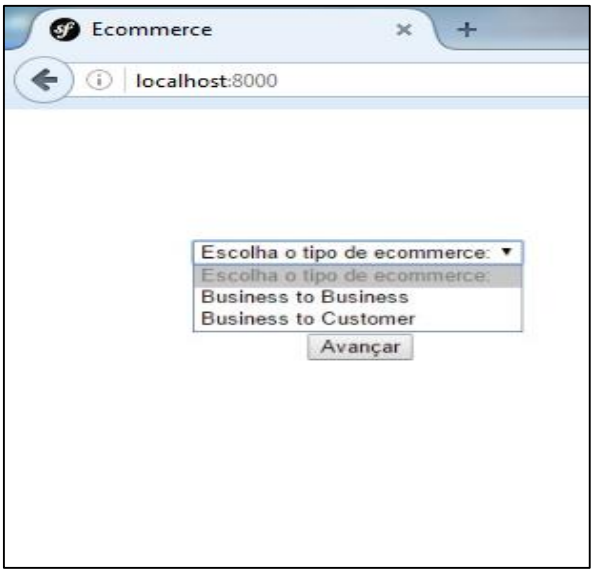

Figura 1. Página inicial da ferramenta

A Figura 1 apresenta apenas o protótipo do que seria o produto final, pois este se encontra em fase de desenvolvimento, apenas a modelagem desta aplicação base foi realizada.

Após a escolha da categoria de e-commerce o usuário escolhe quais módulos a plataforma terá, os módulos fundamentais que já estão implementados obrigatoriamente deverão fazer parte da aplicação, desta forma este modelo contempla módulos extras que poderão ser desenvolvidos e optativamente utilizados pelos usuários. A etapa de isolamento do domínio exigiu grande conhecimento do sistema e demandou grandes esforços dos envolvidos para obter o conhecimento necessário para o desenvolvimento, entretanto, este processo tornou o modelo mais robusto, claro e flexível, possibilitando a adição de funcionalidades extras sem que muitas alterações sejam necessárias.

Como foram encontradas diferenças para o catálogo de produtos entre as categorias B2C e B2B, foram criadas duas implementações diferentes para cada categoria e-commerce. Para o catálogo de produtos B2C foi implementado um catálogo em que o usuário pode visualizar os produtos com informações de preço disponíveis, independente se ele é um usuário que está logado no sistema ou navega anônimamente, como mostra a Figura 2.

Na implementação do catálogo B2B, um usuário tem acesso limitado as informações dos produtos e precisa fazer parte de um grupo de usuários específico para ter acesso aos preços e outras informações importantes, geralmente este grupo de usuários ganha este acesso através de concessões administrativas. A regra de negócio irá depender da empresa que vai gerenciar o e-commerce.

Para o catálogo B2B desta aplicação é verificado se o usuário está logado e se ele pertence ao grupo privilegiado, caso contrário ele não terá acesso as informações referentes ao preço do produto.

A Figura 2 ilustra as diferenças encontradas no catálogo de produtos das categorias durante a listagem dos resultados de uma busca.

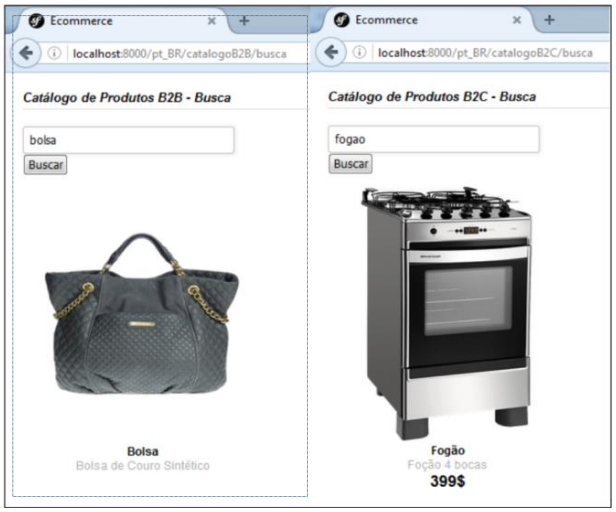

Figura 2. Diferenças do catálogo B2B e B2C na Busca de Produtos

É possível notar que a busca B2B (destacada na imagem) não possui nenhuma informação referente ao preço do produto, enquanto que o catálogo mostrado para a busca B2C apresenta o preço normalmente.

4. CONCLUSÃO

A modelagem e implementação dos módulos login e catálogo de produtos nas categorias B2B e B2C foram realizadas seguindo a metodologia de desenvolvimento baseado em domínio. Primeiramente foram realizados modelos iniciais, em seguida utilizou-se os blocos de construção para isolar o domínio do restante da aplicação, a partir deste ponto prevaleceram as refatorações. A etapa de *design* estratégico do DDD não foi utilizada.

O modelo da ferramenta proposta difere das plataformas de geração de *e-commerce* porque contempla diversas categorias, diferente das outras que atendem apenas uma. O desenvolvimento baseado em domínio proporcionou flexibilidade ao modelo porque funcionalidades podem ser acrescentadas sem que haja grandes mudanças na estrutura do modelo.

Embora a implementação do modelo não esteja completa, pôde-se constar que a utilização do DDD para a criação do modelo inicial foi fundamental, a utilização das etapas e dos blocos de construção possibilitaram um entendimento graduado do domínio do sistema pois focou-se em compreender as funcionalidades do domínio, minimizando os esforços para o desenvolvimento de interfaces e funções menos importantes do sistema. O uso do framework foi importante para o processo de desenvolvimento, toda a camada de infra-estrutura do DDD foi suprida pelo *Symfony*.

5. REFERÊNCIAS

[1] ASCENSÃO, Carlos P. O que é e-Commerce?. 2014. Disponível em: <http://www.gestordeconteudos.com/tabid/3850/Default.aspx>. Acesso em: 24 nov. 2015.

[2] CHAUSSARD, Cristiano. Três erros cometidos por plataformas de e-commerce. 2015. Disponível em: <http://www.guiadeecommerce.com.br/tres-erros-cometidos-por-plataformas-de-ecommerce/>. Acesso em: 09 set. 2016.

[3] DOTSTORE (Mogi das Cruzes). E-Commerce 2016: Projeção de R$56,8 bilhões em faturamento. 2016. Disponível em: <http://site.dotstore.com.br/loja-virtual/e-commerce-2016-projecao-de-r568-bilhoes-em-faturamento/>. Acesso em: 12 maio 2016.

[4] EVANS, Eric. Domain-Drive Design: Tackling Complexity in the Heart of Software. Prentice Hall, 2003.

[5] FABIEN POTENCIER (California) (Org.). SYMFONY. Página do framework. Disponível em: <http://symfony.com/>. Acesso em: 15 mar. 2016.

[6] SILVA, Edson dos Santos. Branding para empresas B2B. Fundação Instituto de Administração, Pós Graduação em Gestão Estratégica de Marcas. São Paulo: Provar, 2013. Disponível em: <http://www.cidademarketing.com.br/2009/sysfotos/tesesmono/tesem_40b478b412cce7278d88fb18869ca034.pdf>. Acesso em: 17 out. 2015.

Análise de modelos de confiança e reputação em sistemas baseados em agentes para alocação de vagas em um estacionamento inteligente

Angelo Bittencourt Marini Filho
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa
Caixa Postal 84.016-210
Ponta Grossa – PR– Brasil
angelobittencourt@gmail.com

Gleifer Vaz Alves
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa
Caixa Postal 84.016-210
Ponta Grossa – PR– Brasil
gleifervaz@gmail.com

RESUMO
Encontrar vagas disponíveis para motoristas estacionarem seus veículos pode ser uma tarefa árdua e desgastante. Outros motoristas que disputam as mesmas vagas podem agir de maneira desleal e não cumprir as normas definidas pelos administradores. Buscando encontrar soluções para tal problema o presente trabalho propõe uma análise de modelos de confiança e reputação em Sistemas Multiagentes, aplicados ao cenário de um estacionamento inteligente, conseguiriam classificar os motoristas e alocar as vagas de maneira justa.

Palavras-chave
Smart Parking; Confiança; Reputação; Sistemas Multiagentes.

ABSTRACT
Finding available spots where drivers will be able to parking their car could be a hard task. Others drivers who are looking for the same spot could act unfairly and don't respect standards set by administrators. Looking for solutions for this issue, this paper provides an analysis of trust and reputation models for Multiagent Systems, applied to smart parking could be able to rank and allocate drivers fairly.

Keywords
Smart Parking; Trust; Reputation; Multiagent System.

1. INTRODUÇÃO
Cidades inteligentes são projetadas por visionários utilizando tecnologias de informação e comunicação com o objetivo de melhorar a qualidade de vida, reduzir desperdícios e melhorar as condições econômicas dos cidadãos das grades cidades, dado que a maioria da população deve viver em ambientes urbanos dentro das próximas décadas [1].

Um dos principais setores no qual cidades inteligentes buscam solucionar problemas é o trânsito. Em [2] é destacado que o Brasil perde em torno de R$300 bilhões por ano com o trânsito das grandes cidades. Nesta mesma publicação, é informado que pessoas que procuram vagas para estacionar causam 30% do trânsito da cidade de São Paulo.

WPCCG'16, Setembro 28, 2016, Ponta Grossa, Paraná, Brasil.

Neste cenário surgem os estacionamentos inteligentes, ou *smart parking*, que buscam por meio do uso de tecnologia facilitar o estacionamento de veículos pelos motoristas, os quais utilizando estacionamentos convencionais podem desperdiçar tempo precioso em lentas filas na busca por vagas.

Uma abordagem possível para modelagem de *smart parkings* se dá por meio da utilização de Sistemas Multiagentes (SMA), sistemas esses compostos por agentes autônomos, os quais interagem entre si e com o ambiente dinâmico no qual estão inseridos para serem capazes de alcançar seus próprios objetivos [3].

O Grupo de Pesquisa em Agentes de Software da UTFPR-PG (GPAS) desenvolve um projeto de pesquisa denominado *MultiAgent Parking System* (MAPS) o qual utiliza SMA aplicado para um estacionamento inteligente. O projeto MAPS tem como objetivo " " [4].

O desenvolvimento dos SMA teve grande contribuição para o aumento do interesse em estudos de confiança e reputação dentro da ciência da computação, gerando a criação de modelos de confiança e reputação que buscam sistematizar a avaliação de agentes para auxiliar em decisões e buscar parceiros confiáveis para interagir [5].

Atualmente no projeto MAPS é utilizada a noção de grau de confiança pela quantidade de vezes que um motorista utiliza o estacionamento (quanto mais o agente faz uso do ambiente, maior será o grau de confiança). Quando dois (ou mais) agentes disputam uma vaga, simplesmente aquele com maior grau de confiança obtém a vaga.

O objetivo desse estudo é analisar modelos de confiança e reputação para que futuramente seja incorporado ao projeto MAPS novos modelos que possibilitem que a experiência dos agentes motoristas no estacionamento seja mais satisfatória e que a alocação das vagas disponíveis seja feita de maneira justa.

Neste trabalho será apresentado na sequência o SMA desenvolvido para o projeto MAPS. A terceira seção mostra os modelos de confiança e reputação e suas características. Em seguida na seção quatro será apresentada comparação entre os modelos expostos na seção 3. E por fim o trabalho apresentará considerações finais a respeito da analise dos modelos, e a identificação da possibilidade da aplicação dos modelos no SMA desenvolvido.

2. PROJETO MAPS
O SMA desenvolvido para o projeto MAPS possui dois tipos de agentes: Os *drivers* que interagem e utilizam o sistema e o agente *manager* que gerencia o sistema. O objetivo do sistema é alocar vagas para os *drivers*, alocação essa feita pelo agente *manager*.

Para alocar a vaga, o agente *manager* deve receber uma solicitação de requisição de vaga e então se existirem mais requisições por vagas do que vagas disponíveis, o *manager* por meio da análise do grau de confiança (ou *degree of trust*) dos *drivers* decidirá para qual dos agentes *drivers* será disponibilizado o recurso, como é ilustrado na Figura 1:

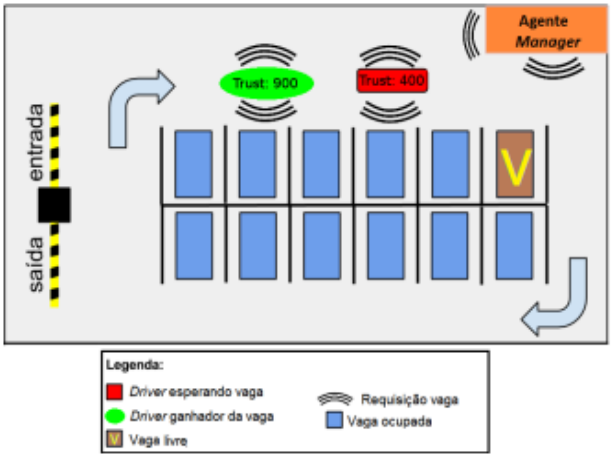

Figura 1. Ilustração Alocação de Vagas MAPS
Fonte: de Castro et al. , 2016 [6]

Drivers novos iniciam com o menor valor de confiança possível, porém para que esses novos agentes, ou agentes com baixo valor de confiança não fiquem por tempo indeterminado esperando, a partir de certo tempo de espera esse *driver* recebe prioridade para receber o recurso.

No momento o projeto MAPS é desenvolvido pensando em estacionamentos privados e fechados, como estacionamentos de shoppings e hospitais. A partir daí serão analisados os modelos de confiança buscando aprimorar o método usado pelo *manager* para alocação de vagas no estacionamento.

3. MODELOS DE CONFIANÇA E REPUTAÇÃO

Na literatura são encontradas diferentes definições de confiança e reputação no contexto de agentes. Segundo Gambetta [7] confiança é a probabilidade subjetiva de que um agente irá executar a tarefa que foi destinada a ele. Em Sabater e Sierra [5] a confiança normalmente é denotada por um valor numérico que indica quão confiável um agente é, e esse valor deve ser usado por outros agentes para decidirem se devem ou não interagir com o agente em questão.

Conforme destacado por Granatyr [8], reputação pode ser definida como uma coleção de opiniões sobre um agente, dadas por outros agentes, ou a expectativa de comportamento deste agente baseada nas suas interações anteriores.

Nesta seção serão apresentados resumidamente cinco modelos de confiança e reputação, SPORAS [9], ReGreT [10], Marsh [11], Travos [12] e FIRE [9], respectivamente.

3.1 Modelo SPORAS

Segundo Huynh et. al. [9], SPORAS é um modelo de reputação centralizado, onde os valores de confiança de todos agentes são guardados por um gerenciador central, onde são mais valorizadas as avaliações recentes e descartadas avaliações antigas.

O algoritmo para a aplicação deste modelo deve comtemplar os seguintes princípios:

1. Usuários novos entram no sistema com o valor mínimo de reputação, que cresce durante sua atividade no sistema.
2. O valor de reputação de um agente antigo nunca é menor do que de um iniciante.
3. Após cada interação o valor de reputação é atualizado.
4. Para agentes com valor de reputação muito alto, as novas avaliações não alteram muito o seu valor a cada interação.
5. Avaliações mais antigas são descartadas, sendo consideradas as avaliações de interações mais recentes.

3.2 Modelo ReGreT

O modelo ReGreT, proposto por Sabater e Sierra [10], caracteriza-se como um modelo de confiança e de reputação, onde três diferentes fontes de informação são levadas em consideração: interações diretas, informações de terceiros e estrutura social. Além disso, o ReGreT tem dois módulos, alimentados pelas informações adquiridas pelo agente, um módulo de confiança e outro para reputação.

O módulo Confiança Direta lida com experiências diretas e como essas experiências podem contribuir para a confiança de terceiros. Junto com o módulo de reputação são capazes de calcular a confiança.

Já o módulo de reputação é dividido em três tipos específicos de reputação, dependendo da fonte da informação utilizada para o cálculo, sendo eles: reputação por testemunha, reputação por vizinhança e reputação do sistema.

Estes módulos trabalham em conjunto para oferecer um modelo de confiança completa com base no conhecimento direto e na reputação.

Neste modelo a confiança é avaliada de maneira totalmente descentralizada onde cada agente por si só é capaz de mensurar o valor de confiança dos demais.

3.3 Modelo Marsh

O modelo Marsh é um dos primeiros modelos de confiança, proposto por Marsh em 1994 [11], onde são consideradas apenas interações diretas entre agentes para calcular a confiança.

Marsh diferencia três tipos de confiança:

• Confiança Básica: é a disposição de um agente confiar em outro, calculada a partir das interações do agente.

• Confiança Geral: A confiança que um agente tem em outro sem levar em conta qualquer situação específica.

• Confiança Situacional: Valor da confiança que um agente tem em relação a outro, baseado em uma situação específica.

3.4 Modelo Travos

Segundo Da Silva [12], o modelo Travos representa a confiança dos agentes por informações recebidas por testemunhas e por meio de interações diretas. Para calcular o nível de confiança de um agente, é feito um cálculo da probabilidade do agente cumprir a tarefa delegada, cálculo realizado aleatoriamente dentro da população de interações.

Quando não é possível determinar a confiança do agente por meio das interações diretas, por conta do intervalo de confiança não atingir um valor mínimo necessário, os agentes devem utilizar as interações indiretas. Neste caso, a confiança é obtida por testemunhas.

3.5 Modelo FIRE

O modelo FIRE proposto por Hyunh et. al. em [9], é um modelo de confiança e reputação integrado, com uma arquitetura de tomada de decisão distribuída entre os agentes. Ele incorpora quatro fontes de informação:

• Confiança por interação: mede a confiança de maneira direta.

• Confiança por papel: Baseado no papel que o agente desempenha o valor de confiança inicial do agente é a média dos agentes com o mesmo papel no sistema.

• Confiança por testemunho: Calcula a reputação por meio do testemunho de outros agentes.

• Confiança certificada: Calculada com base nas referências fornecidas pelo próprio agente avaliado.

Essa variedade de fontes (do modelo FIRE) torna-se importante, visto que, em várias situações nem todas estarão prontamente disponíveis, além de permitir aos agentes combiná-las para lidar com as incertezas do ambiente.

4. COMPARAÇÃO ENTRE OS MODELOS

Os modelos apresentados na seção anterior possuem similaridades, bem como diferenças. Para poder delinear uma comparação entre os modelos aqui são destacadas a natureza da reputação, tipo da reputação, valor da reputação e distribuição da reputação desses modelos.

A natureza da reputação permite identificar a que tipo de entidade a reputação se aplica, e os modelos apresentados são todos semelhantes nesse quesito, onde a reputação se aplica a cada agente individualmente.

O tipo da reputação classifica a origem da informação utilizada na formação da reputação. O modelo SPORAS não distingue a reputação pelo tipo. Todos os outros utilizam a interação direta entre agentes para gerar seu valor de reputação. Além da interação direta o modelo Travos e o ReGreT também utilizam testemunhas para se informar e gerar a reputação do agente. O modelo ReGret também usa o papel e propriedades gerais do agente como fonte de informação. O mais versátil entre os modelos é o FIRE, que além das fontes de informação relatadas anteriormente ainda analisa informações certificadas vindas do próprio agente avaliado.

Para valorizar a reputação dos agentes os modelos utilizam dois tipos de valor, consolidado e detalhado. O valor consolidado valoriza os agentes por meio de pontuação enquanto o detalhado traz os atributos do agente para este valor. Os modelos SPORAS, Marsh e Travos utilizam valor consolidado. Já os modelos FIRE e ReGreT utilizam valor consolidado e valor detalhado para valorar a confiança dos agentes.

A distribuição da reputação identifica se a reputação é formada por uma entidade centralizada da informação, ou se esta pode ser realizada por qualquer agente de maneira descentralizada. Dos modelos apresentados apenas o SPORAS é centralizado, enquanto todos os demais são descentralizados.

A tabela 1 ilustra a comparação entre os modelos estudados conforme as seguintes características:

- Tipo da reputação (TR)
  - ND- Não Distingue
  - ID – Interação Direta
  - IT – Informação por Testemunho
  - P – Papel no sistema
  - IC – Informação Certificada
- Distribuição da Reputação (DR)
  - C – Centralizada
  - D – Descentralizada
- Valor de Reputação (VR)
  - C – Consolidado
  - D – Detalhado
- Tipo do modelo (TM)
  - C- Confiança
  - R – Reputação

	TR	DR	VR	TM
Sporas	ND	C	C	R
Marsh	ID	D	C	C
ReGret	ID, IT e P	D	C e D	R e C
FIRE	ID, IT, P e IC	D	C e D	R e C
Travos	ID e IT	D	C	C

Tabela 1. Comparação Modelos
Fonte: Autoria Própria

5. CONSIDERAÇÕES FINAIS

Analisando os modelos de confiança e reputação descritos na seção anterior, essa seção busca identificar a possibilidade de aplicação desses modelos no projeto MAPS.

O modelo ReGreT não pode ser aplicado pois seu método de classificação da reputação é totalmente descentralizado, e como é utilizado um agente *manager* centralizador para alocar as vagas, esse modelo não aplica-se ao MAPS.

O modelo Travos não seria aplicável ao projeto MAPS, pois nele os cálculos do grau de confiança são feitos de forma subjetiva, o que não seria adequado para solucionar uma disputa de recurso (no caso, vaga do estacionamento) entre dois agentes *drivers*.

O modelo Marsh, utilizando o cálculo da confiança situacional poderia ser utilizado para fazer as médias das avaliações das interações e alocar a vaga para aquele agente *driver* com maior média. Porém se um *driver* utiliza o estacionamento pela primeira vez, e ganha uma nota elevada na sua primeira interação, ele passará na frente de outros agentes que possuem uma média razoável que já estão utilizando o sistema por mais tempo. Isso também pode fazer com que um *driver* com uma avaliação baixa, deixe o sistema e retorne depois subindo seu grau de confiança facilmente.

