O uso de corretores automáticos para o ensino de programação de computadores para alunos de engenharia

Palloma S. S. Brito, Reinaldo S. Fortes

Departamento de Computação – Universidade Federal de Ouro Preto (UFOP)
Campus Universitário Morro do Cruzeiro – Ouro Preto – MG – Brasil

palloma.brito@aluno.ufop.edu.br, reifortes@ufop.edu.br

Abstract. *The high level of abstraction required in computer programming disciplines causes students to become unmotivated, elevating failure and dropout rates. In this context, the present work approaches a teaching methodology that uses automatic code brokers and objective questions as a way to consolidate the subjects addressed, keeping the student informed about their development and mitigating possible learning difficulties. This methodology was applied and evaluated in Engineering classes, demonstrating the positive potential in students' learning and, consequently, in their academic performance in the discipline.*

Resumo. *O alto nível de abstração exigido em disciplinas de programação de computadores causa desmotivação nos alunos, elevando taxas de reprovação e desistência. Nesse contexto, o presente trabalho aborda uma metodologia de ensino que utiliza corretores automáticos de códigos e questões objetivas como forma de consolidar os assuntos abordados, mantendo o aluno informado sobre seu desenvolvimento e atenuando possíveis dificuldades na aprendizagem. Essa metodologia foi aplicada e avaliada em turmas da graduação de Engenharia, demostrando o potencial positivo no aprendizado dos alunos e, consequentemente, em seu desempenho acadêmico na disciplina.*

1. Introdução

De acordo com as Diretrizes Curriculares Nacional dos Cursos de Graduação de Engenharia, disciplinas correlatas a algoritmos e programação devem estar presentes na grade curricular de diversas Engenharias. A inserção de tais disciplinas logo no ciclo básico ocorre, principalmente, devido à necessidade do desenvolvimento do raciocínio lógico do aluno, uma capacidade útil ao longo do curso (Marcussi *et al.* , 2016).

Tais disciplinas são consideradas muito desafiadoras, visto que exigem o desenvolvimento de estratégias de solução de problemas com base lógico-matemática. Na maior parte da vezes, o contato inicial é feito apenas na graduação, sujeitando o aluno a desenvolver o pensamento computacional em um curto espaço de tempo. Vários outros fatores podem dificultar ainda mais o processo de aprendizagem, como por exemplo: uso do inglês como principal idioma das linguagens de programação; desinteresse dos alunos por considerar estas disciplinas como desnecessárias para sua formação enquanto engenheiro; necessidade de se fazer essas disciplinas em conjunto com disciplinas consideradas "críticas", ou difíceis, como cálculo e álgebra.

Diversas ferramentas vêm sendo exploradas no intuito de aprimorar o processo ensino-aprendizagem em disciplinas relacionadas à programação e, ainda, automatizar o

DOI: 10.5753/cbie.sbie.2019.449

trabalho dos docentes com relação à correção de exercícios e emissão de *feedback* aos alunos. Em se tratando dos modelos de ensino atuais, com salas super lotadas, torna-se cada vez mais difícil atender e suprir as dificuldades individuais do aluno. Além disso, elas podem também contribuir para identificar de maneira geral quais conteúdos devem ser melhor explorados devido às dificuldades dos alunos em resolver determinados exercícios.

Nesse trabalho, avalia-se como a introdução de corretores automáticos na metodologia de ensino de programação de computadores para alunos de engenharia contribui para o aprendizado do conteúdo abordado. A ideia é fazer com que os alunos acompanhem sua evolução no curso mediante *feedback* rápidos e explicativos e, também, atenuar os trabalhos dos docentes com relação à correção das inúmeras tarefas e exercícios aplicados. Implementou-se um corretor automático de código voltado para o ensino de programação, que realiza uma série de processamentos da saída gerada pelo código em avaliação, a fim de priorizar os aspectos da lógica de programação em detrimento aos detalhes de saída irrelevantes para o aprendizado. Resultados experimentais indicam de que a metodologia de ensino e as ferramentas utilizadas neste trabalho cumprem com seu objetivo ao auxiliar os alunos no desenvolvimento do raciocínio lógico e ao mantê-los motivados a realizarem as atividades propostas.

