UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TÁGNER FORMANSKI ROSA

ANÁLISE COMPARATIVA DE UM ALGORITMO DE BUSCA E SATISFAÇÃO DE RESTRIÇÕES E ALGORITMO GENÉTICO EM UM SISTEMA PARA GERAÇÃO DE HORÁRIO ESCOLAR

CRICIÚMA

2015

TÁGNER FORMANSKI ROSA

ANÁLISE COMPARATIVA DE UM ALGORITMO DE BUSCA E SATISFAÇÃO DE RESTRIÇÕES E ALGORITMO GENÉTICO EM UM SISTEMA PARA GERAÇÃO DE HORÁRIO ESCOLAR

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel em Ciência da Computação, da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: MSc. Luciano Antunes

CRICIÚMA

2015

TÁGNER FORMANSKI ROSA

ANÁLISE COMPARATIVA DE UM ALGORITMO DE BUSCA E SATISFAÇÃO DE RESTRIÇÕES E ALGORITMO GENÉTICO EM UM SISTEMA PARA GERAÇÃO DE HORARIO ESCOLAR

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Algoritmos.

Criciúma, 23 de novembro de 2015.

BANCA EXAMINADORA

Prof. MSc. Luciano Antunes – (UNESC) - Orientador

Profª MSc. Christine Vieira – (UNESC)

Profª Dra. Merisandra Côrtes de Mattos Garcia – (UNESC)

Dedico esse trabalho aos meus pais que sempre me apoiaram e permitiram que eu concluísse os estudos em busca de um futuro melhor.

AGRADECIMENTOS

Agradeço primeiramente a deus que, me guiou e não me deixou desvirtuar do caminho traçado.

Aos meus pais que sempre me orientam e ajudam, pois sem eles eu não conseguiria concluir minha faculdade.

Agradeço a minha mãe Vanilda Formanski Rosa, por sempre buscar o melhor para mim, sempre lutar para que eu consiga alcançar os meus objetivos. Me dando bastante carinho e amor para que eu consiga fazer as tarefas tranquilamente.

Agradeço ao meu pai Zeli Mario Rosa, por ser o meu melhor amigo, me dar esporros, mais aqueles esporros que só beneficiam o filho, querendo ver sempre o bem.

Agradeço o meu irmão Tiago Rosa, por me ajudar e ser paciente quando necessário.

Também ao meu professor e orientador Luciano Antunes por proporcionar essa experiência incrível e apostar em mim, assim como agradeço o auxilio da banca examinadora e de todos os professores que tive em minha vida, pela educação e carinho que me trataram.

“Às vezes a vida vai-te golpear a cabeça com um tijolo. Não percas a fé.”

Steve Jobs

RESUMO

O objetivo deste estudo foi obter uma análise comparativa de desempenho entre os algoritmos de busca e satisfação de restrições *backtracking* e genético em um protótipo de horário escolar. Para tanto, utilizou-se o software Cronos (algoritmo genético) e desenvolveu-se um protótipo de geração de horário escolar, utilizando o algoritmo *backtracking*. A métrica ou modelo de avaliação da qualidade adotada foi a ISO/IEC 9126 – Parte 1 (NBR 13596), que fornece um modelo de propósito geral. Esta norma define seis amplas categorias de características de qualidade de software que são: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Essas categorias, por sua vez, são divididas em subcaracterísticas. Por meio da aplicação dos critérios estabelecidos, identificou-se que os protótipos utilizando-se o algoritmo de *backtracking* e genético apresentam fatores qualitativos praticamente similares em todas as categorias analisadas, ressaltando-se que no primeiro teste, o protótipo *backtracking* foi mais eficiente na subcategoria relacionada ao tempo, enquanto no segundo teste foi o algoritmo genético. Dessa forma, acredita-se que ambos são viáveis, adequados e suficientemente qualitativos para a aplicação na organização de horários escolares com maior eficiência e menos tempo de espera pelos resultados.

Palavras-chave: Algoritmo *Backtracking*. Algoritmo Genético. Análise Comparativa. Geração de Horário Escolar. Problema Satisfação de Restrições.

ABSTRACT

The objective of this study was to obtain a comparative analysis of performance between search algorithms and satisfaction of backtracking and genetic constraints on a school day prototype. For this, we used the Kronos software (genetic algorithm) and developed a prototype of the school timetable generation, using the backtracking algorithm. The metric or model assessment of quality was adopted ISO / IEC 9126 - Part 1 (NBR 13596), which provides a general purpose model. This standard defines six broad categories of software quality features are: functionality, reliability, usability, efficiency, maintainability and portability. These categories, in turn, are divided into subcharacteristics. Through the application of the criteria, it identified that the prototypes using the backtracking and genetic algorithm have almost similar qualitative factors in all categories analyzed, highlighting that in the first test, the backtracking prototype was more efficient in the subcategory related to the time while the second test was the genetic algorithm. Thus, it is believed that they are viable and sufficiently qualitative suitable for application to the organization school schedules with greater efficiency and less time waiting for results.