O modelo de reputação SPORAS entre os apresentados é o que melhor aplica-se a versão atual do MAPS, pois o agente *manager* age como centralizador, responsável por avaliar o agente *driver* a cada interação. Outra vantagem deste modelo, em relação a sua aplicação ao MAPS está nas avaliações mais recentes serem usadas, enquanto as antigas são descartadas. A única característica diferenciada é que o SPORAS é um modelo de reputação, ao passo que o MAPS utiliza confiança.

O modelo FIRE pode ser considerado o mais robusto dos cinco modelos apresentados anteriormente, com várias abordagens para fazer a avaliação do grau de confiança. Considerando uma possível extensão do projeto MAPS, no caso, uma versão distribuída onde os próprios *drivers* seriam responsáveis pela negociação das vagas. O modelo FIRE poderia ser aplicado, utilizando principalmente a confiança por

interação entre os agentes *drivers*. Porém, como a arquitetura atual do MAPS é centralizada, o modelo FIRE (por completo) não se aplica ao projeto MAPS.

Por fim, é possível dizer que a alternativa (que parece) mais adequada ao MAPS, seria a arquitetura do modelo SPORAS. Porém, ao invés de usar reputação, seria necessário usar o grau de confiança dos agentes.

Como trabalhos futuros é possível destacar: (i) implementação de um modelo de confiança com novas fórmulas para o cálculo do grau de confiança; (ii) testes e simulações do novo modelo de confiança no projeto MAPS.

6. REFERÊNCIAS

[1] C. Stimell. Building Smart Cities. Analytics, ICT, and Design Thinking. 1st. ed. [S.I] CRC Press is an imprint of Taylor & Francis Group, an Informa business, 2016. ISBN 13: 978-1-4987-0277-5

[2] J. Da Silva. Aplicativo de vagas pode reduzir trânsito nas cidades e facilitar sua vida, Disponível: http://www.infomoney.com.br/negocios/startups/noticia/3853310/aplicativo-vagas-pode-reduzir-transito-nas-cidades-facilitar-sua-vida, fevereiro/2015.

[3] M. Wooldridge. An Introduction to MultiAgent Systems. Wiley Publishing, 2nd edition, 2009.

[4] W. Gonçalves; G. Alves. Smart parking: mecanismo de leilão de vagas de estacionamento usando reputação entre agentes. In: Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – WESAAC. [S.l.: s.n.], 2015.

[5] J. Sabater; C. Sierra. Regret: Review on computational trust and reputation models. Bellaterra, Barcelona, Spain, IIIA – CSIC, 2003

[6] L. Castro; G. Alves; A. Pinz. Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking. In: Anais do X Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – WESAAC. Maceió, 2016.

[7] D. Gambetta. Can We trust Trust? In Trust: Making and Breaking Cooperative Relations. Basil Black-well, New York, 1988.

[8] J. Granatyr; V. Botelho; O. Lessing; E. Scalabrin; J. Barthes; F. Enembreck. Trust and Reputation Models for Multiagent Systems. ACM Comput. Surv. 48, 2, Article 27, 2015.

[9] T. Huynh; N. Jennings; N. Shadbolt. An integrated trust and reputation model for open multi-agent systems. Springer Science, LLC 2006.

[10] J. Sabater; C. Sierra. Regret: Reputation in gregarious societies. In: Proceedings of the Fifth International Conference on Autonomous Agents. New York, NY, USA: ACM, 2001.

[11] S. Marsh. Formalising trust as a computation concept – University of Stirling, 1994.

[12] V. Da Silva. Um modelo de confiança certificado baseado em assinatura digital aplicado a sistemas multiagentes.Dissertação (mestrado) – Pontificia Universidade Catolica do Paráná, Curitiba, 2009.

Coloração de arestas distinta nos vértices adjacentes em potências de caminho

Mayara Midori Omai*
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato km 04
Ponta Grossa, Paraná
omai@alunos.utfpr.edu.br

Sheila Morais de Almeida
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato km 04
Ponta Grossa, Paraná
sheilaalmeida@utfpr.edu.br

Diana Sasaki Nobrega
Universidade do Estado do Rio de Janeiro
R. São Francisco Xavier, 524
Maracanã, Rio de Janeiro
diana.sasaki@ime.uerj.br

RESUMO

O Problema da Coloração de Arestas Distinta nos Vértices Adjacentes consiste em, dado um grafo, utilizar o menor número de cores possível para colorir suas arestas de forma que arestas incidentes no mesmo vértice tenham cores distintas e o conjunto de cores incidentes em cada vértice seja diferente dos conjuntos de cores dos seus vizinhos. Nesse trabalho usamos a técnica *pullback* para resolver o Problema da Coloração de Arestas Distinta nos Vértices Adjacentes quando restrito aos grafos potências de caminho P_n^k, com $n \geq 3k$.

Palavras-chave

coloração de arestas distinta nos vértices adjacentes; *pullback*; potência de caminho

ABSTRACT

The Adjacent Vertex Distinguishing Edge-Coloring Problem consists in assigning the minimum number of colors to the edges of a given graph, such that adjacent edges have distinct colors and the color set of each vertex is different of the color sets of its neighbours. In this work we use the pullback technique to solve the Adjacent Vertex Distinguishing Edge-Coloring Problem for the power of a path, P_n^k when $n \geq 3k$.

Keywords

adjacent vertex distinguishing edge-coloring; pullback; power of a path

*Bolsista da Fundação Araucária

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

1. INTRODUÇÃO

Nesse trabalho, denota-se por $G = (V, E)$ um grafo conexo e simples com conjunto de vértices V e conjunto de arestas E.

Em um grafo, dois elementos são adjacentes se: i) são um par de vértices que constituem uma aresta do grafo, ii) são duas arestas que contêm um mesmo vértice, ou iii) se são uma aresta e um dos vértices que a compõem. Classicamente, os problemas de coloração em grafos consistem em determinar o número mínimo de cores necessárias para colorir os elementos de um grafo de forma que elementos adjacentes tenham cores distintas.

O Problema da Coloração de Vértices, por exemplo, consiste em colorir os vértices de um grafo de forma que vértices adjacentes tenham cores distintas utilizando para tal o menor número de cores possível. O Problema da Coloração de Arestas consiste em colorir, com o menor número de cores possível, as arestas de um grafo de forma que arestas adjacentes tenham cores distintas. Outro problema de coloração em grafos bastante conhecido é o Problema da Coloração Total, que consiste em colorir tanto os vértices quanto as arestas de um grafo de forma que quaisquer dois elementos adjacentes tenham cores distintas. Tais problemas possuem diversas aplicações, dentre elas estão os problemas de alocação de recursos, escalonamento de tarefas, projeto de redes elétricas, projetos de circuitos e particionamento de conjuntos [3, 7].

Recentemente, outros problemas de coloração surgiram, impondo mais restrições à coloração de grafos. Dentre as restrições que podem ser impostas, algumas consideram o conjunto de cores de cada vértice, composto pelas cores das arestas que incidem no mesmo. Uma coloração de arestas é distinta nos vértices se o conjunto de cores em cada vértice do grafo é único [2].

Ainda considerando restrições sobre o conjunto de cores dos vértices de um grafo, pode-se relaxar a restrição de que tais conjuntos sejam únicos, exigindo-se apenas que os conjuntos de cores de vértices adjacentes sejam distintos. Tal coloração de arestas é chamada de coloração de arestas distinta nos vértices adjacentes (coloração de arestas DVA). Um exemplo de coloração de arestas DVA pode ser visto na Figura 1. Note que a coloração de arestas do exemplo não é uma coloração de arestas distinta nos vértices, já que existem dois vértices com conjuntos de cores $\{0, 2, 3\}$, entretanto, é uma coloração de arestas distinta nos vértices adjacentes, já que os vértices com mesmo conjunto de cores

não são adjacentes.

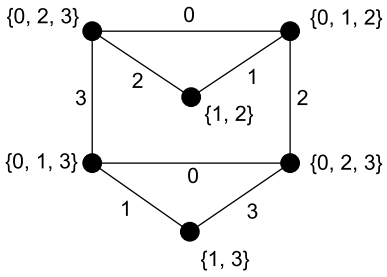

Figura 1: Exemplo de coloração de arestas DVA.

Zhang *et al.* [12] formalizaram e apresentaram os primeiros resultados sobre o Problema da Coloração de Arestas Distinta nos Vértices Adjacentes, que consiste em determinar o menor número de cores necessário para se obter uma coloração de arestas distinta nos vértices adjacentes, chamado de índice cromático distinto nos vértices adjacentes e denotado por $\chi'_a(G)$.

Dentre as classes de grafos nas quais o Problema da Coloração de Arestas DVA está em aberto encontra-se a classe das potências de caminho. Um caminho com n vértices, P_n, é um grafo com conjunto de vértices $V = \{v_0, v_1, \ldots, v_{n-1}\}$ e conjunto de arestas $E = \{(v_i, v_{i+1}), 0 \leq i < n-1\}$. Em um grafo G, a distância entre quaisquer dois vértices u e v é o número de arestas do menor caminho que conecta u a v em G. A k-ésima potência de um caminho com n vértices, denotada por P_n^k, é construída incluindo no grafo caminho, P_n, todas as arestas que conectam pares de vértices que estejam à distância até k no P_n.

Esse artigo apresenta uma solução para o Problema da Coloração de Arestas DVA em potências de caminho P_n^k com $n \geq 3k$.

2. PRELIMINARES

Segundo Zhang *et al* [12] se um grafo G desconexo é composto por n componentes $G_1, G_2, \ldots, G_n$ com pelo menos 3 vértices cada uma, então $\chi'_a(G) = \max\{\chi'_a(G_1), \chi'_a(G_2), \ldots, \chi'_a(G_n)\}$. Portanto, basta considerar os grafos conexos. Note que se um grafo G conexo é composto por um ou dois vértices, o mesmo não possui uma coloração de arestas DVA.

Os lemas a seguir são importantes para a compreensão desse trabalho.

Lema 2.1. (Chartrand e Zhang [3]) Seja K_n um grafo completo com $n \geq 3$. Se n é ímpar, então $\chi'_a(K_n) = n$ e, caso contrário, $\chi'_a(K_n) = n + 1$.

Observe que toda potência de caminho P_n^k com $n \leq k+1$ é isomorfa a um grafo completo e, portanto, para esses casos o Problema da Coloração de Arestas DVA está resolvido, pelo Lema 2.1.

O grau de um vértice v, denotado por $d(v)$, é o número de vizinhos de v no grafo. O grau máximo de G é o maior dos graus dos vértices de G, denotado por $\Delta(G)$. Um vértice v do grafo G é $\Delta(G)$-vértice se $d(v) = \Delta(G)$. O Lema 2.2 apresenta um limite inferior para $\chi'_a(G)$ quando G tem pelo menos dois vértices adjacentes com grau $\Delta(G)$.

Lema 2.2. (Chartrand e Zhang [3]) Se um grafo G possui pelo menos dois vértices de grau máximo adjacentes, então $\chi'_a(G) \geq \Delta(G) + 1$. Caso contrário, $\chi'_a(G) = \Delta(G)$.

Um grafo é indiferença se, e somente se, seus vértices podem ser linearmente ordenados de forma que vértices adjacentes sejam consecutivos. É importante ressaltar que as potências de caminho são subclasse dos grafos indiferença.

Sobre o Problema da Coloração de Arestas em grafos indiferença sabe-se que quando $\Delta(G)$ é ímpar este problema pode ser resolvido utilizando-se $\Delta(G)$ cores [4]. Ainda não é conhecida uma solução para o Problema de Coloração de Arestas em grafos indiferença quando $\Delta(G)$ é par, entretanto, existem soluções parciais. O problema está resolvido, por exemplo, para os grafos split-indiferença [8], indiferença reduzidos [6] e potências de caminho. Quando a potência de caminho P_n^k possui um vértice com grau $n-1$, o problema foi resolvido por Plantholt [9]. Quando $\Delta(P_n^k) < n-1$, basta atribuir as cores $2i-1$ e $2i$ para os caminhos induzidos pelas arestas entre vértices que estão a distância i no P_n ou somente a cor $2i-1$ se tais arestas induzem um emparelhamento. Em relação ao Problema da Coloração Total em grafos indiferença, sabe-se que a conhecida Conjectura da Coloração Total é verdadeira para esses grafos [5]. Segundo a Conjectura da Coloração Total, qualquer grafo simples G com grau máximo $\Delta(G)$ tem uma coloração total com no máximo $\Delta(G) + 2$ cores [1, 11]. Além disso, Figueiredo *et al.* [5] provaram que, quando $\Delta(G)$ é par, existe uma coloração total do grafo indiferença utilizando $\Delta(G) + 1$ cores, uma solução ótima.

Tanto a solução para o Problema da Coloração de Arestas em grafos indiferença com $\Delta(G)$ ímpar, quanto as soluções conhecidas para o Problema da Coloração Total em grafos indiferença, foram obtidas utilizando uma técnica chamada *pullback*. A técnica *pullback* aplicada sobre um grafo G consiste na apropriação das cores das arestas e vértices (se coloridos) de um grafo completo para se obter uma coloração para G. Na Seção 3, apresentamos o índice cromático distinto nos vértices adjacentes do grafo P_n^k, $n \geq 3k$, usando a técnica *pullback*.

Concluimos essa seção apresentando brevemente como o *pullback* é utilizado para a coloração de arestas dos grafos indiferença com grau máximo ímpar.

2.1 Pullback para coloração de arestas

Dado um grafo indiferença G com $\Delta(G)$ ímpar, Figueiredo *et al.* [4] apresentam uma coloração de arestas ótima para G com $\Delta(G)$ cores. A técnica utilizada consiste em se apropriar da coloração de arestas de um grafo completo $K_{\Delta(G)+1}$ para colorir as arestas de G.

Um grafo completo K_n com n par pode ter suas arestas coloridas com $n-1$ cores com o seguinte procedimento. Rotule os vértices do K_n, v_0, v_1, ... , v_{n-2}, d. Atribua a cor $i + j \pmod{n-1}$, para a aresta (v_i, v_j) quando v_i e v_j são distintos de d. Atribua a cor $2i \pmod{n-1}$ para a aresta (v_i, d), $0 \leq i \leq n-2$.

Quando um grafo completo K_n tem n ímpar, não é possível colorir suas arestas com $n-1$ cores. Nesse caso, são necessárias pelo menos n cores e uma coloração ótima pode ser obtida removendo-se o vértice d de um grafo K_{n+1} previamente colorido como descrito. Note que, como todas as arestas incidentes no vértice d têm cores distintas, então na coloração de arestas do K_n com n ímpar o conjunto de cores de cada vértice também é distinto.

Considere um grafo indiferença G com grau máximo ímpar e uma coloração do $K_{\Delta(G)+1}$. A apropriação de cores do $K_{\Delta(G)+1}$ é feita como segue. Por definição, os vértices de G podem ser linearmente ordenados de forma que vértices adjacentes são consecutivos. Rotule os vértices de G, v_0, v_1, ..., $v_{\Delta(G)-1}$, d, v_0, v_1, ..., $v_{\Delta(G)-1}$, d. Pinte cada aresta (v_i, v_j) de G, com a cor da aresta (v_i, v_j) do $K_{\Delta(G)+1}$. Como $K_{\Delta(G)+1}$ é um grafo completo e G foi rotulado somente com os rótulos utilizados na coloração de arestas do $K_{\Delta(G)+1}$, então é possível pintar todas as arestas de G. Note que vértices com o mesmo rótulo estão a uma distância $\Delta(G)+2$ na ordem dos vértices. Então, nenhum vértice v é adjacente a dois vértices distintos que tenham rótulos iguais. Caso contrário, v teria grau $\Delta(G)+1$, o que seria um absurdo. Como nenhum vértice é adjacente a dois outros que tenham rótulos iguais, as arestas incidentes em um mesmo vértice tem cores distintas. Já que não é possível fazer uma coloração de arestas de um grafo G com menos que $\Delta(G)$ cores, essa coloração é ótima.

3. RESULTADOS

Essa seção apresenta a solução do Problema da Coloração de Arestas DVA para a k-ésima potência de caminho com $n \geq 3k$.

A ideia é considerar somente potências de caminho com $\Delta(P_n^k) = 2k$, que tenham pelo menos dois vértices adjacentes de grau máximo. Pelo Lema 2.2, tais grafos têm $\chi'_a(P_n^k) \geq 2k+1$. Vamos utilizar a técnica *pullback* se apropriando das cores de uma coloração de arestas do grafo completo K_{2k+1} feita como descrito na Seção 2.1. Dizemos que uma cor c sobra em um vértice v quando não existe aresta com cor c incidente em v. Observe que em cada vértice v_i do grafo completo K_{2k+1} sobra a cor $2i \mod (2k+1)$. Por construção, os vértices adjacentes com grau máximo no P_n^k terão conjuntos de cores distintos. Então, basta garantir que vértices com mesmo grau e que tenham grau menor que $2k$ não sejam adjacentes. Esta condição é garantida quando a potência de caminho tem pelo menos $3k$ vértices, como apresentado no Lema 3.1.

Lema 3.1. Seja G uma potência de caminho P_n^k. Se $n \geq 3k$, então não existem vértices v_l e v_r que são adjacentes e $d(v_l) = d(v_r) < \Delta(P_n^k)$.

Prova. Seja P_n^k uma potência de caminho tal que $n \geq 3k$. Como toda potência de caminho é um grafo indiferença, seus vértices podem ser linearmente ordenados de forma que vértices adjacentes sejam consecutivos na ordem. Considerando tal ordenação, rotule os vértices do grafo P_n^k, v_0, v_1, ..., v_{n-1}. Sejam $L = \{v_0, v_1, \ldots, v_{k-1}\}$ e $R = \{v_{n-k}, \ldots, v_{n-2}, v_{n-1}\}$ os conjuntos dos k primeiros e dos k últimos vértices nessa ordem. Como $n \geq 3k$, $\Delta(P_n^k) = 2k$. Observe que os vértices dos conjuntos L e R não são $\Delta(P_n^k)$-vértices. Portanto, $2k$ vértices têm grau menor que $\Delta(P_n^k)$. Por construção, quaisquer dois vértices do conjunto L (ou R) tem graus distintos. Então, se existem dois vértices v_l e v_r adjacentes e com graus $d(v_l) = d(v_r) < \Delta(G)$, um deles pertence a L e o outro pertence a R. Dentre os vértices de mesmo grau e que têm grau menor que $\Delta(P_n^k)$, os que estão mais próximos na ordem imposta aos vértices são v_{k-1} e v_{n-k}. Como por hipótese $n \geq 3k$, $n-k \geq 3k-k = 2k$. Então, v_{n-k} é o vértice v_{2k} ou algum vértice a sua direita na ordem linear. Logo, a distância entre quaisquer dois vértices v_l e v_r com $d(v_l) = d(v_r) < \Delta(P_n^k)$ é maior ou igual que a distância entre v_{k-1} e v_{2k}. Então, existem pelo menos k vértices entre v_l e v_r. Portanto, esses vértices não são adjacentes. ■

A seguir provamos que o índice cromático distinto nos vértices adjacentes do grafo P_n^k, com $n \geq 3k$, é $\Delta(P_n^k)+1$.

Teorema 3.2. *Se G é uma potência de caminho P_n^k com $n \geq 3k$, então $\chi'_a(G) = \Delta(G)+1$.*

Prova. É importante lembrar que as potências de caminho são uma subclasse dos grafos indiferença e, portanto, é possível aplicar a técnica *pullback*. Considere uma potência de caminho P_n^k com $n \geq 3k$. Para se obter uma coloração de arestas DVA faça o seguinte procedimento. Obtenha a coloração de arestas do grafo K_{2k+1}. Rotule os vértices do P_n^k: v_0, v_1, v_2, ... ,v_{2k}, ..., v_0, v_1, v_2, ... ,v_{2k}, v_0, ... Pinte cada aresta (v_i, v_j) do P_n^k, com a cor da aresta (v_i, v_j) do K_{2k+1}. Após o procedimento, todas as arestas do P_n^k estão coloridas, já que para qualquer par de rótulos a cor da aresta está definida no grafo K_{2k+1}.

Vamos garantir que a coloração de P_n^k é uma coloração de arestas DVA. Considere os conjuntos $L = \{v_0, v_1, v_2, \ldots, v_{k-1}\}$ e $R = \{v_{n-1}, v_{n-2}, v_{n-3}, \ldots, v_{n-(k-1)}\}$. Por definição, $|L| = k$, $|R| = k$ e, para cada par de vértices v_i e v_j pertencentes ao conjunto L (ou ao conjunto R), tem-se $d(v_i) \neq d(v_j)$. Portanto, os vértices de L (R) possuem conjuntos de cores distintos. Pelo Lema 3.1 sabemos que não existe um vértice $v_l \in L$ que é adjacente a um vértice $v_r \in R$ tal que $d(v_l) = d(v_r) < \Delta(P_n^k)$ quando $n \geq 3k$. Logo, não existem vértices adjacentes com grau menor que $\Delta(P_n^k)$ que tenham conjuntos de cores com a mesma cardinalidade. Obviamente, vértices com grau menor que $\Delta(P_n^k)$ têm conjuntos de cores de tamanhos diferentes dos vértices com grau igual a $2k$ e, portanto, tais conjuntos de cores são diferentes. Por fim, falta garantir que cada par de vértices adjacentes em $V(P_n^k) \setminus (L \cup R)$ possuem conjuntos de cores distintos. Os $n-2k$ vértices em $V(P_n^k) \setminus (L \cup R)$ têm grau $\Delta(P_n^k) = 2k$. Então, em cada um desses vértices o conjunto de cores é o mesmo do vértice correpondente no grafo K_{2k+1}. A coloração de arestas do grafo K_{2k+1} é distinta nos vértices, já que $2k+1$ é ímpar e, portanto, em cada vértice $v_i \in K_{2k+1}$ falta a cor $2i \pmod{2k+1}$, $0 \leq i \leq 2k$. Como vértices de grau máximo com conjuntos de cores iguais estão a distância $2k+1$ no grafo P_n^k, os mesmos não são adjacentes. Portanto, o conjunto de cores de quaisquer dois vértices adjacentes de grau $\Delta(P_n^k)$ é distinto. Assim, para qualquer P_n^k com $n \geq 3k$, é possível obter uma coloração de arestas DVA utilizando $\Delta(P_n^k)+1$ cores. ■

A Figura 2 apresenta uma coloração de arestas DVA para um grafo P_9^3 obtida com a técnica descrita no Teorema 3.2.

