Dissertação de Mestrado

Uma Modelagem Diagnóstica Multidimensional para o Entendimento do Perfil de Alunos Iniciantes Em Programação

Maria Cristina Tenório Cabral Cavalcante

Orientador:

Evandro de Barros Costa

Co-orientador:

Rodrigo de Barros Paes

Maceió, dezembro de 2013.

Maria Cristina Tenório Cabral Cavalcante

Uma Modelagem Diagnóstica Multidimensional para o Entendimento do Perfil de Alunos Iniciantes Em Programação

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Evandro de Barros Costa

Co-orientador:

Rodrigo de Barros Paes

Maceió, dezembro de 2013.

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Maria Auxiliadora G. da Cunha

C376u Cavalcante, Maria Cristina Tenório Cabral.
Uma modelagem diagnóstica multidimensional para o entendimento do perfil de alunos iniciantes em programação / Maria Cristina Tenório Cabral Cavalcante. – 2013.
64 f. : il.

Orientador: Evandro de Barros Costa.
Coorientadora: Rodrigo de Barros Paes.
Dissertação (Mestrado em Modelagem Computacional do Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2013.

Bibliografia: f. 61-64.

1. Ensino à programação. 2. KDD. 3. Perfil de iniciante em programação. I. Título.

CDU: 004.42:372.004.4

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Modelagem Computacional de Conhecimento
Avenida Lourival Melo Mota, Km 14, Bloco 19, Cidade Universitária
57.072-900 Maceió AL Brasil CGC: 24.464.109/0001-48
Telefone: (082) 3214-1364

MODELAGEM COMPUTACIONAL DE CONHECIMENTO

Membros da Comissão Julgadora da Dissertação de Mestrado de Maria Cristina Tenório Cabral Cavalcante, intitulada: "Uma Modelagem Diagnóstica Multidimensional de Perfil de Alunos Iniciantes em Programação", apresentada ao Programa de Pós-Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas em 20 de dezembro de 2013, às 10h00min, no auditório do Instituto de Computação – IC/UFAL.

COMISSÃO JULGADORA

Prof. Dr. Evandro de Barros Costa
UFAL – Instituto de Computação
Orientador

Prof. Dr. Jorge Artur Peçanha de Miranda Coelho
UFAL – Faculdade de Medicina
Examinador

Prof. Dr. Dalton Dario Serey Guerrero
UFCG – Departamento de Sistemas e Computação
Examinador

Maceió, 20 dezembro de 2013.

"*Deus nos concede, a cada dia, uma página de vida nova no livro do tempo. Aquilo que colocarmos nela, corre por nossa conta.*"

(Chico Xavier)

Agradecimentos

"...nossas lembranças permanecem coletivas, e elas nos são lembradas pelos outros, mesmo que se trate de acontecimentos nos quais só nós estivemos envolvidos, e com objetivos que só nós vimos. É porque, em realidade, nunca estamos sós" (HALBWACHS, 1990, p.26). A escrita, assim como as lembranças, também nos é possibilitada pelos outros com os quais compartilhamos nossas vidas. Sendo assim, reservo este espaço para reconhecer e agradecer a todos aqueles que contribuíram para a redação final deste sonho.

Agradeço ao meu orientador, Evandro Costa, pelo apoio incondicional e dedicação à realização deste trabalho. Pelos momentos de reflexão, de aprendizado e de devaneio, que deram a essa árdua jornada um tom de poesia.

Agradeço aos meus amados irmãos pelo apoio constante em minha vida. Pessoas que me fortalecem e me fazem acreditar que sempre vale a pena o esforço.

Ao meu noivo, Romildo, pelo amor, compreensão, carinho e dedicação. Por todas as noites em claro dedicadas à escrita desta dissertação, por entender todas as minhas ausências e por tentar, de todas as maneiras possíveis, prestar auxilio. Te amo.

Aos colegas do Instituto de Computação, Juliana Cavalcante e Marlos Tácio, pelo auxílio na pesquisa.

À minha amiga Priscylla Silva pelas longas conversas, pelos momentos de reflexão, pelo apoio e sorrisos. À Simone Cavalcante pelos momentos de descontração e por não me deixar engordar sozinha, compartilhando sempre do meu lanche.

Os acima citados, e alguns que por ventura eu tenha esquecido de citar o nome, ajudaram-me a compreender exatamente o que Albert Einstein sentiu ao dizer "se podes imaginar, podes conseguir". Muito obrigada.

Resumo

Estudos revelam dificuldades por parte de estudantes no aprendizado de programação para iniciantes. Trata-se de um problema reportado em várias instituições, tal como se observa na literatura sobre o assunto. Neste sentido, o presente trabalho buscou esclarecer possíveis fatores cognitivos relacionados ao desempenho insatisfatório apresentado por muitos estudantes de Ciência da Computação da Universidade Federal de Alagoas, relativamente à atividade de resolução de problemas de programação em um nível iniciante. Neste sentido, procurou-se conhecer quem é este estudante de programação inicial, observando-o em diferentes contextos. Particularmente, investiu-se primeiramente na análise de grupos com vistas à compreensão dos estados cognitivos destes estudantes. Além disso, estudou-se a relação entre os dados de desempenho apresentados pelos estudantes na disciplina de programação com disciplinas assumidas como relacionadas tanto na graduação, quanto no ensino médio expresso no exame de ingresso na universidade. Para tanto, avaliou-se diferentes fontes de dados, que trazem informações quantitativas e qualitativas a respeito do estudante: (i) notas de todos os alunos que cursaram a disciplina entre 2006 e 2013, (ii) notas de disciplinas afins a disciplina de programação entre 2010 e 2013, e (iii) conjunto de dados correspondente ao código-fonte produzidos por uma amostra de alunos em exercícios e avaliações. A partir desses dados, foi possível elaborar um modelo para o diagnóstico de perfil dos iniciantes em programação, que identificou a existência de três grupos de alunos, que possuem características cognitivas que variam de estudantes sem sucesso para os que obtiveram sucesso no curso introdutório.

Abstract

Studies reveal learning difficulties faced by students in taking introductory programming course. This problem has been reported in various institutions, according to the literature on the subject. In this sense, the present study aims to clarify possible cognitive factors related to poor performance shown, particularly, by many students of Computer Science from the Federal University of Alagoas, concerning to the activity of solving programming problems on a beginner level.

In this perspective, we have tried to know, in some cognitive respect, the mentioned students of introductory programming, considering them in different academic contexts. Particularly, we invested primarily in the analysis of groups aiming at understanding the cognitive states of these students. Furthermore, we studied two main kinds of relatioships: (i) the relationship between the performance data presented by the students in the discipline of programming and other related disciplines, as well as, (ii) and in the discipline of programming and the score expressed in the exame to access from high school to the university.

To this end, we evaluated different data sources, bringing quantitative and qualitative information about the student, such as: (i) scores of all students that took this course between 2006 and 2013, (ii) scores related to disciplines, including discipline programming between 2010 and 2013, and (iii) corresponding to the set of source code produced by a sample of students on home works and reviews data. From these data, it was possible to develop a model for diagnostic profile of beginners in programming, which identified the existence of three groups of students who have cognitive characteristics ranging from students without success for those who have obtained success in the introductory programming course.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Neste capítulo será apresentada uma contextualização para este trabalho de mestrado interdisciplinar em Modelagem Computacional de Conhecimento, situando-o na linha de pesquisa modelos computacionais em educação. Além disso, elenca-se uma problemática, envolvendo questões de pesquisa e hipóteses associadas. Prossegue-se então com a proposição de objetivos e uma discussão de relevância da pesquisa, assim como apresentação da estrutura da dissertação.

1.1. Contextualização e Problemática

A presente dissertação trata de uma investigação relacionada ao problema sobre o alto índice de insucesso verificado no desempenho de estudantes em disciplinas introdutórias de programação. No Brasil, trata-se normalmente de uma disciplina que faz parte do currículo definido pela Sociedade Brasileira de Computação (SBC) e pelo MEC, destinada aos cursos de Bacharelado em Ciência da Computação, Sistema de Informação e Engenharia de Computação, sendo tipicamente ofertada no primeiro semestre destes cursos.

Tradicionalmente, a atividade de programar é rotulada como difícil entre a maioria dos estudantes iniciantes, o que torna o desenvolvimento dessa atividade um desafio para o aluno, constituindo-se em um desafio educacional. Assim, realizar a atividade de programar exige que o estudante iniciante possua um conjunto de habilidades consideradas essenciais a essa tarefa, como: bom raciocínio lógico, boa capacidade de resolução de problemas e capacidade de expressão a partir de um alfabeto limitado. Estas necessidades fazem com que o ensino de programação, e consequentemente o aprendizado de programação, torne-se um desafio tanto para professores, quanto para os alunos dos cursos introdutórios. Tais problemas têm sido identificados em toda parte do mundo, sendo um dos exemplos de trabalho o de McGettrick [1], ressaltando que o ensino de programação foi uma atividade vista como um dos sete grandes desafios no ensino de computação.

Em geral, conforme literatura no assunto, alunos iniciantes em programação têm apresentado um baixo nível de aproveitamento em disciplinas introdutórias, o que vem preocupando entidades nacionais e internacionais ligadas à área de ensino da Computação [1].

Particularmente, dados levantados de várias turmas de Programação 1, na Universidade Federal de Alagoas, apontam para elevados índices de reprovação e desistência em disciplinas introdutórias de programação. Isto ocorreu no período de 2006 até 2013 cerca de 35% de alunos matriculados em introdução à programação desistem do curso antes de chegar ao fim do primeiro bimestre. Dos alunos que continuam frequentando as aulas, menos da metade apresenta um desempenho satisfatório, em média apenas 40% desses alunos são aprovados. Problemas como este têm sido identificados em toda parte do mundo. McGettrick [1] relata que em 2005 o ensino de programação foi uma atividade vista como um dos sete grandes desafios no ensino de computação.

As razões causadoras desses índices não são de fato conhecidas, parte desse insucesso é vagamente atribuída à aparente ineficiência na atividade de resolução de problemas apresentada pelos alunos, conforme estudou [2]. Além disso, outro ponto a ser levado em consideração é a variedade de fatores que podem ser relacionados para explicar estas deficiências, onde os estilos cognitivos de cada aluno estão fortemente ligados a tais fatores.

Assim, fica evidente que para o desenvolvimento de intervenções efetivas para a resolução deste problema é necessário conhecer seu fator causador onde, para isso, se faz necessário conhecer o aluno de programação, procurando conhecer suas características cognitivas. Diante da dificuldade de mapear características de cada indivíduo que cursa ou cursou disciplinas introdutórias de programação, mostra-se como solução inicial descobrir características comuns a grupos de alunos novatos.

Deste modo, o escopo deste trabalho consiste em abordar questões que ajudem a conhecer características comuns aos alunos matriculados em introdução à programação da Universidade Federal de Alagoas. Nesta perspectiva, a pretensão de mapear essas características trás a tona uma série de questionamentos a respeito de quem é o aluno de programação e quais os conjuntos de habilidades prévias são realmente necessárias ou fundamentais para o desenvolvimento da habilidade de programar. Com base nisso, o interesse geral desta pesquisa está relacionado a responder aos seguintes questionamentos:

Q_0: Existe uma correlação forte entre o desenvolvimento de habilidades matemáticas e habilidades de programação?

Há, atualmente, uma vasta literatura que versa sobre as habilidades de programação e sua correlação com as habilidades matemáticas.

Q_1: Existe correlação forte entre a capacidade de expressão verbal e as habilidades de programação?

A habilidade de programar está intimamente ligada à habilidade de expressar uma solução estruturada de um problema em uma linguagem finita e específica. Entretanto, não há uma verificação da correlação da capacidade de expressão textual de um aluno e suas habilidades em programação. Necessita-se esta verificação para apurar problemas no ensino aprendizado de programação relacionada às linguagens formais e suas utilizações.

Q_2: Existe correlação forte entre o bom desempenho no exame de admissão na universidade (Enem) e as habilidades de programação?

Q_3: É possível distinguir diferentes grupos de estudantes a partir de suas características cognitivas?

1.2. Objetivos

Este trabalho teve por objetivo geral esclarecer fatores cognitivos referentes ao aluno iniciante em programação que cursa ou cursou Ciência da Computação na Universidade Federal de Alagoas – UFAL, procurando diagnosticar o problema do baixo desempenho dos estudantes nas atividades de resolução de problemas, visando entender as razões disto ocorrer. Ressalta-se aqui que se assume neste trabalho que os alunos matriculados na disciplina de Introdução à Programação da UFAL são semelhantes a estudantes matriculados em cursos similares em outras instituições, nacionais e internacionais. Para isso foi realizado um diagnóstico das características gerais relacionadas aos iniciantes em programação matriculados nesta instituição, tendo em conta um conjunto multidimensional de dados, composto por variáveis quantitativas e qualitativas.

Para realizar o objetivo geral, algumas etapas intermediárias foram necessárias, as quais estão expressas nos seguintes objetivos específicos:

- Conhecer os alunos iniciantes em programação, mapeando assim as possíveis causas de seu baixo rendimento nesta disciplina;

- Identificar grupos de alunos com as mesmas características de aprendizado, no intuito de mapear quais são as dificuldades encontradas por esses alunos e quais as barreiras para o aprendizado da disciplina em questão.

1.3. Relevância

O diagnóstico de características relacionadas aos estilos cognitivos de alunos iniciantes em programação é um passo de grande importância para o ensino de programação. Com isto, espera-se contribuir com subsídios para o desenvolvimento de intervenções que promovam, além do aumento da qualidade do ensino de programação em cursos superiores, o aumento dos índices de desempenho e diminuição da evasão escolar em disciplinas ligadas a programação, podendo inclusive contribuir indiretamente com a formação de profissionais mais qualificados em Ciência da Computação e áreas afins.

Conhecer melhor o aluno é essencial para mapear a origem do problema relacionado ao ensino de programação, visto o alto índice de reprovação e desistência na disciplina de Introdução à Programação nos cursos de nível superior da área de Computação. Isso é primordial para o professor, visto que a partir desse conhecimento torna-se possível a utilização de técnicas de ensino direcionadas aos tipos de alunos matriculados nas disciplinas, o que potencializa o processo de ensino aprendizagem.

1.4. Dissertação

Esta dissertação está escrita seguindo a seguinte estrutura:

- Capítulo 2: Fundamentação Teórica, onde são apresentados conceitos e definições necessários ao entendimento do presente trabalho;
- Capítulo 3: Trabalhos Relacionados, onde são apresentados trabalhos relacionados à proposta desta dissertação, estando subdividido de acordo com a natureza do trabalho, a saber: diagnóstico do problema de programação, intervenções propostas e tipos de iniciantes em programação;
- Capítulo 4: Modelagem Diagnóstica, onde é esquematizado o modelo utilizado para alcançar o diagnóstico para a identificação dos perfis cognitivos dos alunos iniciantes em programação. Além disso, são apresentados os resultados obtidos e uma discussão sobre eles;

- Capítulo 5: Resultados e Discussões, onde os resultados da pesquisa são apresentados;
- Capítulo 6: Conclusão e Trabalhos Futuros, onde os resultados, conclusões e trabalhos futuros são apresentados.

E, por fim, são apresentação as Referências Bibliográficas utilizadas.

2 Fundamentação Teórica

Neste capítulo é apresentada sucintamente uma base conceitual e tecnológica para esclarecer alguns elementos necessários à compreensão deste trabalho.

.

2.1. Habilidades em Programação

O Projeto Político Pedagógico (PPP) dos Cursos de Ciência da Computação definem as habilidades e competências que um profissional formado neste curso deve possuir. O PPP é desenvolvido com base nas diretrizes do Ministério da Educação e Cultura – MEC, e apresenta competências fundamentais a um egresso de Ciência da Computação [3]. Dentre essas competências se destacam:

- Modelar sistemas do mundo real buscando soluções sistematizadas através dos recursos disponíveis da área da Computação, Informática e Comunicações;
- Projetar e construir modelos computacionais, com base científica, para solução de problemas;
- Projetar e implementar sistemas complexos de alta qualidade, os quais requerem soluções computacionais complexas através de algoritmos;
- Gerenciar projetos de desenvolvimento de sistemas computacionais em geral;
- Estar capacitado a desenvolver, implantar e gerenciar sistemas de base tecnológica tais como: redes de computadores, banco de dados, inteligência artificial, sistemas distribuídos e computação científica;

Com base nas competências supracitadas, pode-se perceber a importância do aprendizado de programação em cursos de Ciência da Computação. Neste sentido é definido no PPP que "as disciplinas de Programação formam, com as de Engenharia de Software e a de Interação Homem-Máquina, a espinha dorsal do curso de Ciência da Computação" [3]. De acordo com as recomendações da Sociedade Brasileira de Computação – SBC [4], disciplinas de introdução à programação devem contemplar:

- Desenvolvimento de algoritmos;
- Tipos de dados básicos e estruturados;
- Comandos de uma linguagem de programação;
- Metodologia de desenvolvimento de programas;
- Modularidade e abstração.

Introdução à Programação (disciplina de Programação 1 nesta universidade) é o primeiro contato do aluno com esta área do curso, sendo sua ementa direcionada ao objetivo de desenvolver as habilidades necessárias ao aprendizado de programação. Os pontos abordados por esta disciplina são:

- Resolução de problemas e desenvolvimento de algoritmos;
- Análise do problema;
- Estratégias de solução;
- Representação e documentação;
- Programação de algoritmos usando uma linguagem de programação;
- Estruturação de programas;
- Noções de tipos e estrutura elementares de dados;
- Conceito de recursão e sua aplicação.

2.2. Suporte Estatístico

Para efeitos do presente trabalho fez-se necessário um estudo sobre técnicas de coeficiente de correlações e agrupamento de dados. Deste modo, nas seções seguintes serão tratados os dois tipos de correlação utilizados na pesquisa: Pearson e Ró de Spearman.

2.2.1. Correlações

Em Estatística o termo correlação, também chamado coeficiente de correlação, aponta a força e a direção do relacionamento linear entre duas variáveis aleatórias. Garson [5] define correlação como sendo "uma medida de associação bivariada (força) do grau de relacionamento entre duas variáveis". Para Moore [6] correlação é a forma de "mensurar a direção e o grau de relação linear entre duas variáveis quantitativas". Deste modo, pode-se dizer que correlação se refere à medida de relação entre duas variáveis qualquer. Existem vários coeficientes de correlação que são utilizados de acordo com a natureza do conjunto de dados [6].

O coeficiente de correlação de Pearson é uma medida de associação linear entre variáveis e pode ser obtido dividindo a covariância de duas variáveis pelo produto de seus desvios padrão [7], conforme a seguinte equação:

$$r = \frac{1}{n-1}\sum\left(\frac{xi-\bar{X}}{sx}\right)\left(\frac{yi-\bar{Y}}{sy}\right)$$

Coeficiente de Correlação de Pearson (r).

Já o coeficiente de Ró de Spearman, usa em vez do valor observado, apenas a ordem das observações. Assim, este coeficiente não se mostra sensível a assimetrias na distribuição ou a presença de *outliers*, não exigindo, portanto, que os dados estejam normalizados. Este coeficiente, em geral, é utilizado como alternativa ao de Pearson, quando a população de dados em questão tem a normalidade violada, sendo apropriado para os casos em que os dados não formam uma nuvem bem estabelecida, onde alguns pontos estão muito afastados dos demais, ou em situações onde parece existir uma relação crescente ou decrescente em formato de curva.

O coeficiente de Ró de Spearman pode ser calculado a partir da seguinte equação:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n^3 - n},$$

Coeficiente de Correlação de Ró de Spearman (ρ).

Onde n é o número de pares (x_i, y_i) e d_i é a diferença entre os postos de x e y.