Keywords: Backtracking Algorithm. Genetic Algorithm. Comparative Analysis. School Time generation. Constraint Satisfaction problem.

LISTA DE ILUSTRAÇÕES

Figura 1 - Algoritmo de busca que verifica se a turma já completou a carga horária semanal....22
Figura 2 - Algoritmo de Satisfação de Restrições implementado para o problema analisado....25
Figura 3 - Estrutura de uma busca binária....26
Figura 4 - Estrutura de uma busca linear....27
Figura 5 - Modelo de qualidade para qualidade externa e interna....43
Figura 6 - Modelo de avaliação utilizado....52
Figura 7 - Arquivo xml de persistência do Hibernate....53
Figura 8 - Tela principal do protótipo....54
Figura 9 - Modelo entidade relacionamento....56
Figura 10 - Diagrama de Caso de Uso....57
Figura 11 – Restrição fraca: Tenta intercalar disciplina difíceis com as fáceis....59
Figura 12 – Restrição fraca: Função que define a quantidade que o professor pode lecionar em uma turma no dia....59
Figura 13 – Restrição fraca: Irá selecionar um professor que menos lecionou na disciplina correspondente....59
Figura 14 – Restrição forte: Verificação que garante que uma aula será de 45 min. 60
Figura 15 - Restrição forte: Verificação que garante que uma semana terá 25 aulas de 45 min....60
Figura 16 – Restrição forte: Evita o choque de horário do professor....61
Figura 17 – Situação simulada....61
Figura 18 – Atribuição dos valores do protótipo desenvolvido....62
Figura 19– Processo de retroceder....63
Figura 20 - Processo que retrocede na geração do horário GHE....63
Figura 21 - Horário turma 1-A....65
Figura 22 - Horário turma 2-A....66
Figura 23 - Horário turma 3-A....66
Figura 24 - Horário turma 4-A....67
Figura 25 - Horário turma 6-A....68
Figura 26 - Horário turma 7-A....68
Figura 27 - Horário turma 8-A....68

Figura 28 - Horário turma 9-A..69

LISTA DE QUADROS

Quadro 1 - Características da qualidade do software segundo a ISO/IEC 9126/1 ... 44
Quadro 2 - Comparativo entre os protótipos conforme o critério da funcionalidade .. 70
Quadro 3- Comparativo entre os protótipos conforme o critério da confiabilidade 71
Quadro 4 - Comparativo entre os protótipos conforme o critério da usabilidade 72
Quadro 5 - Comparativo entre os protótipos conforme o critério da eficiência 72
Quadro 6 - Comparativo entre os protótipos conforme critério da manutenibilidade . 74
Quadro 7 - Comparativo entre os protótipos conforme o critério da portabilidade 74

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Programming Interface*
AG	Algoritmo Genético
IA	Inteligência Artificial
IDE	*Integrated DevelopmentEnvironment*
IEC	*International Engineering Consortium*
ISO	Organização Internacional de Normalização
GHE	Geração de Horário Escolar
JPA	Java *Persistence* API
ORM	Mapeamento Objeto-Relacional
PPP	Projeto Político-Pedagógico
PSR	Problema de Satisfação de Restrição
PR	Programação com Restrições
SGBD	Sistemas de Gerenciamento de Banco de Dados
UFLA	Universidade Federal de Lavras
TTP	*Timetabling Problem*

SUMÁRIO

1 INTRODUÇÃO **7**
1.1 OBJETIVO GERAL 8
1.2 OBJETIVOS ESPECÍFICOS 8
1.3 JUSTIFICATIVA 9
1.4 ESTRUTURA DO TRABALHO 10
2 HORÁRIO ESCOLAR 12
2.1 TIMETABLING PROBLEM - TTP 13
2.2 RESTRIÇÕES NO DESENVOLVIMENTO DOS HORÁRIOS ESCOLARES 15
2.3 PROBLEMAS 18
2.4 COMPLEXIDADE 19
3 ALGORITMO 21
3.1 ALGORITMO DE BUSCA 21
3.2 ALGORITMO DE SATISFAÇÃO DE RESTRIÇÕES 22
3.2.1 Busca binária e linear **25**
3.2.2 Algoritmos *backtracking* **27**
3.3 ALGORITMOS GENÉTICOS 29
4 MÉTRICAS DE COMPARAÇÃO 35
4.1 MEDIDAS DIRETAS 37
4.2 MEDIDAS INDIRETAS 37
4.3 TIPOS DE MÉTRICAS 38
4.3.1 Métricas orientadas ao tamanho **38**
4.3.2 Métricas orientadas a função **38**
4.3.3 Métricas de qualidade do produto **39**
4.3.4 Métricas de processo **40**
4.3.5 Métricas objetivas e subjetivas **40**
4.3.6 Métricas primitivas e compostas **41**
4.3.7 Métricas de Halstead **41**
4.4 AVALIAÇÃO DA QUALIDADE SEGUNDO A NORMA iso/IEC 2026 41
4.5 ESCALAS DE MEDIÇÃO 44
5 TRABALHO CORRELATOS **46**