Este artigo está organizado como se segue. A Seção 2 apresenta os principais trabalhos relacionados a esta pesquisa. A Seção 3 aborda a metodologia de ensino utilizada. Na Seção 4 é descrita uma ferramenta de apoio à metodologia. Na Seção 5 os resultados são discutidos. Por fim, a Seção 6 apresenta conclusões e sugestões de trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos introduzem em suas metodologias de ensino ferramentas computacionais no intuito de auxiliar o processo ensino-aprendizagem em disciplinas introdutórias de programação. A seguir, serão apresentadas algumas dessas ferramentas bem como os resultados observados após implantação das mesmas.

Corretores automáticos de código, ou Juízes-online, são ferramentas acessíveis na Internet que permitem automatizar o processo de correção de exercícios de programação. Esses ambientes possuem a descrição de exercícios com entradas e saídas já pré-definidas e abordam os mais variados temas relacionados à programação. A proposta desenvolvida por Jesus *et al.* (2018) parte do pressuposto de que as mensagens de erro emitidas pelos corretores automáticos não são intuitivas. O problema se agrava quando o estudante não possui conhecimento do idioma inglês, visto que grande parte das linguagens de programação emitem *feedback* nesse idioma. Para contornar esse problema, os autores desenvolveram uma ferramenta capaz de emitir mensagens de erros mais sugestivas. Os resultados revelam indícios de que a apresentação de mensagens amigáveis podem melhor guiar os alunos para a correção dos erros. Entretanto, a ferramenta é capaz de detectar e auxiliar apenas em erros relacionados à sintaxe da linguagem abordada.

Raabe *et al.* (2015) apresentam o *framework* de um corretor automático que combina análise estática e dinâmica dos códigos para emitir *feedback* aos alunos relacionados aos erros encontrados. Apesar da ferramenta ter sido avaliada utilizando uma amostra relativamente pequena (23 alunos), constatou-se que o corretor gerou um impacto positivo na dinâmica de resolução e depuração das soluções pelos alunos, entretanto revelou a necessidade de se utilizar mais casos de testes estáticos durante a avaliação.

O URI *Online Judge* é uma ferramenta online que permite praticar a programação e compartilhar o conhecimento, além de fornecer *feedback* em tempo real para os usuários que submeterem exercícios em sua plataforma. Berssanette & Carlos (2018) apresentam uma metodologia de ensino para a disciplina de Algoritmos e Lógica em que utilizam o portal do URI *Online Judge* aliado a técnica de PBL (método de ensino centrado no aluno, consistindo, basicamente, na resolução de exercícios como forma de aprendizagem). Setenta e oito por cento dos alunos que cursaram a disciplina seguindo essa metodologia obtiveram índices satisfatórios.

Os trabalhos citados anteriormente fazem uso de ferramentas online que fornecem *feedback* em tempo real aos alunos, permitindo que eles identifiquem seus erros e os corrijam o mais rápido possível. Entretanto, essas ferramentas são muito rigorosas com relação à padrões de entrada e saída dos problemas, sendo adequadas para competições de programação mas limitadas do ponto de vista didático no ensino de programação. O objetivo da metodologia proposta é valorizar o raciocínio lógico do aluno, através de um corretor automático de código com critérios de avaliação mais justos e menos restritivos em conjunto com questões objetivas, também corrigidas de forma automática.

3. Metodologia de ensino

A metodologia de ensino proposta neste trabalho foi aplicada na disciplina de Programação de Computadores para alunos de variados cursos de engenharia da Universidade Federal de Ouro Preto. A estrutura das aulas é descrita na Figura 1.

As aulas teóricas têm como objetivo apresentar conceitos de programação e os recursos e sintaxe de uma linguagem de programação capazes de resolver problemas computacionais. Primeiramente, o professor apresenta um problema do cotidiano que pode ser resolvido computacionalmente e conduz os alunos a refletirem de qual maneira eles o resolveriam. Em seguida, são apresentados os conceitos necessários para resolvê-lo bem como uma possível solução para o mesmo em uma linguagem de programação.

As aulas práticas têm como objetivo promover a fixação dos conceitos e recursos da linguagem de programação com o intuito de resolver problemas práticos, relacionados ao conteúdo abordado nas aulas teóricas, através de um programa de computador. Estas aulas acontecem nos laboratórios de informática e são divididas em *questões objetivas* e *exercícios práticos*.