Tanto os coeficientes de correlação de Pearson, quanto o de Ró de Spearman, variam de -1 a 1. Os valores dentro deste intervalo sugerem a força da relação entre as duas variáveis e o sinal indica se a direção da correlação é positiva ou negativa. Pode-se dizer que uma correlação é perfeita quando o valor encontrado para o coeficiente é 1 ou -1 e indica que o escore de uma variável pode ser determinado exatamente ao se saber o valor da outra. Já uma correlação de valor 0 indica que não há correlação entre as duas variáveis [8].

Entretanto, dificilmente o valor dessas correlações é igual aos valores extremos possíveis (0 e 1). Desse modo, faz-se necessário definir o tamanho desses coeficientes. Em [9] é estabelecida a seguinte classificação para os escores do coeficiente de correlação:

- $| r | \leq 0,3$; correlação fraca entre os dados;
- $0,4 \leq | r | \leq 0,6$; correlação moderada entre os dados;
- $| r | \geq 0,7$; correlação forte entre os dados;

Existem diversas ferramentas disponíveis que implementam as técnicas de correlação mencionadas. Uma delas é o ambiente MATLAB®, o qual foi utilizado no desenvolvimento dos algoritmos para cálculo de correlações utilizadas no presente trabalho. O MATLAB (MATrix LABoratory) é um ambiente interativo de alta performance direcionado ao cálculo numérico, onde são integrados a análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar. Neste ambiente, problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional.

2.2.2. Análise de *Cluster*

Análise de *Clusters* constitui um conjunto de procedimentos estatísticos que podem ser utilizados para classificar dados por meio da observação das semelhanças e diferenças entre elas, onde, ao final da análise, são gerados n grupos/*clusters* com características similares (Figura 1).

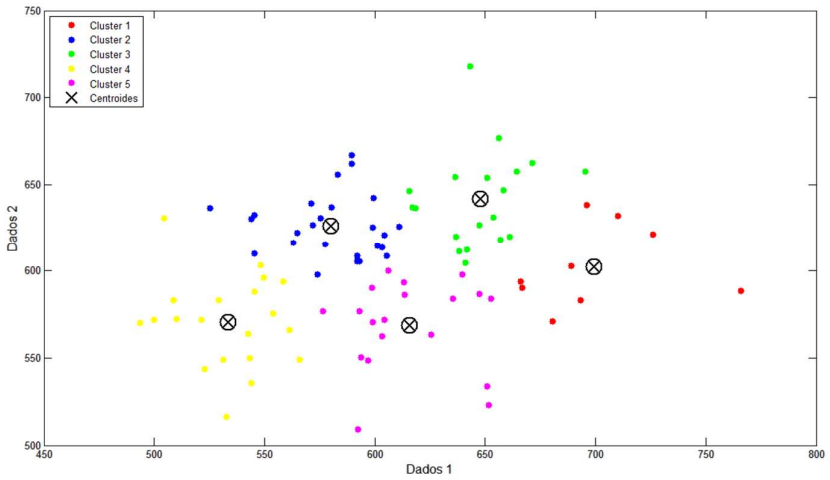


Figura 1: Exemplo da utilização do *k-Means* com 5 *clusters*.

Para definir *cluster*, utiliza-se a definição intuitiva apresentada pelo Professor Marcio Valli [10], onde são considerados dois objetos como pontos de um espaço **p** dimensional, onde cada uma das **p** variáveis representa um dos eixos do espaço. Um sistema de **p** coordenadas dimensionais é definido agora no espaço pelos valores das variáveis por objeto. Assim, uma definição de cluster aceitável é como sendo regiões contínuas que aparecem na massa relativamente grande de pontos no espaço, ou seja, regiões com alta densidade de pontos, separada de outras por regiões com pequena massa relativa (baixa densidade de pontos).

Pode-se visualizar essas divisões de regiões utilizando-se o Diagrama de Voronoi aos centroides de cada *cluster*, como pode ser observado a seguir.

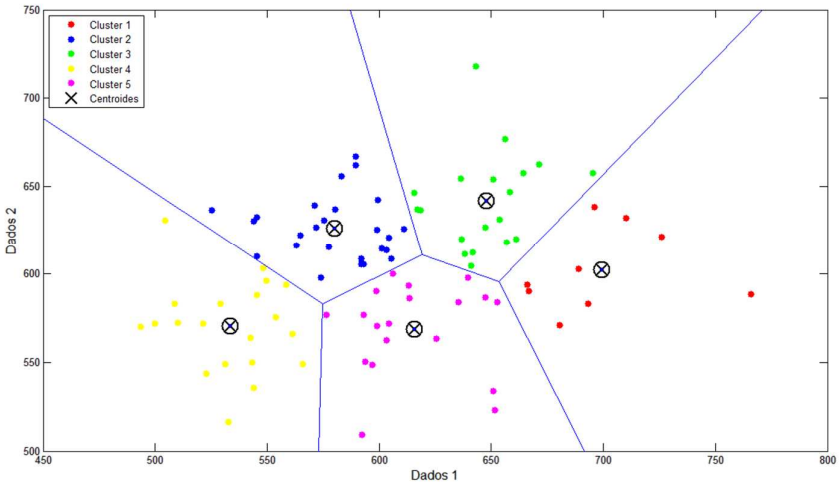


Figura 2: Aplicação do Diagrama de Voronoi aos centroides dos *clusters*.

O método de análise de *cluster* é uma técnica que analisa uma amostra composta por um conjunto de n indivíduos, sobre os quais se tem informação de v variáveis. Na análise, os indivíduos são agrupados em função das informações existentes, onde os grupos são formados por indivíduos muito semelhantes e que sejam muito diferentes dos indivíduos restantes. Existem várias técnicas utilizadas para a análise de *cluster*, dentre elas o *k-Means*, que é uma técnica de agrupamento geométrica bastante difundida e foi desenvolvida a partir do trabalho de Lloyd em 1982 [12], sendo seu principal objetivo classificar as informações a partir dos seus próprios dados. O *k-Means* é baseado na análise e comparação entre os valores numéricos dos dados, proporcionando uma classificação autônoma sem supervisão humana, sendo considerado um algoritmo não supervisionado de mineração de dados.

Existem diversas ferramentas disponíveis que implementam o algoritmo *k-Means*. Neste trabalho foi utilizada uma implementação do algoritmo encontrado no ambiente MATLAB®.

2.3. Ferramentas Computacionais de Apoio ao Aprendizado

Ultimamente têm sido criadas ferramentas computacionais para auxiliar o desenvolvimento de um curso de programação, as quais se prestam a auxiliar tanto estudantes quanto professores, oferecendo uma variedade de serviços. Por exemplo, tais ferramentas servem para os professores cadastrarem problemas para serem resolvidos pelos estudantes, e os tais estudantes por sua vez, submetem suas soluções, as quais receberão algum retorno avaliativo. Em parte, nesse sentido, existem os chamados "juízes", possuindo a função de receber o código, compilá-lo, passar entradas para o programa e fazer uma comparação das saídas do programa com o gabarito do problema.

A seguir são comentadas brevemente duas ferramentas utilizadas para incluir os propósitos descritos, além de oferecerem outras facilidades. Além disso, são apresentadas ferramentas e ambientes de propósitos mais gerais, dando conta de várias iniciativas pretendendo contribuir para facilitar a aprendizagem dos estudantes de programação.

2.3.1. Hoopaloo

Trata-se de uma ferramenta Web desenvolvida na Computação da Universidade Federal de Campina Grande [11]. Nela, podem ser cadastrados os exercícios de programação e seus respectivos testes de unidade, escritos usando o módulo PyUnit de Python.

2.3.2. The Huxley

The Huxley é uma ferramenta web, comercial, utilizada pela Universidade Federal de Alagoas como suporte no ensino de Programação. Esta ferramenta tem como principal funcionalidade o auxílio ao professor, dando suporte para o gerenciamento de exercícios e testes aplicados aos alunos.

O The Huxley é um ambiente para realização de exercícios de programação formado por um vasto banco de questões, cujo grau de dificuldade varia de acordo com o conhecimento e grau de maturidade em programação do usuário indo de iniciante a avançado. Além disso, a ferramenta é munida de um mecanismo de verificação de código, onde cada tentativa de resolução de determinado problema (resposta do problema) é recebida como uma submissão a este verificador, que avalia se o código enviado corresponde à solução para o problema escolhido e apresenta um feedback.

Deste modo, o professor pode criar questionários ou exames neste ambiente direcionado a um grupo específico de alunos, utilizando os problemas do banco de questões e tendo avaliação automática das respostas [12].

2.3.3. Ferramentas e Ambientes de Propósitos Gerais

Muito tem sido investido em pesquisas para a criação de ferramentas de software e abordagens metodológicas no intuito de apoiar o ensino em cursos introdutórios de programação. Em geral, os artigos nessa linha preocupam-se apenas com a criação de uma nova ferramenta, deixando de apresentar testes com a utilização da mesma. Em [13] é apresentada uma revisão sistemática da literatura focada nos trabalhos publicados sobre ensino e aprendizagem de programação para iniciantes, nos últimos dez anos, em dois importantes eventos nacionais: o Simpósio Brasileiro de Informática na Educação (SBIE) e o Workshop de Informática na Escola (WIE).

Esta análise demonstrou que 65% dos artigos publicados nesses eventos propõem ferramentas de software para apoio ao processo de ensino aprendizagem de programação para iniciantes, o que indica a preferência dos pesquisadores nacionais pela criação de novas ferramentas de software. Aproximadamente 23% dos artigos são direcionados a novas metodologias de ensino, 10% preocupa-se com a definição de novas linguagens. Outros tipos de artefatos são responsáveis pelos 2% restantes.

Nesse sentido, há pelo menos quatro décadas pesquisadores vem investindo no desenvolvimento de ambientes virtuais e ferramentas de software dedicadas ao ensino de programação. Uma das primeiras intervenções lançadas nesse sentido foi o Logo, projetada por Daniel G. Bobrow, Wally Feurzeig , Seymour Papert e Cynthia Solomon. LOGO [14] consta de uma linguagem de programação, um dialeto baseado em Lisp1, que foi desenvolvida como uma ferramenta para a aprendizagem de programação. Tem como características a modularidade, extensibilidade, interatividade e flexibilidade, além da comodidade de uma interface gráfica. Essa intervenção foi rotulada pelos seus idealizadores não como uma ferramenta, mas como uma filosofia de ensino [14] [15].

O ensino de programação a partir do LOGO se guia pela perspectiva da utilização da programação como ferramenta para fazer algo. Atividades de programação Logo contêm questões sobre matemática, linguagem, música, robótica, telecomunicações e ciência. Sendo utilizada para desenvolver aplicações multimídia e simulações nessas áreas. Seu projeto foi desenvolvido para atender a diferentes públicos, podendo ser utilizada para o ensino a iniciantes em programação, ensino de programação para crianças, ou mesmo o desenvolvimento de atividades complexas e projetos robustos para programadores experientes.

O LOGO evoluiu e tornou-se alicerce para a construção de vários outros ambientes gráficos para o ensino de programação, como é o caso do LEGO Blocks, LOGO Blocks [14] StarLogo, entre outros. Logo é um dos casos de sucesso no ambiente de ferramentas para o ensino de programação, pois se mostra eficaz e por isso foi adotado por várias instituições em diferentes partes do mundo.

Pesquisadores da Universidade Federal de Alagoas – UFAL lançaram em 2002 um ambiente integrado para auxílio ao ensino introdutório de fundamentos da Computação, o AMBAP [16], cujo objetivo é dar suporte ao ensino das disciplinas de Teoria da Computação, Arquitetura de Computadores e Programação, promovendo a

integração prática dos conceitos destas disciplinas. O AMBAP oferece ao aluno um ambiente simples que promove o entendimento de conceitos básicos da computação, pois dispõe de ferramentas para a construção e execução de programas, depuração e criação de processos que esclarecem com maior representação de detalhes a solução proposta.

No que diz respeito ao ensino de programação, o AMBAP aplica o conceito de níveis de aprendizado onde, num momento inicial, força o aluno a se desprender de aspectos relacionados à linguagem, fazendo com que o foco principal do iniciante seja a resolução de problemas. No nível mais avançado, o ambiente mantém todas as restrições e rigidez de uma linguagem de programação, respeitando regras de sintaxe e semântica. Para auxiliar o aluno na resolução de problemas, o ambiente fornece mecanismos de depuração, simulação de memória e representação gráfica da solução, facilitando o seu entendimento. Provido de uma ferramenta de tutoria, o AMBAP oferece aos iniciantes do curso de Computação uma relação individualizada com o ambiente, o que possibilita ao professor acompanhar o aprendizado do aluno.

Outra ferramenta desenvolvida por pesquisadores brasileiros foi o ASTRAL [17], um ambiente de programação para a produção de algoritmos e estrutura de dados animações com foco no ensino. Este ambiente foi inicialmente desenvolvido entre 1995-1997 no Instituto de Computação da Unicamp e visou atender as necessidades de preparação de diversos exercícios de implementação de estruturas de dados utilizando animações gráficas, de modo homogêneo sob uma interface consistente. A partir das características básicas do ambiente e de sua biblioteca de suporte, foram produzidos diversos destes exercícios que deram origem à primeira experiência de que temos conhecimento da aplicação da abordagem construtiva no ensino de estruturas de dados — o uso de mecanismos de animação para auxiliar na implementação de algoritmos pelos próprios estudantes.

Em [18] é apresentada uma ferramenta computacional didática para apoio ao ensino de algoritmos e estrutura de dados, o TBC-AED (Treinamento Baseado em Computador para Algoritmos e Estruturas de Dados). Esta ferramenta foi desenvolvida com o objetivo de analisar tópicos básicos de programação, onde todo o conteúdo apresentado é acompanhado de processo gráfico passo a passo, formando um repositório didático com visualização gráfica. Os tópicos de introdução à programação abordados

nesse ambiente são: Busca Binária, Métodos de Ordenação (Select Sort, Insert Sort, Bubble Sort, Merge Sort e Quick Sort), Alocação Estática e Dinâmica de Memória (Lista, Fila e Pilha) e Árvore Binária de Busca. Embora seja uma ferramenta de apoio a iniciantes, o TBC-AEB não dispõe de conteúdo sobre estruturas básicas de programação, tais como estruturas condicionais e estruturas de repetição. Além disso, não é apresentado nenhum indício de eficácia da ferramenta no processo de ensino aprendizagem dos alunos iniciantes em programação.

No intuito de criar um ambiente no qual os alunos iniciantes em programação pudessem aprender os conceitos necessários e desenvolver habilidades para a construção de programas de computadores, como estratégias de programação e resolução de problemas, foi desenvolvido o ALICE [19] [20] [21] [22]. O ALICE é um ambiente de programação gráfico interativo em 3D que foi desenvolvido na Carnegie Mellon University, que possibilita que um iniciante em programação desenvolva ambientes interativos em 3D. Oferece um mundo virtual ao aluno, povoado de diversos modelos do mundo real em três dimensões (por exemplo, animais e veículos). Com scripts simples, o usuário do ALICE pode definir a aparência e o comportamento desses objetos. A linguagem que o ambiente utiliza é o Python, nativamente funcional. Os scripts escritos pelos usuários do ALICE podem conter Funções (utilizando a sintaxe do Python), Estruturas de Decisão, Recursão, Estruturas de Repetição e criação de Eventos/Interações.

Graças ao retorno visual que o ALICE proporciona, estudantes conseguem ver imediatamente o resultado das estruturas criadas no script, relacionando as estruturas de programação a cada ação dos objetos da animação, proporcionando maior assimilação dos conceitos e estruturas de programação de computadores. Assim, o ALICE funciona como uma boa técnica para o ensino de programação à iniciantes, conduzindo-os a uma compreensão real do funcionamento das diferentes construções da linguagem [20]

Em 2008 foi lançado o [23], uma ferramenta que utiliza um design visual, enfatizando a manipulação de mídia para a criação de histórias animadas, jogos e apresentações interativas para despertar o interesse dos alunos. Este ambiente é baseado nas ideias dos jogos Logo e LogoBlocks [14]. O ambiente Scratch é formado por uma faixa fixa ao fundo e possui um número de *sprites* de móveis e vários ambientes pré-definidos para que o aluno monte seu cenário. Cada objeto possui um conjunto de

imagens e sons, e a eles scripts e variáveis são associados. Arrastando os objetos para a tela o aluno constrói o código-fonte do projeto estruturando-o em blocos de código, que é exibido à medida que o projeto vai sendo construído. Dessa maneira é possível acompanhar visualmente o que cada bloco de comando realiza. O ambiente possui um grupo de estruturas de controle (IF, IF-ELSE) e um com estruturas de repetição (*Repeat*, *Repeat – Until*) que podem ser manipulados pelo usuário. Além disso, a ferramenta suporta a utilização dessas estruturas de forma aninhada e apresenta ao usuário as ideias de variáveis locais e globais.

Essa ferramenta foi desenvolvida com o objetivo de auxiliar jovens entre 8 e 18 anos a aprender a programar. A priori ela foi distribuída como opção de entretenimento em uma *lan house*. Para analisar a eficácia do ambiente, foram coletados durante dois anos dados produzidos pelos usuários do Scratch, como: (i) arquivos de resumo dos projetos criados, contendo informações textuais como data, nome do autor, tipo de projeto, quantidade de comandos utilizados, tipos de comando e número total de pilhas; (ii) uma equipe formada por estudantes de graduação e pós-graduação acompanhou semanalmente o uso da ferramenta, realizando entrevistas semanais com seus usuários, o que possibilitou o acompanhamento do aprendizado em programação desses jovens. A partir da análise desses dados e dos projetos desenvolvidos utilizando o Scratch pode-se observar que pessoas sem qualquer conhecimento em programação compreenderam conceitos iniciais importantes, tais como o de variável, estruturas condicionais, estruturas de controle e funções. Não foram encontrados estudos que avaliam a utilização do Scratch em turmas de cursos iniciais de programação.

Outras ferramentas encontradas na literatura [24] [25] [26].

2.3.4. Enem

Criado em 1998, o Exame Nacional do Ensino Médio (Enem) tem o objetivo de avaliar o desempenho do estudante em relação aos seus conhecimentos na escolaridade básica. Participam do exame alunos que estão concluindo ou que já concluíram o ensino médio em anos anteriores. O exame avalia o aluno em cinco grandes áreas: Redação, Matemática e suas tecnologias, Ciências da Natureza e suas tecnologias, Ciências Humanas e suas tecnologias, Linguagens, Códigos e suas tecnologias.

As grandes áreas supracitadas são formadas pelas seguintes habilidades do ensino médio:

a) Redação: a produção textual de acordo com a norma culta da língua, com tema pré-definido pelo exame;
b) Matemática e suas tecnologias: resolução de problemas matemáticos com aplicações práticas no cotidiano, cujo conteúdo corresponde a todo o ensinado no ensino fundamental e médio;
c) Ciências da Natureza e suas tecnologias: corresponde as habilidades desenvolvidas nas disciplinas de Física e Química do ensino médio. As questões propostas são problemas multidisciplinares aplicados a situações do dia-a-dia;
d) Linguagens, Códigos e suas tecnologias: corresponde as habilidades em línguas. A prova é composta por questões relacionadas à língua Portuguesa e uma estrangeira, de escolha do candidato.

O Enem é realizado para avaliar a qualidade do ensino médio no país e seu resultado tem sido utilizado como parâmetro de seleção o ingresso no ensino superior, através do SiSU – Sistema de Seleção Unificada. Na maioria nas universidades públicas, a nota do Enem é o critério de seleção para o ingresso nos cursos superiores. Já nas universidades privadas, o exame é utilizado para seleção de bolsas de estudo, através do PROUNI – Programa Universidade para Todos. A UFAL utiliza o Enem como forma de ingresso.