5.1 DEFINIÇÃO E IMPLEMENTAÇÃO DE UMA FUNÇÃO DE AVALIAÇÃO PARA UM SISTEMA DE GERAÇÃO DE GRADE hORÁRIA QUE UTILIZA COMO MÉTODO DE BUSCA ALGORITMO GENÉTICO 46
5.2 GERAÇÃO DA GRADE HORÁRIA DO CURSO DE ENGENHARIA DE PRODUÇÃO DA UFPR ATRAVÉS DE PROGRAMAÇÃO LINEAR BINÁRIA.....47
5.3 AMBIENTE DE OTIMIZAÇÃO NA WEB: UMA APLICAÇÃO EM TIMETABLING47
5.4 UM ALGORITMO HÍBRIDO BASEADO EM ALGORITMOS MEMÉTICOS E RECONEXÃO POR CAMINHOS PARA RESOLUÇÃO DO PROBLEMA DE HORÁRIO ESCOLAR 48
5.5 GERAÇÃO AUTOMÁTICA DE HORÁRIOS ESCOLARES UTILIZANDO ALGORITMOS DE SATISFAÇÃO DE RESTRIÇÕES.....49
5.6 UMA SOLUÇÃO DO PROBLEMA DE HORÁRIO ESCOLAR VIA ALGORITMO GENÉTICO PARALELO.....50
6 TRABALHO DESENVOLVIDO.....51
6.1 METODOLOGIA.....51
6.2 PROTÓTIPO GHE – GERAÇÃO DE HORÁRIO ESCOLAR.....52
6.2.1 Funcionamento do GHE.....54
6.2.2 Modelagem de bancos de dados GHE.....55
6.2.3 Caso de uso GHE 56
6.2.4 Descrevendo e exemplificando a utilização do algoritmo *Backtracking*...57
6.2.4.1 Variáveis de entrada do algoritmo *backtraking*.....58
6.2.4.2 Restrições estabelecidas para o algoritmo *backtraking*.....58
6.2.4.3 Funcionamento do algoritmo *backtraking* 61
6.2.5 Testes de implementação no protótipo desenvolvido 64
6.2.6 Resultados obtidos pelo GHE 64
6.2.7 Resultados obtidos pelo software Cronos.....67
6.3 Resultados obtidos.....69
6.3.1 Análise comparativa entre os dois algoritmos 69
6.3.1.1 Funcionalidade 70
6.3.1.2 Confiabilidade.....71
6.3.1.3 Usabilidade 71
6.3.1.4 Eficiência.....72
6.3.1.5 Manutenibilidade 73
6.3.1.6 Portabilidade.....74

7 CONCLUSÃO 76
REFERÊNCIAS 78

1 INTRODUÇÃO

A formulação de quadros de horários é uma tarefa necessária e inevitável no cotidiano de qualquer instituição de ensino. De maneira geral, o processo consiste em gerar uma tabela associando professores, turmas e disciplinas em determinados horários do dia com suas respectivas restrições. Para que um horário seja considerado viável, é preciso respeitar as restrições de disponibilidade dos professores e atender suas preferências.

À medida que o número de turmas aumenta e mais variáveis são levadas em consideração, o processo se transforma em uma incógnita, ou até mesmo em um problema difícil de ser resolvido manualmente, devido ao seu caráter combinatório. Essa dificuldade motiva a criação de ferramentas computacionais que permitam gerar automaticamente quadros de horários de maior qualidade, satisfazendo as restrições necessárias para cada professor, e podendo esta tarefa de forma mais rápida do que o processo manual.

Problemas como este descrito acima, e também que envolvem Inteligência Artificial (IA), podem ser vistos como problemas de satisfação de restrições, na qual o objetivo é descobrir se existe algum estado de problema que satisfaça a um determinado conjunto de restrições.

Segundo Cordenonsi, Aramburu e Almanca (2003), as implementações que utilizam restrições, tratam problemas combinatoriais com o objetivo de restringir o espaço de busca, diminuindo o gasto de memória e de tempo de processamento, conseguindo com isso maior eficiência. O uso do algoritmo de satisfação de restrições foi proposto para diminuir o espaço de busca na execução dos programas. A execução utiliza um mecanismo de controle que elimina os caminhos inválidos, isto é, caminhos que provavelmente não levem ao resultado final, deste modo, minimiza-se o processamento desnecessário e o retrocesso.