Questões objetivas correspondem à aplicação de perguntas de múltipla escolha disponibilizadas através do Moodle e que permitem avaliar o entendimento do aluno com relação ao pensamento lógico na resolução de problemas, à estrutura sintática da linguagem de programação e ao fluxo de execução dos programas. A avaliação dessas questões é feita de maneira automática através de recursos oferecidos pelo próprio Moodle.

Figura 1. Estrutura das aulas da disciplina de programação de computadores.

Os exercícios práticos, por sua vez, correspondem à codificação de soluções de problemas. Estes problemas normalmente exploram situações do cotidiano que podem ser resolvidas computacionalmente através de um algoritmo. São propostos aos alunos entre 3 e 6 exercícios em cada aula prática. Dado o volume de exercícios e ao elevado número de alunos, torna-se impraticável que o professor corrija todos os exercícios. Portanto, para avaliar os exercícios práticos, foi desenvolvida uma ferramenta que permite fazer a correção automática baseando-se em critérios pré definidos, detalhado na Seção 4.

Para estimular os alunos a estarem presentes nas aulas práticas e realizarem as atividades no horário proposto, foram definidas penalizações nas notas para aqueles que não entregam as atividades durante a aula. Os alunos podem entregar os exercícios resolvidos mesmo que estejam ausentes na aula até um prazo final que excede o horário de término da aula. São aplicadas penalidades fixas pela ausência em aula e penalidades graduais para as entregas posteriores ao horário de término da aula.

4. Corretor automático de código

Os corretores automáticos citados anteriormente avaliam apenas a sintaxe da linguagem ou avaliam aspectos gerais da execução do programa. Especificamente em relação à saída originada pela execução da solução do aluno, eles apenas avaliam se ela está exatamente igual à saída esperada, eventualmente apresentando um percentual geral de similaridade. Do ponto de vista pedagógico, este tipo de *feedback* ao aluno limita consideravelmente a capacidade de auxilia-lo no aprendizado e em valorizar os aspectos mais relevantes da solução do problema em relação aos conceitos de programação em avaliação.

Portanto, diferentemente dos juízes automáticos disponíveis, o corretor implementado neste trabalho avalia o grau de similaridade entre a resposta correta para um exercício e a resposta obtida ao executar o programa do aluno, particionando a saída e ponderando a pontuação de acordo com a importância e o grau de dificuldade de cada um de seus seguimentos. As subseções a seguir descreverão detalhadamente as três etapas do corretor automático de códigos: (1) pré-processamento, (2) correção e (3) *feedback* ao aluno.

4.1. Pré-processamento

Antes de iniciar a correção propriamente dita é necessário definir casos de teste que serão utilizados na validação de cada solução submetida pelo aluno. Cada caso de teste é composto por um arquivo de entrada e um arquivo de critérios. Os arquivos de entrada são arquivos texto que simulam a entrada de dados para um programa enquanto os de critério são arquivos texto que definem a saída esperada para cada arquivo de entrada bem como as pontuações associadas a cada segmento de saída. Os critérios são definidos levando em consideração que existem elementos presentes na saída que são mais relevantes que outros. Por exemplo, em questões que possuem em sua saída mensagens de texto padrão e cálculos matemáticos, a presença deste último deve ser melhor pontuada, uma vez que mensagens de texto padrão não exigem nenhum raciocínio lógico para implementação, apenas devem ser exibidas.

Destaca-se que, normalmente os exercícios disponibilizados em plataformas de competições de programação, como o URI *Online Judge* citado anteriormente, apresentam padrões mais "objetivos" para entrada e saída, eliminando quaisquer mensagens textuais consideradas não relevantes para a lógica de solução do problema, tais como "Digite o valor de X:" para uma entrada, ou "As raízes da equação são:" para uma saída

calculada. Novamente, do ponto de vista pedagógico, para alunos que estejam iniciando em programação, ou alunos de cursos que não estejam diretamente relacionados à Computação, como os cursos de Engenharia, por exemplo, esta abordagem não é adequada, pois torna a programação de computadores algo ainda mais abstrato e artificial.