3 Trabalhos Relacionados

Neste capítulo são apresentados trabalhos relacionados ao estudo de problemas na aprendizagem de programação, considerando-se um curso introdutório. Assim, apresentam-se desde trabalhos que serviram de apoio ao desenvolvimento desta dissertação, aos que são considerados correlacionados mais relacionados por abordar aspectos das questões de pesquisa descritas no Capítulo 1, podendo ser de algum modo comparado aos resultados obtidos na presente pesquisa.

3.1. Resolução de Problemas e Programação

É comum o desempenho insuficiente em disciplinas de programação ser relacionados à habilidade de resolução de problemas. Muitos autores relatam que alunos

com baixa capacidade para resolver problemas são fadados ao insucesso em disciplinas iniciais de programação, como pode ser observado em [27].

Sheard *et. al* [28] realizaram uma análise dos trabalhos de pesquisa relacionados ao ensino de programação publicados entre 2005 e 2008 nos seguintes eventos: ICER – *International Computing Education Research Workshop*, SIGCSE – *Technical Symposium on Computer Science Education*, ITiCSE – *Innovation and Technology in Computer Science Education*, ACE – *Australasian Computing Education Conference, Baltic Sea Conference on Computing Education Research* (*Koli Calling*) e na conferencia anual New *Zealand's National Advisory Committee on Computing Qualifications* (NACCQ). Dos 164 artigos avaliados 40% teve como foco de pesquisa habilidade / aptidão / entendimento do curso de programação, 35% realizaram pesquisas em técnicas de ensino / aprendizagem / avaliação, 9% dedicaram-se ao estudo de ferramentas de ensino / aprendizagem / avaliação, ensino / aprendizagem teorias e modelos tiveram 9% dos artigos submetidos e os 7% restante trataram de assuntos como currículo do curso de computação, acessibilidade, pesquisa entre outros.

George Polya em seu livro "*How to Solve It*" [29] descreve problema como sendo "um obstáculo que se deve transpor" e processo de resolução de problemas para ele é um conjunto de passos bem definidos e iterativos que se deve adotar para transpor o problema, ou seja, alcançar uma solução. São os passos para resolver um problema:

a) Compreender o problema: onde se deve fazer perguntas a respeito do problema, identificar qual é a incógnita, quais são os dados do problema e quais as condições;
b) Construção de uma estratégia de resolução: nesta etapa devem-se encontrar as conexões entre os dados e a incógnita, se necessário utilizar problemas auxiliares ou particulares;
c) Execução da estratégia;
d) Revisão da solução: verifica-se se o resultado alcançado resolve o problema, em caso negativo reinicia-se o processo.

Nesse sentido, Lister *et al.* [2] sugere que baixa habilidade de resolução de problemas seja uma explicação popular para a dificuldade apresentada pelos iniciantes em programação e propõe uma avaliação alternativa onde o as dificuldades em aprender a programar podem ser ocasionadas por duas razões: frágil compreensão dos princípios

de sintaxe e semântica de uma linguagem de programação e fragilidade na habilidade de articular correlações para alcançar a solução de um problema.

Como "baixa habilidade de resolução de problemas" Lister *et al*. Define como a capacidade resumida ou incapacidade para realizar as seguintes aços: (i) compreender o problema; (ii) dividir o problema em sub problemas (problemas menores); (iii) traduzir os sub problemas em pequenas soluções; (iv) integrar estas soluções, recompondo assim o problema; e (v) avaliar a solução final e interar o processo caso seja necessário.

Para esta verificação foram realizados dois testes com estudantes selecionados de sete países distintos, onde o primeiro teste avaliou a capacidade desses estudantes de prever o resultado da execução de um pequeno pedaço de código, constatando assim sua compreensão de sintaxe e semântica. O segundo teste consistiu em fornecer ao aluno um código pronto com algumas lacunas e solicitar que ele completasse esses espaços corretamente.

Há, entretanto, uma similaridade aos passos descritos como ideais para o bom desempenho durante a elucidação de questões de programação estudadas por iniciantes, conforme propõe [30]. Deste modo, é possível observar uma confusão entre os conceitos de resolução de problemas e de programação, pois é sutil o limiar entre essas atividades, onde o primeiro resume-se a realizar o segundo utilizando uma linguagem específica. Em [31] é feita uma investigação a respeito do desenvolvimento das habilidades de resolver problemas a partir da aquisição de conhecimentos em programação. Neste trabalho é proposto que o ensino de programação é um método de ensino de resolução de problemas. Uma conclusão similar é alcançada por Davies em [32]:

> "*modelos emergentes de comportamento programação sugerem um processo de resolução de problemas incrementais onde a estratégia é determinada por episódios de resolução de problemas localizados e frequente reavaliação da solução*".

Um experimento realizado por R. Jonhson [33] identificou que o escore da capacidade de resolução de problema de um grupo de alunos aumentou após um curso introdutório de programação. O objetivo de R. Johnson era estabelecer uma correlação entre habilidades de resolução de problemas em função das habilidades de programação. Para isso ele aplicou um teste de raciocínio lógico a um grupo de alunos no início e no

final de um curso de introdução há programação. Os resultados apontam que alunos que tinham alcançado boa pontuação no teste inicial apresentaram um bom desempenho na disciplina de programação e no segundo teste de raciocínio lógico alcançaram pontuação significativamente maior. Já alunos com baixa pontuação no primeiro teste de raciocínio lógico obtiveram desempenho razoável na disciplina de programação e aumentaram seu escore no segundo teste. Esses resultados apontam que desenvolver a habilidade para programação desenvolve também as habilidades necessárias para resolução de problemas.

Deste modo, pode-se afirmar que habilidades de resolução de problemas e habilidades de programação estão intimamente relacionadas, onde o sucesso em programação depende fortemente das capacidades de resolução de problemas apresentadas pelo iniciante.

3.2. Correlatos para Iniciantes em Programação

Grande tem sido o esforço para descobrir qual a forma que um indivíduo aprende a programar, ou qual o processo cognitivo utilizado para realizar esta tarefa, qual a melhor ferramenta ou método utilizar em aula.

Um grupo de iniciantes em programação é formado por indivíduos variados, onde suas habilidades, motivação e conhecimento prévio definem o sucesso ou fracasso em um curso introdutório. Algumas pesquisas buscaram classificar esses iniciantes de acordo com o seu nível de desempenho e definiram tipos de novatos em programação, outras levaram em consideração seu grau de motivação e aspectos psicológicos envolvidos no processo de aprendizagem.

Um dos pioneiros na tentativa de classificar o aluno para conhecê-lo melhor foi Mayer *et al*. [34]. Seu trabalho descreve a pluralidade de indivíduos em um grupo de iniciantes em programação e leva em consideração para o sucesso em programação, além de medidas de inteligência geral, fatores como origem do individuo, habilidades, níveis de motivação e frustração.

Para Robins *et al* [35] um dos fatores mais fortes que predizem o sucesso em disciplinas de introdução a programação é o nível em que um aluno pretende chegar seu grau de motivaçã. Robins *et al* realizou um estudo em um universidade australiana onde a disciplina introdutória de programação é uma eletiva para todos os cursos da

universidade, sendo obrigatória apenas para os cursos diretamente ligados à Computação. Nesse estudo foram aplicados questionários a todos os alunos matriculados no curso entre o ano tal e o tal, onde foram realizadas questões relativas à área de estudo original do aluno, qual o curso universitário no qual estava matriculado, qual seu objetivo na disciplina, qual seu conhecimento em programação e quais as suas expectativas com o curso de programação. Ao final da disciplina, percebeu-se que os alunos que obtiveram sucesso foram os que tinham como objetivo ser bom na disciplina, ou seja, realmente aprender a programar. Dessa forma, a primeira classificação a respeito de iniciantes em programação foi estabelecida pelo seu grau de motivação.

Uma segunda classificação é apresentada por Perkins classifica os novatos em programação de acordo com seu comportamento. Ele distingue dois tipos de novatos: *stoopers* e *moovers*. Onde *stoopers* ao se depararem com um problema ou com a ausência de direção clara para realizar determinada tarefa, para. Como quem se sente incapaz de solucionar o problema por conta própria [36]. Os *moovers* são estudantes que, mesmo após alcançar alguma solução, continuam tentando, experimentando e modificando seu código. Em geral, esse grupo de alunos utiliza o *feedback* recebido de forma positiva, fazendo uso eficaz dele. Entretanto, há de ser considerado ainda um subtipo de *moovers*, denominado "*moovers extremos*". Os *moovers* extremos são iniciantes que não são capazes de rastrear/executar mentalmente o seu programa, fazendo alterações aleatórias em seu código. Esse grupo de alunos, tal qual os *stoopers*, apresentam pouca chance de sucesso no estudo de programação.

Em [37] é apresentado uma relação entre estilos cognitivos, personalidade do indivíduo e fase de desenvolvimento de um programa, sendo consideradas fases de desenvolvimento as seguintes etapas: representação do problema, planejamento de solução, codificação e depuração. Catherine [37] apresenta cinco estilos cognitivos:

- Analítico/Holístico;
- Pensamento Divergente;
- Área de Dependência/Independência;
- Lateralização Hemisférica;
- Impulsividade/Reflexividade;

Apesar da relação, não fica determinado se o estilo cognitivo é definidor de um bom ou mau programador, nem quais técnicas podem ser aplicadas para o desenvolvimento da capacidade de programar em indivíduos que apresentam um dos estilos cognitivos acima citados.

3.3. Síntese do Capítulo

Neste capítulo foram apresentados trabalhos de apoio ao desenvolvimento da presente pesquisa, trazendo uma discussão sobre resolução de problemas circunstanciada à atividade de programação. Além disso, discutiu-se com mais detalhes os principais trabalhos que se relacionam diretamente com os problemas tratados nesta dissertação, os quais abordam questões sobre correlações e agrupamento de dados relacionados às atividades desenvolvidas pelos estudantes.

4 Modelagem Diagnóstica

Neste capítulo serão apresentados os objetivos da pesquisa bem como o caminho percorrido sua execução. Além disso, serão expostos detalhes sobre a seleção dos dados, as técnicas utilizadas para agrupamento e classificação, que resulta em um diagnóstico inicial sobre características de iniciantes em programação.

4.1. Problemática

Tradicionalmente, a atividade de programar é rotulada como difícil entre a maioria dos iniciantes, o que torna o desenvolvimento dessa atividade um desafio para o aluno. Assim, realizar a atividade de programar exige que o aluno iniciante possua um conjunto de habilidades consideradas essenciais a essa tarefa, como: bom raciocínio lógico, boa capacidade de resolução de problemas e capacidade de expressão a partir de um alfabeto limitado.

Essas necessidades fazem com que o ensino de programação, e consequentemente o aprendizado de programação, se torne um desafio tanto para professores, quanto para os alunos dos cursos introdutórios. Na tentativa de amenizar esse problema, várias soluções vêm sendo propostas, que variam de intervenções metodológicas ao uso de ferramentas de *software* como base para o ensino de programação, tal como discutido na seção 2. No entanto, essas ferramentas têm sido desenvolvidas sem considerar qualquer

característica sobre o iniciante em programação. Sendo, portanto, um conjunto de ferramentas de modelagem genérica que visam auxiliar um público específico, mesmo sem possuir qualquer heurística sobre ele. A isso se atribui a falta de notícias sobre a eficácia de tais ferramentas.

Desse modo, esta pesquisa visa conhecer quem é o estudante de programação inicial matriculado na Universidade Federal de Alagoas, observando-o em diferentes contextos. Embora o experimento tenha um caráter local, o seu desenvolvimento foi substanciado na literatura internacional. Além disso, acredita-se que os alunos matriculados na disciplina de introdução à computação da UFAL sejam semelhantes a estudantes matriculados em cursos similares em outras instituições, nacionais e internacionais.

4.2. Metodologia Para Execução da Pesquisa

O desenvolvimento desta pesquisa se deu através da análise de conjuntos de dados, qualitativos e quantitativos, de alunos que cursaram Ciência da Computação na Universidade Federal de Alagoas no período de 2006 a 2013. Esta análise teve como objetivo identificar características comuns a grupos de alunos matriculados na disciplina de Introdução à Programação (Programação 1) nesta instituição. Para tanto, foram selecionados dados de duas fases da vida do egresso: anterior ao ingresso na universidade e o período de curso da disciplina de introdução à programação. Para a análise qualitativa, foram considerados apenas indivíduos matriculados no período 2010 – 2013, visto que os dados utilizados são provenientes do ambiente *The Huxley,* que começou a ser utilizado nesta instituição a partir de 2010.

Os dados que apresentam informações relativas à vida acadêmica do aluno antes do ingresso na universidade são suas notas do ENEM – Exame Nacional do Ensino Médio, que é utilizado como meio de ingresso em cursos superiores da UFAL. Os dados relativos ao período do curso de introdução a programação referem-se às médias dos alunos nas disciplinas cursadas no mesmo período que introdução à programação, quais sejam: Geometria Analítica, Programação 1 (introdução à programação), Laboratório de Programação, Internet e Web, Inglês Instrumental e Fundamentos de Matemática. Ainda relativo ao período de curso da disciplina, serão avaliados os códigos-fontes produzidos pelos alunos em exercícios da disciplina de introdução à programação.

4.2.1. Conjunto de Dados para Análise e Seleção dos Valores Observados

Os dados utilizados para a realização da pesquisa foram (i) as notas do ENEM, as notas da disciplina de Programação 1 (equivalente a introdução à programação em outros cursos de nível superior), notas das disciplinas consideradas correlatas e cursadas no mesmo período que Programação 1 e código-fonte produzidos pelos alunos de Programação 1.

Ocorreram alterações na grade curricular e do regime do curso de Ciência da Computação na UFAL durante o período dos dados coletados. Fazendo necessários alguns esclarecimentos:

I. Entre os anos de 2006 e 2009, o regime do curso era anual e as disciplinas ofertadas no primeiro ano de curso eram Cálculo 1, Álgebra Linear, Inglês Instrumental, Introdução à Computação, Programação 1, Laboratório de Programação, Probabilidade e Estatística e Lógica aplicada à programação;
II. Entre os anos de 2010 e 2011, o curso passou a ter regime semestral. As disciplinas estudadas no primeiro período eram Cálculo 1, Geometria Analítica, Inglês Instrumental, Introdução à Computação, Laboratório de Programação e Programação 1;
III. Em 2012 foram realizadas alterações na grade curricular do primeiro semestre, sendo definido como disciplinas iniciais Fundamentos de Matemática, Geometria Analítica, Inglês Instrumental, Introdução à Computação, Laboratório de Programação e Programação 1.
IV. O ambiente The Huxley começou a ser utilizado em julho de 2010.

Durante a catalogação dos dados foram identificadas algumas anomalias, para sua correção, foram definidos critérios de seleção, tais quais foram utilizados em [38]. Esses critérios têm como objetivo excluir da amostra observações com ruído. Os critérios de seleção aplicados na amostra foram os seguintes:

CS_1) Reprovação por falta;

CS_2) Aproveitamento de estudo por equivalência[1];

CS_3) Trancamento da disciplina;

CS_4) Desempenho geral igual à zero, que foi considerado como desistência.

Durante a aplicação dos critérios de seleção foram tomadas algumas decisões *ad hoc*, tais quais seguem:

D_1) As observações dos anos de 2006 até 2009 não apresentam desempenho no processo seletivo conhecido como SISU, que utiliza o ENEM como nota de ingresso na universidade. Nesse período foi utilizado o PSS – Processo Seletivo Simplificado, sendo o ENEM adotado nesta instituição apenas a partir do ano de 2010. Deste modo, optou-se por utilizar as observações relativas ao período de 2006 – 2009 apenas quando forem verificadas correlações entre as disciplinas do semestre de introdução à programação, e do semestre posterior a este;

D_2) A disciplina de Geometria Analítica não entrou na amostra no período que vai de 2011 a 2013. Nesse intervalo de tempo houve um número alto de desistências e trancamentos, que foram atribuídos à ausência de professor no inicio da disciplina nos três períodos consecutivos.

D_3) Foram consideradas disciplinas correlatas ao ensino de programação as seguintes: Cálculo 1, Fundamentos de Matemática, Lógica aplicada à programação, Laboratório de Programação e Introdução à Computação. Tendo sido as quatro primeiras selecionadas pelo seu caráter matemático e estimulo a resolução de problemas e, a última, por trazer ao aluno conceitos necessárias para auxiliar o aluno no desenvolvimento das abstrações utilizadas em introdução à programação.

Após a aplicação desses critérios de seleção, os dados se apresentaram da seguinte maneira:

Ano Letivo	**Quantidade de Indivíduo**
2006	58
2007	59
2008	73
2009	40
2010	66

[1] Aproveitamento de estudo por equivalência: ocorre quando o aluno cursou a disciplina em outra instituição ou curso de nível superior e teve esse crédito aproveitado em sua grade, deixando de cursar a disciplina no curso/instituição atual.

2011	72
2012	63
2013	29
Total	**460**

Tabela 1: Total de observações utilizadas na amostra relativa a análise Quantitativa.

Ano Letivo	**Quantidade de Indivíduo**
2010	60
2011	70
2012	63
2013	29
Total	230

Tabela 2: Total de observações utilizadas na amostra relativa a análise Qualitativa.

4.3. Execução do estudo

A presente pesquisa foi dividida em oito etapas:

I. O estudo das correlações entre as disciplinas do ensino médio (que caracterizam a vida pregressa do iniciante) e a disciplina de introdução à programação, sendo:
    a. Ciências Humanas e suas Tecnologias x Programação 1;
    b. Ciências da Natureza e suas Tecnologias x Programação 1;
    c. Línguas x Programação 1;
    d. Matemática x Programação 1;
    e. Redação x Programação 1.

II. O estudo das correlações entre introdução à programação e as disciplinas cursadas no mesmo período que ela. As correlações estudadas serão:
    a. Fundamentos de Matemática x Programação 1;
    b. Laboratório de Programação x Programação 1;
    c. Internet e Web x Programação 1;
    d. Introdução à Computação x Programação 1.

III. Geração de *clusters* utilizando o algoritmo *k-Means* no conjunto de dados relacionados à vida pregressa do iniciante (dados do ENEM);

IV. Geração de *clusters* utilizando o algoritmo *k-Means* no conjunto de dados relacionados ao período letivo onde foi realizado o curso introdutório de programação;

V. Comparação dos *clusters* obtidos nos passos III, IV e V para a geração de um único *cluster*;

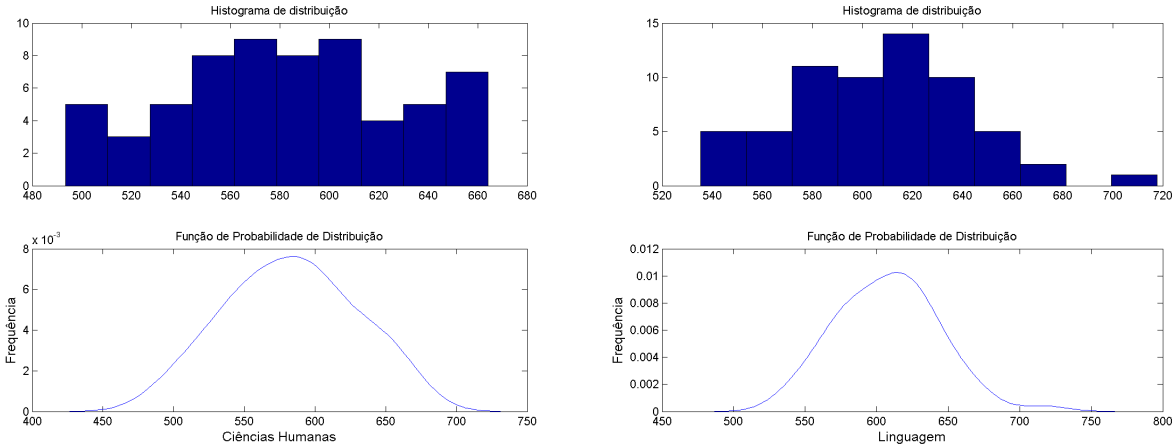
VI. Análise do código-fonte produzido por uma amostra de cada *cluster* identificado;

VII. Geração de um diagnóstico a respeito de características quantitativas e qualitativas dos iniciantes em programação.

Além dos pontos acima, foi verificado o percentual de aprovação, reprovação e evasão do curso de Ciência da Computação. A seguir será apresentada, em detalhes, a execução de cada uma dessas etapas.