O algoritmo de busca pode ser compreendido como aquele que toma em consideração um problema, sendo este sua entrada, dando origem a uma solução, buscando sempre aquela que melhor atende as demandas estabelecidas. É aplicado sobre problemas complexos e cujo resolução não se dá pela aplicação de técnicas de programação convencionais, principalmente as de natureza puramente numérica (RUSSEL; NORVING, 2004).

O algoritmo de satisfação de restrições encampa um problema

representado por variáveis e pelas restrições que sobre elas incidem. Para Cordenonsi, Aramburu e Almanca (2003) alguns problemas da Inteligência Artificial (IA) podem ser compreendidos como problemas de satisfação de restrições, visando encontrar uma solução que atenda as restrições existentes.

Por algoritmos de *backtracking* destacam-se aqueles que atuam com tentativa e erro, também a chamada busca com retrocesso, decompondo os processos em tarefas e subtarefas limitadas que são exploradas de forma exaustiva (POTROS, 2015).

Os algoritmos genéticos, por sua vez, buscam uma solução adequada ao problema estabelecido, partindo de uma população inicial da qual são selecionados os melhores representantes para que uma nova população passe a existir. Com isso, a cada substituição surgem soluções mais adequadas (FILITTO, 2008).

1.1 OBJETIVO GERAL

Obter uma análise comparativa de desempenho entre os algoritmos de busca e satisfação de restrições *backtracking* e genético em um protótipo de horário escolar.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) compreender o funcionamento do processo de organização dos algoritmos de busca de satisfação de restrições *backtracking* e genético;
b) identificar as etapas no processo de organização e elaboração de um horário escolar;
c) desenvolver um protótipo com base no algoritmo *backtracking*;
d) analisar o desempenho do algoritmo *backtracking* e algoritmo genético;
e) esclarecer as métricas de comparação existentes, apontando a mais apropriada para o protótipo desenvolvido.

1.3 JUSTIFICATIVA

No início do ano letivo perde-se um tempo considerável para a formulação do quadro de horários escolares consistente, sem coincidências de horários entre as disciplinas, turmas e professores. Quando este processo é realizado manualmente, acarreta uma demora acentuada, bem como maior possibilidade de apresentação de problemas.

Assim sendo, pode-se destacar que um quadro de horários escolares consistente, sem coincidências de horários, permite que os professores possam ter mais confiabilidade e segurança para planejar e organizar suas atividades do cotidiano.

Uma forma de melhorar o desenvolvimento dos horários escolares é a aplicação dos algoritmos de busca de satisfação de restrições de backtracking, bem como os algoritmos genéticos. Os algoritmos genéticos buscam as melhores soluções, eliminando aquelas que não atendem as necessidades, enquanto os algoritmos backtracking buscam soluções viáveis com menor custo de processamento, fator que diferencia o tempo de execução entre eles.

Segundo Cordenonsi, Aramburu e Almanca (2003) faz-se necessária a utilização do algoritmo de busca de satisfação de restrições para diminuir o espaço de busca na execução, pois a execução utiliza mecanismos que eliminam caminhos inválidos, ou seja, caminhos que não levam a resultado nenhum. Deste modo minimiza o gasto de memória e de tempo de processamento, conseguindo com isso uma maior eficiência.

Leite, Carneiro e Carvalho (2006) destaca que o algoritmo genético é baseado nos mecanismos de seleção natural e da genética. Sua atuação se dá por meio da identificação de uma população de soluções e seleção das melhores, descartando aquelas de menor eficiência. A cada nova avaliação, soluções mais adequadas são geradas, porém, com uma demanda maior de tempo.

Atualmente, a formulação de horários escolares é muito complexa, considerando-se que existem restrições que não podem ser ignoradas, como a disponibilidade dos professores, o número de aulas por turma ou a repetição de aulas em um mesmo período (aulas geminadas) e, assim, é preciso que sejam encontradas as soluções mais adequadas para a instituição de ensino, para os professores e, principalmente para os alunos. Sob este prisma, surge a percepção

de que é necessário desenvolver uma alternativa para a redução do tempo de formulação de horários escolares construídos manualmente, apresentando-se uma opção computacional para este fim.

É preciso considerar que quanto mais rapidamente são encontradas as soluções para os problemas propostos, menores os custos associados a esse processo e, assim, é essencial que se destaque que os algoritmos de backtracking buscam as soluções que demandam dos menores custos de processamento, diferenciando o tempo de execução dos processos. Nesse sentido, realizar a comparação entre os algoritmos de backtracking e genéticos é realizada com o intuito de verificar as questões de organização, eficiência e desempenho de ambos e, assim, selecionar aqueles que melhor respondem às demandas dos usuários.