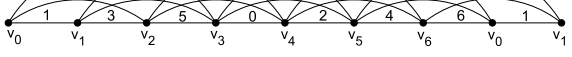

Figura 2: Coloração de arestas DVA do grafo P_9^3

A Figura 3 apresenta uma coloração de arestas do grafo P_7^3. Note que $n < 3k$, e que os vértices com rótulos 2 e 4 são adjacentes e apresentam conjunto de cores iguais. Portanto, a mesma técnica não pode ser aplicada diretamente para se obter uma coloração de arestas DVA quando $n < 3k$.

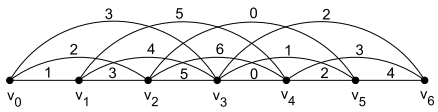

Figura 3: Coloração de arestas do grafo P_7^3

4. CONCLUSÃO E TRABALHOS FUTUROS

Zhang *et al.* [12] conjecturam que $\Delta(G) \leq \chi'_a(G) \leq \Delta(G)+2$ para todo grafo G simples e conexo com pelo menos 3 vértices que não seja um C_5. Em *The avd-edge-coloring conjecture for some split graphs* [10], os autores provaram que tal conjectura é válida para os grafos split-indiferença. Os split-indiferença são grafos indiferença com até três cliques maximais. E nesse trabalho, provamos que essa conjectura é válida para as potências de caminho P_n^k com $n \geq 3k$. Essas duas classes de grafos têm interseção não-vazia, mas nenhuma delas contém propriamente a outra.

Como o índice cromático distinto nos vértices adjacentes de grafos completos, K_n, é conhecido e toda potência de caminho P_n^k com $n \leq k+1$ é um grafo completo, resta considerar os casos em que $k+1 < n < 3k$.

Nesses casos, sabemos que há vértices v_l e v_r que são adjacentes e têm graus $d(v_l) = d(v_r) < \Delta(P_n^k)$.

Pretende-se considerar também variações da coloração de arestas distinta nos vértices adjacentes, como a coloração de arestas distinta nos vértices, a coloração total distinta nos vértices e a coloração total distinta nos vértices adjacentes.

5. REFERÊNCIAS

[1] M. Behzad. *Graphs and their chromatic numbers*. PhD thesis, Michigan State University, 1965.

[2] A. C. Burris and R. H. Schelp. Vertex-distinguishing proper edge-colorings. *Journal of graph theory*, 26(2):73–82, 1997.

[3] G. Chartrand and P. Zhang. *Chromatic graph theory*. CRC press, 2008.

[4] C. M. de Figueiredo, J. Meidanis, and C. P. de Mello. On edge-colouring indifference graphs. In *Latin American Symposium on Theoretical Informatics*, pages 286–299. Springer, 1995.

[5] C. M. de Figueiredo, J. Meidanis, and C. P. de Mello. Total-chromatic number and chromatic index of dually chordal graphs. *Information processing letters*, 70(3):147–152, 1999.

[6] C. M. de Figueiredo, J. Meidanis, C. P. de Mello, and C. Ortiz. Decompositions for the edge colouring of reduced indifference graphs. *Theoretical computer science*, 297(1):145–155, 2003.

[7] T. R. Jensen and B. Toft. *Graph coloring problems*, volume 39. John Wiley & Sons, 2011.

[8] C. Ortiz, N. Maculan, and J. L.Szwarcfiter. Characterizing and edge-coloring split-indiference graphs. *Discrete applied mathematics*, 82:209–217, 1998.

[9] M. Plantholt. The chromatic index of graphs with a spanning star. *Journal of graph theory*, 5(1):45–53, 1981.

[10] A. d. M. Vilas-Bôas and C. P. de Mello. The avd-edge-coloring conjecture for some split graphs. *Matemática contemporânea*, 44:1–10, 2015.

[11] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.*, 3:25–30, 1964. (in Russian).

[12] Z. Zhang, L. Liu, and J. Wang. Adjacent strong edge coloring of graphs. *Applied mathematics letters*, 15(5):623–626, 2002.

Sistema para Detecção e Reconhecimento de Placas de Limite de Velocidade

Felipe Paes Gusmão
Universidade Tecnológica Federal do Paraná
Av Monteiro Lobato, s/n - Km 04
Ponta Grossa - PR - Brasil
felipegusm1@gmail.com

Ionildo José Sanches
Universidade Tecnológica Federal do Paraná
Av Monteiro Lobato, s/n - Km 04
Ponta Grossa - PR - Brasil
ijsanches@utfpr.edu.br

RESUMO

Neste artigo, descrevemos o trabalho em andamento objetivando o desenvolvimento de uma metodologia para detecção e reconhecimento de placas de sinalização trânsito de limite de velocidade, adquiridas por câmera de vídeo convencional acoplada a um veículo em movimento. Utilizou-se uma coleta de dados, treinamento por aprendizagem de máquina por meio do algoritmo de Viola e Jones, detecção das placas de sinalização de limite de velocidade e um estudo sobre os trabalhos atuais encontrados na literatura. Os resultados obtidos são: um banco de imagens de vídeo específico para treinamento e testes de outros métodos de detecção e as etapas de construção da metodologia.

Palavras-chave

Processamento de Imagens; Visão Computacional; Aprendizagem de Máquina; Dectecção e Reconhecimento de Placas de Sinalização de Trânsito; Sistema de Apoio ao Motorista.

ABSTRACT

In this paper we describe the work in progress aiming the development of a methodology for speed-limit traffic signs detection and recognition acquired by conventional camera coupled to a moving vehicle. It was used a set of data collected in thoroughfares, machine learning based on Viola-Jones algorithm, detection of the speed-limit traffic signs and a study of the most recent work in area. The results are: a data set of images and videos for this specific purpose and the steps to construct a methodology for detection and recognition of speed-limit traffic signs.

Keywords

Image Processing; Computer Vision; Machine Learning; Traffic Signs Detection and Recognition; Driver Support Systems.

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

1. INTRODUÇÃO

Os sistemas de apoio ao motorista (DSS – *Driver Support Systems*) juntamente com a Visão Computacional vêm fornecendo auxílio em situações onde o motorista necessita ajuda na captura de informações do cenário. Os principais trabalhos em DSS são: detecção de obstáculos, sistemas integrados, detecção de marcas em estradas e detecção de placas de sinalização de trânsito.

A detecção e reconhecimento de placas de sinalização de trânsito se faz importante na segurança de trânsito, oferecendo informações não captadas naturalmente pelo motorista. Um grupo de placas de trânsito o qual mostra atenção da área são as placas de limite de velocidade.

No Brasil, a aplicação das placas de limite de velocidade nas vias é padronizada pelo Conselho Nacional de Trânsito (CONTRAN) [7].A forma padrão de um sinal de limite de velocidade é circular, com fundo na cor branca, símbolo na cor preta, orla na cor vermelha e letras na cor preta, como pode ser observado na Figura 1.

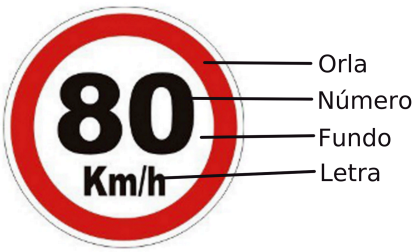

Figure 1: Exemplo de placa de limite de velocidade.

Os limites de velocidades são estabelecidos visando a segurança dos condutores e dos pedestres nas vias públicas. O não cumprimento deste limite pode acarretar acidentes graves e também infrações de trânsito. Ao trafegar em ruas de grandes centros urbanos, os motoristas se deparam com uma grande quantidade de veículos, pedestres, sinalização vertical e horizontal e redutores de velocidade, ou seja, uma grande quantidade de informação que pode não ser observada pelo condutor.

O número de mortes em acidentes de trânsito alerta para a criação e implantação de métodos para detecção de placas de trânsito no Brasil. São mais de 42 mil em 2013, segundo dados do Departamento de Informática do SUS [27].

Os métodos de detecção e reconhecimento de placas de

sinalização de trânsito (MDPS) podem ser classificados em uma, duas e/ou três categorias: baseado na cor, forma e aprendizado. Utilizando aprendizagem de máquina com detecção e reconhecimento automático de objetos, o algoritmo de Viola e Jones [25] pode ser utilzado para detecção e reconhecimento de placas de sinalização.

Métodos que implementam identificação de objetos em imagens apresentam duas etapas principais: a fase de detecção, onde é determinado se o objeto está presente no contexto e a posição especial deste objeto dentro da imagem; e a fase de reconhecimento na qual o objeto é identificado dentro de um grupo de outros objetos.

Os trabalhos que utilizam MDPSs baseados na forma são: Loy e Barnes [18] com um detector baseado em polígonos regulares (círculo, triângulo, quadrado e hexágono). Paulo e Correia [21] descrevem um identificador baseado na detecção de bordas aplicando o detector de ROI (*Region of Interest*) de Harris e Stephens [14]. Gavrila [13] apresenta um detector baseado na transformada da distância (DT - *distance transform*). Moutarde *et al.* [19] descrevem um detector utilizando a transformada de Hough para detecção de objetos circulares e um algoritmo para detecção de objetos retangulares criado pelos autores.

Os trabalhos que utilizam MDPSs baseados em cor procuram reconhecer a placa com base na análise das cores e suas respectivas posições na imagem. São encontrados na literatura análises em diferentes modelos de cores, como RBG [1, 10, 24, 4], HSV [20, 22, 17, 11, 8] e CIECAM97 [12].

Dentre os trabalhos que utilizam aprendizagem de máquina estão: Chen *et al.* [5] com treinamento AdaBoost [16] e identificação de áreas homólogas por meio de características Haar; Jeon *et al.* [15] com um método constituindo dos algoritmos: seleção da área de interesse, segmentação de cores e variância local. E outros trabalhos os quais utilizam o algoritmo de Viola e Jones [3, 2, 6, 26, 9, 23], obtendo taxas de acerto entre 90 e 96%.

Este artigo propõe a construção de uma metodologia para detecção e reconhecimento de placas de limite de velocidade a partir dos MDPSs existentes na literatura e o algoritmo de Viola-Jones com um conjunto de imagens de vídeo obtidas por câmeras convencionais.

2. METODOLOGIA

A Figura 2 ilustra as quatro etapas deste trabalho: coleta e organização dos dados, treinamento, identificação de positivos e execução do método. A seguir, serão detalhadas cada etapa deste processo.

2.1 Coleta e Organização dos Dados

Foi realizada uma coleta de vídeos com uma câmera convencional acoplada a um veículo em movimento em vias urbanas. No total, foram coletados mais de duas horas de vídeo com 44 placas de limite de velocidade (positivos) e 29 placas diversas.

Os vídeos foram realizados nas cidades de Flórida, Maringá e Ponta Grossa (Paraná) com as velocidades: 15, 20, 40, 60 e 80 Km/h. Também foram diferenciados quanto à condição climática, sendo 70% das imagens realizadas no período diurno e tempo ensolarado, 20% no período noturno e tempo seco e 10% no período diurno com tempo chuvoso.

2.2 Treinamento

O algoritmo de Viola-Jones [25] originalmente foi utilizado para a detecção de faces, porém é possível realizar um treinamento para que seja possível detectar outros tipos de objetos ou formas. O algoritmo funciona em três etapas:

1. Treinamento baseado em *AdaBoost*, que seleciona as principais características da imagem;
2. Representação da imagem em uma forma intermediária que permite o cálculo das características usando poucas operações com o auxílio de pequenos retângulos;
3. Combinação de classificadores em cascata que são utilizados para o descarte rápido de áreas desnecessárias.

Após a aquisição das imagens de sinalização das placas de limite de velocidade, será realizada a detecção da região da placa, baseado no treinamento de máquina, utilizando a biblioteca OpenCV (*Open Source Computer Vision*). Em seguida, as imagens serão segmentadas para permitir o reconhecimento da placa.

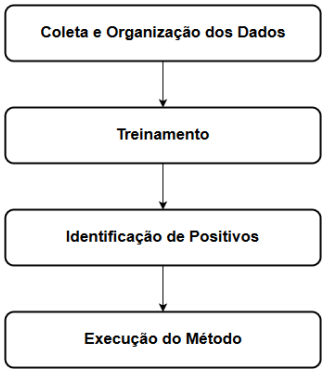

Figure 2: Diagrama das etapas do algoritmo.

2.3 Identificação de Positivos

Os positivos representam objetos (neste caso, placas de limite de velocidade) presentes nos vídeos, estes podem ser:

- Verdadeiro positivo: o método identifica presença de placas de limite de velocidade e há placas de limite de velocidade no frame;
- Falso positivo: o método identifica presença de placas de limite de velocidade, porém não há placas de limite de velocidade no frame.

A identificação de positivos possui o objetivo principal de fornecer a posição da possível placa para que elimine processamento desnecessário na imagem.

2.4 Execução do Método

Após o treinamento, obtém-se um algoritmo que busca reconhecer as placas de limite de velocidade utilizando as imagens obtidas na fase de coleta e organização de dados.

A identificação das placas de limite de velocidade de acordo com a execução do método será armazenada em uma tabela de resultados para posterior avaliação, assim como o tempo de execução do treinamento, a quantidade de falsos positivos e verdadeiros positivos.

3. RESULTADOS E CONCLUSÃO

Os resultados já alcançados pelo trabalho foram a coleta e a organização dos dados. A etapa de treinamento está em desenvolvimento e as seguintes etapas serão desenvolvidas. Após a execução das quatro etapas descritas na seção de metodologia, será possível realizar comparações com outros trabalhos que identificam placas de limite de velocidade.

Este trabalho propõe o desenvolvimento de uma metodologia para a detecção e reconhecimento de placas de limite de velocidade nas vias de trânsito brasileiras.

4. REFERÊNCIAS

[1] M. Bénallal and J. Meunier. Real-time color segmentation of road signs. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 3, pages 1823–1826. IEEE, 2003.

[2] K. Brkić, Z. Kalafatić, A. Pinz, et al. Generative modeling of spatio-temporal traffic sign trajectories. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 25–31. IEEE, 2010.

[3] K. Brkic, A. Pinz, and S. Šegvic. Traffic sign detection as a component of an automated traffic infrastructure inventory system. *Stainz, Austria, May*, 2009.

[4] A. Broggi, P. Cerri, P. Medici, P. P. Porta, and G. Ghisio. Real time road signs recognition. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 981–986. IEEE, 2007.

[5] L. Chen, Q. Li, M. Li, and Q. Mao. Traffic sign detection and recognition for intelligent vehicle. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 908–913, June 2011.

[6] S.-Y. Chen and J.-W. Hsieh. Boosted road sign detection and recognition. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 7, pages 3823–3826. IEEE, 2008.

[7] CONTRAN. Manual brasileiro de sinalização de trânsito, 2007.

[8] A. De la Escalera, J. M. Armingol, and M. Mata. Traffic sign recognition and analysis for intelligent vehicles. *Image and vision computing*, 21(3):247–258, 2003.

[9] S. Escalera, P. Radeva, et al. Fast greyscale road sign model matching and recognition. *Recent Advances in Artificial Intelligence Research and Development*, 2(1):69–76, 2004.

[10] L. Estevez and N. Kehtarnavaz. A real-time histographic approach to road sign recognition. In *Proceedings of the IEEE southwest symposium on image analysis and interpretation*, pages 95–100, 1996.

[11] C.-Y. Fang, S.-W. Chen, and C.-S. Fuh. Road-sign detection and tracking. *Vehicular Technology, IEEE Transactions on*, 52(5):1329–1341, 2003.

[12] X. W. Gao, L. Podladchikova, D. Shaposhnikov, K. Hong, and N. Shevtsova. Recognition of traffic signs based on their colour and shape features extracted using human vision models. *Journal of Visual Communication and Image Representation*, 17(4):675–685, 2006.

[13] D. M. Gavrila. Traffic sign recognition revisited. In *Mustererkennung 1999*, pages 86–93. Springer, 1999.

[14] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[15] W. J. Jeon, G. A. R. Sanchez, T. Lee, Y. Choi, B. Woo, K. Lim, and H. Byun. Real-time detection of speed-limit traffic signs on the real road using haar-like features and boosted cascade. In *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, page 93. ACM, 2014.

[16] A. Z. Kouzani. Road-sign identification using ensemble learning. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 438–443. IEEE, 2007.

[17] W.-J. Kuo and C.-C. Lin. Two-stage road sign detection and recognition. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1427–1430. IEEE, 2007.

[18] G. Loy and N. Barnes. Fast shape-based road sign detection for a driver assistance system. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 70–75. IEEE, 2004.

[19] F. Moutarde, A. Bargeton, A. Herbin, and L. Chanussot. Robust on-vehicle real-time visual detection of american and european speed limit signs, with a modular traffic signs recognition system. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 1122–1126. IEEE, 2007.

[20] P. Paclık, J. Novovičová, P. Pudil, and P. Somol. Road sign classification using laplace kernel classifier. *Pattern Recognition Letters*, 21(13):1165–1173, 2000.

[21] C. F. Paulo and P. L. Correia. Automatic detection and classification of traffic signs. In *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS'07. Eighth International Workshop on*, pages 11–11. IEEE, 2007.

[22] A. Ruta, Y. Li, and X. Liu. Detection, tracking and recognition of traffic signs from video input. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 55–60. IEEE, 2008.

[23] R. Timofte, K. Zimmermann, and L. Van Gool. Multi-view traffic sign detection, recognition, and 3d localisation. *Machine Vision and Applications*, 25(3):633–647, 2014.

[24] S. Varan, S. Singh, R. Sanjeev Kunte, S. Sudhaker, and B. Philip. A road traffic signal recognition system based on template matching employing tree classifier. In *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, volume 3, pages 360–365. IEEE, 2007.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[26] J. Vitria et al. Fast traffic sign detection on greyscale images. *Recent Advances in Artificial Intelligence Research and Development*, page 209, 2004.

[27] J. J. Waiselfisz. Mapa da violência 2013: homicídios e juventude no brasil. 2013.

A Middleware for Using PIC Microcontrollers and Jason Framework for Programming Multi-Agent Systems

João Victor Guinelli
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - Campus Nova Friburgo
Av. Gov. Roberto Silveira 1900
Nova Friburgo-RJ, Brasil
CEP 28.635-000
joao.silva@cefet-rj.br

Carlos Eduardo Pantoja
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - Campus Maria da Graça
Rua Miguel Ângelo 96 -
Maria da Graça-RJ, Brasil
CEP 20.785-220
pantoja@cefet-rj.br

ABSTRACT

This paper presents a middleware using PIC microcontrollers for programming robotic agents controlled by a Multi-Agent Systems alongside with Jason Framework. The middleware is a hardware side library developed in C for PIC and is based on the Javino protocol. The proposed middlewarel allows a communication between hardware and software and aims to be used together with Javino for software side. A simple example is presented to show the basic functioning of a robot architecture using the middleware.

Keywords

Middleware; Multi-Agent Systems; Robotics

1. INTRODUCTION

Nowadays, a robot can be understood as an intelligent agent, which is able to perceive, reason and act into a real environment [7]. Similarly, agents can be defined as virtual or physical entities that are capable of perceiving or acting into an environment, where they can be situated. So, in robotics it is reasonable to use Multi-agent Systems (MAS) approach and agent-oriented programming languages. Furthermore, MAS approach can be used for complex and distributed problems.

However, embedding BDI agents in robotics is not a trivial task since their reasoning cycle can generate undesirable delays into the robot execution in real environments [8]. Some works tries to embed MAS in platforms using BDI frameworks such as [1] and [4]. In [1] it is proposed a robotic platform that moves from one point to another based on the GPS position. The prototype uses the JASON framework [3] but the reasoning agent is not embedded in the robot vehicle. On the other hand, [4] extends the Jason in order to control Lego robots that are able to follow lines on the floor and avoid obstacles. However, the extension does not allow embedding MAS because Lego platform does not have sufficient space for a MAS system.

In [5] it is presented a middleware that is responsible for helping the communication between low-level hardware and high-level programming languages and it is used alongside with Jason for programming embedded MAS. However, the Javino middleware is dedicated only to ATMEGA microcontrollers. Therefore, the objective of this paper is to present the Javic middleware for communication between PIC microcontrollers and the Jason framework. The Javic uses a library developed for the PIC side and it the JAVA side library of Javino. This middleware aims to provide the usage of PIC microcontrollers in embedded MAS since PIC is one of the most used microcontrollers in industry and automation applications.

This paper is structured as follows: section 6 analyzes some related works; section 3 presents the robotic-agent architecture; in section 4 the Javic middleware is explained; section 5 shows how to embed Jason and use Javic middleware to communicate with PIC microcontrollers; in section 6 a case study is presented; and the section 7 presents the conclusions and future works.

2. RELATED WORK

In this section it is discussed some related works, which also make use of multi-agent programming languages to control robotic platforms. In [1], the authors propose an integration between Jason and Arduino in order to control grounded-vehicles. To implement the communication between the Arduino board and Jason's, the authors used the RxTx library. Using that approach, it is possible to use Jason to operate any kind of vehicle, however, Jason is not truly embed since it is running in a external computer that communicates itself with the vehicle through transmitters and receivers. Besides that, no protocol is used to perform the communication between the software and hardware-side, what can cause failures on the system since data loss and interferences are not handled. Differently, this work can truly embed Jason in a Raspberry board, and we use Javic, that implements the same protocol implemented by Javino, to perform the communication between the software and the hardware-side.

In [4] the author extend Jason framework, through the implementation of internal actions, to make possible the communication between physical agents and the Lego Mindstorm NXT toolkit. That communication is performed using Bluetooth and LeJOS as the middleware between Jason and Lego. However, according to the author, this communication is kind of slow, and and the robotic platform is tied to Lego Mindstorm NXT. On the other hand, the agent-robotic architecture presented in this paper can be used in any robotic platform that uses Java-based languages in the software-side, and ATMEGA or PIC microcontrolers in the hardware-side.