4.3.1. Avaliações Quantitativas

Durante a realização das etapas I e II foi necessário à aplicação de uma técnica para o cálculo do coeficiente de correlação. Na definição de qual coeficiente de correlação utilizar, os dados de cada disciplina foram estudados através de um histograma de distribuição, onde foi possível relacionar ao mesmo uma função de probabilidade de distribuição, que é a curva que mostra onde os dados estão ocorrendo com maior frequência, conforme Figura 3, Figura 4, Figura 5.



(a) (b)

Figura 3: Histograma e Função de Probabilidade de Distribuição Dados do ENEM : (a) Ciências Humanas, (b) Linguagem.

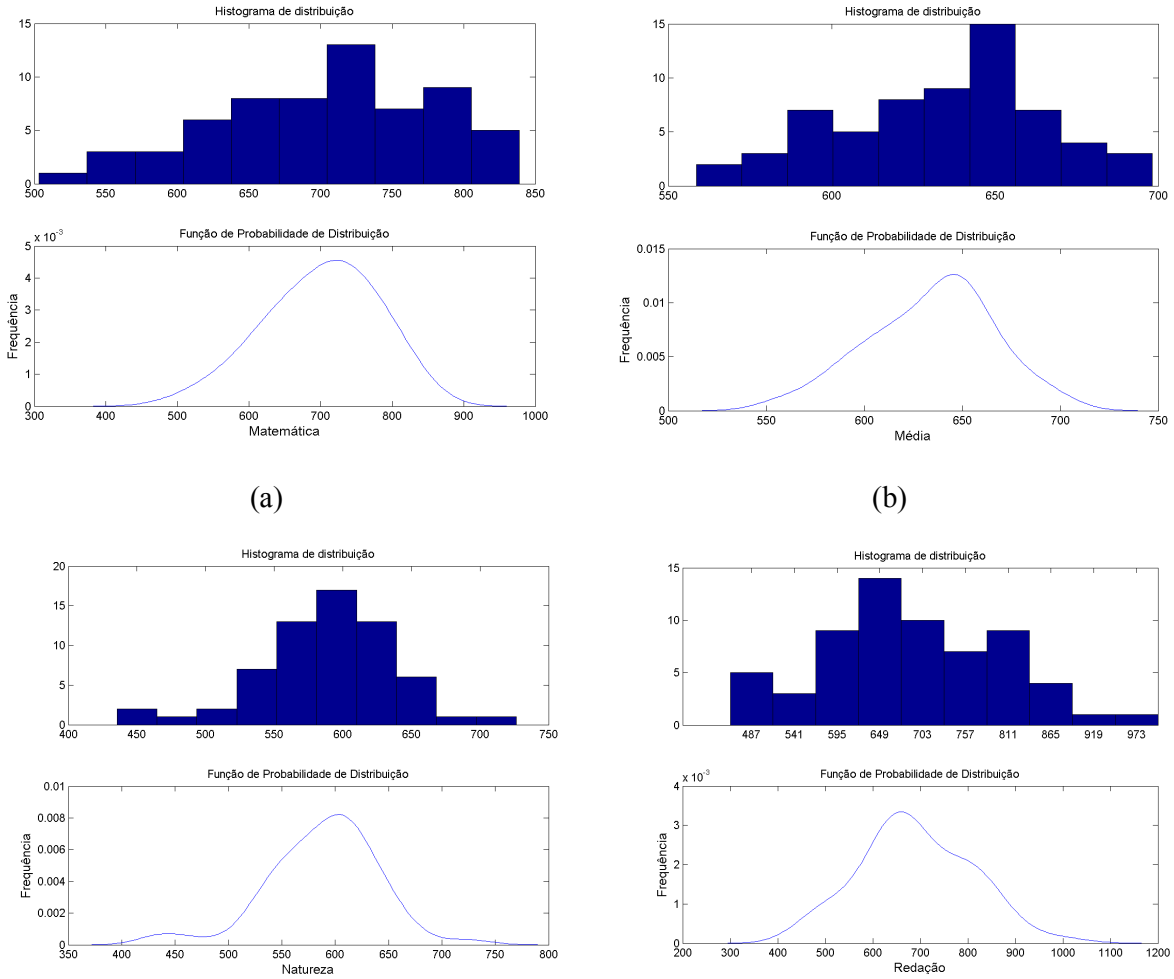


Figura 4: Histograma e Função de Probabilidade de Distribuição Dados do Enem : (a) Matemática, (b) Média, (c) Ciência da Natureza e (d) Redação.

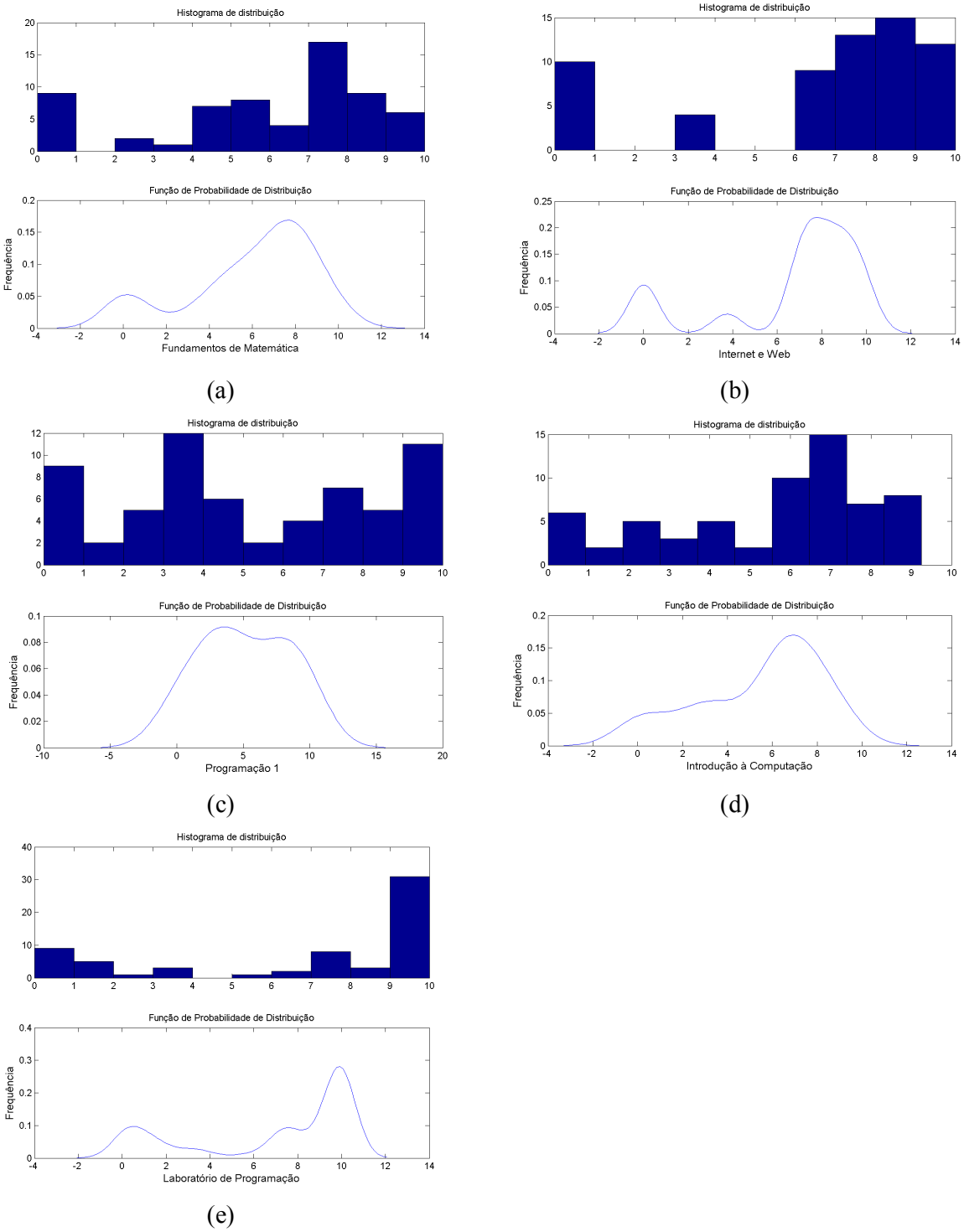


Figura 5: Disciplinas: (a) Fundamentos de Matemática, (b) Internet e Web, (c) Programação 1, (d) Introdução à Computação e (e) Laboratório de Programação.

De acordo com os gráficos apresentados nas figuras acima, os dados não seguem uma distribuição normal. Devido a este fato, foi utilizado o coeficiente de correlação de Ró de Spearman (ρ) para realizar as verificações de correlações dos dados.

4.3.1.1. Estatística do Curso de Introdução à Programação

Para fundamentar a importância desta pesquisa, foram levantados os quantitativos relacionados aos índices de aprovação, reprovação e evasão no curso de computação, conforme Tabela 3.

	2006	2007	2008	2009	2010	2011	2012
Aprovados	53.8%	55.7%	59.7%	46.8%	48.8%	46.5%	35.6%
Reprovados	21.8%	12.9%	12.5%	12.7%	9.3%	18.2%	26.7%
Desistentes	24.4%	31.4%	27.8%	40.5%	41.9%	35.3%	37.7%

Tabela 3: Percentuais de Aprovação, Reprovação e Evasão da disciplina de Programação 1.

4.3.1.2. Correlações

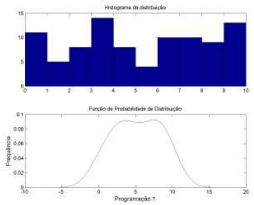
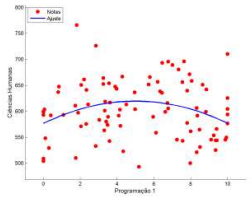
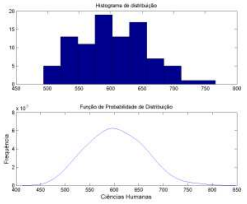
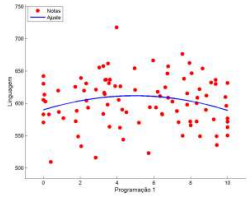
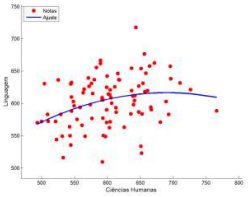
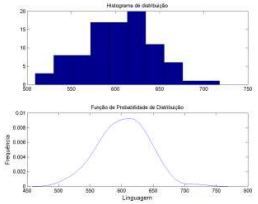
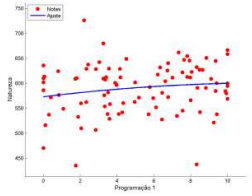
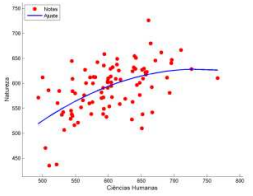
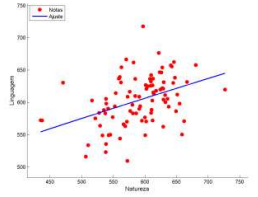
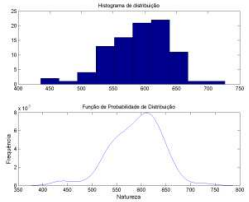
A priori foram verificadas as correlações existentes entre as áreas de estudo do Enem e a disciplina de Programação 1. Em seguida, verificou-se as correlações entre as Programação 1 e as disciplinas cursadas no mesmo período estabelecidas como correlacionadas ao ensino de programação.

Conforme especificado no item anterior, a primeira etapa para a geração dos grupos de alunos é a verificação das correlações entre o desempenho dos indivíduos da amostra nas áreas de estudo abordadas no ENEM e na disciplina de Programação 1. Para isto, foi utilizado o coeficiente de correlação de Ró de Spearman (ρ). A tabela 4 abaixo apresenta os índices de correlação dessas disciplinas.

Os coeficientes de correlação obtidos nessa análise demonstram fraca ou nenhuma relação entre o desempenho do aluno nas grandes áreas do Exame Nacional do Ensino Médio e o desempenho na disciplina de Programação 1. Conforme pode ser observado na Tabela 4, os coeficientes de correlação verificados variaram de fraco a moderado, não apresentando nenhuma correlação forte. As correlações que apresentaram maior fator de correlação foi Ciências da Natureza e suas tecnologias x Ciências Humanas,

com ρ = 57,5% de correlação e Ciências da Natureza e suas tecnologias x Matemática, com ρ = 54,8% de correlação. .

Na segunda parte da análise, foram verificadas as correlações entre Programação 1 e as disciplinas consideradas correlatas, cursadas no período inicial do curso. Essas correlações foram realizadas considerando as alterações curriculares e regimentais da instituição, sendo dividida em três grupos: (i) correlações para o período de 2006 a 2009, (ii) correlações para o período 2010 a 2011 e (iii) correlações para o período 2012 a 2013. As Tabelas Tabela 5, Tabela 6 e Tabela 7 apresentam os resultados dessas correlações.

 Programação 1	ρ = -0.0347 valor de p = 0.743	ρ = -0.0490 valor de p = 0.643	ρ = 0.1630 valor de p = 0.121	ρ = 0.2196 valor de p = 0.035	ρ = 0.0205 valor de p = 0.8463
	 Ciências Humanas	ρ = 0.3102 valor de p < 0,05	ρ = 0.5214 valor de p < 0,05	ρ = 0.2666 valor de p = < 0,05	ρ = -0.1046 valor de p = 0.3211
		 Linguagem	ρ = 0.4285 valor de p < 0,05	ρ = 0.2192 valor de p < 0,05	ρ = -0.1712 valor de p = 0.1028
			 Ciências Natureza	ρ = 0.5629 valor de p < 0,05	ρ = -0.3510 valor de p < 0,05

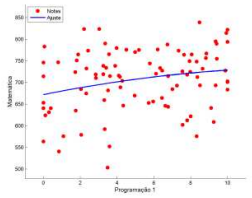
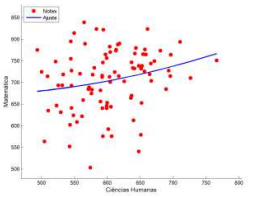
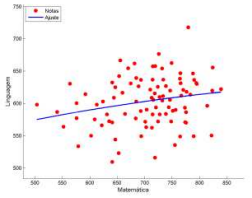
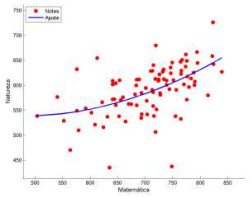
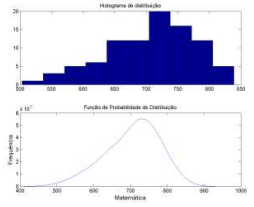
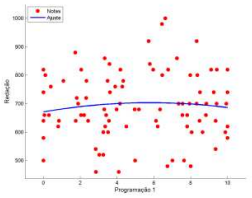
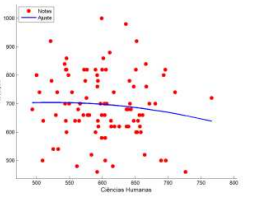
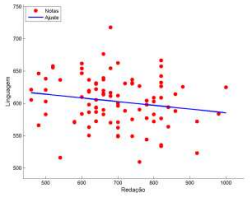
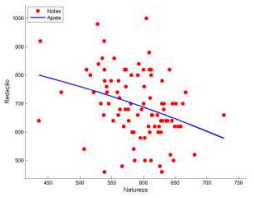
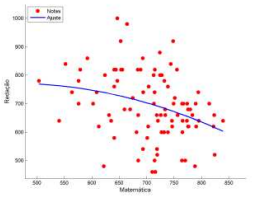
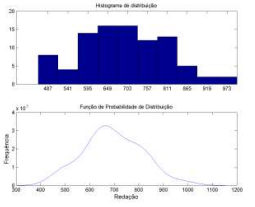
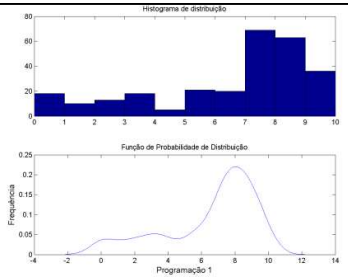
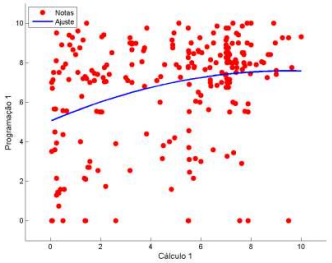
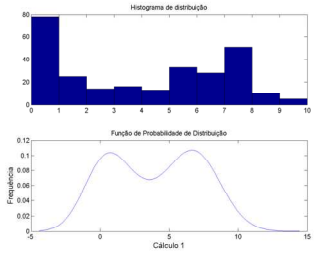
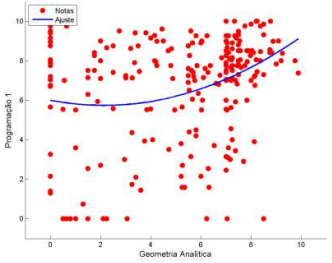
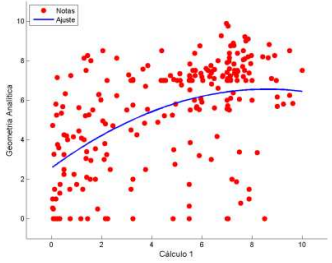
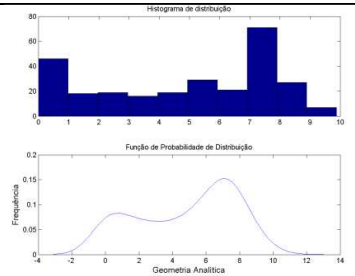
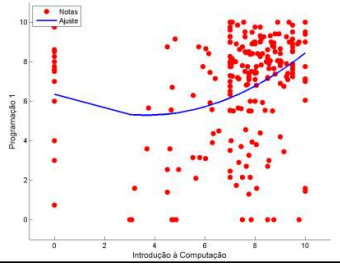
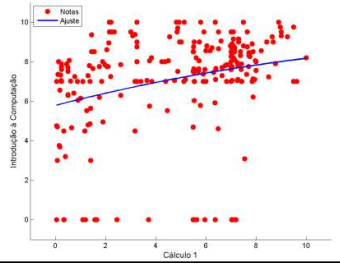
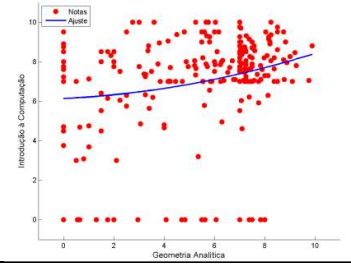
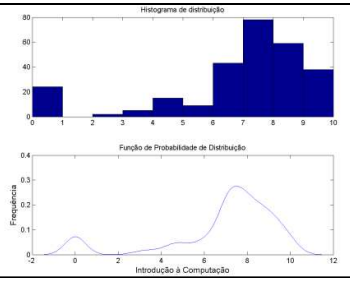
				 Matemática	ρ = -0.3234 valor de p < 0,05
					 Redação

Tabela 4: Correlações entre o desempenho na disciplina de Programação 1 e o desempenho das áreas de verificação do Enem relativos aos anos de 2012 e 2013.