No presente trabalho procedeu-se da comparação dos algoritmos de backtracking e genéticos, com vistas a compreender as diferenças e semelhanças entre ambos, além de identificar aspectos sobre a eficiência, usabilidade, qualidade, tempo de execução, e, diante dessas informações, verificar qual o algoritmo que melhor atende às demandas de construção do horário escolar, em uma situação genérica.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em sete capítulos. No primeiro capítulo mostra-se uma breve descrição sobre o trabalho desenvolvido, permitindo tomar conhecimento do objetivo geral e dos objetivos específicos, como também saber qual é a justificativa para a realização deste trabalho.

O segundo capítulo aborda uma visão da importância em relação à geração do horário escolar organizado de forma automatizada, em substituição à organização manual. São apresentadas as restrições, problemas e complexidades existentes na geração de horários escolares adequados as demandas das instituições de ensino.

No terceiro capítulo são apresentadas as principais informações a respeito dos algoritmos, o algoritmo de busca, o algoritmo de satisfação de restrições, algoritmos de *backtracking*, algoritmos genéticos, de busca linear e binária.

O quarto capítulo encampa as métricas de comparação, sua conceituação e características, bem como medidas diretas, indiretas, os tipos de métricas e as escalas de medição, além de aspectos sobre a avaliação da qualidade do software segundo a norma ISO/IEC 9126.

No quinto capítulo são apresentados os trabalhos correlatos que possuem relacionamento com este trabalho, visando esclarecer a relevância do tema dentro dos estudos acadêmicos recentes.

No sexto capítulo é caracterizada a metodologia adotada para a realização do estudo e demonstrado o desenvolvimento do protótipo para geração de horário escolar, a metodologia aplicada no desenvolvimento do estudo e os resultados obtidos no trabalho.

O último capítulo deste trabalho destaca as conclusões de todo o desenvolvimento realizado, desde conhecimentos obtidos até as dificuldades encontradas no trabalho, bem como as propostas de trabalhos futuros a serem realizados.

Por fim, estão listadas todas as referências consultadas para que a base teórica deste trabalho pudesse ser construída de forma clara, coerente com os objetivos e bem organizada.

2 HORÁRIO ESCOLAR

Segundo Amaral (2013), formular ou criar um horário escolar pode ser considerado uma etapa da revisão do Projeto Político-Pedagógico (PPP). Afinal, antes de realizá-la, é preciso seguir a mesma sequência de procedimentos usados para a atualização do documento: a análise das matrículas e dos dados de aprendizagem e a avaliação dos recursos e da estrutura da escola. Esse é um bom começo, porém alguns imprevistos certamente aparecerão no meio do caminho.

Moura et al (2004) esclarecem que a organização de uma agenda escolar torna-se muito mais fácil com o auxílio de uma ferramenta computacional, já que sem ela o desenvolvimento deve ser feito manualmente, o que consome tempo, a disponibilidade de uma pessoa ou equipe para o desenvolvimento dos horários. Ainda assim, os riscos de erros ou de ocorrência de problemas torna-se muito mais acentuado, em função da necessidade de proceder do sistema de tentativa e erro.

> Ao longo dos últimos 40 anos, aproximadamente, a comunidade científica esforça-se para buscar uma solução computacional que auxilie em um antigo, longo e duro processo: a alocação de recursos sob restrições. Um dos problemas típicos desta natureza é o problema de alocação de horários escolares (*timetabling*). O casamento de interesses entre disponibilidades de salas, recursos audiovisuais, professores e alunos ao longo de determinados períodos da semana é tarefa árdua e consumidora de tempo. Contudo, é essencial e periódica em instituições de ensino, seja semestral ou anualmente, já que todas as atividades são pautadas sobre o período letivo. (TERRA; RADAELI, 2007, p. 95).

No intuito de melhor compreender o problema, é essencial esclarecer que dentro de uma escola encontram-se diferentes turmas, cada uma com suas disciplinas específicas e um número de aulas que varia conforme o currículo e o ano dessas turmas. Assim, as aulas precisam ser organizadas para preencher uma semana, em todos os horários de cada período. Multiplicando-se os horários do período e os dias da semana pelo número de turmas em uma escola, obtém-se o número total de aulas que serão ministradas em determinado período. Como cada professor tem seu número de aulas, e que muitos atendem mais de uma escola, a geração de horários deve considerar essas peculiaridades, de forma a organizar os horários escolares da melhor forma para a escola e para os professores que nela atuam (CISCON et al, 2005).

De modo semelhante, Lima Júnior e Correa (2010) afirmam que alocar professores e disciplinas de acordo com as grades horárias de cada escola, sem que

ocorram problemas diversos nessa organização trata-se de uma atividade de alta dificuldade, considerando-se que isso demanda de tempo e muito envolvimento com a agenda escolar para que a organização seja adequada a escola, aos alunos e aos professores ao mesmo tempo.