3. ROBOTIC-AGENT ARCHITECTURE

The development of the Javic middleware was motivated mainly by its possible application in robotic-agents using embedded MAS, which can be able to control different devices connected to different microcontrollers. In fact, Javic allows the design of robotic-agents using PIC microcontrollers while Javino uses the ATMEGA microcontrollers. Despite of the possibility of a selection between these microcontrollers, both hardware-side libraries communicate with the same software-side library, allowing the usage of both microcontrollers into the same robot. In order to clarify this approach, we propose a robotic-agent architecture for using along with any microcontroller or agent-oriented programming language with components adherent to the protocol implemented by Javino and Javic.

A robotic-agent designed using the proposed architecture contains microcontrollers of different types, each of them controlling sensors and actuators, and a central core responsible for the reasoning, based on information perceived by sensors, and it is able of controlling actuators. For interconnecting the hardware and the software of the robot, it is necessary to use middleware.

In Figure 1, it is possible to see the architecture of a robot using different microcontrollers, such as PIC, ATMEGA, INTEL or any other, as long as this hardware uses a middleware compliant to the same protocol used in Javino and Javic when communicating. For each microcontroller there are several actuators and sensors connected. The hardware-side libraries used are Javic for PIC and Javino for ATMEGA. However the architecture is extensible, so it is possible to implement a hardware-side library for INTEL8051 for example (named Javintel) or any other library for other microcontroller.

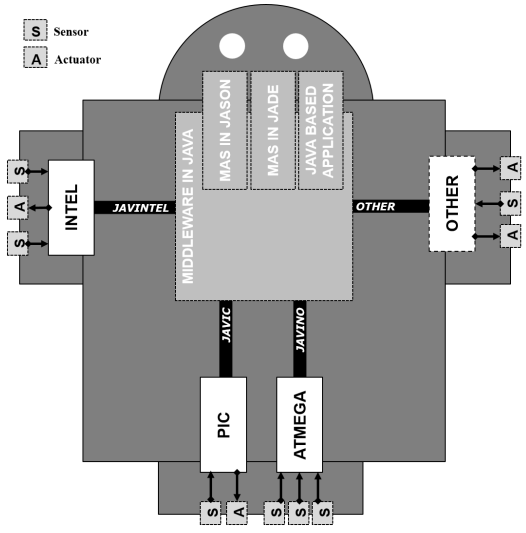

Figure 1: A robotic agent architecture using Javino and Javic middleware.

In the context of software-agents, the core of the robot needs to execute an application programed in Java or any other framework based on that language, such as MAS in Jason or JADE [2]. However, if another programming language or framework were employed, it is necessary to develop the software-side of the protocol of Javino. The Javino software-side library is responsible for gathering all the perceptions from the microcontrollers and, after that, these perceptions has to be processed as beliefs depending on the MAS framework chosen. Based on the beliefs processing, the MAS can control the actuators through Javino library, which communicates with the microcontroller sending the actions to be performed in real world.

Therefore, depending on the strategy chosen for the implementation of MAS, the agent may exchange information with only one microcontroller at time or be programmed to gathers all perceptions and process them all together. This means that, for example, importing Javino into the Jason's reasoning cycle, when the agent is perceiving the real world environment, it will catch only the perceptions that came from the microcontroller (in the serial port set on Javino) that is being used in that moment. The same thing occurs when it is necessary to manipulate an actuator, the agent only has access to the microcontroller connected to the selected in Javino in that moment, but the agent can use Javino at real time to change the port where it is communicating.

4. JAVIC MIDDLEWARE

The Javic is a middleware that implements the same protocol as Javino in order to allow a communication using the serial port between a software in Java and PIC microcontrollers. The Javic middleware is composed of a library for PIC (hardware-side) and another one for Java (software-side).

For this, it was developed a library for PIC devices, which uses the C language. Depending on the type of the PIC, the amount of available memory can interfere on the functioning of this version of the library. The library was developed to work in PIC with at least 256 bytes of RAM memory because the size of the message is up to 256 bytes. It is possible to use the PIC 18F, 24F, 30F and 33F without restrictions. The PIC from family 16F should have at least 256 bytes of RAM and for family 12F it is not possible to use this version of Javic (however a version with a short version of Javic could be implemented if it is desirable). Table 1 shows a summary of PIC specifications.

Table 1: Amount of available memory of PIC families

Devices	RAM (bytes)	ROM (bytes)
12FXXX	64 – 128	256
16FXXX	128 – 256	256
18FXXX	256 – 512	256
24FXXX	24K – 96K	–
30FXXX	512 – 8K	1024 – 4092
33FXXX	8K – 30K	–

As the Javic is an implementation of the protocol for hardware-side, in order to maintain a pattern between the previous existent implementation and possible next ones, the methods presents in the Javic are the same present in the Javino for ATMEGA microcontrollers.

In the software-side of the middleware, it is used the Javino library for Java since it is not bound to the type of the microcontroller. It uses serial communication for exchanging messages. Then, when the hardware-side library sends a message (using PIC or ATMEGA), the software-side library receives and verifies the correctness of the content. If there is no data loss during the transmission, the message

is delivered. When the software-side library needs to send a message to the other side, it is necessary to inform the port where the target device is connected (it is not possible to send the same message for all the devices at the same time).

With the development of the Javic, nowadays there is tree libraries that implement the Javino protocol: one for the software-side that uses the Java language, and others two for the hardware-side, one for ATMEGA, and another one for PIC microcontrollers.

5. EMBEDDING JASON + JAVIC

For embedding any MAS in robots, it is necessary to program and interfere in different abstraction layers using several hardware (sensors, actuators, microcontrollers, boards and operational systems) and software (middleware and programming languages). These layers should be independent depending of the robot architecture and project. The modularity of the layers provides a maintainable and scalable architecture with high cohesion and low coupling. Besides, it should provide a fault tolerance mechanism for avoiding a total block of the MAS in case of malfunctioning of a component in real time constraints.

For using Jason and Javino, or Javic, for embedding MAS into any robotic platform (Figure 3), it is necessary to interfere in four layers accordingly to [5]:

1. **Hardware**. Firstly, this layer represents the physical robot, where all the components are interconnected. The sensors and actuators are plugged on the microcontrollers that are connected on serial ports of the selected board. The board should be able to host an operational system where the MAS will run. The selected board was the Raspberry Pi. Therefore, the operational system of the selected board was adapted to automatically run the MAS as soon as it starts;

2. **Microcontroller.** The microcontroller is responsible for controlling the sensing and acting into the real world. In this layer, all desirable actions should be programmed as procedures that are called in return of received serial messages. Besides, the perceptions has to be sent through serial port. The microcontroller used was the PIC 18F4520;

3. **Middleware.** The middleware is responsible for exchanging the messages between the microcontroller layer and the agent layer. For this, the middleware should be imported in both layers (microcontroller and agent). Javino and Javic were used as the middleware, Javino in the software-side and Javic in the hardware-side.

4. **Agent.** In this layer, the MAS is programmed. The agent-oriented programming language used is Jason, where in the simulated environment the Javino library is responsible for exchanging messages with the microcontroller.

6. PRELIMINARY TEST

In order to test the proposed architecture, we present a basic test using Javic. We will use Jason Framework to develop the MAS because it is based on BDI, which is a cognitive model is responsible for the reasoning of the system. Our purpose with this section is to prove that the architecture can work properly using the middleware for communicating with a MAS. So, we live for further work a performance analysis and tests in a complex scenario.

Here, we show an example using PIC 18F4520 connected to an ultrasonic sensor and a led as actuator. We choose this PIC version because of the amount of memory available for mounting Javic's messages. In this case, the agent should turn on the light when it realizes that an obstacle is approaching. In this example, the PIC microcontroller was programmed based on methods that can be activated when a serial message arrive from an agent. The same occurs for the perceptions, once the agent requests them based on its needs. However, it is possible to program the microcontrollers to send the perceptions from time to time. The program code of the microcontrollers can be seen as follows (we use C to represent the PIC program code):

```
while(true){
  receivedMessage = javic.getMsg();
  if(receivedMessage="getPercepts")
    getPercepts();

  if(receivedMessage="turnOn")
    turnOn();

  if(receivedMessage="turnOff")
    turnOff();
}

void getPercepts() {
  //code for getting the distance from
    ultrasonic sensors
}

void turnOn() {
  //code for turning on the led
}

void turnOff() {
  //code for turning off the led
}
```

In Jason, we optioned to use the simulated environment in Java to provide the external actions to be performed by the agent in real world (using actuators) and the perceptions processing from the microcontrollers. The environment code can be seen as follows:

```
if(action.toString().equals("refresh")){
  this.javino.port = "COM8";
  this.javino.sendMsg("getPercepts");
  if(this.jBridge.availablemsg()){
    this.msg = "dist(pic," + this.javino.
    getmsg() + ")";
    addPercept(Literal.parseLiteral(this.
    msg);
  }
}

if(action.toString().equals("lightOnPic")){
  this.javino.port = "COM8";
  this.javino.sendmsg("turnOn");
}

if(action.toString().equals("lightOffPic"))
    {
  this.javino.port = "COM8";
  this.javino.sendmsg("turnOff");
}
```

The agent uses Jason's external actions to get perceptions and to activate the actuators. The external action activates the software side of the middleware that sends a serial message to the PIC microcontroller which get the perceptions or execute an action. The agent code can be seen as follows:

```
+!moving: dist(pic, X) & value(20) & X>J <-
   refresh;
   lightOffPic;
   !moving.

+!moving: dist(pic, X) & value(20) & X<=J
    <-
   refresh;
   lightOnPic;
   !moving.
```

7. CONCLUSIONS

This paper presented the Javic middleware, which implements the same protocol as Javino and allow the communication between Java-based languages and PIC microcontrolers. For future works, we will develop a tiny version of Javic for PIC with RAM memory below 256 bytes, reducing the length of the message in the protocol to 128 bytes. Besides, we will also provide a library for Javintel middleware, in order to provide a communication between INTEL8051 microcontrollers and the Java language. Then, the architecture will provide a selection among several microcontrollers used at industry.

The use of embedded MAS in real-time scenarios arises several performance issues as the time of processing perceptions and the response time that the robotic should proper act. So, the architecture should be analyzed in a complex scenario case study with real-time constraints. We also intend to use ARGO [5] for testing the Javic middleware in embedded scenarios.

8. REFERENCES

[1] R. S. Barros, V. H. Heringer, C. E. Pantoja, N. M. Lazarin, and L. M. de Moraes. An agent-oriented ground vehicle's automation using jason framework. In *ICAART (2)*, pages 261–266, 2014.

[2] F. Bellifemine, G. Caire, and D. Greenwood. *Developing multi-agent systems with JADE*. Wiley series in agent technology. John Wiley, 2007.

[3] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons, 2007.

[4] A. S. Jensen. Implementing lego agents using jason. *arXiv preprint arXiv:1010.0150*, 2010.

[5] N. M. Lazarin and C. E. Pantoja. A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. *9th Software Agents, Environments and Applications School*, 2015.

[6] C. E. Pantoja, M. F. Stabile Jr, N. M. Lazarin, and J. S. Sichman. Argo: A customized jason architecture for programming embedded robotic agents. *Third International Workshop on Engineering Multi-Agent Systems (EMAS 2016)*, 2016.

[7] S. Russel and P. Norvig. Inteligência artificial. *Editora Campus*, page 26, 2004.

[8] F. R. Santos and J. Hubner. Avaliação do uso de agentes no desenvolvimento de aplicações com veículos aéreos não-tripulados. 2015.

LuBras: Uma Arquitetura de um Dispositivo Eletrônico para a comunicação Libras-Língua Portuguesa Utilizando o Javino

Vinicius Souza de Jesus
CEFET/RJ, Rua Miguel Ângelo 96 , Maria da Graça, 20785 - 223, RJ, Brasil
(21) 99338-4561, 55
vinicius_gu_07@hotmail.com

Fabian Cesar P. B. Manoel
CEFET/RJ, Rua Miguel Ângelo 96 , Maria da Graça, 20785 - 223, RJ, Brasil
(21) 98812-1517, 55
fabiancpbm@gmail.com

Carlos Eduardo Pantoja
CEFET/RJ, Rua Miguel Ângelo 96 , Maria da Graça, 20785 - 223, RJ, Brasil
(22) 99809-2894, 55
pantoja@cefet-rj.br

Leandro Marques Samyn
CEFET/RJ, Rua Miguel Ângelo 96 , Maria da Graça, 20785 - 223, RJ, Brasil
(21) 99177-4832, 55
Leandro.samyn@cefet-rj.br

ABSTRACT

Aiming to teach and assist in the Portuguese language communication and Brazilian Language of Signals (LIBRAS), this work presents two gloves that interact directly with users and are responsible for displaying a series of instructions made by controllers (Arduino). Through the proposed architecture, LED glove and sleeve of flexible resistors were created: The first receives messages in Portuguese and turn them into light signals; The second receives a start command for read, captures hand movements and sends messages in Portuguese. The Javino is a middleware that performs transmission of data of the software for controller and vice versa. The software is situated on a minicomputer (Raspberry) and it uses the serial communication as a channel of communication with the hardware.

Keywords

Libras; communication; raspberry; Javino; arduino; serial communication.

RESUMO

Com o objetivo de ensinar e auxiliar na comunicação língua portuguesa e Língua Brasileira dos Sinais (LIBRAS), este trabalho apresenta duas luvas que interagem direto com os usuários e são responsáveis por exibir uma série de instruções realizadas pelos controladores (Arduino). Por intermédio da arquitetura proposta, foram criadas a luva de *LED* e a luva de resistores flexíveis: A primeira recebe mensagens em português e as transformam em sinais luminosos; A segunda recebe um comando de inicialização de leitura, capta os movimentos das mãos e envia mensagens em português. O Javino é um *middleware* que realiza a transmissão de dados do *software* para o controlador e vice-versa. O *software* está situado em um minicomputador (Raspberry) e este utiliza a comunicação serial como um canal de comunicação com o *hardware*.

Palavras Chave

Libras; comunicação; raspberry; Javino; arduino; comunicação serial.

1. INTRODUÇÃO

A tecnologia assistiva é um importante ramo a ser explorado, pois é encarregada de contribuir para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência e consequentemente promover independência e inclusão social [3].

Protótipos automatizados voltados para a tecnologia assistiva, em sua maioria, possuem controladores para coordenar tarefas e comandos a serem executados. Contudo, a pessoa com deficiência precisa interagir diretamente com o protótipo, tornando necessária uma interface gráfica. Porém para a interface gráfica conversar com os controladores é preciso utilizar os *middleware*, programas responsáveis de realizar a comunicação entre diferentes plataformas de hardware e software. O Javino [2] é um middleware para a troca de mensagens entre softwares programados em linguagens de alto nível e placas microcontroladas programadas em linguagens de baixo nível, como o Arduino, por meio da porta serial da placa. O Javino também realiza a verificação de erros nas mensagens através de um protocolo de comunicação.

Existem alguns projetos assistivos que são responsáveis por tentar realizar a comunicação LIBRAS-Língua Portuguesa. O primeiro projeto é um bracelete que registra mensagens de gestos criados por uma pessoa surda e os traduzem em sons na língua portuguesa utilizando a comunicação *bluetooth*. Porém o trabalho realiza somente uma via da comunicação [6]. O segundo é um protótipo de uma luva equipada com acelerômetros nas pontas dos dedos, que através da leitura dos três eixos cartesianos, identificam os movimentos das mãos. A luva também está interligada a uma placa que decodifica esses movimentos e os transformam em áudio. Assim como o primeiro trabalho, este somente realiza uma via da comunicação [7].

O objetivo deste trabalho é desenvolver um protótipo capaz de viabilizar a comunicação entre surdos e ouvintes possibilitando uma maior independência para o surdo, uma vez que, torna-se desnecessária a utilização de um tradutor presente. Para atingir o objetivo foram criados dois protótipos de luvas. A primeira, a luva de LEDs, é responsável por traduzir em sinais luminosos algo que foi escrito em português, assim o leigo em LIBRAS pode fazer o sinal referente. Somente com esta luva, a comunicação fica unilateral, pois o surdo não conseguirá responder ou transmitir nenhuma informação. Portanto, foi criada a segunda luva,

responsável pela tradução dos sinais em LIBRAS para língua portuguesa.

Para a confecção do trabalho serão utilizados LED SMD e RGB, dois CI 74HC595 (para multiplexar as portas digitais do Arduino), dois Arduino Lilypad, duas Raspberry, cinco sensores flexíveis, um CI 4051 (para multiplexar as portas analógicas do Arduino) e um acelerômetro. As luvas serão construídas baseadas em uma arquitetura que não deixa lacunas na troca de informações, pois opera nas duas vias de comunicação, tanto da linguagem falada para a gestual quanto para a gestual para a falada.

2. REFERENCIAL TEÓRICO

Nesta seção serão apresentados de forma detalhada os princípios do *middleware*, Javino e microcontroladores como os PIC e o ATMEGA.

2.1 Javino

O Javino [2] é um protocolo de comunicação, implementado em duas bibliotecas e utilizado como *middleware* entre hardware e software. O mesmo fornece confiabilidade entre o emissor e o receptor por ter um processo de verificação da mensagem. A mensagem é composta por três campos: preâmbulo, tamanho de campo e conteúdo da mensagem. O primeiro é formado por quatro caracteres hexadecimais que são usados para identificar o início da mensagem. O segundo é estruturado por dois caracteres hexadecimais que são utilizadas para calcular a extensão de dados. Finalmente, o último é estruturado pelo conteúdo de escrita, que pode ser de até 255 bytes. Durante este processo, ambos, o preâmbulo de campo e tamanho são usados em conjunto, a fim de evitar a perda de informação e, por uma questão prática, o Javino monta automaticamente a mensagem final.

Quando uma mensagem é enviada de um hardware para um software de alto nível, ao lado do software implementado, o Javino emula a biblioteca em Java e começa a escutar a porta serial (aguardando a mensagem) e se houver alguma informação, ele armazena e analisa se é parte do preâmbulo esperado. Assim, este processo é repetido até que a mensagem tenha sido completamente recebida. O Javino descarta todas as informações, enquanto um preâmbulo válido não é confirmado. Caso contrário, ele verifica o valor de tamanho de campo, a fim de identificar o comprimento da mensagem. Devido a isso, é possível, evitar erros de cálculo ou definir onde uma mensagem começa e onde termina. Depois de terminar todo este processo, a mensagem é disponibilizada no lado do software. O mesmo processo funciona em uma comunicação entre software e hardware.

2.2 Microcontroladores

Os microcontrolares são circuitos integrados capazes de armazenar instruções lógicas, aritméticas e de tomada de decisão, permitindo ser programado para realizar tarefas específicas. Para seu funcionamento integral, circuitos auxiliares podem ser necessários. Por exemplo, filtragem de sinais, como alimentação e transmissão de dados, regulagem de frequência (*clock*), periféricos de entrada e saída de sinais digitais ou analógicos. Os microcontroladores *Programmable Interface Controller* (PIC) possuem diversas famílias diferenciadas entre si de acordo com a memória, a velocidade de processamento e arquitetura [4].

Os PIC utilizam basicamente duas arquiteturas: RISC e CISC. A RISC realiza pequenas instruções em um ciclo de *clock* e a CISC realiza grandes instruções em variados ciclos de *clock*. Portanto na arquitetura CISC os microcontroladores necessitam de mais memória. Os ATMEGA estão presentes em inúmeras placas microcontroladas comerciais, como o Arduino [5]. A escolha do controlador se dá principalmente pela capacidade de processamento que o projeto exigir. Os microcontrolares são circuitos integrados capazes de armazenar instruções lógicas, aritméticas e de tomada de decisão, permitindo ser programado para realizar tarefas específicas. Para seu funcionamento integral, circuitos auxiliares podem ser necessários. Por exemplo, filtragem de sinais, como alimentação e transmissão de dados, regulagem de frequência (*clock*), periféricos de entrada e saída de sinais digitais ou analógicos. Os microcontroladores *Programmable Interface Controller* (PIC) possuem diversas famílias diferenciadas entre si de acordo com a memória, a velocidade de processamento e arquitetura [4].

3. ARQUITETURA DO PROTÓTIPO

Nesta seção será apresentada a arquitetura do protótipo das luvas. A arquitetura das mesmas consiste em um fluxo de informações entre hardware e software podendo ser iniciado tanto pelo software quanto pelo hardware e a garantia da entrega correta da mensagem fica a cargo do Javino (Figura 1).

Em [1] foi desenvolvida uma interface gráfica que auxiliava na primeira comunicação entre LIBRAS e língua portuguesa, porém percebeu-se uma necessidade de uma maior relação entre usuário e hardware, pois a troca de informações ficava limitada a um programa e não existia nada físico. Então o objetivo deste trabalho é desenvolver luvas que realize a comunicação entre LIBRAS e língua portuguesa.

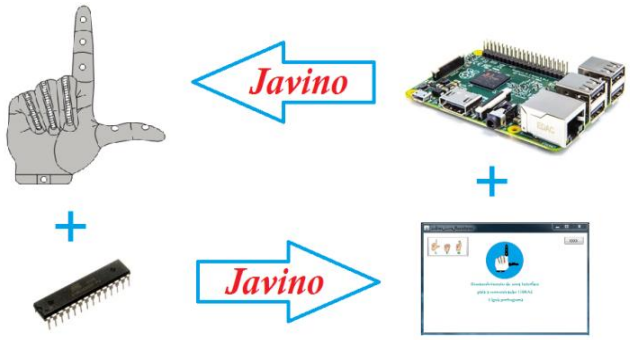

Figura 1. Fluxo de informação da arquitetura das luvas.