.

 Programação 1	ρ = 0,2640 valor de p < 0,05	ρ = 0,2623 valor de p < 0,05	ρ = 0,2958 valor de p < 0,05	ρ = 0.5686 valor de p < 0,05
	 Cálculo 1	ρ = 0,5651 valor de p < 0,05	ρ = 0,3499 valor de p < 0,05	ρ = 0,2462 valor de p < 0,05
		 Geometria Analítica	ρ = 0,2419 valor de p < 0,05	ρ = 0,2219 valor de p < 0,05
				ρ = 0,3987 valor de p < 0,05

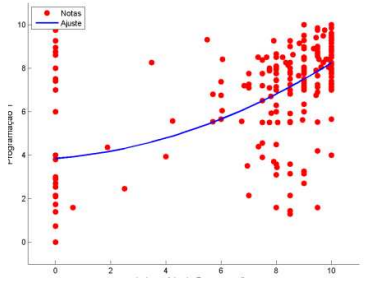
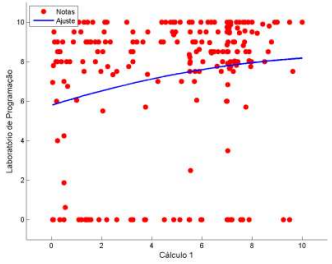
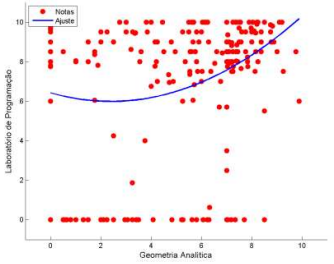
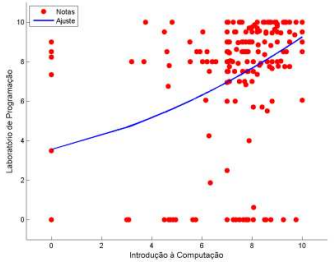
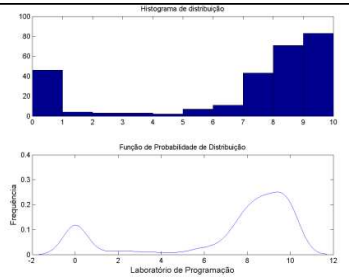
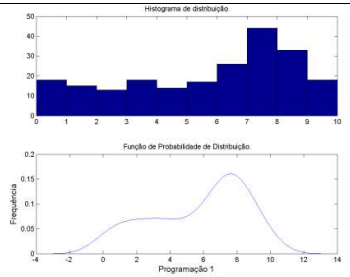
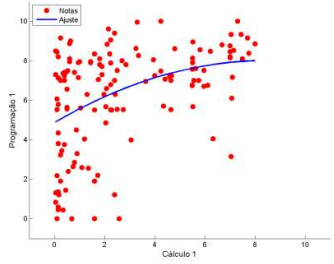
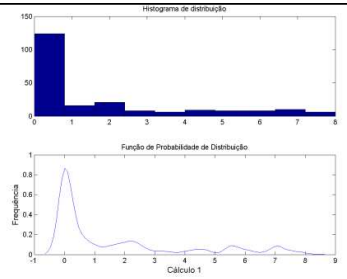
			Introd. à Computação	
				 Lab. Programação

Tabela 5: Correlações entre o desempenho na disciplina de Programação 1 e o desempenho as disciplinas ditas correlatas no período de 2006 a 2009.

 Programação 1	ρ = 0,3614 valor de p < 0,05	ρ = 0,4608 valor de p < 0,05	ρ = 0,2192 valor de p < 0,05	ρ = 0,4838 valor de p < 0,05
	 Cálculo 1	ρ = 0,5651 valor de p < 0,05	ρ = -0,1238 valor de p = 0,0693	ρ = -0,0977 valor de p = 0,1525

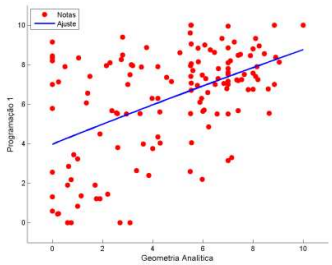
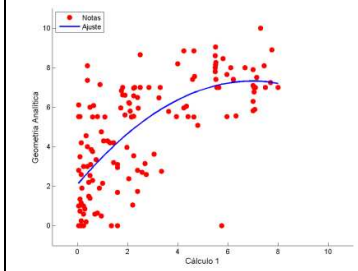
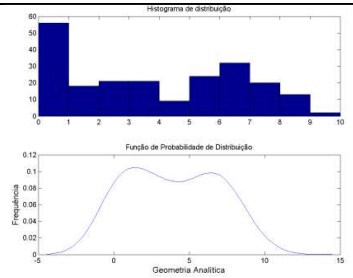
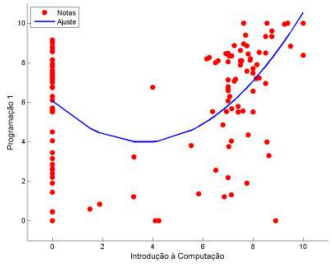
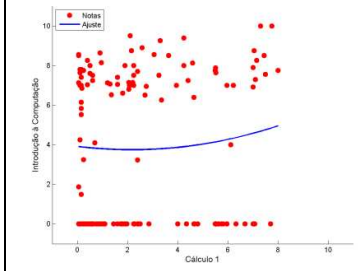
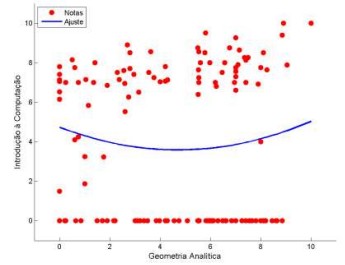
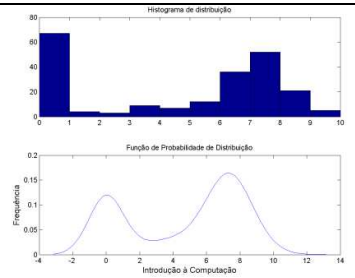
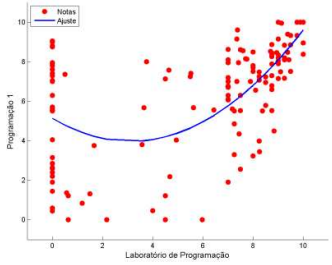
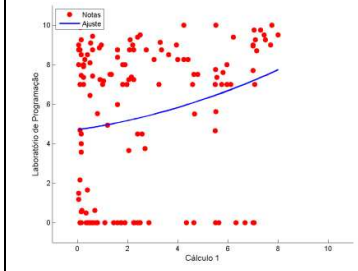
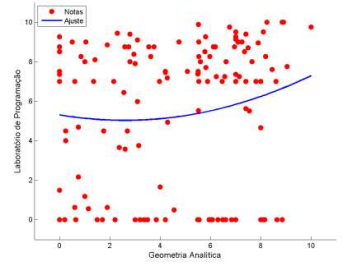
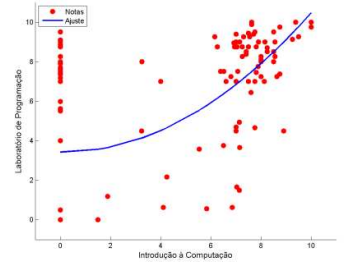
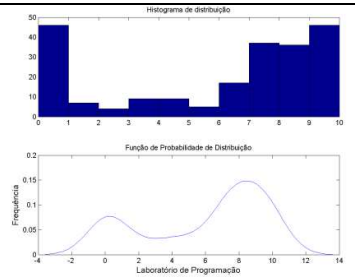
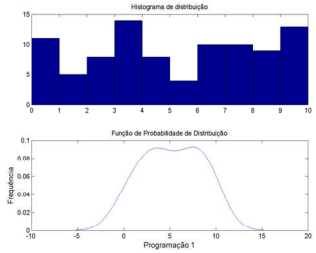
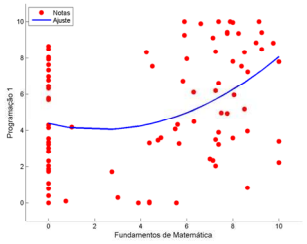
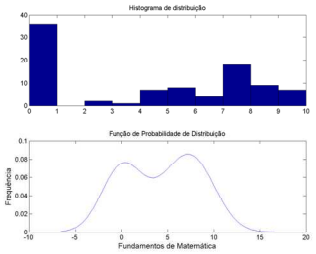
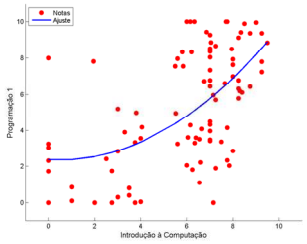
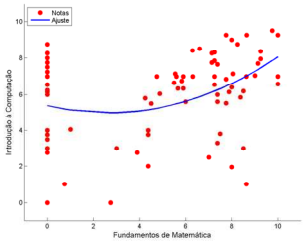
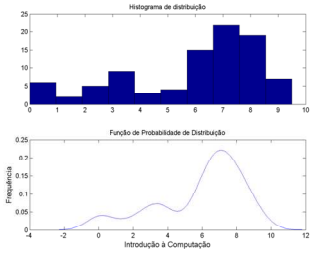
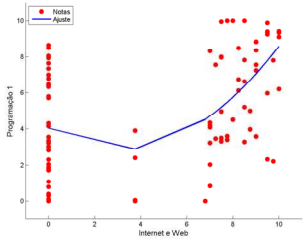
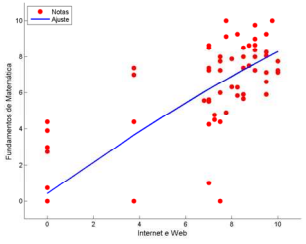
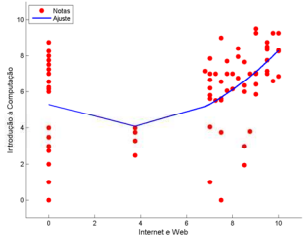
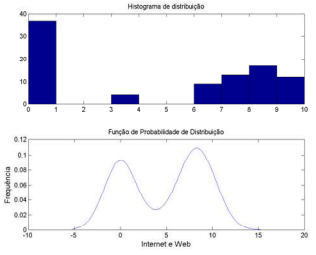
		 Geometria Analítica	ρ = 0,0596 valor de p = 3833	ρ = 0,1399 valor de p < 0,05
			 Introd. à Computação	ρ = 0,5533 valor de p < 0,05
				 Lab. Programação

Tabela 6: Correlações entre o desempenho na disciplina de Programação 1 e o desempenho nas disciplinas ditas correlatas no período de 2010 a 2011 .

 Programação 1	ρ = 0,5845 valor de p < 0,05	ρ = 0,3267 valor de p < 0,05	ρ = 0,4541 valor de p < 0,05	ρ = 0,5217 valor de p < 0,05
	 Fund. de Matemática	ρ = 0,6265 valor de p < 0,05	ρ = 0,6162 valor de p < 0,05	ρ = 0,3576 valor de p < 0,05
		 Intro. à Computação	ρ = 0,8506 valor de p < 0,05	ρ = 0,1866 valor de p = 0.0748
				ρ = 0,2876 valor de p = 0,054

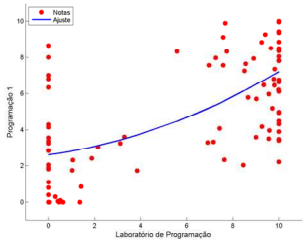
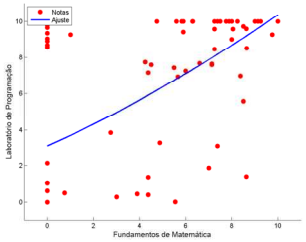
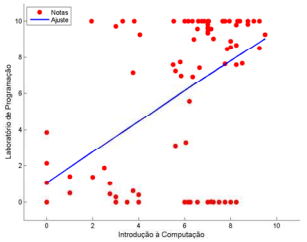
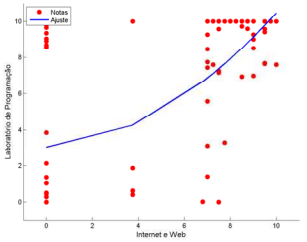
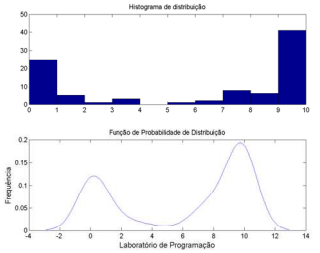
			Internet e Web	
				 Lab. Programação

Tabela 7: Correlações entre o desempenho na disciplina de Programação 1 e o desempenho nas disciplinas ditas correlatas no período de 2012 a 2013.

Os resultados das correlações apresentaram um coeficiente variável de fraco a moderado, em nenhuma das correlações estabelecidas para os períodos estudados ouve coeficiente de correlação forte, mesmo entre disciplinas consideradas complementares, como é o caso de Programação 1 e Laboratório de Programação. Entretanto, pode-se destacar alguns valores relevantes de correlação que aparecem nas Tabela 5, 6 e 7, como pode ser observado a seguir.

- 2006 a 2009
  - Cálculo 1 x Geometria Analítica: ρ = 56,5%
  - Programação 1 x Laboratório de Programação: ρ = 56,9%
- 2010 a 2011
  - Cá1culo 1 x Geometria Analítica: ρ = 56,5%
  - Programação 1 x Laboratório de Programação: ρ = 48,4%
  - Introd. à Computação x Laboratório de Programação: ρ = 55,3%
- 2012 a 2013
  - Programação 1 x Introd. à Computação: ρ = 57,8%
  - Introd. à Computação x Internet e Web: ρ = 66,4%
  - Programação 1 x Internet e Web: ρ = 66,0%
  - Introd. à Computação x Internet e Web: ρ = 66,4%

Confirmando que existe uma correlação moderada.

4.3.1.3. Análise de Cluster

A análise de *cluster* corresponde as etapas IV e V deste estudo. Após observação do desempenho na disciplina de Programação 1 e no Enem, optou-se por definir inicialmente três centróides para a formação dos primeiros grupos de alunos. Os centróides definidos podem ser aplicados tanto para os dados anteriores ao ingresso na universidade, quanto para os dados coletados após o início do curso, sendo os grupos classificados de acordo com o seu desempenho nas disciplinas, sendo:

- G1 - Bronze: grupo de alunos com desempenho ruim;
- G2 - Prata: grupo de alunos com desempenho que varia de mediano a bom;
- G3 - Bronze: grupo de alunos com desempenho excelente.

A segunda parte do estudo estabelece a busca por grupos de alunos no contexto dos dados, através da utilização de análise de *cluster,* utilizando a técnica de agrupamento *k-Means*. Os *clusters* obtidos a partir da execução do *k-Means* no conjunto de dados relacionados ao Enem não foram informativos, pois a distribuição da amostra é bastante irregular, conforme pode ser verificado nas Figuras 10, 11, 12, 13, 14, 15, 16, 17 e 18.

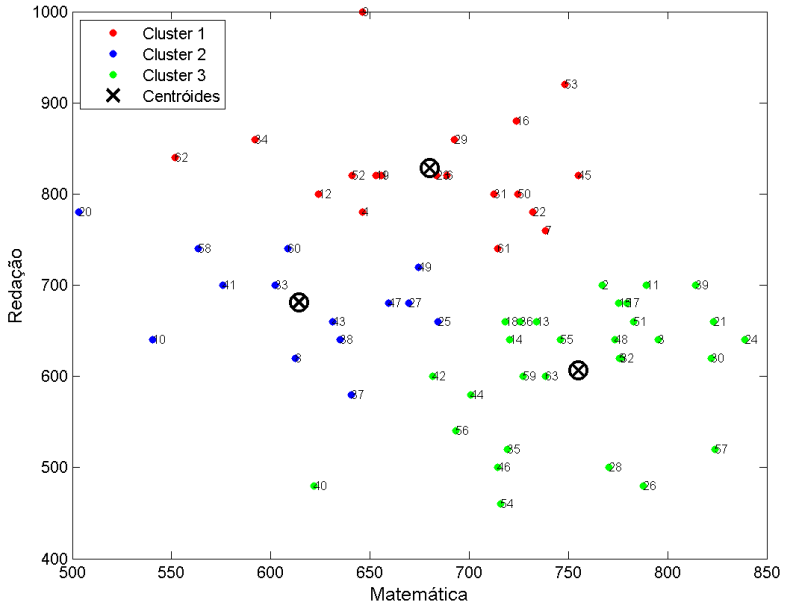


Figura 6: *Clusters* gerados entre a relação Redação e Matemática.

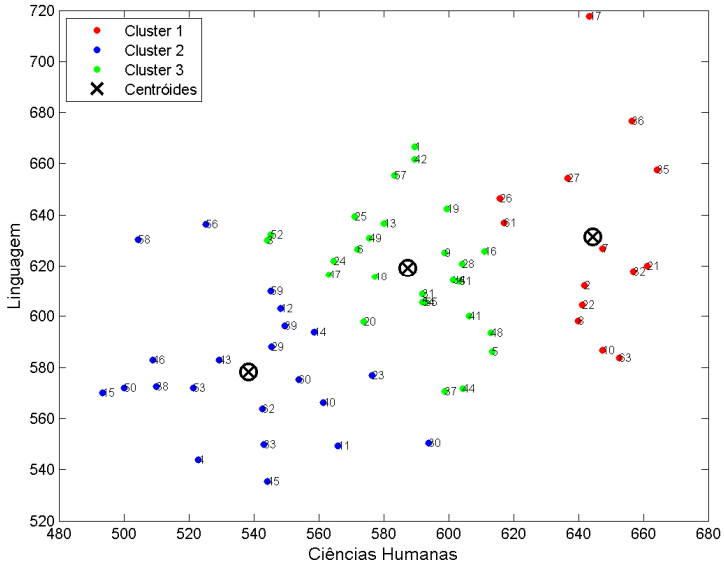


Figura 7: *Clusters* gerados entre a relação Linguagem e Ciências Humanas

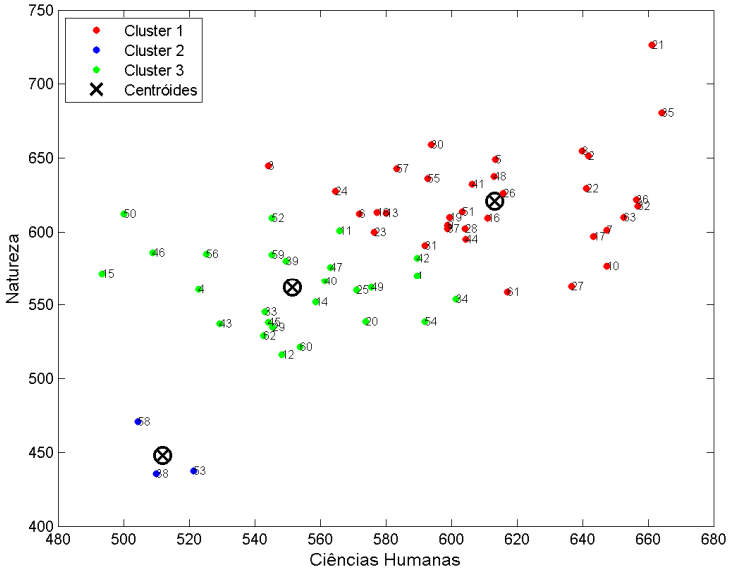


Figura 8: *Clusters* gerados entre a relação Ciências da Natureza e Ciências Humanas

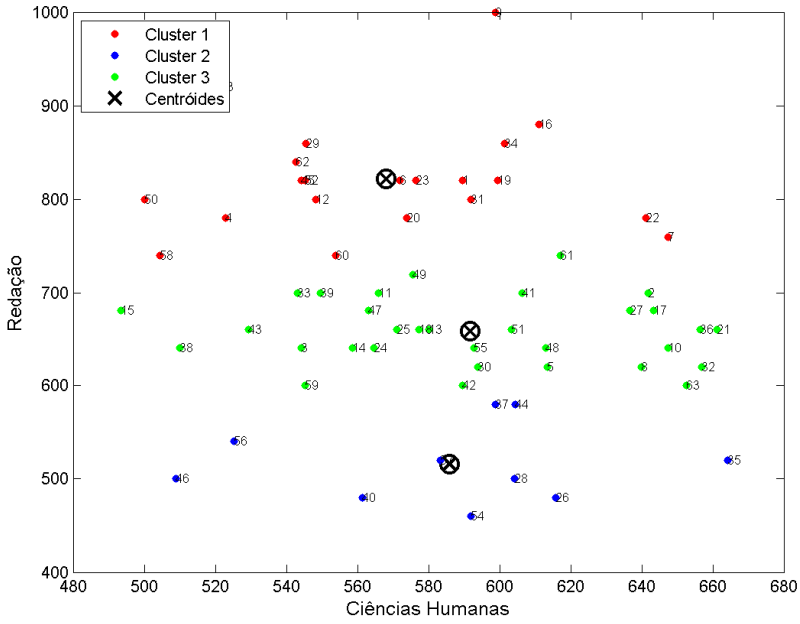


Figura 9: *Clusters* gerados entre a relação Redação e Ciências Humanas.