Ciscon et al (2005) destacam que na expressiva maioria das instituições escolares, quando do início de um novo período letivo, o desenvolvimento dos horários escolares é um problema que demanda de tempo e envolvimento da equipe para que seja realizado. Para os autores, a formação de horários escolares requer que sejam arranjados horários para que professores e alunos se encontrem em períodos pré-estabelecidos, visando satisfazer diferentes restrições existentes.

Para Xavier, Silva e Costa (2014), quando se trata da organização de horários em uma escola pequena, com poucas turmas e poucos professores, o problema pode não parecer tão relevante, porém, em instituições nas quais o número de turmas e professores é elevado, a organização dos horários escolares é muito difícil e laboriosa, sendo necessário analisar uma ampla gama de critérios e situações.

Torres-Ovalle *et al* (2014) destacam que o desenvolvimento de horários com a devida alocação de recursos trata-se de uma dificuldade muito presente na realidade de diversas e diferentes instituições, principalmente aquelas que precisam oferecer atividades específicas, com horários delimitados, disponibilizando um professor ou instrutor para sua realização.

Martins (2010) acredita que o desenvolvimento dos horários escolares não é uma atividade burocrática, conduzida por exigência e sem grande valia para a instituição, na verdade, trata-se de uma atividade prática que impacta diretamente sobre a organização e eficiência das aulas, permitindo que as características didáticas e pedagógicas das aulas sejam alcançadas com maior facilidade.

2.1 TIMETABLING PROBLEM - TTP

O Problema de Geração de Horários Escolares, conhecido na literatura como *Timetabling Problem (TTP)*, trata da definição dos horários para todas as aulas de uma escola, considerando um conjunto limitado de horários e satisfazendo diversas restrições em prol do professor e aluno. A solução manual do problema além de ser trabalhosa e lenta, pode ocasionar soluções de qualidade muito ruim.

Esforços no sentido de aperfeiçoar a elaboração de quadros de horário têm sido na maioria das vezes fracassada. Mesmo encontrar um quadro viável de horários é um problema difícil, dificultando o uso de técnicas exatas para instâncias de ordem mais elevada.

O TTP pode ser descrito como uma lista organizada, de modo geral desenvolvida em forma de tabela, que informações sobre os horários escolares em períodos, como horas e dias de aula, apresentando a disciplina e o professor que deverá ministrá-la em cada período (LIMA JÚNIOR; CORREA, 2010).

Pode-se definir TTP como a alocação (combinada ou não) de recursos necessários para a execução de um conjunto de tarefas, objetivando-se o melhor uso possível destes recursos. Em grande maioria, estes problemas, incluindo-se a geração de grade horária, utiliza como base o fator tempo para a execução individual de suas tarefas. Para a solução destas tarefas pode-se citar métodos como a programação matemática, métodos heurísticos ou técnicas da área de IA (ALENCAR, 2015).

Souza (2013) destaca que o TTP trata-se da necessidade de alocar aulas, professores e os demais recursos dos quais uma instituição dispõe, considerando-se que existem restrições a serem consideradas dentro dessa atividade, sempre com intuito de satisfazer ao máximo os objetivos que se deseja alcançar. Há um conjunto de quatro parâmetros que precisam ser considerados no TTP, T (*times*) sendo este o conjunto finito de horários, R (*resources*) sendo este o conjunto finito de recursos; M (*meetings*) sendo este o conjunto finito de encontros e C (*constrains*) sendo este o conjunto de restrições que precisam ser identificadas e consideradas de forma integral.

Na concepção de Santos (2008) que caracteriza o TTP como sendo um problema que se dá no intuito de determinar horários para o encontro letivo entre professores e classes, respeitando a disponibilidade dos professores, os horários das classes, um encontro por professor em cada horário. Trata-se de um problema de difícil resolução que costuma ter melhores resultados quando da aplicação de programas computadorizados.

Simão (2013) esclarece que o TTP trata-se da ação de alocar, diante de todas as restrições que possam surgir, os recursos da instituição dentro do espaço de tempo disponível para as atividades escolares, permitindo o maior alcance dos

objetivos didáticos. O TTP deve se basear em três elementos essenciais, quais sejam: os professores, as turmas e as aulas a serem ministradas.

Martins (2010) esclarecem que o TTP deve ser compreendido como uma situação que engloba diversas variantes, devendo associar todas elas da melhor forma possível, trazendo soluções para a necessidade de aulas, de acordo com a disponibilidade de recursos, como salas e professores, fato que demanda de uma análise pormenorizada de todos os fatores envolvidos, pois se algum deles for ignorado, torna-se impossível solucionar o problema de organização das aulas.