A Luva de LEDs é composta por 14 LED SMD, 4 LED RGB, 2 multiplexadores 74HC595, 1 arduino Lilypad, 1 cabo FTDI e 1 Raspberry. Os LED SMD estão posicionados nas juntas dos dedos indicando que se for acionado, a junta referente deverá ser flexionada. Três LED RGB estão alocados nos dedos e um no pulso. Aqueles alocados nos dedos têm duas possibilidades de acionamento: Verde, indica para juntar com o dedo mais próximo; Vermelho, informa para separar. No LED do pulso existem as cores Verde, Vermelho e Azul, que são responsáveis pela movimentação (descer, subir e girar) respectivamente. Os Multiplexadores são para aumentar a quantidade de portas digitais do arduino, utilizando 3 portas, pode-se controlar 8. O arduino é o controlador. O cabo FTDI é para conectar o Lilypad com a Raspberry, um minicomputador que proporcionou a interação com a interface gráfica. A luva LEDs poder ser vista na Figura 2.

Figura 2. O protótipo da luva de LEDs.

O fluxo de informações na luva de LED (Figura 1) ocorre em um processo em cadeia. Uma mensagem é digitada na interface gráfica, que passa para a biblioteca Javino realizar o transporte para o Arduino. Uma vez feito isso, o Arduino interpretará a mensagem, armazenará em um vetor de caracteres e passará por uma sequência de estruturas de decisão. Quando encontrar a correspondente, chamará a função de acionamento dos LED.

A Luva de resistores é constituída de 5 resistores flexíveis, 1 multiplexador CD4051, 1 acelerômetro, 1 Arduino Lilypad, 1 cabo FTDI e 1 Raspberry. Os sensores flexíveis estão alocados um em cada dedo, onde, através deles, pode-se entender o ato de flexionar um dedo. O multiplexador é para aumentar a quantidade de portas analógicas do Arduino, utilizando 3 portas digitais pode-se fazer a leitura de 8 portas analógicas. O acelerômetro é responsável pelas leituras dos eixos X, Y e Z, e identificar se a mão está em pé, deitada ou em movimentação. O Arduino e a Raspberry possuem a mesma função que na luva de LED.

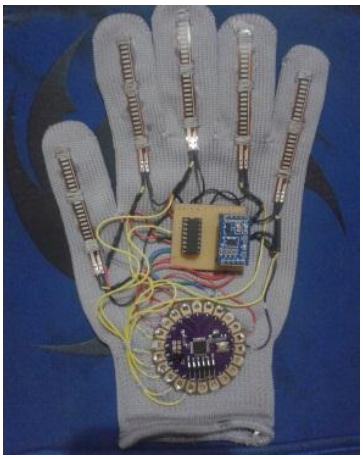

Figura 3. O protótipo da luva de resistores.

Para iniciar a leitura dos movimentos, precisa-se clicar no botão LER. Feito isso, o Javino enviará uma mensagem para o Arduino habilitar a leitura. A letra correspondente ao sinal em LIBRAS realizado aparecerá imediatamente na interface gráfica. A luva de resistores pode ser vista na Figura 3.

4. TRABALHOS RELACIONADOS

Nesta seção serão apresentados dois trabalhos com o foco na comunicação entre LIBRAS e língua portuguesa. O primeiro projeto é um bracelete que os traduz os movimentos dos braços em comandos de voz na língua portuguesa utilizando a comunicação *bluetooth*. Porém o trabalho realiza somente uma via da comunicação [6].

O segundo é um protótipo de uma luva equipada com acelerômetros nas pontas dos dedos que identificam os movimentos das mãos e os transformam em áudio. Assim como o primeiro trabalho, este somente realiza uma via da comunicação. [7]. A arquitetura proposta não deixa lacunas na troca de informações, pois opera nas duas vias de comunicação, tanto da linguagem falada para a gestual quanto da gestual para a falada.

5. CONCLUSÃO E TRABALHOS FUTUROS

O projeto partiu do pressuposto que os surdos não sabiam língua portuguesa e os ouvintes não sabiam LIBRAS. Foi utilizado na interface gráfica referente ao surdo LIBRAS ou símbolos intuitivos. Também, o projeto não necessita de nenhuma aquisição de dados externos, como internet, dando uma maior autonomia ao projeto. Além disso, o fato de ser móvel permite uma maior acessibilidade.

Os protótipos das luvas são os principais itens da arquitetura, pois elas estão em contato direto com o usuário. Já o Javino, realizou a interação hardware-software e também controlou o fluxo de informações, garantindo a integridade das mensagens. O programa armazenado no microcontrolador identifica a mensagem recebida e a utiliza para o acionamento dos LED referentes à entrada na interface gráfica da luva de LED. No caso da luva de resistores, o programa identifica a tensão lida, transforma em letras da língua portuguesa e envia para sua interface gráfica, com o Javino realizando a transmissão serial. Como trabalhos futuros, é necessário acrescentar o comando de voz.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] V. S. Jesus, Y. Silva, C. Eduardo, and L. M. Samyn. Desenvolvimento de uma interface para a comunicação LIBRAS - língua portuguesa. In Anais da V Mostra Nacional de Robótica (MNR 2015), 2015.

[2] N. M. Lazarin and C. E. Pantoja. A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. 9th Software Agents, Environments and Applications School, 2015.

[3] M. A. Oliveira. O que significa ser surdo? Conhecendo um pouco do que significa ser surdo através de discussão do filme seu nome é jonas. RVCSD - Revista Virtual de Cultura Surda e Diversidade, 2009.

[4] F. Pereira, Microcontroladores Pic – Progamação em C, 2003.

[5] M. McRoberts, Arduino Básico, 2015.

[6] Em http://www.surdosol.com.br/professor-da-uea-cria-equipamento-que-auxilia-na-comunicacao-de-surdos-em-amazonas/ acessado no dia 27 de setembro de 2016.

[7] Em http://g1.globo.com/mato-grosso/noticia/2015/10/jovens-criam-luva-que-decodifica-libras-e-transforma-em-audio-em-mt.html acessado no dia 27 de setembro de 2016.

Novo Escalonador para Rede LTE

Renê P. de Oliveira
UTFPR – Ponta Grossa
Av. Monteiro Lobato
018996086540
renepomilio@gmail.com

Lourival A. Góis
UTFPR – Ponta Grossa
Av. Monteiro Lobato
04232204800
gois@utfpr.edu.br

Augusto Foronda
UTFPR – Ponta Grossa
Av. Monteiro Lobato
04196383381
foronda@utfpr.edu.br

RESUMO
LTE é uma tecnologia celular desenvolvida para prover comunicações entre terminais com aplicações de voz, vídeo e dados com Qualidade de Serviço. A especificação do LTE deixa o desenvolvimento do escalonador, que é o responsável por selecionar o usuário que vai transmitir em um determinado instante, em aberto. Vários escalonadores existem na literatura, como RR, PF e MLWDF. Como nenhum escalonador garante o *delay* solicitado pelo usuário e maximiza o número de terminais no sistema, este trabalho propõe um novo escalonador com estes objetivos. Será demonstrada a ideia principal do novo modelo e alguns resultados analíticos do mesmo.

Palavras Chave
LTE, Escalonador, *Delay*.

ABSTRACT
LTE is a mobile technology developed to provide communication between terminals with voice, video and data applications with Quality of Service. The specification of LTE makes the development of the scheduler open. The scheduler is responsible for selecting the user that will transmit at a given moment and there are several schedulers in the literature, such as RR, PF and MLWDF. These schedulers cannot guarantees the delay requested by the user and maximizes the number of terminals in the system, and then this paper proposes a new scheduler with these goals. The main idea of the new model and some of the analytical results will be demonstrated.

Keywords
LTE, Escalonador, *Delay*.

1. INTRODUÇÃO
O crescimento do mercado de dispositivos móveis que usam pacotes de banda larga, fez com que as operadoras de telefonia móvel começassem a migrar para uma tecnologia mais eficiente de transmissão de pacotes de dados, fornecendo Qualidade de Serviço (QoS) para as aplicações dos usuários. O LTE (*Long Term Evolution),* mais conhecido como rede 4G, surgiu para suprir essa demanda de crescimento fazendo com que os terminais tivessem uma transmissão de dados com baixo *delay*, alto *throughput*, oportunidade de transmissão para todos os usuários e baixa perda de pacotes [1].

A tecnologia LTE tem como objetivo satisfazer as comunicações de Voz sobre IP (VoIP), redes sociais, *streaming* de vídeos em tempo real e transferência de arquivos para os terminais. Para isso, a arquitetura do LTE estabelece que a estação base, conhecida como *Evolved NodeB* (eNodeB), tenha um escalonador de pacotes para escolher qual terminar terá o direito de transmitir em determinado instante [2].

Alguns algoritmos escalonadores de pacotes são: *Round Robin* (RR), *Proportional Fair* (PF) e *Modified Largest Weighted Delay First* (M-LWDF). Estes escalonadores funcionam na camada *Medium Access Control* (MAC) e cada um tem uma característica diferente. Mas nenhum deles provê o atraso solicitado pelo usuário e ao mesmo tempo maximiza o número de terminais no sistema [3].

O objetivo deste trabalho é o desenvolvimento de um novo escalonador de pacotes com base no escalonador PF e escalonador *latency rate* (LR) [6], com o objetivo de garantir o atraso solicitado pelo usuário e maximizar o número de usuários no sistema.

2. ASPECTOS DO LTE
2.1 Topologia da Rede LTE
A topologia da rede LTE desenvolvida pelo 3GPP (*Third Generation Partnership Project*) é composta pelas E-UTRAN (*Evolved Universal Terrestrial Rádio Access Network*) [4] que contém a estação base eNodeB e é responsável por gerenciar toda a comunicação com os terminais UEs, como pode ser observado na Figura 1.

A eNodeB é responsável pelo implementação e execução dos escalonadores de pacotes. Os escalonadores RR, PF, M-LWDF serão considerados neste trabalho e usados para comparação com o novo escalonador proposto. Os resultados de *delay* e *throughput* serão usados para analisar o desempenho do novo modelo.

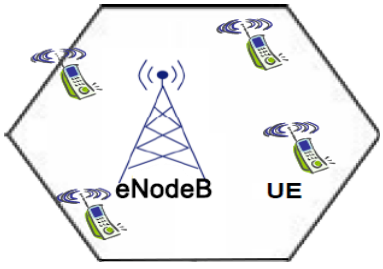

Figura 1. Topologia rede LTE

WPCCG'16, Setembro 28, 2016, Ponta Grossa, Paraná, Brasil.

2.2 Round Robin

O RR define um intervalo de tempo para cada processo, mantendo o mesmo intervalo constante de transmissão para cada usuário no sistema. Embora isso faça com que o eNodeB não tenha esforço nenhum para definir qual usuário vai ser selecionado, a desvantagem é que este escalonador não considera o estado do enlace nem o atraso requisitado pelo usuário [7].

2.3 Proportional Fair

O PF seleciona o usuário com a melhor taxa de dados instantânea com relação a sua taxa média de dados, requisitando um esforço maior do eNodeB para informar os UEs sobre suas posições de *slot*. Este escalonador tenta alocar os usuários de maneira mais justa considerando o estado do enlace, conforme a equação (1) abaixo. Mas também não considera o atraso requisitado pelo usuário [8].

$$\frac{r_j(t)}{\mathrm{R}_j(t)} \quad (1),$$

onde $r_j(t)$ é a taxa do canal de usuário j e $R_j(t)$ é a taxa média do canal do usuário j.

2.4 Modified Largest Weighted Delay First

O MLWDF é utilizado para suportar vários usuários com diferentes requisitos de QoS, priorizando determinado usuário com maior atraso de pacotes e com melhores condições de canal com relação à média, conforme equação (2). Embora este escalonador considere o *delay* para a seleção dos usuários, não é o atraso especificado pelo usuário e também não maximiza o número de terminais no sistema [5].

$$w_{(i)} = -\frac{log\delta_i}{\tau_i} W_t \frac{r_{(i)}}{\mathrm{R}_{(i)}} \quad (2),$$

onde W_t representa o primeiro pacote que será transmitido na fila a exceder o limite de atraso, τ_i é o limite de atraso do pacote e δ_i é a probabilidade máxima de atraso do pacote.

3. DESCRIÇÃO DO NOVO MODELO

A Figura 2 ilustra uma rede LTE com a proposta do novo escalonador, que é baseado em um escalonador LR modificado e utiliza o algoritmo do balde de fichas. A abordagem básica consistirá que o balde de fichas formata o tráfego de entrada e o escalonador seleciona uma determinada UE para transmitir em cada *time slot*.

O balde de fichas vai determinar seus parâmetros (tamanho e taxa) a partir dos dados do tráfego de entrada. Com os parâmetros do balde, mais a taxa física do meio e o *delay* solicitado pelo usuário vai ser calculado um valor para cada usuário usando uma função. Com isso é possível determinar o número máximo de usuários para satisfazer um determinado *delay* [6].

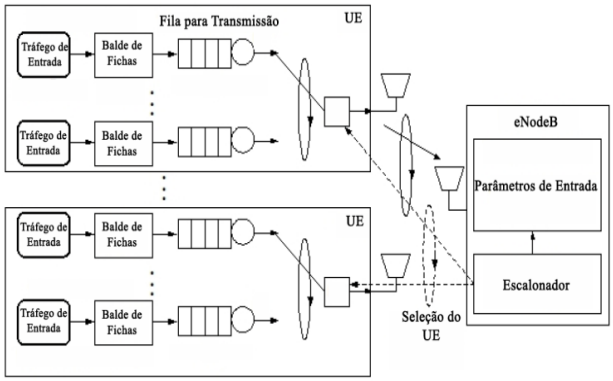

Figura 2. Rede LTE com novo escalonador

4. RESULTADOS ANALÍTICOS

Esta seção demonstra alguns resultados analíticos preliminares do novo modelo. A Figura 3 mostra o tráfego de entrada usado na simulação.

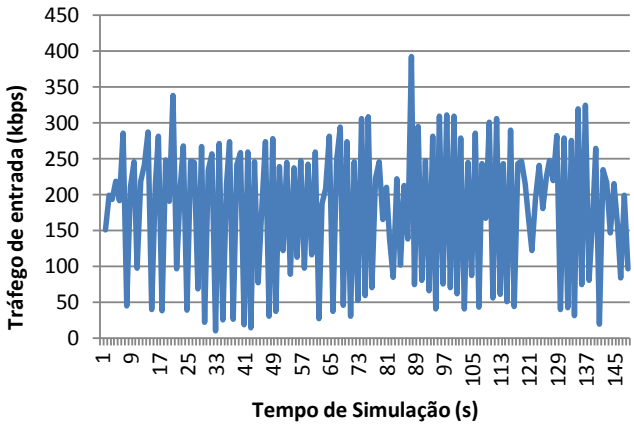

Figura 3. Tráfego de Entrada.

A partir desse trafego de entrada foram calculados os valores do balde de fichas que podem ser vistos na tabela 1.

Tabela 1. Parâmetros do Balde de Fichas

Tamanho do Balde de Fichas (bits)	Taxa do Balde de Fichas (kbps)
15000	400

Com os parâmetros do balde de fichas, mais a taxa física do meio e o atraso solicitado pelo usuário, com o novo escalonador pode-se determinar a taxa do usuário para atender o atraso e a quantidade de usuários no sistema, como pode ser visto na Tabela 2.

Tabela 2. Parâmetros de Entrada e Saída

Parâmetros de Entrada	
Taxa Física do Meio (bps)	36000000
Delay (s)	0,2
Tamanho do Balde de Fichas (bits)	15000
Taxa do Balde de Fichas (bps)	400000
Parâmetros de Saída	
Taxa do Usuário (bps)	400000
Número de Usuários	80

Com a taxa do usuário, será desenvolvida uma nova função baseado na equação (1), que vai ser criada para selecionar o usuário que pode transmitir em cada *time slot*.

5. AGRADECIMENTOS

Agradeço a CAPES pelo apoio financeiro na concessão da bolsa de estudos.

6. CONCLUSÃO

Futuros testes que serão realizados poderão comprovar que o escalonador proposto, a partir da implementação do algoritmo no simulador poderá suprir a necessidade do usuário garantindo o atraso solicitado e maximizando o número de usuários no sistema. Até o momento foi desenvolvido o modelo analítico para o LTE, aplicado o tráfego de entrada e obtido os parâmetros de saída, para serem testados após a implementação do modelo proposto no simulador.

7. REFERÊNCIAS

[1] Luiz P, Luiz V, Juan Rogriguez, Pedro Capelastegui, Guilem, Hernández-Sola, Lorena Calavia, Antonio Marqués, Borja Iribarne, Amandor Pozo and Antoine De Poorter, "Network Convergence and QoS for future muiltimidea services in the VISION project", Computer Networks 56, 1183-1199, DOI:10.1016/j.comnet.2011.11.018, 2012.

[2] D.McQueen, "The momentum behind LTE adoption," *IEEE Communication Magazine*, vol. 47, no. 2, pp. 44-45. Feb. 2009.

[3] Erik Dhlman, Stefan Parkvall and Joahn Sköld, "4G LTE/LTE-Advanced for Mobile Broadband", Copyright © 2011.

[4] 3GPP, *Tech. Specif. Group Radio Access Network Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN), 3GPP TS* 25.913.

[5] R. Basukala, H. M.Ramli, and K. Sandrasegaran, "Performace analysis of EXP/PF and M-LWDF in downlink 3GPP LTE system," in *Proc. Of First Asian Himalayas Int. Conf. on Internet. AH-ICI*, Kathmandu, Nepal, Nov. 2009.

[6] D. Stiliadis and A. Varma. *Latency-Rate Servers*: *A General Model for Analysis of Traffic Scheduling Algorithms*. In IEEE-ACM Transactions on Networlink, v. 6, pages 611-624, Oct 1998.

[7] Claude Simon and Geert Leas "Round-Robin scheduling for timeva ying channel with limited feedback", IEEE 10th workshop on signal processing advances in wireless communication, 2009, ISBN-978-1-4-4244-3695-8, DOI: 10.1109/SPAWC.2009.5161781, Pp(s): 231-234.

[8] H. Seo and B. G. Lee, "A proportional-fair power allocation scheme for fair and efficient multiuser OFDM systems," in Proc. of IEEE GLOBECOM, Dallas, TX, USA, Dec. 2004.

Diversidade dos conjuntos independentes maximais em alguns grafos não-simpliciais

Aleffer Rocha
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato, s/n - Km 04 CEP
84016-210 - Ponta Grossa - PR - Brasil
aleffer@alunos.utfpr.edu.br

Sheila Morais de Almeida
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato, s/n - Km 04 CEP
84016-210 - Ponta Grossa - PR - Brasil
sheilaalmeida@utfpr.edu.br

RESUMO

Seja $\mathcal{F}$ uma família de conjuntos. A diversidade de $\mathcal{F}$, $\Upsilon(\mathcal{F})$, é a quantidade de cardinalidades diferentes dos conjuntos contidos em $\mathcal{F}$. Denota-se por $M(\Upsilon(\mathcal{F}))$ a classe dos grafos cuja diversidade dos conjuntos independentes maximais é $\Upsilon(\mathcal{F})$. Reconhecer os grafos da classe $M(t)$ para um dado t é um problema Co-NP-completo. Entretanto, sabe-se que é possível reconhecer eficientemente os grafos que pertencem a $M(1)$ e os grafos simpliciais que pertencem a $M(2)$. Nesse trabalho apresentamos classes de grafos não-simpliciais que pertencem a $M(2)$ e que podem ser reconhecidas eficientemente.

Palavras-chave

conjunto independente maximal; diversidade; prisma complementar

ABSTRACT

Let $\mathcal{F}$ be a family of sets. The diversity of $\mathcal{F}$, $\Upsilon(\mathcal{F})$, is the amount of different cardinalities on the sets of $\mathcal{F}$. The class of graphs whose diversity of the maximal independent set equals $\Upsilon(\mathcal{F})$ is denoted by $M(\Upsilon(\mathcal{F}))$. Recognizing the $M(t)$ class for a given t is a Co-NP-complete problem. However, it is possible to recognize the $M(1)$ graphs and the simplicial graphs in $M(2)$ efficiently. In this work we present non-simplicial graphs belonging to $M(2)$ that can be recognized by polynomial algorithms.

Keywords

maximal independent set; diversity; complementary prism

1. INTRODUÇÃO

Nesse trabalho, os grafos considerados são simples, ou seja, não possuem laços, nem arestas múltiplas. Um conjunto independente em um grafo G é um conjunto I tal que os vértices contidos em I não são adjacentes em G. Um conjunto independente de G é maximal se não está propriamente contido em outro conjunto independente de G. Esse trabalho aborda o problema de se determinar a quantidade cardinalidades distintas de conjuntos independentes maximais em um grafo G.

Seja $\mathcal{F}$ uma família de conjuntos. Definimos a diversidade de $\mathcal{F}$, $\Upsilon(\mathcal{F})$, como sendo a quantidade de cardinalidades diferentes dos conjuntos contidos em $\mathcal{F}$. Por exemplo, se $\mathcal{F} = \{\{1,4,5\},\{1,5,6\},\{1,4,7,9\},\{1,2,3,4\},\{2,3,4,5,7\}\}$, então a diversidade de $\mathcal{F}$ é $\Upsilon(\mathcal{F}) = 3$, já que existem conjuntos com três cardinalidades distintas: conjuntos de tamanhos 3, 4 e 5.

Denota-se por $M(t)$ a classe dos grafos cuja diversidade dos conjuntos independentes maximais é t. Reconhecer os grafos da classe $M(t)$ para um dado t é um problema Co-NP-completo [2]. Mesmo quando $t = 1$, determinar se um grafo pertence a $M(1)$ é um problema NP-difícil [3]. Entretanto, quando se restringe o problema a algumas classes de grafos específicas, existem algoritmos polinomiais para sua solução. Por exemplo, é possível determinar eficientemente quais os grafos simpliciais que pertencem a $M(2)$ [1]. Um grafo G é simplicial, se cada vértice v do grafo G satisfaz uma das seguintes condições: 1) todos os vizinhos de v são adjacentes entre si (nesse caso dizemos que v é simplicial); ou 2) v não é simplicial, mas tem um vizinho que é simplicial. Nesse trabalho apresentamos classes de grafos que não são simpliciais e cuja diversidade de seus conjuntos independentes maximais pode ser determinada eficientemente.