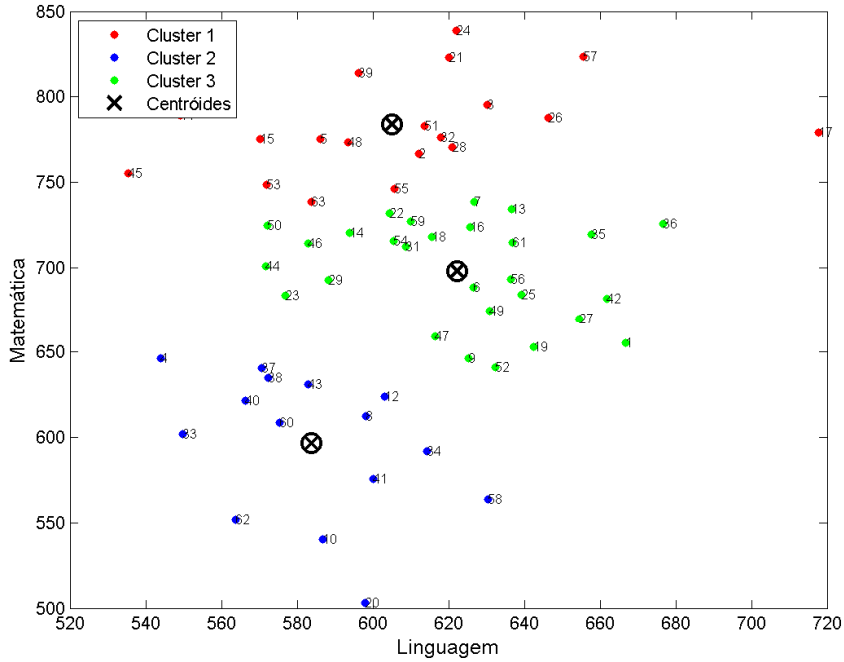


Figura 10: *Clusters* gerados entre a relação Matemática e Linguagem

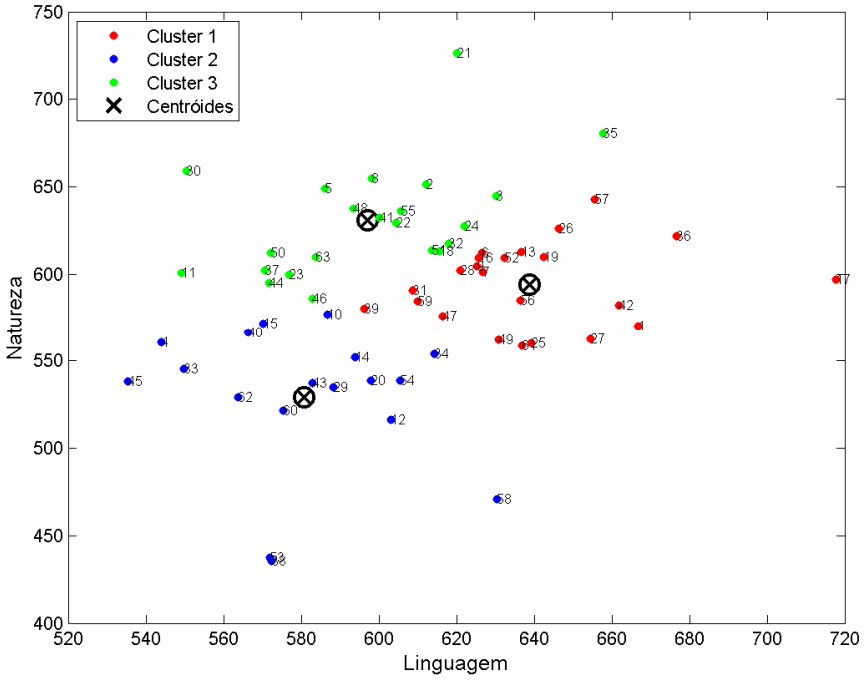


Figura 11: *Clusters* gerados entre a relação Ciências da Natureza e Linguagem

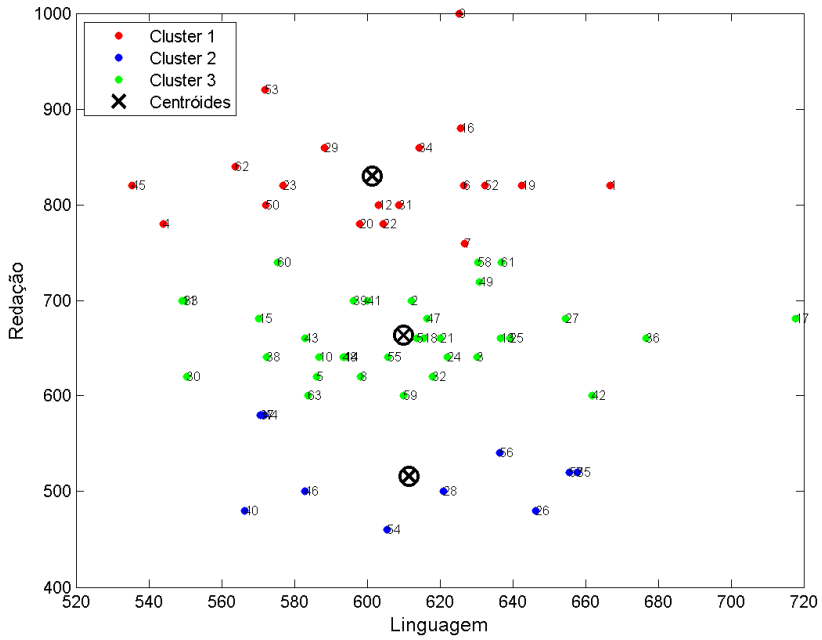


Figura 12: *Clusters* gerados entre a relação Redação e Linguagem

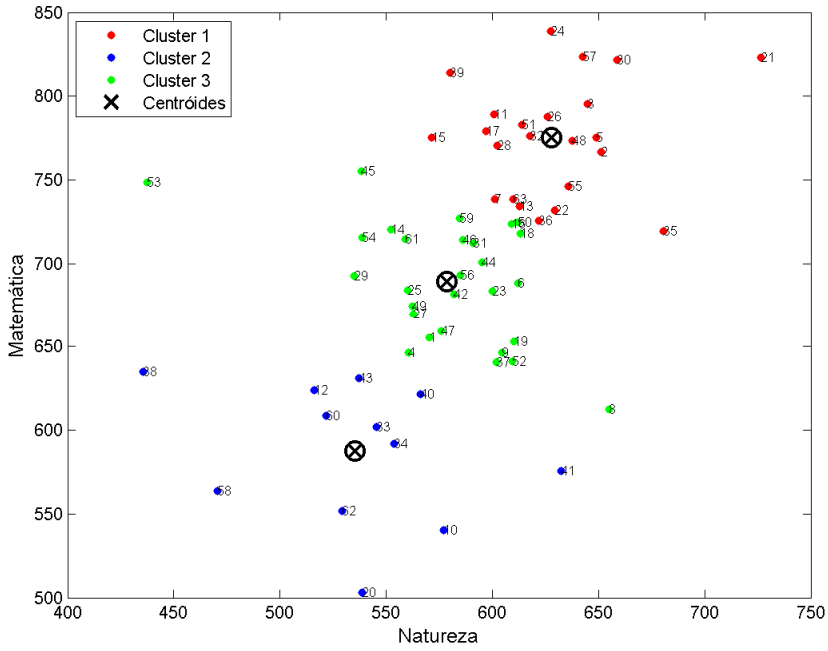


Figura 13: *Clusters* gerados entre a relação Matemática e Ciências da Natureza

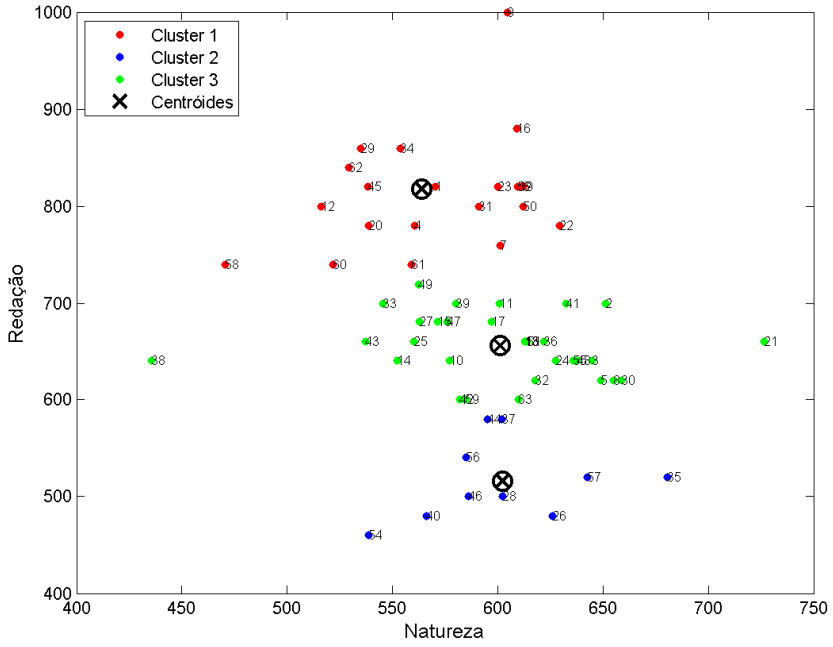


Figura 14: *Clusters* gerados entre a relação Redação e Ciências da Natureza

Com os gráficos mostrados acima, percebe-se claramente que o método *k-Means* dividiu de forma aproximada os dados, onde pode-se encontrar alunos com boas notas em uma prova e ruim em outra, deixando assim inconclusivo a análise. Abaixo segue o percentual de alunos em cada um dos *clusters* de acordo com a relação Disciplina 1 por Disciplina 2.

	GA-C1	IC-C1	GA-IC	C1-LP	LP-GA	IC-LP	P1-C1	P1-GA	P1-IC	P1-LP
Cluster 1	34.8	34.8	18.3	50.4	9.6	19.1	20.9	7.4	9.1	18.7
Cluster 2	10.9	55.7	52.2	28.3	61.3	22.2	23.5	73.9	20.9	62.6
Cluster 3	54.3	9.6	29.6	21.3	29.1	58.7	55.7	18.7	70.0	18.7

Tabela 8: Classificação dos alunos do período de 2006 a 2009 em *clusters* para a análise das Provas nas disciplinas do curso de Ciência da Computação.

	C1-GA	C1-IC	IC-GA	C1-LP	LP-GA	IC-LP	P1-C1	P1-GA	P1-IC	P1-LP
Cluster 1	29.7	16.7	26.1	31.2	26.8	39.9	53.6	49.3	44.9	63.0
Cluster 2	31.9	35.5	41.3	46.4	26.8	28.3	24.6	22.5	36.2	18.1
Cluster 3	38.4	47.8	32.6	22.5	46.4	31.9	21.7	28.3	18.8	18.8

Tabela 9: Classificação dos alunos do período de 2010 a 2011 em *clusters* para a análise das Provas nas disciplinas do curso de Ciência da Computação.

	LP-P1	FM-P1	IW-P1	IC-P1
Cluster 1	38.0	31.5	32.6	46.7
Cluster 2	25.0	28.3	27.2	26.1
Cluster 3	37.0	40.2	40.2	27.2

Tabela 10: Classificação dos alunos do período de 2012 a 2013 em *clusters* para a análise das Provas nas disciplinas do curso de Ciência da Computação.

	Li-Ch	Na-Ch	Na-Li	Ma-Ch	Ma-Li	Ma-Na	Re-Ch	Re-Li	Re-Na	Re-Ma
Cluster 1	33.7	26.1	44.6	28.3	34.8	45.7	34.8	31.5	35.9	37.0
Cluster 2	38.0	44.6	31.5	38.0	43.5	39.1	50.0	53.3	48.9	47.8
Cluster 3	28.3	29.3	23.9	33.7	21.7	15.2	15.2	15.2	15.2	15.2

Tabela 11: Classificação dos alunos do período de 2012 a 2013 em *clusters* para as Provas do Enem.

	P1-Ch	P1-Li	P1-Na	P1-Ma	P1-Re
Cluster 1	33.7	31.5	30.4	32.6	31.5
Cluster 2	38.0	41.3	45.7	40.2	53.3
Cluster 3	28.3	27.2	23.9	27.2	15.2

Tabela 12: Classificação dos alunos do período de 2012 a 2013 em *clusters* para a comparação das Notas do Enem com a disciplina de Programação 1.

Os dados mostrados nas Tabelas Tabela 8, Tabela 9, Tabela 10, Tabela 11 e Tabela 12 estão em % e as siglas representam as seguintes informações:

- Disciplinas da Grade de Ciência da Computação
  - C1: Cálculo 1;
  - GA: Geometria Analítica;
  - IC: Introdução à Computação;
  - LP: Laboratório de Programação;
  - IW: Internet e Web;
  - FM: Fundamentos de Matemática Computacional; e
  - P1: Programação 1.
- Disciplinas do Enem

- Ch: Ciências Humanas e suas tecnologias;
- Li: Linguagem e suas tecnologias;
- Na: Ciências da Natureza e suas tecnologias;
- Ma: Matemática e suas tecnologias; e
- Re: Redação.

Deste modo, neste primeiro teste não foi possível verificar a formação de grupos que atendessem ao padrão acima estabelecido. Portanto, seus resultados não serão incluídos no conjunto de informações utilizados para o diagnóstico de características de iniciantes em programação.

Em relação aos *clusters* identificados nas relações entre as disciplinas que se relacionam no período, os resultados foram bem informativos, como pode ser observado nas Figuras Figura 15, Figura 16, Figura 17 e Figura 18.

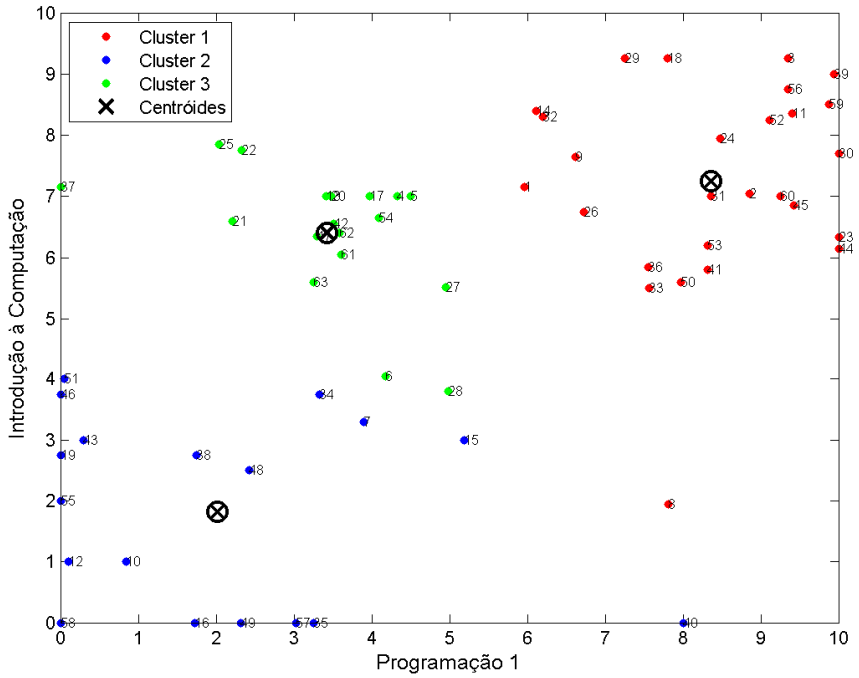


Figura 15: *Clusters* gerados entre a relação Introdução à Computação e Programação 1.

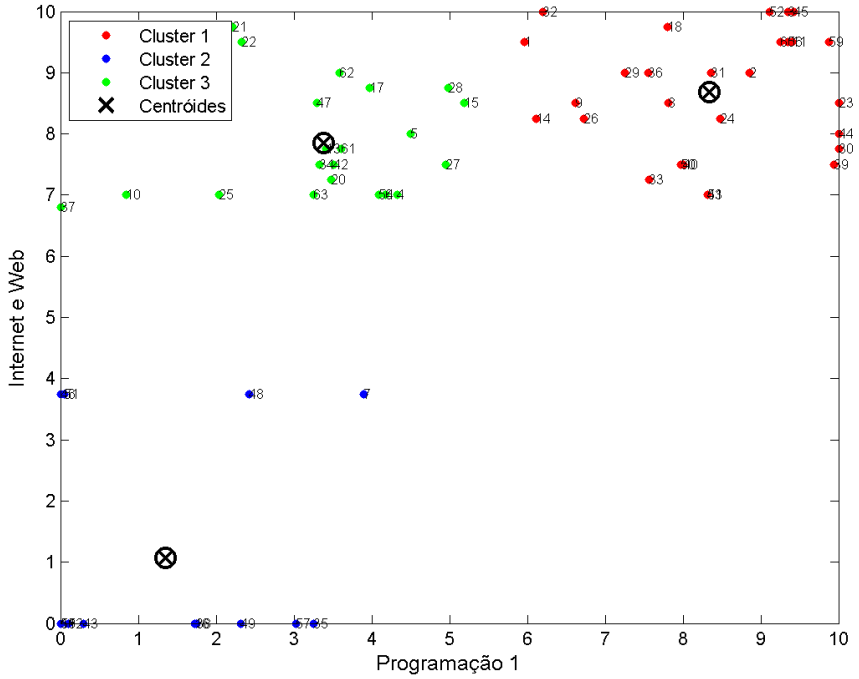


Figura 16: *Clusters* gerados entre a relação Internet e Web e Programação 1.