Xavier, Silva e Costa (2014) destacam o TTP como um problema matemático de otimização combinatória, cujo estudo é essencial para que se compreenda o quanto esse problema afeta o andamento adequado das atividades didáticas dentro da escola, sempre tendo em mente que inúmeras variáveis estão envolvidas, como turmas, professores, disciplinas, dias da semana, horários de aula, preferência dos professores, todas elas atuando diretamente sobre a dificuldade ou facilidade da organização dos horários dentro de cada instituição de ensino.

2.2 RESTRIÇÕES NO DESENVOLVIMENTO DOS HORÁRIOS ESCOLARES

Segundo Cordenonsi, Almaça e Aramburu (2003) o tipo de restrições que envolvem um problema varia bastante, dando origem a uma série se variantes do TTP. Essas restrições podem envolver as preferências pessoais, características desejadas no *layout* do quadro de horários, disponibilidade de recursos, distância entre as escolas que o professor irá lecionar, a existência de mais de um professor designado para uma mesma disciplina e diversas restrições que podem ter dependendo da instituição.

> O conjunto de restrições varia com as características da instituição e dos envolvidos no processo de desenvolvimento, geralmente os coordenadores. A montagem do horário é feita, na maioria dos casos, de forma manual e leva um considerável tempo na sua elaboração (FREITAS et al, 2014, p. 04).

É preciso recordar que diferentes instituições apresentam variantes peculiares do problema, de acordo com o tipo de aulas, de alunos e de professores que recebem. Estas variantes, e forma sumarizada, podem ser descritas como (CISCON et al, 2005):

a) *schooltimetabling*: nesse caso necessita-se de um sequenciamento semanal de aulas dentro de uma escola, de modo que não ocorra a definição de mais de uma aula ao mesmo tempo;
b) *coursetimetabling*: refere-se a um sequenciamento semanal de aulas dentro de uma universidade, com o intuito de evitar cursos com alunos em comum em períodos simultâneos;
c) *examinationtimetabling*: encampa o sequenciamento de exames de alguns cursos dentro de uma universidade, visando que não ocorram exames simultâneos de cursos que apresentam alunos em comum, de forma que esses exames sejam o mais espalhados possível.

> [...] a diferença entre essas três formas de classificação está na quantidade do conjunto finito de alguns parâmetros. O Timetabling Escolar possui quantidade menor de recursos porque uma turma já possui sala especificada e alunos normalmente predeterminados. O que torna sua complexidade reduzida em relação ao Timetabling Universitário, que possui um conjunto grande de restrições, pois disciplinas e turmas não possuem sala fixa e alunos de uma turma não são predeterminados, entre outras restrições. Já no Timetabling de Exames as restrições são mínimas e dizem respeito as avaliações de cada disciplina, em que dias e horários devem ocorrer ou não ocorrer, entre outras (SOUZA, 2013, p. 27).

A grande dificuldade de Kotsko (2003) em criar uma modelagem especifica para a escola é uma possível alteração de dados de professores, como por exemplo, a disponibilidade. Caso ocorra alguma alteração, as restrições relativas a essas modificações têm que ser rescrita dificultando o trabalho. Assim uma interface que gerem todas estas restrições facilita o trabalho de qualquer professor.

A alocação dos horários das disciplinas influencia fortemente a atividade de uma instituição de ensino durante um período letivo, sendo necessário que todos os envolvidos se adaptem à mesma. Uma grande diversidade de problemas associados a esse assunto já foram relatados e diferentes soluções foram propostas, sendo estas tipicamente dependentes das situações específicas de uma instituição. Assim, não há uma solução universal que possa ser adotada em todos os casos (HAMAWAKI, 2005).

Wren (1996) define o problema da geração de grade horária escolar como um conjunto de elementos dentro de padrões de tempo ou espaço, no qual um conjunto de objetivos deve ser totalmente ou quase totalmente atendido, satisfazendo um conjunto de restrições da forma mais completa possível. Fernandes

et al (2002) classifica as restrições para o problema da grade horária escolar em restrições fortes e fracas:

a) violação de restrições fortes (a alocação de um professor em duas salas diferentes em um mesmo horário): impedem a geração de uma solução válida;
b) violação de restrições fracas: apesar de não impedirem a geração de soluções válidas, podem afetar negativamente a qualidade da solução (e.g., a violação de regras de preferência dos professores por determinados horários pode gerar insatisfação de alguns professores).

De forma semelhante, Ciscon et al (2005) esclarecem que as restrições fortes garantem a viabilidade de uma solução, abrangendo os conflitos de simultaneidade de aulas para professores e turmas. As restrições fracas, por sua vez, não levam a inviabilidade da solução, porém, é preciso que ambas sejam avaliadas e consideradas no desenvolvimento do horário escolar.