Uma dessas classes é a dos grafos k-partidos completos com pelo menos três vértices. Um grafo é k-partido se, e somente se, seu conjunto de vértices pode ser particionado em k conjuntos independentes disjuntos. Considere um grafo k-partido com partição $[P_0, P_1, \ldots, P_{k-1}]$. Se cada vértice da parte P_i for adjacente a todos os vértices das partes P_j, quando $i \neq j$, o grafo k-partido é completo.

Outras classes para as quais apresentamos o número de diversidade do conjunto de conjuntos independentes maximais são os prismas complementares de grafos k-partidos completos e prismas complementares de grafos split completos. O prisma complementar de um grafo G, denotado por $G\overline{G}$, é a união disjunta dos grafos G e $\overline{G}$ com a inclusão de uma aresta entre cada par de vértices correspondentes de G e $\overline{G}$, onde $\overline{G}$ é o complemento do grafo G[1]. Um grafo é split se seu conjunto de vértices pode ser particionado em uma clique e um conjunto independente, onde a clique é um conjunto de vértices dois a dois adjacentes. Um grafo split

[1]O conceito de complemento de um grafo é definido na Seção 2.

é completo se cada vértice da clique é adjacente a todos os vértices do conjunto independente.

Grafos split completos são simpliciais, uma vez que a vizinhança de qualquer vértice do conjunto independente constitui uma clique e, portanto, todo vértice do conjunto independente é simplicial e todo vértice da clique é vizinho de um vértice simplicial. Além disso, para um grafo split completo $G = [Q, S]$, onde Q é a clique e S é o conjunto independente, quando $|S| > 1$, há somente dois tamanhos de conjuntos independentes maximais: $|S|$ e 1, uma vez que cada vértice da clique é um conjunto independente máximal em G. Por outro lado, o prisma complementar de um grafo split completo não é um grafo simplicial e nesse trabalho mostramos que tais grafos pertencem a $M(2)$.

2. DEFINIÇÕES BÁSICAS E RESULTADOS ANTERIORES

Um grafo $G = (V, E)$ é dito vazio se o mesmo possui um conjunto $V = \{v_0, v_1, \ldots, v_{n-1}\}$ de vértices e um conjunto $E = \emptyset$ de arestas.

Como já definido na introdução, um grafo é k-partido completo se é um k-partido cujos vértices de partes distintas são necessariamente adjacentes. Nesse trabalho, o grafo k-partido completo com partes de tamanhos n_0, n_1, ..., n_{k-1} é denotado por $K_{[n_0;n_{k-1}]}$. Um grafo $G = [X, Y]$ é bipartido se seus vértices podem ser particionados em dois conjuntos independentes distintos, X e Y. Observe que um grafo bipartido é um grafo k-partido com $k = 2$. Um grafo bipartido completo, $K_{m,n}$, é um grafo bipartido $G = [X, Y]$, com $|X| = m$ e $|Y| = n$, tal que cada vértice em X é adjacente a todos os vértices de Y. Um grafo completo K_n é um grafo com n vértices dois a dois adjacentes.

O complemento de um grafo $G = (V, E)$ é o grafo $\overline{G} = (V, \overline{E})$, composto por todos os vértices de G, os quais são adjacentes em $\overline{G}$ se e somente se não são adjacentes em G.

Seja G um grafo com $V(G) = \{v_0, v_1, \ldots, v_{n-1}\}$ e $\overline{G}$ seu complemento com $V(\overline{G}) = \{u_0, u_1, \ldots, u_{n-1}\}$, onde u_i é o vértice de $\overline{G}$ correspondente a v_i em G. O prisma complementar de G é o grafo $G\overline{G} = (V(G) \cup V(\overline{G}), E(G) \cup E(\overline{G}) \cup M)$, tal que $M = \{(v_i, u_i), 0 \leq i < n\}$.

Como já mencionado, um vértice é simplicial se sua vizinhança é uma clique. Um simplex é um subgrafo de G induzido por um vértice simplicial e seus vizinhos. Em [1] Barbosa e Hartnell provam o seguinte teorema.

TEOREMA 1. *Um grafo simplicial G pertence a $M(2)$ se e somente se os vértices de G podem ser particionados em conjuntos não-vazios A e Q tal que os vértices em A pertençam a exatamente um simplex e aqueles em Q a exatamente $k > 1$ simplexos onde os vértices de Q formam uma clique.*

3. RESULTADOS

Os resultados estão divididos em duas partes: determinação da diversidade de conjuntos independentes maximais em grafos k-partidos completos e em prismas complementares de k-partidos completos e split completos.

3.1 Grafos k-partidos completos

Com o objetivo de facilitar a compreensão da prova para o caso dos grafos k-partidos completos, o Lema 1 apresenta primeiro o caso em que $k = 2$. A seguir, generalizamos tal resultado para o caso em que $k > 2$.

LEMA 1. *O grafo $K_{m,n} \in M(2)$ se, e somente se, $m \neq n$, caso contrário, $K_{m,n} \in M(1)$.*

PROVA. Sejam $[X, Y]$ a partição de $K_{m,n}$ tal que $|X| = m$ e $|Y| = n$, e I um conjunto independente maximal do $K_{m,n}$. Considere que $I \cap X \neq \emptyset$. Então existe um vértice v em $I \cap X$ e nenhum vértice de Y pertence a I, pois são todos adjacentes a v. Como os vértices de X são um conjunto independente e I é maximal, $X = I$. Logo, $|I| = m$ quando $v \in X$. De mesmo modo podemos concluir que existe um outro conjunto independente maximal $I' = Y$ com tamanho n. Como todo vértice de X é adjacente a todo vértice de Y, então não existe um conjunto independente com vértices de ambos os conjuntos. Portanto I e I' são os únicos conjuntos independentes maximais em $K_{m,n}$. Conclui-se que, quando $m \neq n$, o grafo $K_{m,n} \in M(2)$, caso contrário, $K_{m,n} \in M(1)$. □

TEOREMA 2. *Se G é um grafo k-partido completo com partição $\mathcal{F} = [A_0, A_1, A_2, \ldots, A_{k-1}]$, então $G \in M(\Upsilon(\mathcal{F}))$.*

PROVA. Seja G um grafo k-partido completo com partição $\mathcal{F} = [A_0, A_1, A_2, \ldots, A_{k-1}]$. Considere um vértice $v \in A_i$ e seja I um conjunto independente maximal que contém v. Como v é adjacente a $V(G) \setminus A_i$, S só pode conter vértices de A_i. Como A_i é um conjunto independente e I é maximal, $I = A_i$. Então cada vértice pertence a um único conjunto independente maximal. Então o conjunto de conjuntos independentes maximais distintos em G é exatamente $\mathcal{F}$. Portanto o número de conjuntos independentes maximais de tamanhos diferentes é $\Upsilon(\mathcal{F})$. □

3.2 Prismas complementares

Da mesma forma, esta seção apresenta primeiro a diversidade dos conjuntos independentes maximais de prismas complementares dos grafos bipartidos completos. Em seguida, o resultado é generalizado para grafos k-partidos completos com $k > 2$. Para apresentação da prova, é útil a seguinte observação.

OBSERVAÇÃO 1. $K_n \in M(1)$.

TEOREMA 3. *O grafo $K_{m,n}\overline{K_{m,n}} \in M(2)$ se, e somente se, $m \neq n$, caso contrário, $K_{m,n}\overline{K_{m,n}} \in M(1)$.*

PROVA. Seja $G = K_{m,n}\overline{K_{m,n}}$ o prisma complementar de $K_{m,n}$, tal que $m \neq n$. O grafo G pode ser decomposto em três subgrafos disjuntos nas arestas: $K_{m,n}$, $\overline{K_{m,n}}$ e o subgrafo bipartido $B = [V(K_{m,n}), V(\overline{K_{m,n}})]$. Sejam $[X, Y]$ uma partição em dois conjuntos independentes dos vértices do subgrafo de G isomorfo ao $K_{m,n}$ e $[X', Y']$ uma partição em duas cliques dos vértices do subgrafo de G isomorfo ao $\overline{K_{m,n}}$. Pelo Lema 1, as partes X e Y são conjuntos independentes maximais do $K_{m,n}$. Então, X é um conjunto independente de G, mas não é maximal já que X não é adjacente a nenhum vértice de Y'. Como Y' induz um grafo completo K_n, pela Observação 1, no máximo um vértice de Y' pode fazer parte de qualquer conjunto independente maximal de G. Seja u um vértice de Y'. Então, $I = X \cup \{u\}$ é um conjunto independente de G e é maximal já que todos os vértices em $X' \cup Y$ são adjacentes a algum vértice de X e todo vértice de Y' é adjacente a u. Outros conjuntos independentes maximais podem ser obtidos removendo-se um vértice $v \in X$ do conjunto I e inserindo seu correspondente do conjunto X'. Observe que, como X' induz um grafo

completo K_m, apenas 1 vértice de X' pode pertencer a I, pela Observação 1. Como a substituição de um vértice de X por um vértice de X' não altera o tamanho do conjunto independente maximal, ainda temos apenas conjuntos independentes maximais de tamanho $|X|+1$.

De forma análoga, Y é um conjunto independente de G, mas não é maximal já que vértices de Y não são adjacentes a vértices de X'. Como X' é uma clique, apenas um vértice de X' pode pertencer a qualquer conjunto independente maximal de G. Como cada vértice de $X \cup Y'$ é adjacente a algum vértice de Y, $I = Y \cup \{u\}$ tal que $u \in X'$ é um conjunto independente maximal. Então todo conjunto independente maximal de G têm tamanho $|Y|+1$. Outros conjuntos independentes maximais podem ser obtidos substituindo-se um vértice de Y pelo seu correspondente em Y', mas como apenas um vértice de Y' pode pertencer ao conjunto independente maximal, todos esses conjuntos independentes maximais têm tamanho $|Y|+1$. Portanto, todo conjunto independente maximal tem tamanho $m+1$ ou $n+1$ e $G \in M(2)$ se $m \neq n$ ou, caso contrário, $G \in M(1)$. □

TEOREMA 4. *Seja G o prisma complementar de $K_{[n_0;n_{k-1}]}$. Então $G \in M(\Upsilon(\mathcal{F}))$, onde $\mathcal{F} = \{n_0, n_1, \ldots, n_{k-1}\}$.*

PROVA. Considere um grafo k-partido completo $K_{[n_0;n_{k-1}]}$ e seja G o seu prisma complementar. Os vértices de G podem ser particionados em cliques Q_i e conjuntos independentes P_i, $0 \leq i < k$, onde cada parte P_i é um conjunto independente maximal do $K_{[n_0;n_{k-1}]}$, para todo $0 \leq i < k$. Além disso, a cada P_i corresponde uma clique Q_i e cada vértice $u_{i,j} \in Q_i$ é adjacente a $v_{i,j} \in P_i$, $0 \leq i < k$.

Seja I um conjunto independente de G que contém P_i. Então $Q_i \cap I = \emptyset$. Por outro lado, qualquer vértice $u_{t,j} \in Q_t$, com $t \neq i$, pode pertencer a I e se G não contém nenhum desses vértices, não é maximal. Então seja $u_{t,j} \in Q_t$, $t \neq i$ um vértice em I. Nenhum outro vértice de Q_t pode pertencer a I. Adotando o mesmo raciocínio, para toda clique $Q_z \neq Q_i$, tem-se $|Q_z \cap I| = 1$ quando I é maximal. Então os vértices de uma parte P_i qualquer unidos com um vértice de cada clique Q_z, $z \neq i$, são um conjunto independente maximal. Portanto, nesses casos, $|I| = |P_i| + k - 1$.

Existem três formas de criar outros conjuntos independentes maximais:

1. substituir exatamente um vértice de P_i pelo seu correspondente em Q_i;
2. substituir o vértice $u_{t,j}$ de $Q_t \cap I$ por outro vértice $u_{t,y}$, $y \neq j$;
3. realizar o mesmo processo de construção de I a partir de outra parte de $K_{[n_0;n_{k-1}]}$ que não seja P_i.

Observe que as duas primeiras alterações criam novos conjuntos independentes maximais com o mesmo tamanho de I, apenas substituindo um vértice por outro. Por outro lado, a terceira alteração pode criar conjuntos independentes maximais com cardinalidades diferentes de $|I|$, desde que o tamanho da parte escolhida seja diferente do tamanho de P_i.

Portanto, a quantidade de conjuntos independentes maximais de tamanhos diferentes em G é igual ao número de tamanhos diferentes das partes do $K_{[n_0;n_{k-1}]}$, ou seja, $G \in M(\Upsilon([n_0; n_{k-1}])$. □

TEOREMA 5. *Se $G\overline{G}$ é prisma complementar de um grafo split completo $G = [Q, S]$, então $G\overline{G} \in M(2)$ quando $|S| > 1$ e $G\overline{G} \in M(1)$, caso contrário.*

PROVA. Seja $G\overline{G}$ o prisma complementar de um grafo split completo, G, particionado em 4 subconjuntos de vértices: Q, uma clique de G; S, um conjunto independente de G; Q' e S', cujos vértices pertencem ao subgrafo $\overline{G}$ e correspondem, respectivamente, aos vértices de Q e S.

Observe que S e Q' são conjuntos independentes em $G\overline{G}$ e não existe aresta entre vértices de S e Q'. Então, $I = S \cup Q'$ é um conjunto independente de $G\overline{G}$ de tamanho $|S| + |Q|$. Tal conjunto independente é maximal, já que todo vértice de S' é adjacente a algum vértice de S e todo vértice de Q é adjacente a algum vértice de Q'.

Agora, resta considerar os casos em que algum vértice de Q ou S' pertence a um conjunto independente maximal I em $G\overline{G}$. Primeiro, suponha que um vértice $v \in Q$ pertence a I. Então nenhum outro vértice de Q ou S pertence a I. Então, todos os vértices de Q', exceto o correspondente a v, pertencem a I. Como nenhum vértice de S pertence a I, algum vértice de S' deve pertencer a I e apenas um, já que S' é uma clique. Portanto, I é um conjunto independente maximal de tamanho $|Q| + 1$.

O último caso é quando um vértice $v \in S'$ está no conjunto independente maximal I. Como S' é uma clique, nenhum outro vértice de S' pertence a I. Há dois subcasos: 1) algum vértice de S pertence a I, ou 2) nenhum vértice de S pertence a I.

Subcaso 1: se algum vértice de S não correspondente a v pertence a I, então, nenhum vértice de Q pertence a I, todo vértice de Q' pertence a I, e todo vértice de S não correspondente a v pertence a I. Nesse caso, $|I| = |S| + |Q'| = |Q| + |S|$.

Subcaso 2: se nenhum vértice de S pertence a I. Então, ou algum vértice de Q pertence a I ou todos os vértices de Q' pertencem a I. Se algum vértice $u \in Q$ está em I, todos os vértices de Q' menos o correspondente a u pertencem a I. Em ambos os casos $|I| = |Q| + |S|$.

Portanto, $G\overline{G}$ tem dois possíveis tamanhos de conjuntos independentes maximais: $|Q| + |S|$ e $|Q| + 1$. Logo, quando $|S| > 1$, $G\overline{G} \in M(2)$ e, caso contrário, $G\overline{G} \in M(1)$. □

4. CONCLUSÃO

Nesse trabalho, pela primeira vez foi definido o conceito de diversidade de um conjunto de conjuntos. Além disso, determinamos a diversidade dos conjuntos independentes maximais das seguintes classes de grafos não-simpliciais: k-partidos completos de ordem pelo menos três, prismas complementares de k-partidos completos e de split completos. O estudo da diversidade de conjuntos independentes maximais em prismas complementares não-simpliciais se mostrou um campo fértil para o desenvolvimento de pesquisa. Entretanto acreditamos que classes não definidas a partir do conceito de partição em cliques e/ou conjuntos independentes sejam muito mais difíceis de se trabalhar. Para trabalhos futuros é interessante considerar prismas complementares de grafos bipartidos e splits não completos.

5. REFERÊNCIAS

[1] R. Barbosa and B. Hartnell. Some problems based on the relative sizes of the maximal independent sets in a graph. *Congressus Numerantium*, 131(5):115–121, 1998.

[2] Y. Caro, A. Sebő, and M. Tarsi. Recognizing greedy structures. *Journal of Algorithms*, 20(1):137–156, 1996.

[3] D. Tankus and M. Tarsi. The structure of well-covered graphs and the complexity of their recognition problems. *Journal of Combinatorial Theory, Series B*, 69(2):230–233, 1997.

Propriedades do Conjunto Dominante Mínimo no Produto Lexicográfico

Filipe Rodrigues Pereira da Silva
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato km 04
Ponta Grossa, Paraná
filipesilva@alunos.utfpr.edu.br

Sheila Morais de Almeida
Universidade Tecnológica Federal do Paraná
Av. Monteiro Lobato km 04
Ponta Grossa, Paraná
sheilaalmeida@utfpr.edu.br

RESUMO

O Problema do Conjunto Dominante Mínimo é encontrar o menor conjunto de vértices D tal que todo vértice pertença a D ou seja vizinho de um vértice de D. Sabe-se que o Problema do Conjunto Dominante Mínimo é NP-completo. Neste trabalho investigamos o Problema do Conjunto Dominante Mínimo no produto lexicográfico de grafos. Devido a uma famosa conjectura de V. G. Vizing, este problema tem sido abordado com mais frequência no produto cartesiano de grafos, havendo poucos resultados em relação ao produto lexicográfico. Determinamos o número de dominação mínimo para alguns grafos resultantes do produto lexicográfico e apresentamos um limite superior justo para o número de dominação de qualquer produto lexicográfico.

Palavras chave

Conjunto Dominante; Produto Lexicográfico; Grafos k-partidos completos

ABSTRACT

The Minimum Dominating Set Problem is to find the least vertex set D such that every vertex belongs to D or it is adjacent to a vertex in D. It is known that the Minimum Dominating Set Problem is NP-complete. In this work we investigate the Minimum Dominating Set Problem on lexicographic product of graphs. Due to a famous conjecture of V. G. Vinzing, this problem has been more frequently studied on cartesian products of graphs and few results are related to the lexicographical product. We determine the domination number for some lexicographical product graphs and we present an upper bound for the domination number of any lexicographical product graph.

Keywords

Dominating set; Lexicographical Product; Complete k-partite Graphs

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

1. INTRODUÇÃO

Seja $G = (V(G), E(G))$, um grafo G com conjunto de vértices $V(G)$ e conjunto de arestas $E(G)$, formado por pares não ordenados de G. Nesse trabalho, G é um grafo simples e não orientado. O número de vértices de G, ou ordem G, é denotado por $|V(G)| = n$, e o número de arestas de G é denotado por $|E(G)| = m$. Cada aresta do conjunto $E(G)$, constituída pelos vértices v e w, é denotada por vw e dizemos que v e w são adjacentes. A vizinhança aberta de um vértice $v \in V(G)$ consiste no conjunto de vértices adjacentes a v e é denotada por $N(v) = \{w \in V(G) : vw \in E(G)\}$. A vizinhança fechada de v consiste no conjunto de vértices adjacentes a v inclusive v e é denotada por $N[v] = N(v) \cup \{v\}$. Um vértice v em um grafo G é universal se é vizinho de todos os outros vértices de G, ou seja, $|N[v]| = |V(G)|$.

Este trabalho aborda o problema de determinar o tamanho do conjunto dominante mínimo em grafos resultantes do produto lexicográfico. Dado um grafo qualquer, G, um conjunto de vértices $D \subseteq V(G)$ é chamado de conjunto dominante de G se todo vértice de G pertence a D ou é adjacente a um vértice de D. Quando um vértice v pertence a um conjunto dominante D ou é adjacente a um vértice de D, diz-se que v é dominado por D ou que D domina v. A cardinalidade mínima de um conjunto dominante $D \in V(G)$ é chamada de número de dominação e é denotada por $\gamma(G)$. O Problema do Conjunto Dominante Mínimo consiste em encontrar o número de dominação de um dado grafo G. O Problema do Conjunto Dominante Mínimo é um problema clássico de decisão NP-completo [4]. O produto lexicográfico de dois grafos G e H é o grafo $G \bullet H$ cujo conjunto de vértices é formado pelo produto dos conjuntos $V(G) = \{x_1, \ldots, x_n\}$ e $V(H) = \{y_1, y_2, y_3, \ldots, y_k\}$ e dois vértices (x_1, y_1) e (x_2, y_2) são adjacentes em $G \bullet H$ se $x_1x_2 \in E(G)$, ou se $x_1 = x_2$ e $y_1y_2 \in E(H)$.

Diversos resultados parciais foram publicados considerando o Problema do Conjunto Dominante Mínimo em produtos de grafos. Hartnell e Rall [2] mostraram a relação do número de dominação entre os produtos lexicográfico, categórico, forte e cartesiano. Em 1968 V. G. Vizing [6] propôs uma conjectura que determina um limite inferior para o conjunto dominante mínimo do produto cartesiano entre dois grafos. Em [1] são apresentados resultados parciais da conjectura de Vizing, considerando os grafos *claw-free* e classe A. Thanin [5] apresenta o número de dominação para o grafo resultante do produto lexicográfico entre um ciclo e um grafo qualquer. Zhang [7] apresenta o número de dominação do produto lexicográfico entre G e H em que G é um grafo simples e H um grafo com vértice universal. Neste trabalho, nós

apresentamos o número de dominação do grafo resultante do produto lexicográfico $G \bullet H$, em que G é um grafo com vértice universal e H é um grafo simples qualquer. Como o produto lexicográfico não é uma operação comutativa, esse novo resultado junto com aquele apresentado em Zhang [7] garante que é possível determinar eficientemente o número de dominação de qualquer grafo resultante do produto lexicográfico $G \bullet H$ quando G ou H tem vértice universal. Além disso, determinamos o número de dominação do proguto lexicográfico de um grafo k-partido completo por um grafo simples qualquer. Nesse caso, em particular, ambos os grafos podem não ter vértice universal. Por fim, apresentamos um limite superior justo para o número de dominação de produtos lexicográficos de grafos simples e conexos.