4.2. Correção automática

As soluções implementadas pelos alunos para resolver os problemas propostos em cada aula prática, ou atividade extraclasse, são submetidas em uma tarefa do Moodle. O corretor automático implementado é executado em modo offline, sendo assim, os arquivos submetidos pelos alunos são primeiramente armazenados localmente de forma padronizada e, em seguida, o corretor automático é executado. A Figura 2 apresenta o fluxograma de execução do corretor automático e os detalhes são descritos a seguir.

Primeiramente, o código do aluno é executado para o caso de teste em análise. Através de redirecionamento do sistema operacional são fornecidas as entradas para o caso de teste e é gerada a saída do aluno. Na ocorrência de algum erro léxico ou sintático ou algum erro de execução, a saída do aluno não é gerada, mas a ocorrência é identificada pelo corretor e uma mensagem de erro é retornada em lugar da saída do aluno. Laços infinitos são identificados por critério de tempo de execução, caso a execução exceda um tempo limite é considerado que o erro ocorreu. No caso do programa concluir a execução sem erros, um arquivo textual contendo a saída resultante da execução do programa do aluno é gerado e avaliado.

Para calcular a pontuação do aluno, mede-se o grau de similaridade entre a saída esperada e a saída obtida considerando os critérios estabelecidos. Utiliza-se uma função de comparação que, baseada na Distância *Levenshtein* (Levenshtein, 1966), retorna um valor representando o percentual de similaridade entre as strings comparadas. Em síntese, a Distância *Levenshtein* corresponde a uma distância de "edição", baseada na quantidade

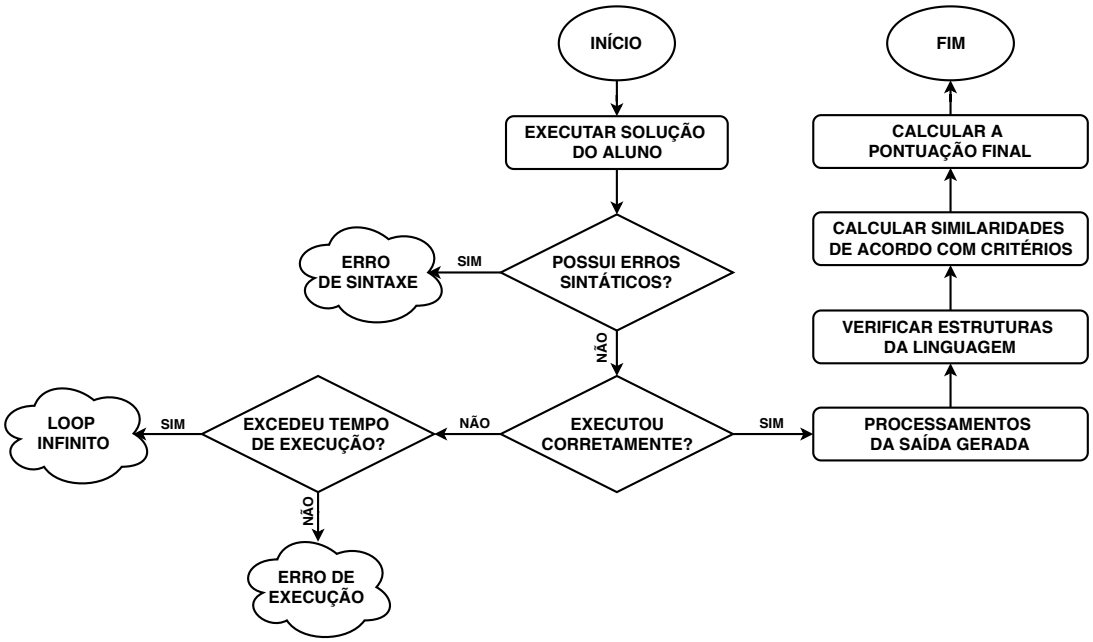


Figura 2. Fluxograma de execução do corretor automático.

de caracteres que precisam ser substituídos, inseridos ou excluídos para tornar as duas strings em comparação iguais. A pontuação final é definida pela média dos critérios estabelecidos ponderada pelas similaridades calculadas.

Visando priorizar as questões lógicas da programação em detrimento às questões menos relevantes para o aprendizado, a saída do aluno passará por uma série de processamentos antes de ser comparada à saída esperada, conforme descrito a seguir.