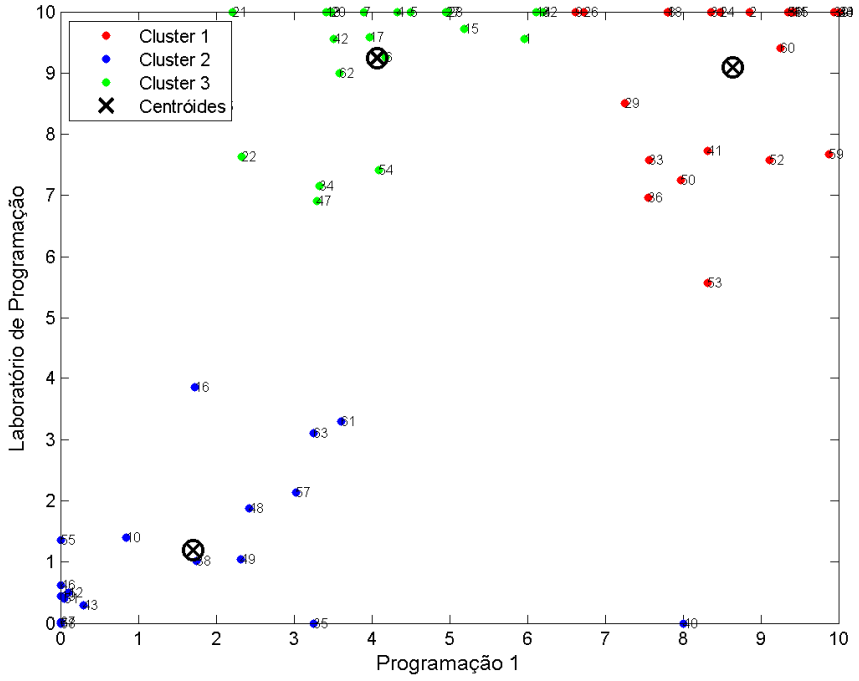


Figura 17: *Clusters* gerados entre a relação Laboratório de Programação e Programação 1.

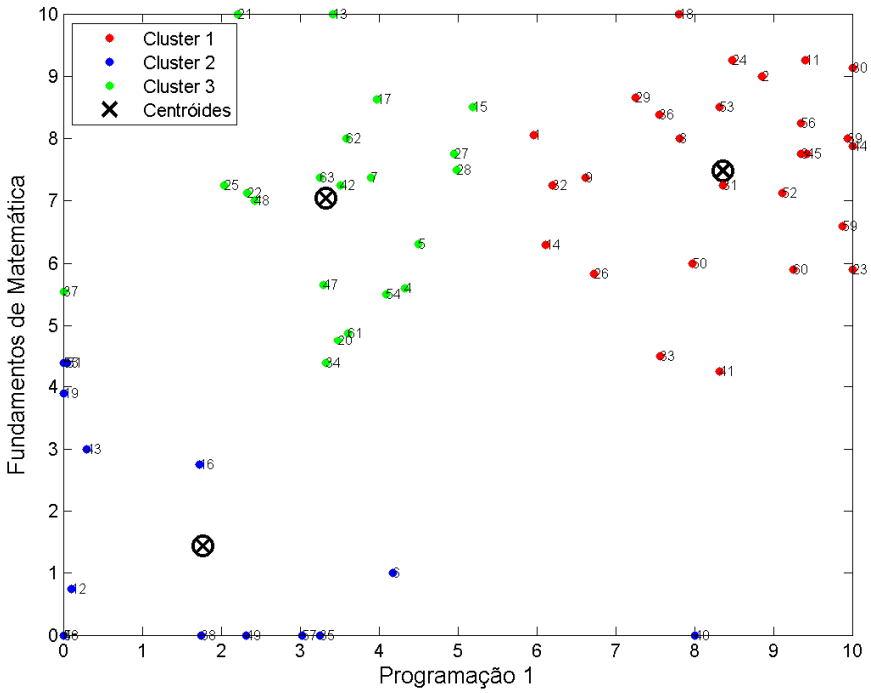


Figura 18: *Clusters* gerados entre a relação Fundamentos de Matemática e Programação 1.

Nota-se nesses resultados que os três grupos definidos inicialmente se mostraram suficientes para a definição dos grupos de alunos iniciantes. A partir da clusterização obtida na análise dos dados relativos à turma de 2012 – 2013, pode-se observar três grupos bem divididos de alunos, atendendo à caracterização relacionado ao desempenho (Ouro, Prata e Bronze).

Os indivíduos classificados identificados com desempenho ruim, grupo Bronze, obedeceram a uma constante em relação às notas de todas as disciplinas cursadas no período, onde seu intervalo de aproveitamento varia de 0 a 4 em todas as disciplinas investigadas. Já os que se classificaram no grupo Prata, correspondente a desempenho mediano – bom, apresentaram desempenho variado, onde o limite inferior foi superior a 4 e o superior foi 8. Apenas na disciplina do Laboratório de Programação alguns indivíduos desse grupo ficaram com nota superior a 8. Este foi considerado um evento curioso, posto que alguns dos alunos com notas 9 em laboratório, por exemplo, obtiveram coeficiente 4 em Programação 1. Onde laboratório é a parte prática da disciplina de Programação 1. Já os indivíduos do terceiro grupo se mantiveram com

desempenho acima de 7 em todas as disciplinas, sendo considerado isto um desempenho excelente no curso.

4.3.2. Avaliações Qualitativas

A avaliação qualitativa será realizada a partir do conjunto de códigos-fontes de algoritmos submetidos à plataforma The Huxley como solução de exercícios avaliativos propostos pelo professor. Para a análise qualitativa foi utilizada uma amostra de estudantes que cursaram Programação 1 no período de 2010 à 2013. Essa amostra foi selecionada de acordo com o grupo em que foi classificado na clusterização. Buscou-se analisar alunos que estiveram presentes no mesmo grupo em todas as correlações.

Devido à ausência de uma taxonomia de problemas para iniciantes em programação, a seleção das questões-problema utilizadas na análise foi realizada levando em consideração os seguintes parâmetros:

- Questões comuns a todos os participantes da amostra;
- Questões que contivessem estruturas de repetição;
- Questões que contivessem fluxos de decisão;
- Questões consideradas problemas bem formados.

A avaliação das soluções apresentadas pelos alunos para as questões-problemas foi realizada de forma manual. Os resultados foram separados por grupo, como segue:

G1 - Bronze

Os códigos apresentados por esses estudantes apresentam faltas graves, tanto em relação à linguagem (domínio de sintaxe e semântica) quanto ao próprio problema. A maioria das soluções escritas demonstra que o aluno não sabe resolver o problema (independente da linguagem utilizada), que não traça um plano ou um modelo mental para escrever a solução e que não possui conhecimento em linguagem. Além disso, deve-se considerar a falta de legibilidade e qualidade na solução apresentada. Abaixo é apresentado um exemplo de código-fonte produzido por indivíduos desse grupo:

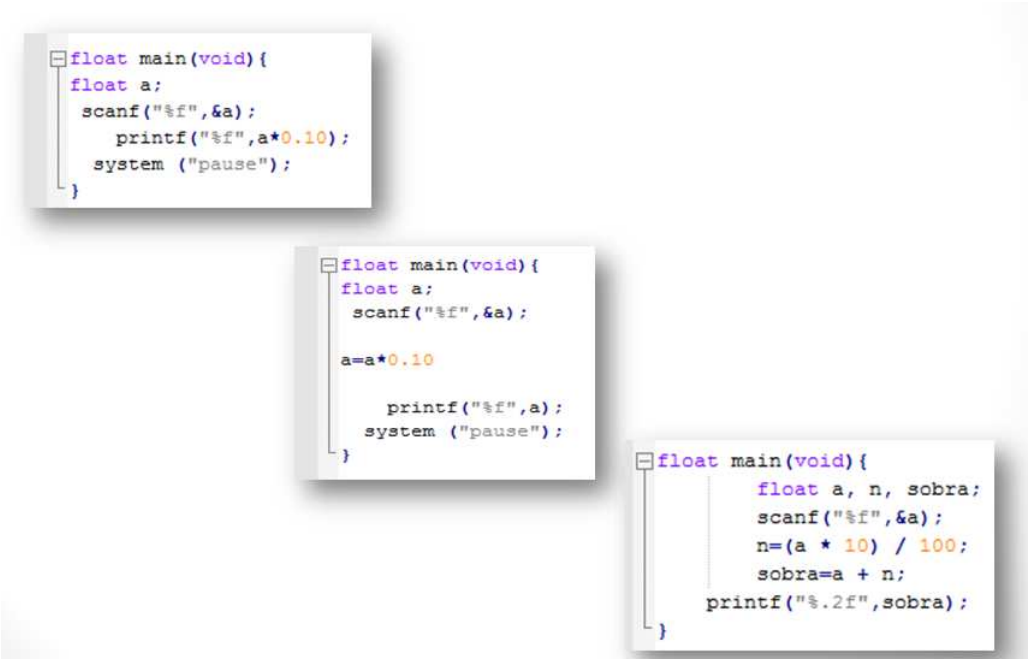
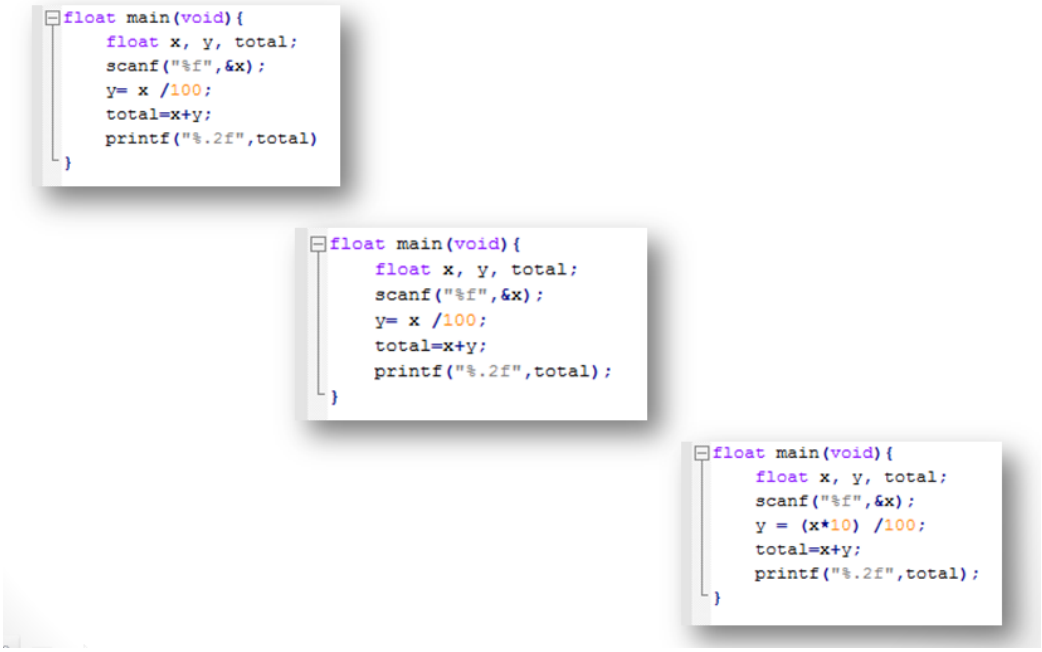


Figura 19: Código produzido por um iniciante em programação classificado como Bronze. Os trechos de códigos-fonte representam diferentes tentativas de resolução do mesmo problema no ambiente The Huxley.

G2 - Prata

Os códigos apresentados por esses alunos apresentam algumas fragilidades em relação à sintaxe e a semântica da linguagem utilizada. Em geral esses alunos possuem alguma habilidade para resolução de problemas, embora seu conhecimento seja frágil. Seus códigos tem algum indício de planejamento, ou seja, suas soluções não são escritas a esmo. Entretanto, desatenções são cometidas e são necessárias várias tentativas até chegar à solução ideal. Além disso, esse grupo apresenta soluções complexas para problemas simples, e em alguns casos percebeu-se certa dificuldade com algumas estruturas. Abaixo é apresentado um exemplo de código-fonte produzido por indivíduos desse grupo, onde podem ser verificadas algumas dessas características.



```
float main(void){
    float x, y, total;
    scanf("%f",&x);
    y= x /100;
    total=x+y;
    printf("%.2f",total)
}
```

```
float main(void){
    float x, y, total;
    scanf("%f",&x);
    y= x /100;
    total=x+y;
    printf("%.2f",total);
}
```

```
float main(void){
    float x, y, total;
    scanf("%f",&x);
    y = (x*10) /100;
    total=x+y;
    printf("%.2f",total);
}
```

Figura 20: Código produzido por um iniciante em programação classificado como Prata. Os trechos de códigos-fonte representam diferentes tentativas de resolução do mesmo problema no ambiente The Huxley.

G3 - Ouro

Esses alunos apresentam um bom mapeamento problema-solução e realizam poucas tentativas para a resolução das questões propostas. Seu código, de uma maneira geral é limpo, com poucas falhas. Não foi verificada a hesitação em utilizar algum tipo de estrutura específica e suas soluções apresentam qualidade, no que tange o código-fonte. Abaixo é apresentado um exemplo de código-fonte produzido por indivíduos desse grupo, onde podem ser verificadas essas características.

```
float main(void){
    float x, y, valor;
    scanf("%f",&x);
    y = (x * 10) /100;
    valor = x + y;
    printf("%.2f",valor);
}
```

Figura 21: Código produzido por um iniciante em programação classificado como Ouro. O trecho de código-fonte representa uma única tentativa de resolução do mesmo problema no ambiente The Huxley

5 Resultados e Discussão

Neste capítulo serão apresentados os resultados da pesquisa, bem como uma discussão sobre eles. Além disso, serão respondidas as questões de pesquisa.

5.1.1. Parte I – Análise das Correlações Entre o Desempenho nas Disciplinas do Enem e o Desempenho na Disciplina de Programação 1

Foi realizado um estudo sobre a experiência acadêmica prévia do aluno que ingressou no curso de Ciência da Computação com o objetivo de verificar se o desempenho do aluno neste período poderia influenciar, ou ser um fator determinante, no processo de aprendizado de disciplinas introdutórias de programação. Este estudo, que corresponde à segunda parte desta pesquisa, verificou o fator de correlação no desempenho dos novatos no Enem.

Os resultados obtidos a partir dessa análise apresentaram os seguintes resultados:

- Programação 1 x Ciências Humanas: $\rho = -0,35$;
- Programação 1 x Linguagem: $\rho = -0,049$;
- Programação 1 x Ciências da Natureza: $\rho = 0,163$;
- Programação 1 x Matemática: $\rho = 0,22$;
- Programação 1 x Redação: $\rho = 0,021$.

A partir desses dados, pode-se afirmar que o desempenho das disciplinas no Enem não é preditor de sucesso em disciplinas iniciais de programação. Todas as relações analisadas apresentaram fator de correlação fraco, incluindo as disciplinas relacionadas à matemática e raciocínio lógico, sendo este um resultado não esperado. Uma explicação plausível para estes resultados é o processo seletivo adotado pela instituição, o SiSU.

Para ingresso na universidade pelo SiSU, o aluno escolhe o curso após saber seu desempenho e a média de corte para o curso escolhido. Sendo assim, a decisão em ingressar no curso de Ciência da Computação, em boa parte dos casos ocorre devido ao fato de sua nota não ser suficiente para ingresso em outro curso. Assim, grande parte dos alunos que estão no curso de Ciência da Computação podem nunca ter se imaginado um cientista da computação, ou um excelente programador, faltando-lhe motivação para o desenvolvimento de habilidades nessa área. Motivação, conforme apresentado por Robins *et al*. é uma característica que prediz o sucesso em aprender a programar [35].

A análise desta correlação está em consonância com modelo utilizado por [39], que investigou essas correlações na *National University of Ireland Maynooth*, na Irlanda. Entretanto, em seu estudo, Bergin concluiu que naquela instituição estudantes com bom desempenho em disciplinas relacionadas à ciência em geral tem um fator de correlação significativo com introdução à programação, com exceção apenas da disciplina de Química, que apresentou um baixo índice de correlação.

Outra correlação investigada nesta pesquisa refere-se ao desempenho em Programação 1 e as disciplinas correlatas cursadas no mesmo período, onde os resultados foram:

- 2006 a 2009
    - Programação 1 x Cálculo 1: $\rho = 0{,}264$
    - Programação 1 x Laboratório de Programação: $\rho = 0{,}569$
    - Programação 1 x Geometria Analítica: $\rho = 0{,}262$
    - Programação 1 x Introd. à Computação : $\rho = 0{,}296$
- 2010 a 2011
    - Programação 1 x Cálculo 1: $\rho = 0{,}361$
    - Programação 1 x Laboratório de Programação: $\rho = 0{,}484$

    - Programação 1 x Geometria Analítica: ρ = 0,461
    - Programação 1 x Introd. à Computação : ρ = 0,219
- 2012 a 2013
    - Programação 1 x Introd. à Computação: ρ = 0,327
    - Programação 1 x Fundamentos de Matemática: ρ = 0,585
    - Programação 1 x Internet e Web: ρ = 0,454
    - Programação 1 x Laboratório de Programação: ρ = 0,522

Nesta análise pode-se observar um coeficiente de correlação que varia entre fraco e moderado. O que remete a duas conclusões: (i) o desempenho do aluno em programação está relacionado ao seu desempenho em disciplinas que exigem habilidades similares e que (ii) esses alunos tendem a seguir uma frequência de desempenho. Assim, alunos com baixo aproveitamento em Programação 1 tende a ter baixo desempenho em Cálculo 1, Fundamentos de Programação, Internet e web.

Era esperado que o desempenho em Programação 1 fosse fortemente relacionado ao desempenho em Laboratório 1, visto que uma disciplina complementa a outra. Entretanto, isso não foi verificado.

5.1.2. Parte III – Análises dos Clusters Obtidos

Na etapa de verificação foi utilizado o *k-Means* para a geração dos grupos, e foram definidos três centroides para realização da classificação. Os resultados obtidos nessa etapa foram similares para todos os intervalos temporais avaliados. Os períodos apresentaram as mesmas características de *cluster*, mesmo com variação nas disciplinas e no regime do curso.

Os centroides definidos foram rotulados como Ouro (cor vermelha), Prata (cor verde) e Bronze (cor azul) para caracterizar o desempenho dos alunos na disciplina. Os gráficos gerados a partir do k-Means seguem uma distribuição constante, onde, em todos os períodos, os grupos Ouro e Bronze podem ser verificados facilmente e seus componentes não invadem a fronteira do outro grupo. Além disso, a população presente nesses grupos é constante. O grupo Prata apresentou um desempenho variante. Seus integrantes ficaram distribuídos no gráfico, tendo o centroide localizado na

maioria das amostras no quarto quadrante.

Os alunos classificados como Ouro têm, em sua maioria, desempenho maior que 7 em todas as disciplinas, e concentram-se no primeiro quadrante do gráfico. Os alunos que foram classificados como Bronze tem o desempenho variante, indo de 0 à 5 em todas as análises realizadas. O grupo Prata apresentou características interessantes na análise de *cluster*, onde sua população ficou distribuída entre o primeiro, o terceiro e o quarto quadrante do gráfico, tendo suas notas variantes ente 6 e 10. Em percentuais, aproximadamente 37% doa alunos foram classificados como Bronze, 26% dos alunos como Prata e 37% como Ouro.

Esses resultados demonstram que os alunos classificados como Ouro são alunos que se identificam com a estrutura e as disciplinas do primeiro período do curso de computação. Os alunos classificados como Bronze apresentam desempenho insatisfatório em todas as disciplinas do curso. Já os alunos de Prata te rendimento regular e que varia de acordo com a disciplina. Percebeu-se que em alguns momentos alguns indivíduos desse grupo classificam-se como Ouro, alguns momentos como Bronze. Atribui-se, portanto, a esse grupo a característica "esforço". O desempenho apresentado por eles induz à dedução de que embora não possuam características necessárias para serem classificados como Ouro, esforçam-se a todo o momento para manter seu nível no curso e aprender o possível para continuar indo razoavelmente bem nas disciplinas.