2. CONJUNTO DOMINANTE

Em meados de 1850, na Europa, entusiastas do xadrez propuseram o problema de determinar o número mínimo de rainhas que podem ser posicionadas em um tabuleiro convencional de xadrez (8×8) de modo que todas as casas (quadrados) são atacadas por uma rainha ou são ocupadas por uma rainha. A Figura 1 mostra cinco rainhas posicionadas de forma que todas as casas sejam atacadas ou ocupadas. Sabe-se que este é o número mínimo necessário de rainhas para dominar o tabuleiro [3].

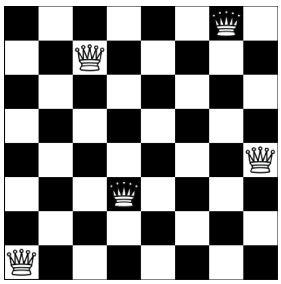

Figura 1: Uma possível solução do problema das cinco rainhas.

Tal problema pode ser modelado como o Problema do Conjunto Dominante Mínimo. Considere que cada casa do tabuleiro é um vértice de um grafo G e existe aresta entre dois vértices v_i e v_j se, e somente se, é possível deslocar a rainha entre as casas correspondentes a v_i e v_j com um único movimento. Se o conjunto de casas onde as rainhas estão posicionadas é o conjunto dominante mínimo do tabuleiro, então não existe nenhuma casa que não seja dominada por alguém do conjunto dominante ou que não faça parte do conjunto dominante.

2.1 Propriedades do produto lexicográfico

Como apresentado na introdução, o produto lexicográfico de dois grafos G e H é o grafo $G \bullet H$ cujo conjunto de vértices é formado pelo produto dos conjuntos $V(G) = \{x_1, \ldots, x_n\}$ e $V(H) = \{y_1, y_2, y_3, \ldots, y_k\}$ e dois vértices (x_1, y_1) e (x_2, y_2) são adjacentes em $G \bullet H$ se $x_1x_2 \in E(G)$, ou se $x_1 = x_2$ e $y_1y_2 \in E(H)$. Por exemplo, seja $P_3 = (V(P_3), E(P_3))$, com $V(P_3) = \{v_1, v_2, v_3\}$ e $E(P_3) = \{v_1v_2, v_2v_3\}$; e seja $C_3 = (V(C_3), E(C_3))$, com $V(C_3) = \{u_1, u_2, u_3\}$ e $E(C_3) = \{u_1u_2, u_2u_3, u_1u_3\}$. A Figura 2 mostra o produto lexicográfico $P_3 \bullet C_3$. As linhas contínuas mais finas representam arestas entre vértices do conjunto $\{v_i\} \times V(C_3)$. Observe que, para cada $v_i \in V(P_3)$, o conjunto $\{v_i\} \times V(C_3)$ induz um subgrafo isomorfo ao C_3. As linhas contínuas mais escuras evidenciam que cada vértice (v_i, u_k) é adjacente a todos os vértices (v_j, u_l), desde que $v_iv_j \in E(P_3)$ e $k \neq l$. As linhas intermitentes são as demais arestas do tipo (v_j, u_l) com $v_iv_j \in E(P_3)$ e $k \neq l$.

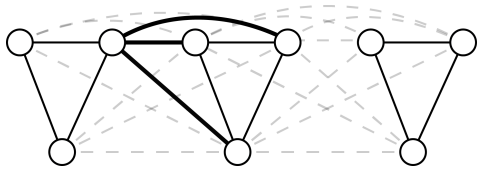

Figura 2: Grafo resultante de $G \bullet H$.

Considere o produto lexicográfico $G \bullet H$, onde $V(G) = \{v_1, v_2 \ldots, v_{|V(G)|}\}$. Para cada $v_i \in G$, seja H_{v_i} o subgrafo de $G \bullet H$ induzido por $\{v_i\} \times V(H)$. Se G é conexo, v_i é adjacente a algum vértice $v_j \in V(G)$, $i \neq j$. Pela definição de produto lexicográfico, (v_j, u) é adjacente a todos os vértices em H_{v_i}, qualquer que seja o vértice u do conjunto $V(H)$. Visto que todos os vértices de H_{v_i} possuem arestas incidentes em todos os vértices de cada subgrafo H_{v_j} quando $v_iv_j \in V(G)$, usamos duas linhas para representar as conexões entre todos os vértices de H_{v_i} e H_{v_j}. Os subgrafos H_{v_i} de $G \bullet H$ induzidos por $\{v_i\} \times V(H)$ são representados por círculos. A Figura 3 ilustra a representação do produto lexicográfico $P_3 \bullet C_3$. Perceba que, com esse tipo de representação gráfica, pode-se ter uma noção da estrutura do grafo $G \bullet H$ independentemente de qual seja a estrutura do grafo H.

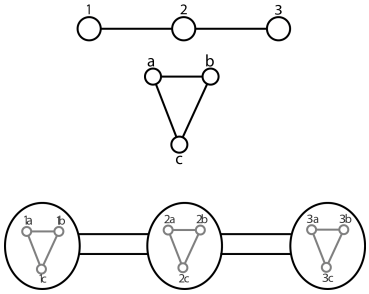

Figura 3: Grafos P_3, C_3 e representação gráfica do grafo $P_3 \bullet C_3$.

Um expressivo resultado de Zhang sobre número de dominação em produtos lexicográficos de grafos é apresentado no Teorema 1, a seguir.

TEOREMA 1. *Zhang [7] Se G é um grafo simples e H é um grafo com $\gamma(H) = 1$, então $\gamma(G \bullet H) = \gamma(G)$.*

Quando G tem um conjunto dominante de tamanho 1, o vértice que domina $V(G)$ é chamado de vértice universal. Observe que, como o produto lexicográfico não é comutativo, o Teorema 1 não estabelece um número de dominação mínimo para o caso em que G é um grafo com vértice universal

e H um grafo simples com $\gamma(G) \geq 2$. Para este caso, apresentamos o Teorema 2, no qual provamos que $\gamma(G \bullet H) = 2$ quando $\gamma(H) \geq 2$.

TEOREMA 2. *Seja G é um grafo com vértice universal e H um grafo simples com $\gamma(H) \geq 2$. Se $|V(G)| \geq 2$, então $\gamma(G \bullet H) = 2$, caso contrário $\gamma(G \bullet H) = \gamma(H)$.*

PROOF. Seja G um grafo com vértice universal e H um grafo simples com $\gamma(H) \geq 2$. Se $|V(G)| = 1$, $G \bullet H$ é isomorfo ao grafo H e, portanto, $\gamma(G \bullet H) = \gamma(H)$. Considere então que $|V(G)| \geq 2$. Seja v o vértice universal em G. Então, para todo vértice $u \in V(H)$, o vértice $(v, u) \in G \bullet H$ domina todos os vértices (w, u) tal que $w \in G$ e $w \neq v$. Então, basta escolher qualquer vértice $(v, u) \in V(G \bullet H)$, tal que v seja vértice universal em G para dominar $V(G \bullet H) \setminus S$, onde $S = \{(v, u') : uu' \notin E(H)\}$. Como H não tem vértice universal, existe pelo menos um vértice u' não adjacente a u em H. Observe que $S \neq \emptyset$ e S é um subconjunto de vértices de H_v. Para dominar S, escolha qualquer vértice (v', z) tal que $v' \neq v$ e $vv' \in E(G)$. Como G tem vértice universal e ordem pelo menos 2, $vv' \in E(G)$. Então (v', z) é adjacente a todos os vértices do subgrafo H_v. Logo, (v', z) domina S. Portanto, $\gamma(G \bullet H) = 2$. □

Nos casos em que nenhum dos grafos possui vértice universal, ainda é possível determinar o número de dominação no produto lexicográfico para algumas classes bem estruturadas, como os grafos k-partidos completos. Um grafo é k-partido completo quando seu conjunto de vértices pode ser particionado em k conjuntos de forma que dois vértices são adjacentes se, e somente se, não pertencem ao mesmo conjunto da partição. O Teorema 3 apresenta o número de dominação entre um grafo k-partido completo e um outro grafo qualquer.

TEOREMA 3. *Se G é um grafo k-partido completo, $k \geq 2$, e H é um grafo simples sem vértice universal, $\gamma(G \bullet H) = 2$.*

PROOF. Seja G um grafo k-partido completo, $k \geq 2$ e H um grafo simples sem vértice universal. Considere a partição $[P_1, P_2, \ldots, P_k]$ de G em k conjuntos independentes disjuntos. Como G tem pelo menos duas partes, então dois vértices são necessários e suficientes para dominar G: um vértice em $v_1 \in P_1$ domina $V(G) \setminus P_1$ e um vértice em $v_2 \in P_2$ domina P_1. Então, $D = \{v_1, v_2\}$ é um conjunto dominante mínimo em G. Seja u um vértice qualquer em $V(H)$. Considere os vértices (v_1, u) e (v_2, u) pertencentes a $G \bullet H$. Observe que (v_1, u) domina todo vértice (v_i, w) tal que $v_i \notin P_1$, qualquer que seja $w \in V(H)$. Note que, nenhum vértice (v_i, w) com $i \neq 1$ tal que $v_i \in P_1$ é dominado por (v_1, u). Como H não tem vértice universal, existe pelo menos um vértice w não-adjacente a u em H. Para todo w tal que $uw \notin E(H)$, o vértice $(v_1, w) \in G \bullet H$ não é dominado por (v_1, u). Seja S o subconjunto dos vértices de $V(G \bullet H)$ não dominados. Note que tais vértices pertencem a subgrafos H_{v_i} tais que $v_i \in P_1$. Como $\{v_1, v_2\}$ é um conjunto dominante em G, $v_2 \notin P_1$. Logo, qualquer vértice $(v_2, w) \in V(G \bullet H)$, é adjacente a todos os vértices de H_{v_i} tal que $v_i \in P_1$. Portanto, (v_2, w) domina S. Logo, $\gamma(G \bullet H) = 2$. □

Considerando o grafo resultante de qualquer produto lexicográfico, apesar de não determinarmos o seu número de dominação, estabelecemos um limite superior justo para o mesmo. Este resultado é apresentado no Teorema 4.

TEOREMA 4. *Sejam G e H grafos simples e conexos, então $\gamma(G \bullet H) \leq 2\gamma(G)$.*

PROOF. Sejam $D = \{v_{t_1}, v_{t_2}, \ldots, v_{t_{\gamma(G)}}\}$ um conjunto dominante mínimo em G e $V(H) = \{u_1, u_2, \ldots, u_{|V(H)|}\}$. Para cada $v_i \in G$, seja H_{v_i} o subgrafo de $G \bullet H$ induzido por $\{v_i\} \times V(H)$. Cada vértice (v_{t_i}, u_1) tal que $v_{t_i} \in D$ domina todos os vértices $(x, z) \in G \bullet H$ tal que $(v_{t_i}, x) \in E(G)$. Falta dominar os vértices dos subgrafos H_{v_i} tais que $v_i \in D$ e $N(v_i) \cap D = \emptyset$. Como o grafo G é conexo, v_i é adjacente a algum vértice $x \in G$. Portanto, (x, u_1) é adjacente a todos os vértices em H_{v_i}. Então, para cada H_{v_i} tal que $v_i \in D$, é necessário incluir no máximo um vértice no conjunto dominante de $G \bullet H$ para dominar os vértices de H_{v_i}. Portanto, $\gamma(G \bullet H) \leq 2\gamma(G)$. □

3. CONCLUSÃO

Neste trabalho determinamo o número de dominação mínimo do produto lexicográfico entre um grafo com vértice universal e um outro qualquer e entre dois grafos sem vértice universal, onde o primeiro é um k-partido completo. Também definimos um limite superior para o número de dominação de $G \bullet H$, quando G e H são simples e conexos. Note que esse limite é justo pelo fato de que existem grafos cujo número de dominação mínimo é igual a $2\gamma(G)$ como no caso do produto lexicográfico de um C_6 por qualquer grafo com $\gamma(G) \geq 2$. É interessante notar que quanto menos componentes de tamanho 1 houverem no subgrafo induzido pelo conjunto dominante mínimo de G, menor será a cardinalidade do conjunto dominante de $G \bullet H$ obtido pela técnica do Teorema 4. Observe que quando todo vértice do conjunto dominante mínimo $D \subseteq V(G)$ possui um vizinho em D, então $\gamma(G \bullet H) = \gamma(G)$. Considerando esta observação é interessante investigar nos trabalhos futuros quais os grafos que sempre possuem um conjunto dominante mínimo sem componentes de tamanho 1. Uma caracterização de tais grafos seria um resultado relevante.

4. REFERENCES

[1] B. Bresar;et.al. Vizing's conjecture: A survey and recent results. *Journal of Graph Theory*, 69(1):46–76, 2011.

[2] B. Hartnell and D. F. Rall. *Domination in graphs, advanced topics*. Marcel Dekker, Inc., 1998. Domination in cartesian products: Vizing's Conjecture (Chapter 7), 163–189.

[3] T. W. Haynes. *Fundamentals of domination in graphs*. Marcel Dekker, Inc., New York, 1 edition, 1998.

[4] D. S. J. Michael R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1 edition, 1979. p. 190, problem GT2.

[5] T. Sitthiwirattham. Domination on lexicographical product of cycles. *Far East Journal of Mathematical Sciences (FJMS)*, 75(1):193–200, 2013.

[6] V. G. Vizing. *Some unsolved problems in graph theory*. Uspehi Nauk, 23 edition, 1968.

[7] J. L. X. Zhang and J. Meng. Domination in lexicographic product graphs. *Ars Combin*, 101(1):251–256, 2011.

Alocação de recursos geograficamente distribuídos em clusters homogêneos

Wagner Senger
Universidade Tecnológica Federal do Paraná
Ponta Grossa-PR, Brasil
wagnersenger@gmail.com

Lourival Aparecido de Góis Departamento de Pós Graduação em Ciência da Computação
Universidade Tecnológica Federal do Paraná
Ponta Grossa-PR, Brasil
gois@utfpr.edu.br

RESUMO

Baseado na ideia de aproveitamento de recursos já existentes mas pouco ou mal utilizados, o presente artigo tem como objetivo propor um modelo de clusterização dos mesmos, de forma que todos os elementos pertencentes ao ambiente global possam ser dispostos em grupos homogêneos entre si, permitindo assim uma igual possibilidade de aproveitamento de recursos, desde os mais básicos até aqueles com capacidade mais avançada. Neste estudo é apresentada uma forma costumeira de criação de grupos com base na proximidade entre os elementos através do algoritmo k-means, e também a sua alternativa, objeto do estudo, que alia um conceito de redes neurais simples a um método de agrupamento baseado em um algoritmo hierárquico de ligação pelo elemento mais distante.

Palavras-chave

Grade Computacional; Cluster; Balanceamento de Carga; k-means, algoritmo hierárquico

ABSTRACT

Based on the idea of the utilization of the resources already existed but bad or low used, this paper has as objective propose a model of clusterization of them, in the way that all elements that belongs to the global environment can be placed in homonegeous groups, allowing and equal possibility of use of the resources, since the most basic until that with high capacity. In this research is presented a way of creating groups based on the closeness between the elements using the algorithm k-means, and his alternative, object of this paper, that join a concept of simple neural networks to an method of grouping based in an hierarchical algorithm of linkage by the farthest element.

Keywords

Computational Grid; Cluster; Load Balance; k-means, hierarchical algorithm

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

1. INTRODUÇÃO

A cada dia mais e mais equipamentos são produzidos e outros tantos são esquecidos por seus compradores. Diversos são os motivos para isso, mas o fato é que muitos deles acabam sendo substituídos por uma próxima geração de equipamentos, que futuramente acabará tendo o mesmo destino. O que se vê são milhares de recursos que já tiveram um custo altíssimo tendo seu uso diminuído, mesmo máquinas atuais em que seus donos desprenderam muito do seu tempo para encontrar aquela com configuração ideal para suas necessidades sendo subutilizadas e deixando grande parte do seu poder computacional sem uso, não por terem sido supervalorizadas no momento da aquisição mas por estarem desligadas ou simplesmente ociosas em boa parte do tempo, seja em um momento de descanso ou mesmo no intervalo do café. Estas máquinas, se somado seu poder computacional para atender uma demanda poderiam ser capazes de se assemelhar aos mair poderosos computadores.

Suponhamos a seguinte situação, digamos que seu computador represente um poder de processamento de 10, e este tenha sido adquirido em virtude do seu computador anterior já não suprir suas necessidades, ele possuía um poder de 5 enquanto você precisava de 7, por isso você foi levado a comprar esta máquina com um poder maior do que precisava já pensando em uma necessidade futura. Pois bem, e se neste cenário você passasse a necessitar de um poder de 12, o que haveria de fazer? Compraria uma máquina de poder 15 pensando em uma necessidade futura, porém, você já possui 15 de processamento em sua casa, 10 da atual e 5 da antiga. Não seria interessante se fosse possível somá-las? Agora, vamos expandir um pouco nosso pensamento para quantas dessas máquinas se encontram abandonadas, muitas que funcionam perfeitamente, porém já não são mais utilizadas por diferentes motivos, sem levar em consideração que são recursos com um valor já investido, e a sua reutilização não implicaria em um gasto extra.

Identificar a melhor forma de aproveitamento destes recursos ociosos não é uma tarefa fácil e tem sido alvo de vários estudos na área da ciência da computação, pois além de se implementar a alocação destes recursos é necessário também que haja a gerência da sua utilização. Por isso muitos estudos têm apontado para o conceito de alocação em clusters, que consiste em agrupar elementos de acordo com alguma métrica preestabelecida.

Clusterização é uma técnica mais primitiva em que não se é feita correlação com o número de estruturas ou grupos que serão gerados, mas sim um agrupamento com base em similaridades ou diferenças entre os elementos [4]. Agrupar os

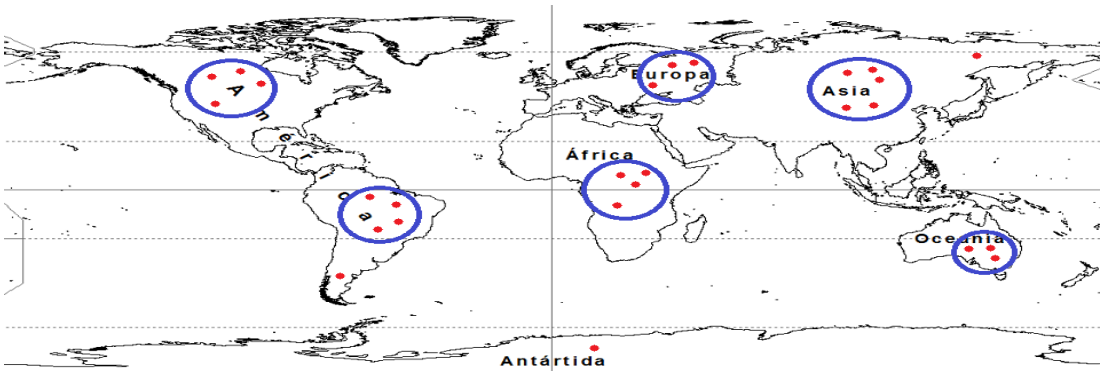

Figura 1: Aplicação de k-means em recursos espalhados geograficamente.

elementos proporciona uma melhor forma de gerenciamento, assim como também é possível identificar aqueles recursos que não se enquadram em grupo algum.

Muitas são as formas com que a clusterização vem sendo aplicada para se organizar os recursos, e em várias situações os clusters são criados a partir de alguma similaridade entre os recursos disponíveis. Esta similaridade costuma ser utilizada em algoritmos que geram o agrupamento com base na aproximação destes dados, e desta forma o cluster consegue reunir elementos de um poder computacional similar por exemplo. A característica escolhida depende do foco do estudo, sendo em alguns casos primordial e em outros uma característica desnecessária.

Para que alguns recursos não sejam sobrecarregados e outros permaneçam com pouca ou nenhuma carga de utilização é importante a aplicação de um balanceamento de carga, e neste quesito a clusterização é um dos principais métodos de balanceamento, permitindo que para uma visão externa o ambiente de cluster se apresente como um sistema único [7].

Com este estudo pretendemos aplicar a clusterização de modo a criar clusters com recursos internos distintos entre si, porém ao serem observados de fora apresentarão homogeneidade entre eles, de forma que futuramente quando a escolha de um cluster for necessária para se executar alguma tarefa, qualquer um seja apto a receber a solicitação.

2. ESTUDO EXECUTADO

A partir do que foi apresentado anteriormente, podemos perceber a necessidade de construção de um mecanismo de agrupamento de recursos. Este mecanismo deve ser feito não somente para identificação e agrupamento dos recursos, mas também para que a gerência sobre eles se torne mais simples, e a partir de então possam ser analisadas melhor as condições para aplicação do processamento das informações e balanceamento de carga.

Este método porém deve ser organizado e por isso não pode aleatoriamente distribuir os elementos, caso contrário não seria possível atribuir a homogeneidade proposta no estudo. Existem algumas características em cada elemento que são particularmente interessantes para o estudo em questão, como localização geográfica, capacidade de processamento, percentual de uso, capacidade de memória RAM e capacidade de armazenamento.

Alguns estudos como o efetuado por Norouzi e Akbarpour [7] utilizam como métrica de agrupamento a proximidade entre os elementos, causando um resultado como o demonstrado na Figura 1, em que os elementos estão agrupados segundo a sua localização geográfica. Isso faz com que os pontos tenham uma homogeneidade entre si, porém os clusters gerados não são homogêneos sob vários outros aspectos como: os grupos possuem elementos em posições terrestres diferentes; não possuem o mesmo número de elementos; a latência de respostas das redes pode apresentar uma discrepância devido a distância geográfica; os elementos agrupados em um dos clusters podem possuir um poder de processamento muito maior que os outros. Podemos perceber visualmente a diferença entre os grupos, onde por exemplo o grupo da Europa se apresenta diferente do grupo da Ásia mesmo em número de elementos, o que acaba destruindo a homogeneidade almejada.