Eliminam-se as linhas em branco. As linhas em branco podem causar desordenação da saída fazendo com que partes da resposta do aluno sejam desconsideradas em situações em que seu raciocínio lógico está correto mas a correlação entre as linhas de saída obtidas fica incorreto em relação à saída esperada.

Eliminam-se a acentuação de caracteres. Um caractere sem a acentuação correta pode provocar penalidades significativas à solução do aluno devido ao cálculo de similaridade, embora ele possa ter aplicado o raciocínio lógico correto para resolver o problema.

Eliminam-se caracteres especiais. Os caracteres especiais, sobretudo em exercícios de programação, são requisitados quando se deseja construir uma saída mais atrativa, não possuindo, na maioria dos casos, relação com a lógica a ser desenvolvida.

Eliminam-se espaços em branco. É muito comum que os alunos imprimam espaços em branco diferente do esperado na elaboração da saída de sua solução. Devido à irrelevância desta questão em relação à lógica de programação, optou-se por eliminar também os espaços em branco.

Eliminam-se os sinais de pontuação, exceto o ponto final. A presença e a falta dos sinais de pontuação na saída do aluno não é considerada relevante na avaliação da lógica desenvolvida, por isso são descartados. O motivo de se manter o ponto final é porque em situações que envolvem cálculos matemáticos com números reais, por exemplo, é necessário identificar as partes inteira e fracionada.

O texto é convertido para letras minúsculas. Salvo casos especiais, pode-se considerar que uma mensagem escrita em letra maiúscula não é diferente dessa mesma mensagem escrita em letra minúscula, por isso todos os caracteres alfabéticos são substituídos por seu corresponde em minúsculo.

Tais processamentos se justificam devido aos alunos, em sua maioria recém concluintes do ensino médio, não estarem habituados a essa nova forma de pensamento, em que padrões especificados devem ser seguidos à risca. Em versões preliminares do corretor automático observou-se que mesmo quando a lógica existente para resolução do problema estava correta, a pontuação obtida era muito penalizada pelo simples fato do formato das mensagens de saída não ter sido seguido rigorosamente. Do ponto de vista pedagógico, essas penalizações podem ser consideradas injustas e acabar desmotivando os alunos e/ou prejudicando seu desempenho na disciplina. Para ilustrar o efeito da ausência destes processamentos, a Figura 3 mostra o efeito de uma linha em branco inserida na saída do aluno e o impacto causado pelo seu processamento na pontuação computada. Considerando que as duas linhas possuem o mesmo peso na avaliação, sem eliminar a linha em branco o aluno recebeu pontuação apenas em relação à linha 1, atribuindo-lhe metade da nota, enquanto que, com a eliminação da linha em branco, a comparação entre a saída esperada e a saída obtida lhe atribui nota total.

Saída Esperada	Saída Obtida	Pontuação
Linha 1: PERÍMETRO DO TRIÂNGULO = 28 m Linha 2: ÁREA DO TRIÂNGULO = 36.6606 m^2	Linha 1: PERÍMETRO DO TRIÂNGULO = 28 m Linha 2: Linha 3: ÁREA DO TRIÂNGULO = 36.6606 m^2	5,0

(a) Correção sem a eliminação de linhas em branco.

Saída Esperada	Saída Obtida	Pontuação
Linha 1: PERÍMETRO DO TRIÂNGULO = 28 m Linha 2: ÁREA DO TRIÂNGULO = 36.6606 m^2	Linha 1: PERÍMETRO DO TRIÂNGULO = 28 m Linha 2: ÁREA DO TRIÂNGULO = 36.6606 m^2	10,0

(b) Correção com a eliminação de linhas em branco.

Figura 3. Correção de uma saída em relação ao processamento de linhas em branco.

Para determinadas situações é necessário verificar o uso de certas estruturas da linguagem que não podem ser identificadas apenas pela análise da saída da execução do programa. Atualmente tratamos apenas o uso de funções quando solicitadas explicitamente no enunciado do exercício. A detecção de funções é feita percorrendo o arquivo de código fonte do aluno e identificando palavras reservadas da linguagem que devem ser utilizadas quando se deseja definir uma função. Apesar de ser uma abordagem ingênua, o corretor consegue tratar situações em que o aluno sequer se preocupa em criar funções, mesmo que o enunciado deixe claro que isso deve ser feito. Observe que, atualmente, este é o único critério de avaliação aplicado ao código fonte do aluno, todos os demais aplicam-se à processamentos e análises da saída gerada pela execução do código fonte.