Um fato interessante e inesperado observado foi a inversão verificada no gráfico de Laboratório de Programação e Programação 1. Essas disciplinas são diretamente complementares. Laboratório de Programação existe com o intuito de ensinar na prática o que é visto conceitualmente em Programação 1. Os alunos obtiveram notas altas em Laboratório e continuaram com o rendimento baixo em Programação 1, levantando o seguinte questionamento: Porque um aluno que apresenta um baixo índice de aproveitamento na parte teórica obtém um bom desempenho na parte prática da mesma disciplina?

Para responder esta pergunta é necessária uma investigação mais detalhada. A priori, desconfia-se que variáveis relacionadas ao ambiente estejam provocando este evento. Podem ser levados em consideração alguns pontos, tais como:

- As disciplinas são lecionadas por professores diferentes;

- As turmas de Laboratório são menores do que a de Programação 1 (a turma de Programação 1 é dividida em duas turmas de Laboratório);
- Pode haver falta de comunicação entre os dois professores;
- O método adotado para aulas e avaliação pode ser diferente;

Os resultados deste *cluster* estão em consonância com os resultados obtidos através da análise de correlação entre as disciplinas de Programação 1 e Laboratório. Onde o coeficiente de correlação apresentou correlação moderada, quando era esperado que fosse forte.

5.1.3. Parte IV – Avaliação do Código-fonte produzido por iniciantes

Foram avaliados os códigos-fontes produzidos por iniciantes em programação que cursaram Programação 1 na UFAL nos períodos de 2010 à 2013. A análise do código levou em consideração a classificação de grupo definida pela análise de cluster. Assim, verificou-se as seguintes características dos códigos produzidos por cada grupo:

I. G1 – Bronze:
   a. Dificuldade em expressar uma solução. Em alguns casos é como se não houvesse compreensão do problema;
   b. Soluções parecem não planejadas;
   c. Dificuldades em utilizar a linguagem;
   d. Códigos com pouca legibilidade;
   e. Hesitação em utilizar estruturas aninhadas.

II. G2 – Prata:
   a. Dificuldade na utilização de algumas estruturas da linguagem;
   b. Aparente desatenção. Muitos erros de compilação referentes ao esquecimento da utilização de um ";" por exemplo;
   c. Códigos com alguma legibilidade;
   d. Hesitação na utilização de estruturas aninhadas;
   e. Hesitação na utilização de estruturas complexas.

III. G3 – Ouro:

a. Código apresenta bom grau de legibilidade;
b. Poucos erros de compilação;
c. Boas soluções.

5.1.4. Discussão

Os grupos definidos classificam os alunos de acordo com seu desempenho, onde este foi estabelecido a partir das notas e da qualidade de solução produzida durante o período letivo da disciplina. Os grupos Bronze e Ouro foram bem caracterizados pelo desempenho, sendo o Bronze para alunos de baixo desempenho e o Ouro alunos de alto desempenho. Já o grupo Prata apresenta características de desempenho variante. A população que forma esse grupo não é constante e em algumas disciplinas alguns indivíduos variam de grupo, hora migrando para o Bronze, hora para o Ouro. Isso indica que esses alunos possuem grandes chances de se tornarem efetivamente do grupo Ouro.

Acredita-se que o grupo Prata é o mais necessitado de intervenções que objetivam melhorar o aprendizado de programação, pois é formado por alunos que já apresentam indícios de habilidades para programar. Além disso, pelo esforço percebido no decorrer da disciplina, nas tentativas de resolução dos exercícios e nos erros pode-se afirmar que estes alunos seriam mais sensíveis a qualquer intervenção no sentido de melhorar suas habilidades.

Com base nos dados analisados, pode-se afirmar que o desempenho dos iniciantes em programação possui uma correlação moderada com as disciplinas relacionadas do mesmo período que introdução à programação e uma fraca correlação com as disciplinas estudadas para o Enem. Afirma-se ainda que o bom desempenho em Línguas e Redação não possui correlação com habilidades de programação, sendo a capacidade de expressão verificada nessa avaliação sem influência para o desenvolvimento das habilidades de programação.

Dados os resultados supracitados podem ser consideradas questões importantes relacionadas, diretamente ou indiretamente, ao ensino-aprendizado de disciplinas introdutórias de programação. Essas questões que dizem respeito aos estudantes, aos professores e ao ambiente.

Foi possível observar que os alunos apresentam baixo desempenho em disciplinas relacionadas às áreas que exigem raciocínio lógico matemático, como Álgebra e Cálculo. Esse fato remete a problemas anteriores ao ensino superior que influenciam diretamente no desempenho nos aluno. Em geral, as escolas de ensino médio do Brasil preocupam-se muito em preparar o aluno para ser aprovado nos exames para ingresso nas universidades, estimulando-os a decorar fórmulas e desenvolver técnicas para responder as questões de tais exames, deixando de lado o desenvolvimento do raciocínio matemático. Os alunos "pré-vestibulandos" não são ensinados a deduzir equações ou entender o que elas realmente fazem e para que foram desenvolvidas. São treinados apenas a aplicá-las.

Essa prática útil para alcançar aprovação do vestibular e, por conseguinte o ingresso na universidade é demasiado prejudicial para o desenvolvimento acadêmico dos estudantes no ensino superior. Além do déficit de conhecimento, o aluno chega ao ensino superior apresentando carências relacionadas ao desenvolvimento de raciocínio lógico, fragilidade no conhecimento matemático (em introdução à programação, os exercícios clássicos estão diretamente relacionados a essas habilidades) e diminuto interesse em desenvolver coisas novas, novos raciocínios. Em geral percebe-se que o estudante que acaba de ingressar na universidade sobre um choque recebe um choque em relação ao paradigma de ensino.

As questões acima citadas são relacionadas diretamente ao aluno egresso nos cursos superiores e, em especial, aos matriculados em Ciência da Computação. Mas, além dos problemas do aluno devem ser considerados outros fatores importantes no ensino introdutório de programação. Um segundo fator a ser considerado é o ambiente no qual será ministrado o curso. Para efeitos desta dissertação entende-se por ambiente o espaço físico utilizado, os equipamentos disponíveis para o ensino-aprendizagem (computadores e recursos de mídia e áudio disponível para a aula), o professor que lecionará a disciplina e o conteúdo do curso introdutório.

No que se refere aos recursos físicos do ambiente, é necessário ressaltar a necessidade da disponibilidade de um computador por aluno, de um ambiente limpo e seguro, propício ao estudo. A UFAL oferece essa estrutura aos seus alunos. Com o objetivo de potencializar o ensino a disciplina foi dividida em duas: aulas teóricas e práticas. Nas aulas práticas as turmas são divididas em duas, ficando menos alunos em

sala, o que faz com que o professor possa dedicar maior atenção ao grupo de alunos presente.

Entretanto, foi observado na análise que os alunos da disciplina prática apresentam desempenho estranhamente superior ao da disciplina teórica. O que faz com que seja levantada a seguinte questão: Como um indivíduo poder ter baixo aprendizado teórico e alto desenvolvimento prático de um mesmo conteúdo?

Uma resposta plausível a essa questão é o professor. As disciplinas (que são complementares) são ministradas por professores diferentes. Que apresentam um diferente grau de ensino e avaliação dos iniciantes, embora ambos os professores utilizem o mesmo ambiente (físico e de software) para lecionar. Após os resultados das correlações, foi realizada uma entrevista não estruturada com alguns professores que lecionam as disciplinas (teórica e prática) na universidade e pode-se constatar que não há um contato estreito entre eles durante o período do curso onde é ministrada a disciplina. Ou seja, não há um planejamento estratégico para o ensino teórico e prático de introdução à programação.

Outrossim, o conteúdo das disciplinas introdutórias de programação é denso e extenso e o tempo para seu aprendizado curto. Além de preocupar-se com o desenvolvimento do raciocínio lógico na resolução de problemas, a necessidade de abstrair conceitos que não possuem forma ou modelo, os alunos ainda enfrentam as limitações dos ambientes de programação e a necessidade de aprendizado de uma linguagem nova.

6 Conclusão e Trabalhos Futuros

Nesta dissertação foi apresentado um estudo com o para o entendimento inicial do perfil dos alunos iniciantes em programação da Universidade Federal de Alagoas. Para tanto, foi utilizada uma sistemática de análise multidimensional, valendo-se da investigação de três fontes de dados: (i) notas do Enem utilizadas para o ingresso na universidade, (ii) desempenho escolar no primeiro ano/período do curso e (iii) código-fonte produzido pelos alunos durante o curso de Programação 1.

Os resultados desta análise foram suficientes para definir uma taxonomia inicial

de estudantes de programação: Ouro, Prata e Bronze. Essa classificação agrupa os alunos de acordo com o seu desempenho em dois momentos acadêmicos, quais sejam seu ensino médio e o começo do ensino superior. Como foi visto no Capítulo anterior, as correlações referentes às questões anunciadas no Capítulo 1 foram respondidas e analisadas, trazendo algumas surpresas em relação ao que era esperado, como a fraca correlação entre as disciplinas de Matemática e Programação 1, Cálculo 1 e Programação 1 e Fundamentos Matemáticos e Programação 1. Outro resultado não esperado foi a correlação moderada entre Programação 1 e Laboratório de Programação. Dado o caráter complementar dessas disciplinas esperava-se encontrar uma correlação forte.

Além disso, observaram-se ainda, alguns fatores coadjuvantes no ensino introdutório de programação que devem ser levados em consideração no estudo dos problemas de iniciantes, tais como interação dos professores de disciplinas complementares, conhecimento adquirido antes do ingresso no ensino superior, ambiente utilizado como ferramenta de apoio e conteúdo.

Há muitas possibilidades de trabalho futuro. Mas, um trabalho para ser realizado imediatamente diz respeito à necessidade de melhor avaliar e validar a abordagem proposta. Além disso, devem-se automatizar procedimentos para permitir a análise de código, usando técnicas de mineração de texto.

7 Bibliografia

[1] A. McGettrick, R. Boyle, R. Ibbet, J. Lloyd, G. Lovegrove and K. Mander, Grand Challenges in Computing - Education, Sanford Stree, 2005.

[2] R. Lister, E. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Mostrom, K. Sanders, O. Seppala, B. Simon and L. Thomas, "A Multi-National Study of Reading and Tracing Skills in Novice," *ACM SIGCSE Bulletin,* vol. 36, no. 4, pp. 119-150, 2004.

[3] M. d. M. Braga, F. P. D. da Costa, A. D. Junior e F. C. de Albuquerque, *Projeto Pedagógico - Curso de Ciência da Computação,* Universidade Federal de Alagoas, 2006.

[4] *Diretrizes Curriculares - MEC Consulta Publica,* 2011.

[5] G. D. Garson, "Statnots: Topics in Multivariate Analysis," 2009. [Online]. Available: http://faculty.chass.ncsu.edu/garson/PA765/statnote.htm. [Acesso em 18 Novembro 2013].

[6] D. Moore, The Basic Practice of Statistics, W. H. Freeman, 2006.

[7] R. M. O'Brien, "The Use Of Pearson's R With Ordinal Data," *American Sociological Review,* vol. 44, pp. 851-857, 1979.

[8] D. B. Figueiredo Filho e J. A. Da Silva Júnior, "Desvendando os Mistérios do Coeficiente de Correlação de Pearson (r)," *Política Hoje,* vol. 18, n. 1, pp. 115-146, 2009.

[9] C. P. Dancey e J. Reidy, Estatística sem Matemática para Psicologia: Usando SPSS para Windows, Porto Alegre: Artmed, 2006.

[10] M. Valli, "Análise de Cluster," *Augusto Guzzo Revista Academica,* vol. 4, pp. 77-87, 2002.

[11] M. R. d. Nascimento, *Um Estudo sobre a Eficácia do Ensino à Distância de Programação para Alunos Iniciantes,* Campina Grande, 2011.

[12] R. de Barros Paes, R. Malaquias, M. Guimarães e H. Almeida, "Ferramenta para a Avaliação de Aprendizado de Alunos em Programação de Computadores," em *II Congresso Brasileiro de Informática na Educação (CBIE 2013)*, Campinas - SP, 2013.

[13] V. C. Aureliano Oliveira e P. Cabral de Azevedo Restelli Tedesco, "Ensino-aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIEe WIE," *23º Simpósio Brasileiro de Informática na Educação,* 26-30 Novembro 2012.

[14] A. Begel, "LogoBlocks:A Graphical Programming Language for Interacting with the World.," Unpublished Advanced Undergraduate Project Report, MIT Media Lab, 1996.

[15] "Logo: A Philosophy And Implementation," *LEARNING, PROJECT LIGHTHOUSE IN THAILAND: GUIDING PATHWAYS TO POWERFUL,* pp. 2-

99, 1999.

[16] E. S. de Almeida, J. D. Herrera, L. J. da S. Filho, H. Oliveira de Almeida, E. De Barros Costa, B. L. Vieira e M. D. de Melo, "Um Ambiente Integrado para auxílio ao Ensino de Ciência da Computação," *Colabor@ - Revista Dtal da CVA-Ricesu,* Vols. %1 de %22 - n 8ª, 2002.

[17] I. C. Garcia, P. J. de Rezende e F. C. Calheiros, "Astral: Um Ambiente para Ensino de Estruturas de Dados através de Animações de Algoritmos," *CLEI - V Congresso Iberoamericano de Educação Superior em Computação,* n. V, 1996.

[18] R. Pereira dos Santos e H. Xavier Costa, "TBC-AED: Um Software Gráfico para Apresentação de Algoritmos e Estruturas de Dados aos Iniciantes em Computação e Informática," *INFOCOMP – JOURNAL OF COMPUTER SCIENCE,* pp. 41-50, 2006.

[19] M. Consway, S. Audia, T. Burnette, D. Cosgrove e K. Christiansen, "Alice: lessons learned from Building a 3D System for Novices," em *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000.

[20] S. Cooper, W. Dann e R. Pausch, "ALICE: A 3D Tool For Introductory Programming Concepts".

[21] S. Cooper, W. Dann e R. Pausch, "Teaching Objects-first In Introductory Computer Science".

[22] W. Dann, D. Cosgrove, D. Slater e D. Culyba, "Mediated Transfer: Alice 3 to Java".

[23] J. Maloney, K. Peppler, Y. B. Kafai, M. Resnick e N. Rusk, "Programming by Choice: Urban Youth Learning Programming with Scratch," *Proceedings of the 39th SIGCSE technical symposium on Computer science education,* pp. 367-371, Março 2008.

[24] K. Schubert Vargas e J. Martins, "Ferramenta para Apoio ao Ensino de Introdução à Programação," em *XIV Seminario de Cumputação*, 2005.

[25] E. Falk Anderson e L. McLoughlin, "Critters in the Classroom: A 3D Computer-Game-Like Tool for Teaching Programming to Computer Animation Students," em *Proceeding SIGGRAPH '07 ACM SIGGRAPH 2007 educators program*, New York, 2007.

[26] R. de Santiago e R. L. Scaranto Dazzi, *Ferramenta de apoio ao ensino de algoritmos.,* 2004.

[27] E. Lahtinen, K. AlaMutka e H. Jarvinen, "A Study of the Difficulties of Novice Programmers".

[28] J. Sheard, Simon, M. Hamilton e J. Lönnberg, "Analysis of Research into the Teaching and Learning of Programming," *ICER '09 Proceedings of the fifth international workshop on Computing education research workshop,* pp. 93-104, 10-11 Agosto 2009.

[29] G. Pólya, "How to solve it" A New Aspect of Mathematical Method., vol. Segunda Edição, Rio de Janeiro: Interciência, 1977.

[30] E. Soloway, "Leaning to Program = Learning To Construct Mechanisms And Explanations," *Communications of the ACM,* vol. 29, 1986.

[31] M. C. Linn, "The cognitive consequences of programming instrution in

classrooms," *Educational Researcher,* pp. 14(5): 14-29, 1985.

[32] S. P. Davies, "Models and theories of programming strategy," *International Journal of Man-Machine Studies,* pp. 237-267, Agosto 1993.

[33] J. R. Johnson, "Can Computer Programming Improve Problem-Solving Ability?," *ACM SIGCSE Bulletin ,* vol. 22 , 1990.

[34] R. E. Mayer, "Different Problem-Solving Competencies Estabilished in Learning Computer Programming With And Without Meaningful Models," *Journal of Educational Psychology,* vol. 67, no. 6, pp. 725-734, 1975.

[35] Robins, Anthony; Rountree, Janet; Rountree, Nathan, "Learning and Teaching Programming: A Review and Discussion," *Computer Science Education,* vol. 13, pp. 137-172, 2003.

[36] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin and R. Simmons, "Conditions of Leanrning in Novice Programmers," *Journal of Educational Computing Research,* vol. 2, 1986.

[37] C. Bishop-Clark, "Cognitive Style, Personality, and Computer Programming.," *Computers in Human Behavior,* vol. 11, no. 2, pp. 241-260, 1995.

[38] M. T. Silva, M. C. Tenório Cabral Cavalcante e E. De Barros Costa, "Explorando Correlações em Programação: um estudo focado no processo seletivo e em disciplinas correlatas.," em *II Congresso Brasileiro de Informática na Educação, XXIV Simpósio Brasileiro de Informática na Educação*, Campinas - SP, 2012.

[39] S. Bergin e R. Reilly, "Programming: Factors that Influence Success," em *SIGCSE' Special Interest Group on Computer Science Education*, St. Louis, Missouri, 2005.

[40] M. de Raadt, M. Toleman e R. Watson, "Incorporating Programming Strategies Explicitly into Curricula," em *roc. Seventh Baltic Sea Conference on Computing Education Research*, Kolin National Park, Finland, 2007.

[41] T. H. C. d. Castro, *Sistematização da Aprendizagem de programação em Grupo,* Rio de Janeiro, 2011.

[42] C. Rich, *Inspecion Methods In Programming: Clichés and Plans,* 1987.

[43] H. C. Lande, *Natural Language Tutoring And The Novice Programmer,* Pittsburgh, 2004.

[44] N. Pennington e B. Grabowski, "The Tasks Of Programming," J. M. Hoc, T. R. G. Green, R. Samurcay, e D. J. Gilmore, Londres, 1990.

[45] D. W. Valentine, "CS Educational Research: A Meta-Analysis of SIGCSE Technical Symposium Proceedings," *SIGCSE - ACM Special Interest Group on Computer Science Education,* 3 - 7 Março 2004.

[46] D. Radošević, T. Orehovački e A. Lovrenčić, "New Approaches and Tools in Teaching Programming," em *Central European Conference on Information and Intelligent Systems CECIIS 2009*, Varaždin - Croacia, 2009.

[47] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Delvin e J. Peterson, "A Survey of Literature on the Teaching of Introductory," *ACM SIGCSE Bulletin,* vol. 39, n. 4, pp. 204-223, 2007.

[48] C. D. Hundhausen, S. A. Douglas e J. T. Stasko, "A Meta-Studyof AlgorithmVisualizationE¡ectiveness," *Journal of Visual Languages and Computing,* pp. 259-290, 2002.

[49] S. Garner, P. Haden e A. Robins, "My Program is Correct But it Doesn't Run: APreliminary Investigation of Novice Programmers' Problems," em *Australasian Computing Education Conference, Research and Practice in Information Technology*, 2005.

[50] M. d. Raadt, "A Review of Australasian Investigations into Problem Solving and the Novice Programmer," *Computer Science Education,* vol. 17, pp. 201-2013, 2007.

[51] R. Poter e P. Calder, "A Pattern-Based Problem-Solving Process For Novice Programmers," *ACE '03 Proceedings of the fifth Australasian conference on Computing education,* vol. 20, pp. 231-238, 2003.

[52] J. McCarthy, "Basis For a Mathematical Theory Of Computation," *Proceedings of the Western Joint Computer Conference,* 1961.

[53] A. McGettrick, R. Boyle, R. Ibbet, J. Lloyd, G. Lovegrove and K. Mander, Grand Challenges in Computing - Education, Sanford Stree, 2004.