O algoritmo utilizado para tal geração de grupo é o k-means, o qual possui seu funcionamento baseado na distância [2], de forma que para se gerar o cluster os pontos devem estar a uma distância aproximada de um mesmo centro [8], o qual é calculado pelo algoritmo. Este algoritmo é muito utilizado devido a possuir uma vasta literatura e pela consolidação que recebeu devido aos vários anos de uso. Por considerar apenas a distância, este algoritmo não é sendo justo com a criação dos grupos, desconsiderando critérios importantes.

Ainda neste exemplo podemos observar outra situação como os recurso situados na Antártida, leste da Ásia e sul da América, por não possuírem nenhum outro recurso próximo acabam por não serem incluídos em grupo algum, mesmo que estes tivessem uma alta capacidade e pudessem contribuir para algum grupo. Estes recursos são chamados de ruídos, ou então como utilizado na mineração de dados, *outliers*, e tem este nome porque são elementos que não se enquadram em alguma classificação específica. Não significa que todos os recursos devam ser aproveitados, pode acontecer de que algum não tenha capacidade suficiente para contribuir em nenhum grupo e, portanto, deva ser descartado. Porém essa análise deve ser mais minuciosa, coisa que simplesmente aplicando a métrica da distância pode deixar a desejar. Por isso neste caso a aplicação do algoritmo k-means não se apresenta como a melhor opção.

O ponto central é a forma como a medida de similaridade será calculada para que se reflita a homogeneidade dos grupos [1], quando se utiliza apenas uma medida para criação

Figura 2: Neurônio Artificial.

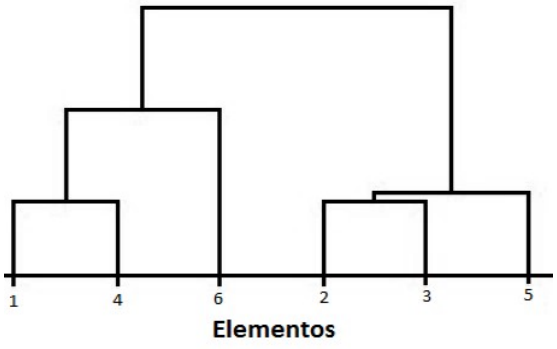

Figura 3: Algoritmo Hierárquico.

dos grupos não se dá abertura para que haja homogeneidade entre eles. Porém, quando um conjunto de regras é utilizado, existe a possibilidade de haver um balanceamento de modo que seja gerada uma compensação em atributos fracos de um recurso por seus atributos fortes. O que se pretende fazer é a utilização da posição geográfica como parte de uma relação de parâmetros para criação do cluster e não como a única métrica para isso.

Criar um grupo nesses moldes pode permitir um melhor aproveitamento de um recurso considerado fraco em certo aspecto e balanceá-lo em um grupo com outro elemento de propriedades mais vantajosas.

Outro ponto que pode ser favorecido nesta forma de utilização é em relação aos ruídos. Diferente do exemplo anterior, onde perderiam espaço devido a não se enquadrar em algum grupo pela sua localização incompatível com outros recursos, neste caso eles podem vir a ser aproveitados já que outras características podem lhes tornar interessantes.

Características diferentes também inferem valores diferentes, dessa forma não podemos simplesmente comparar um recurso com valor X de distância a um outro recurso de um valor Y de capacidade de processamento. É necessário que todos os elementos possuam a mesma forma de medida, permitindo assim que apesar de distintos estes valores possam contribuir igualmente no resultado final.

Nesse estudo será aplicado uma medição baseada em um conceito da rede perceptron, comum em redes neurais. Este algoritmo utiliza um dos formatos mais simples de rede neural, a qual considera que existe apenas um neurônio, e para ele existem várias entradas, cada uma com um peso diferente, e, por fim, é produzida uma saída [3].

É possível visualizar na Figura 2 a estrutura do neurônio, as entradas x0, x1 e x2 possuem pesos respectivos w0, w1 e w2, os quais passam pelo processo de somatório e produzem uma saída. Da mesma forma utilizaremos várias entradas para compor o valor do nosso recurso, as entradas serão referentes a valores de características como capacidade de processamento, localização geográfica, disponibilidade, quantidade de memória RAM, etc.

Um recurso com maior capacidade de processamento possuirá um peso mais alto para este item, já se o recurso tiver pouca disponibilidade para o uso possuirá um valor mais baixo nesta entrada. Por fim todos estes valores serão somados e produzirão um resultado que será a identificação da capacidade do recurso. Os valores que deverão ser atribuídos para cada item, como capacidade de processamento, que fará a composição do valor final, deverão ser testados e ajustados de modo que cada um tenha o peso correto de acordo com a sua influência no potencial do recurso.

Tendo o valor composto de cada recurso já é possível passar a construção da clusterização. Da mesma forma que não é possível agrupar elementos pela localização geográfica também não poderemos agrupá-los pela similaridade do valor final, pois se o fizermos criaremos grupos de recursos com baixa capacidade, e grupos de alta capacidade, o que foge da proposta do estudo, já que os grupos não atingiriam a homogeneidade almejada.

A alternativa avaliada como plausível para a proposta é a utilização de um algoritmo hierárquico de ligação por vizinho mais distante. Esse tipo de algoritmo é capaz de agrupar elementos de valores distantes [6], promovendo um balanceamento entre o resultado final, já que os grupos tendem a possuir tanto recursos de alto poder computacional quanto de baixo poder.

Como podemos observar na Figura 2 os elementos 1, 4 e 6 são unidos no mesmo grupo, já os elementos 2, 3 e 5 são dispostos em outro, desta forma o grupo esquerdo ficaria com valor final 11 enquanto o grupo direito ficaria com valor 10, alcançando a homogeneidade dos grupos como proposto.

Existe ainda a possibilidade de recursos com valores muito inferiores serem adicionados em algum grupo, para estes ruídos deverão ser criadas regras específicas para que sejam tratados independentemente. Vão existir casos em que esses elementos não terão condições de auxiliar o cluster e deverão ser descartados. Porém pesquisas sobre este tema deverão definir ainda como será o seu tratamento.

Desconsiderando os ruídos o formato de balanceamento acaba por gerar clusters com características semelhantes [5], suas composições tendem a ter um formato mais próximo uns dos outros e por isso seu poder total deve ser nivelado. Ainda assim existirão situações em que os clusters ficarão com pequenas disformidades entre eles, pois atingir a perfeita sintonia pode não vir a ser possível devido a disparidade que a conjunção de várias características pode gerar. Por isso há de ser estabelecida uma margem acima ou abaixo para que esses sejam considerados com iguais, coisa que só com testes mais exaustivos poderemos definir melhor.

3. CONCLUSÃO

Clusterização é um assunto que a cada novidade também desperta um novo desafio. Os meios tradicionais disponíveis nem sempre apresentam a melhor solução para o objetivo

desejado nas aplicações e, em várias situações, ao corrigirem uma falha acabam abrindo uma brecha para outro questionamento.

Clusterizar recursos de forma que todos tenham a mesma possibilidade de serem aproveitados é uma área que ainda possibilita uma grande variedade de estudos, assim como a utilização final destes recursos disponíveis.

Este, apesar de ser um estudo em andamento com fatores a ainda serem ajustados, pode proporcionar um ambiente mais balanceado promovendo a homogeneidade sugerida sem elevar os custos da rede. Sua eficácia ainda depende de mais estudos, porém a forma diferenciada de aplicação permite nos deixar com uma boa perspectiva.

4. REFERÊNCIAS

[1] L. Gómez-Chova, L. Bruzzone, G. Camps-Valls, and J. Calpe-Maravilla. Semi-supervised remote sensing image classification based on clustering and the mean map kernel. *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, July 2008.

[2] Y. Hanmin, L. Hao, and S. Qianting. An improved semi-supervised k-means clustering algorithm. *IEEE Conference Publications*, pages 41–44, May 2016.

[3] S. Haykin. *Redes Neurais*. Bookman, 2001.

[4] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Pearson Education, Inc., Upper Saddle River, New Jersey, 2007.

[5] R. Massin, C. J. L. Martret, and P. Ciblat. Distributed clustering algorithms in group-based ad hoc networks. *Signal Processing Conference (EUSIPCO), 2015 23rd European*, September 2015.

[6] S. Mehrotra and S. Kohli. Comparative analysis of k-means with other clustering algorithms to improve search result. *Signal Processing Conference (EUSIPCO), 2015 23rd European*, September 2015.

[7] M. Norouzi and S. Akbarpour. Data processing in grid systems by using cluster algorithms. *IEEE*, pages 309–312, September 2013.

[8] N. Yildirim and B. Uzunoglu. Association rules for clustering algorithms for data mining of temporal power ramp balance. *2015 International Conference on Cyberworlds (CW)*, February 2016.

A proposal of an architecture for a Smart Parking based on intelligent agents and embedded systems

Lucas Fernando Souza de Castro
Universidade Tecnológica Federal do Paraná
PPGCC - Programa de Pós-Graduação em Ciência da Computação
Av. Monteiro Lobato - km4
Ponta Grossa, Brazil
lcastro@alunos.utfpr.edu.br

Gleifer Vaz Alves
Universidade Tecnológica Federal do Paraná
PPGCC - Programa de Pós-Graduação em Ciência da Computação
Av. Monteiro Lobato - km4
Ponta Grossa, Brazil
gleifer@utfpr.edu.br

André Pinz Borges
Universidade Tecnológica Federal do Paraná
PPGCC - Programa de Pós-Graduação em Ciência da Computação
Av. Monteiro Lobato - km4
Ponta Grossa, Brazil
apborges@utfpr.edu.br

ABSTRACT

A smart city presents interactions between human beings and technological objects. Moreover, it is an environment where technology helps the human life, becoming it easier and practical. The smart city has different branches, such as: governance, people living, economy, mobility, and traffic. The traffic in a big city is one of the most stressful and costly thing to manage and organize, where specifically the management of parking lots is a hard task indeed. This paper aims to propose an architecture for a smart parking, which is an intelligent parking lot that uses technology to help drivers get a parking spot. The proposed architecture will be based on a multiagent system through JaCaMo Framework and embedded systems using Arduino and Raspberry Pi.

Keywords

smart parking; multiagent system; jacamo; embedded system

1. INTRODUCTION

It is well known that drivers spend a considerable time searching for a parking spot. For example, in New York City, 40% of the time that a driver spends while drives the car is to look a spot [10]. Moreover, in cities, there are many private parking lots where a driver can pre-book a spot even before of get off home. Most of the parking lots have an electronic system to manage the spots and their price. Although there are computer solutions to get a parking spot or check if it is available [4] [12], as far as we know, it is uncommon parking systems that have a smart system used to manage and allocate the parking spots in an autonomous way.

Some solutions to smart parking are based on multiagent system (MAS), which is a system composed of agents able to use and share resources in a dynamic environment [14]. In the smart parking there is a dynamic and complex environment due to many drivers and cars running at same time and even competing for the same resource, a spot. Therefore, the multiagent system could be applied to solve the smart parking matter. Actually, there are some solutions to smart parking based on a multiagent system, for instance Naples - Italy has been used as a scenario to apply a MAS able to provide a negotiation among drivers to get a parking spot [5]. Finally, a multiagent system is known for providing such social interaction among the agents, hence, in the smart parking context the drivers could compete or share a resource [6].

The present work aims to propose an architecture designed to smart parking based on a multiagent system developed through the JaCaMo framework and embedded systems (Arduino and Raspberry Pi). This architecture will be divided into three stages: `mobile, core, and embedded`. The `core` part was previous developed as a MAS called MAPS (**M**ulti**A**gent **P**arking **S**ystem), able to receive a request from driver agents and allocate a spot in a private parking lot. However, the MAPS was not applied in a real parking lot. Thus, this paper proposes the architecture that will use the MAPS and extend it to become applicable and workable in a real parking lot.

2. MAPS PROJECT

The MAPS Project was created to allocate parking spots for drivers in an autonomous way. The system has been developed using the JaCaMo Framework, which it is a tool to develop multiagent systems [1]. The framework is composed by Jason (Agent programming language), Cartago (Framework to develop artifacts for the multiagent system) and Moise (Tool to create a social organization).

The process of allocating a spot for a driver is based on a trust degree, which is a value responsible for showing the agent driver commitment before the smart parking. The figure 1 shows how the trust degree is used to decide which driver shall obtain the requested spot. The winner driver received the spot because it has a greater trust degree ($trust = 870$) than the another driver ($trust = 290$). The current version of the trust degree is calculated based on how many times the driver uses the parking lot. Each

WPCCG '16 Setembro 28, 2016, Ponta Grossa, Paraná, Brasil

time the driver uses the parking lot its trust degree increases in 20 points.

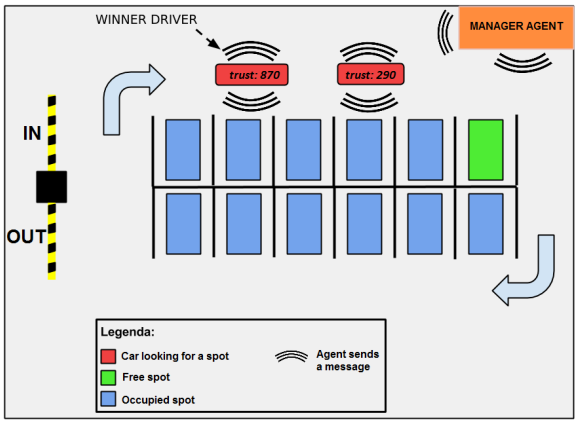

Figure 1: MAPS General representation

The trust degree value is stored by the manager agent. Every single driver agent has its own trust degree value, moreover, there is a database system created to store this value and update it whereas the driver agent uses the smart parking [7] [3]. Although the MAPS Project has been developed through JaCaMo framework, we use just the JaCa (Jason + Cartago) technology so far. Therefore, one of the possible extensions for the MAPS project is the implementation of social organization through Moise.

3. THE ARCHICTETURE

In order to develop the architecture, we shall face a complex situation, which different systems and technologies must work together appropriately. The figure 2 shows the general view of the architecture with its three main parts.

The architecture could be splitted into three stages: `mobile, core and embedded`. The `mobile` will be able to receive a request from the driver through an Android app and send to the `core`. The `core`, like the work suggests, is the kernel of the system, which is capable of managing the requests coming from the drivers and giving to them a parking spot. Finally, the `embedded` will be able to control the spots in the smart parking and inform the `core` how is going on the environment at the parking lot. The `embedded` is composed by a Raspberry Pi and an Arduino. The Raspberry Pi will be responsible for managing the Arduino boards, which will have sensors running. These sensors are going to check the presence of a driver in a spot and control the parking lot.

3.1 Architecture and its connections

Besides the architecture is composed of the stages (`mobile, core and embedded`), there are connections between each part of the system. Every connection will be based on the Java technology because it is done and ready to use. The connection among the `mobile, core` and `embedded` will be created using TCP/IP Java sockets. Further, the connection inside of the `embedded` will be done by the ARGO platform, which is a tool used to communicate the Jason language with the Arduino and Raspberry pi boards through a serial communication [11].

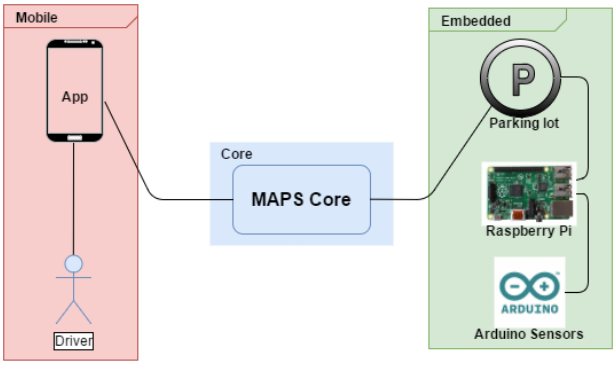

Figure 2: Proposed Architecture - MAPS

The connection using TCP/IP Java Sockets will be used to communicate among the stages because each state is separate of the another. The reason of choose Java socket is because the framework JaCaMo is based on Java language, thus it will be easier to use the Java to implement the connections among the stages.

3.2 Architecture and its stages

Implementing this architecture will be divided into subgoals because they are independent to implement and each one has a different impact on the architecture. The first sub-goal is to extend the current MAPS Core developed using JaCaMo Framework to a new version which supports the requests from an Android app. This feature will be develop using the JaCa-Android [13]. The JaCa-Android tool is a project that provides the integration between Jason+Cartago and the Android platform [9]. The Android app will be able to provide to the driver the ability to request a parking spot to the MAPS Core. In Brazil, there are some new cars which support the usage of Android auto on streets. Moreover, it is possible to extend the JaCa-Android to develop an app to the Android auto platform [8].

As far as about the `embedded`, the second sub-goal is create an integration between the agents (in Jason) and Artifacts (in Cartago) with embedded platforms, like Raspberry pi and Arduino.

As explained before and showed in figure 1, allocating a parking spot depends on of trust degree value, a greater value will receive a parking spot faster than a lower value. The current way to calculate the trust degree is increment the trust degree value every single time the driver uses the system independently how long it stayed in the parking lot. Then, the third sub-goal is to develop a new way to calculate the driver trust degree and reward the driver to reinforce the use of this smart parking. This new trust degree will be showed in advance:

```
trustDegree(D) = trustDegree(D) + a + t + u
```

- **attendance (a)**: Value that shows how the driver is attending the parking lot. This value increases, according to with the frequency of the driver uses the parking;
- **time of usage (t)**: This value shows how long the driver spent in the parking lot;

- **usage (u)**: It shows the behavior of the driver inside of the parking lot. There will be social rules developed in Moise that govern the parking lot. For example, the driver will receive a specific parking spot and he cannot park in another parking spot. Otherwise, the drivel will be sanctioned.

4. FINAL REMARKS

As expected results, in Brazil, there are many cities waiting for smart parking technologies due to the growing of the urban traffic. The MAPS Project tries to bring some light to this issue offering to drivers the ability of a parking spot near where they are going at with a good price and availability. Moreover, there are rewards to the drivers who use the MAPS, they will gain a higher trust degree value. This value impacts the spot allocation time and consequently they could receive a spot faster. Finally, the smart parking is a part of a context called smart city, where there are many things to optimize and the parking lot is one and most important of them [2] [10]. Moreover, we hope this proposed architecture for the MAPS will offer a whole solution for a smart parking to manage the parking spots in different layers (`mobile, core` and `embedded`).

The future work is improving the proposed architecture to become more functional and effective. Therefore, to achieve that is important to decentralize the management of the system. The management of the smart parking is done by a central agent called manager. The manager is a JaCaMo agent. In the current version of MAPS, the manager works in MAPS core, however, to this new version of MAPS we intend to decentralize the manager agent to have many manager agents running at same time. There will be a hierarchy of the manager agents. In the MAPS Core, on the one side there will be managers able to receive a request for a spot and pick a parking lot for this spot, on the other side there will be another manager agent in every parking to manage the parking spots. Thereafter, it is going to be possible to support the negotiation of parking spots among driver agents. Hence, the risk of a fail will decrease whereas there will be more than one manager agent. The current version of MAPS could fail if the manager fails.

Finally, it is important to extend the MAPS Core to support more than one parking lot (scalability). This extension will be done through the multi-workspace concept in the Cartago. This main goal of this feature is provide to the driver multiple choices to park his car and the ability to the managers control multiple parking lots in order to approximate a real scenario.

5. ACKNOWLEDGMENTS

Thanks for the support of CAPES by means of the masters scholarship.

6. REFERENCES

[1] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi. Multi-agent oriented programming with jacamo. *Sci. Comput. Program.*, 78(6):747–761, June 2013.

[2] A. Caragliu, C. D. Bo, and P. Nijkamp. Smart cities in europe. *Journal of Urban Technology*, 18(2):65–82, 2011.

[3] L. F. S. Castro, G. V. Alves, and A. P. Borges. Utilização de grau de confiança entre agentes para alocação de vagas em um Smart Parking. In *Anais do X Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – X WESAAC*, 2016.

[4] C. Di Napoli, D. Di Nocera, and S. Rossi. Negotiating parking spaces in smart cities. In *Proceeding of the 8th International Workshop on Agents in Traffic and Transportation, in conjunction with AAMAS*, 2014.

[5] C. Di Napoli, D. Di Nocera, and S. Rossi. Using negotiation for parking selection in smart cities. In Y. Demazeau, F. Zambonelli, J. Corchado, and J. Bajo, editors, *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, volume 8473 of *Lecture Notes in Computer Science*, pages 331–334. Springer International Publishing, 2014.

[6] D. Di Nocera, C. Di Napoli, and S. Rossi. A Social-Aware Smart Parking Application. In *Proceedings of the 15th Workshop "From Objects to Agents"*, volume 1269, 2014.

[7] F. Ducheiko, L. F. S. Castro, and G. V. Alves. Implementação de recursos em um smart parking baseado em sistemas multiagente. In *Anais do X Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações – X WESAAC*, 2016.

[8] Google and A. Auto. Android auto. Available on <https://www.android.com/intl/eng-us/auto>, 2016.

[9] JACAMO. The JaCaMo approach. Available on <http://jacamo.sourceforge. net/?page_id=40>, 2011.

[10] A. Koster, F. Koch, and A. L. Bazzan. Incentivising crowdsourced parking solutions. In J. Nin and D. Villatoro, editors, *Citizen in Sensor Networks*, volume 8313 of *Lecture Notes in Computer Science*, pages 36–43. Springer International Publishing, 2014.

[11] N. M. Lazarin and C. E. Pantoja. A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. In *9th Software Agents, Environments and Applications School (WESAAC)*, 2015.

[12] Parkingedge. Best parking. Available on <http://www.bestparking.com>, 2013.

[13] A. Santi. JaCa-Android. available on <http://jaca-android.sourceforge.net, 2011.

[14] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition, 2009.