4.3. *Feedback* ao aluno

Após a correção dos casos de teste para os exercícios da tarefa, inicia-se a etapa de geração de *feedback* ao aluno. Esta etapa é muito importante para consolidar o aprendizado, pois é através da identificação de seus erros e acertos é que o aluno adquire experiência e, consequentemente, aprimora seu conhecimento.

Como a versão atual do corretor automático é executada em modo offline, optou-se por gerar um arquivo em *Portable Document Format* (PDF). Um arquivo padrão em LaTeX foi construído e é preenchido com todas as informações referentes às correções da tarefa de cada aluno. Os arquivos PDF originados são então enviados por *e-mail*, individualmente para cada aluno, também de forma automatizada. Um arquivo PDF gerado para um aluno possui todas as informações a respeito da correção, contendo principalmente: (a) síntese do resultado geral do aluno contendo as notas em cada questão e as penalidades aplicadas; (b) síntese do resultado parcial de cada questão; e (c) detalhamento do resultado de cada questão de acordo com os critérios estabelecidos.

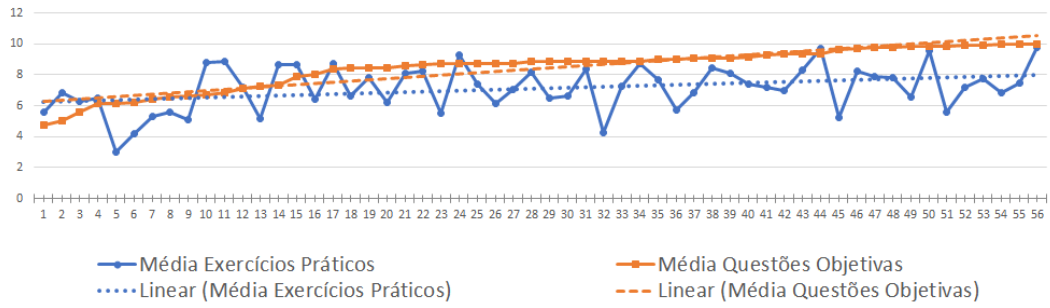
5. Resultados

A proposta metodológica apresentada foi aplicada durante um semestre letivo para quatro turmas de Engenharia de uma universidade pública federal, totalizando 68 alunos envolvidos. Para efeito avaliativo, relacionou-se o desempenho dos alunos nas atividades práticas (segmentando em questões objetivas e questões práticas) e nas avaliações teóricas. Foi aplicado também um questionário para avaliarmos o efeito dessa metodologia do ponto de vista do aluno após a conclusão do semestre letivo.

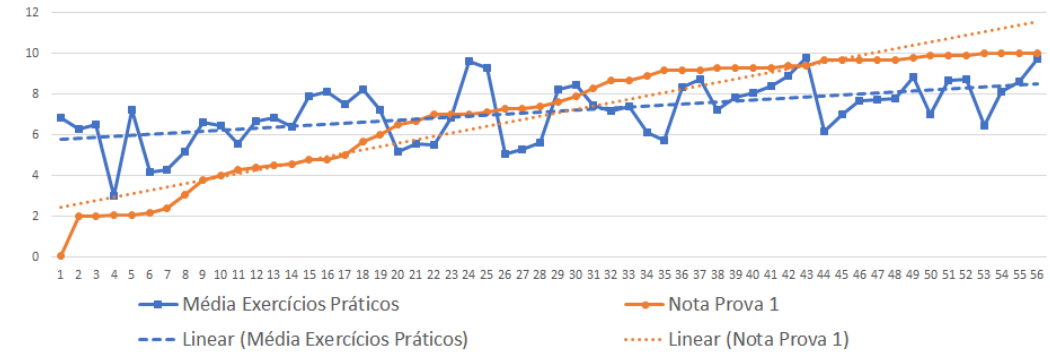
A Figura 4 apresenta gráficos comparativos entre as notas dos alunos nas diferentes atividades avaliativas da prova 1. Alunos infrequentes ou que não realizaram as atividades avaliativas foram removidos. Por limitação de espaço os resultados para a prova 2 e sobre as médias finais da disciplina não serão apresentados. No entanto, os mesmos padrões podem ser observados nestes resultados.

A Figura 4a compara as notas médias em questões objetivas e exercícios práticos de cada aluno relacionadas à prova 1. Os alunos foram ordenados pela sua nota nas questões objetivas. As linhas de tendência crescentes denotam que alunos que resolvem corretamente as questões objetivas tendem a se saírem bem também na resolução dos exercícios práticos. A Figura 4b compara as notas médias nas provas escritas e exercícios práticos para cada aluno, também relacionadas à prova 1. Novamente, temos linhas de tendência crescentes, indicando que alunos que se saem bem na resolução dos exercícios práticos tendem a ter também, um bom desempenho nas avaliações teóricas.

Alguns casos nos resultados observados geram questionamentos. Como justificar, por exemplo, alunos que obtêm um bom desempenho nas atividades práticas mas se saem mal nas avaliações teóricas? Algumas hipóteses foram levadas para trabalhos futuros: (1) *o aluno copiou soluções sem se importar em entendê-las?*, (2) *o aluno consegue realizar as tarefas no computador mas no papel não?*, (3) *fatores físicos e psicológicos afetam o desempenho em avaliações escritas?*, (4) *o auxílio dos instrutores nas aulas práticas, que não ocorrem durante as provas escritas, influencia nos resultados?*.



(a) Exercícios práticos *vs.* Questões objetivas



(b) Prova escrita *vs.* Exercícios práticos

Figura 4. Gráficos comparativos de desempenho dos alunos na prova 1. O eixo x representa os alunos e o eixo y suas notas. Linhas pontilhadas representam linhas de tendência.

Um questionário de opinião foi aplicado para avaliar, sob o ponto de vista dos alunos, como o uso da metologia proposta impactou em seu desempenho na disciplina. Vinte e três alunos responderam ao questionário. A Tabela 1 apresenta o percentual de respostas obtido para as principais perguntas, as respostas permitidas eram valores inteiros entre 1 e 5, quanto maior o valor, maior a satisfação do aluno. As perguntas apresentadas são: (1) *Questões objetivas ajudam na fixação de conteúdo teórico?*, (2) *Questões práticas ajudam na fixação de conteúdo teórico?*, (3) *Satisfação com resultado gerado pelo corretor automático*, (4) *As penalidades aplicadas motivam a realização das tarefas no prazo?*.

A maior parte dos alunos que responderam ao questionário obtiveram um bom desempenho na disciplina, tanto nas atividades práticas quando na resolução das provas teóricas. Além disso, eram alunos frequentes e consideravam a aplicação de questões objetivas e exercícios práticos como importantes para a fixação do conteúdo teórico.

A avaliação do resultado gerado pelo corretor automático dividiu opiniões. Uma das hipóteses para as avaliações negativas se deve ao fato da ferramenta trabalhar em modo offline. Ou seja, só depois de um tempo os alunos obtêm a resposta se suas implementações estavam corretas ou não. Outra hipótese pode ser a falta de compreensão dos resultados gerados pelo corretor ou discordância das avaliações feitas pelo mesmo.

As penalizações aplicadas motivaram muitos os alunos a estarem presentes nas aulas e a realizarem os exercícios dentro do horário proposto, mas não agradou a todos. Uma estratégia futura a ser explorada é a de bonificar os alunos que se dedicarem na resolução dos exercícios durante as aulas em lugar de aplicar penalidades.

Em uma pergunta adicional, onde os alunos responderam sobre o que foi mais útil para eles na disciplina, podendo selecionar múltiplas opções de respostas, conforme apresentado na Figura 5, observa-se que o percentual mais expressivo de respostas foi para as aulas práticas, ocorrendo em $69,6\%$ das respostas. Além deste resultado destacar a importância das aulas práticas no processo de aprendizado, dentre as vários opções de resposta à disposição, esta é justamente a opção onde as correções automáticas foram diretamente aplicadas, reforçando ainda mais sua utilidade no aprendizado dos alunos.

6. Conclusões e Trabalhos Futuros

A proposta metodológica descrita neste trabalho baseou-se no uso de corretores automáticos para exercícios práticos e questões objetivas a fim de auxiliar o processo ensino-aprendizagem de alunos da Engenharia da Universidade Federal de Ouro Preto na disciplina de programação de computadores.

Tabela 1. Questionário de opinião aplicado aos alunos. Respostas são valores inteiros entre 1 e 5, quanto maior o valor, maior a satisfação do aluno.

Perguntas \ Respostas (%)	1	2	3	4	5
(1) Quanto às questões objetivas	$4,3$	$8,7$	$34,8$	$30,4$	$21,7$
(2) Quanto às questões práticas	0	0	$17,4$	$17,4$	$65,2$
(3) Quanto ao Corretor automático	13	$26,1$	$26,1$	$34,8$	0
(4) Quanto às penalidades aplicadas	13	$4,3$	$26,1$	$17,4$	$39,1$

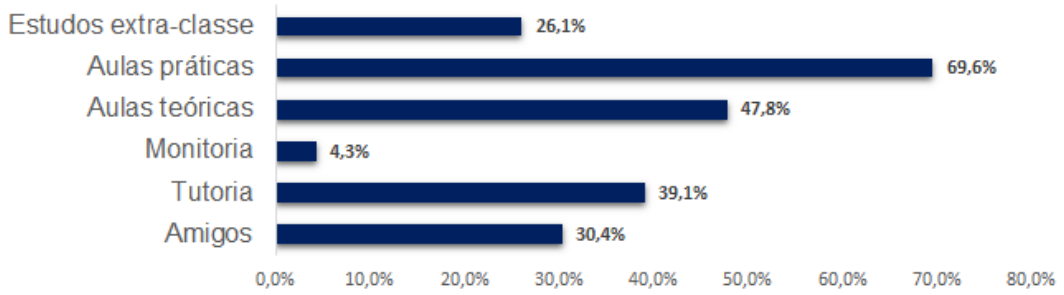


Figura 5. Respostas à pergunta *O que mais te ajudou na disciplina?*.

A metodologia trouxe importantes resultados, sobretudo no que diz respeito ao rendimento acadêmico dos alunos que a seguiram. Foi possível comprovar, através de análises numéricas do desempenho dos alunos e questionário de opinião, que a metodologia proposta foi bem sucedida.

Como sugestões de trabalhos futuros destaca-se a necessidade de se adaptar o corretor automático para ser executado de forma online, possibilitando um *feedback* mais rápido e útil para promover melhor o aprendizado dos alunos. A aplicação de bonificação ao invés de penalidades em relação às tarefas a serem cumpridas, fundamentadas através de técnicas de gamificação. Por fim, o levantamento e resposta a novas hipóteses relacionadas à utilidade da metodologia e ao rendimento acadêmico dos alunos.

Agradecimentos

O presente trabalho foi realizado com apoio da Pró-Reitoria de Extensão (PROEX) da Universidade Federal de Ouro Preto (UFOP), projeto PRJ1359.

Referências

Berssanette, J. H. de F., & Carlos, A. 2018. Uma Proposta de Ensino de Programação de Computadores com base na PBL utilizando o portal URI Online Judge. *Anais do Simpósio Ibero-Americano de Tecnologias Educacionais*, 348–354.

Jesus, G., Santos, K., Conceicao, J., Ribeiro, E., & Neto, A. 2018. Avaliação de uma abordagem para auxiliar a correção de erros de aprendizes de programação. BRAZILIAN SYMPOSIUM ON COMPUTERS IN EDUCATION, 2018, Fortaleza. Anais do SBIE 2018 (Proceedings of the SBIE 2018).

Levenshtein, Vladimir Iosifovich. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, **10**(8), 707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

Marcussi, L. D., Guedes, K., Dal M. F., Rafael G., Santiago F., Robertino M., & Junior, C. R. B.i. 2016. PESQUISA NO ENSINO DE ALGORITMOS E PROGRAMAÇÃO NAS ENGENHARIAS: ESTUDOS E RESULTADOS PRELIMINARES. *In: Simpósio de Engenharia de Produção*.

Raabe, A., de Jesus, E. A., Hodecker, A., & Pelz, F. 2015. Avaliaçao do feedback gerado por um corretor automático de algoritmos. *Page 358 of: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 26.