> Restrições fortes, que obrigatoriamente devem ser satisfeitas: Essa definição é baseada no fato de quadros de horários somente serem considerados viáveis, caso respeitem todas as restrições deste tipo. Um exemplo comum de restrição forte é a necessidade de inexistência de conflitos entre os horários associados a determinado professor.
> Restrições fracas, que não são obrigatórias, mas desejáveis: Elas não interferem na factibilidade das soluções, contudo, são úteis como indicador da qualidade das mesmas. Um exemplo de restrição fraca é a quantidade de dias por semana em que são divididas as aulas de uma disciplina. É interessante, como requisito pedagógico, que aulas (definição 3.1) de uma disciplina não sejam ministradas em dias consecutivos. Contudo, uma solução que não possua essa característica continua sendo viável, apesar de apresentar uma baixa qualidade, segundo esse critério. (MARTINS, 2010, p. 34).

Sobre as restrições fortes e fracas, Simão (2013) destaca que restrições fortes, caso não sejam satisfeitas a solução proposta não será válida. No caso das restrições fracas, estas sendo satisfeitas, a solução se torna mais adequada, porém, caso isso não ocorra, a solução ainda poderá ser válida.

De modo a compreender se uma restrição é forte ou fraca, inviabilizando ou não uma solução, alguns pontos devem ser considerados, conforme as explicações de Ciscon et al (2005) a seguir:

No caso de **colisão por professor**, trata-se de uma restrição que leva em conta a definição de um professor para mais de uma aula ao mesmo tempo.

As **janelas** são restrições que abordam a existência de horários vagos em determinado dia.

A **restrição por aulas isoladas** encampa aulas que ocorrem isoladamente na grade.

O critério de **bloco de disciplinas** refere-se a condição didática das disciplinas, visando que estejam dispostas em bloco de duas.

O critério de **vários blocos por dia** abrange o fato de que dois blocos de uma disciplina podem ocorrer no mesmo dia.

A **média de aulas por dia** refere-se ao total de horários por dia e sua relação coma média de aulas, tomando como base a carga horária semanal de cada turma.

No caso da **preferência dos professores**, leva-se em conta o fato de que os professores talvez atuem em outras escolas e, assim, é preciso que seu horário em uma escola não impacte em outro lugar de atuação. O desenvolvimento de uma tabela com os horários de preferência do professor é muito importante.

2.3 PROBLEMAS

Ferreira e Glazar (2005) aduzem que o problema pode ser dividido como busca e otimização, sendo que a busca se refere ao fato de alcançar uma organização que satisfaça as restrições, enquanto a otimização visa produzir a melhor organização de horários, abrangendo até as restrições mais críticas e de difícil solução. Os autores destacam que é preciso evitar conflitos de horário, o excesso de aulas geminadas, aulas únicas, janelas nos horários dos professores, aulas geminadas quebradas pelo recreio, excesso de dias que o professor deve ir à escola, aulas não consecutivas de um professor em uma turma no mesmo dia. Tomando-se por base esses critérios, é possível obter uma organização que beneficia todos os envolvidos e respeita as demandas didáticas da formulação de horários em qualquer escola.

Alves (2010) afirma que os problemas de *timetabling* têm chamado atenção dos estudiosos ao longo dos anos, considerando-se que são de difícil solução, demandam de grande envolvimento e investimento, não apenas financeiro mas de tempo dos envolvidos, além de apresentarem uma importância prática muito relevante, já que atingem a todos envolvidos dentro da instituição escolar.

Simão (2013) esclarece que problemas de busca são aqueles cujas restrições são todas fortes, de modo que sem sua total satisfação não é possível encontrar uma solução válida.

Santos (2008) descreve os problemas de busca como aqueles que apresentam objetivo de obter uma solução viável, que atenda a todas as demandas e considere todas as variantes envolvidas.

Os problemas de otimização apresentam restrições leves, de modo que podem ser satisfeitos, ainda que nem todas as variáveis sejam completamente atendidas (SIMÃO, 2013).

Martins (2010) esclarece que problemas de otimização podem ser resolvidos por meio da quantificação e qualidade das soluções encontradas, comparando-se diferentes soluções até ser possível selecionar aquela que melhor atende às demandas existentes.

Os problemas de otimização são descritos por Santos (2008) como aqueles que têm por objetivo encontrar soluções para restrições rígidas, levando à soluções aceitáveis.

> [...] tanto no caso de busca como no caso de otimização, é feita a definição do problema base que, no caso da busca, é decidir se existe uma solução; e no caso da otimização é decidir se existe uma solução com um dado valor de função objetivo. Nos casos de busca são utilizados algoritmos de busca e nos casos de otimização são utilizadas heurísticas e algoritmos de otimização (SOUZA, 2013, p. 26).

É preciso que uma solução (*s*) seja avaliada com base na componente de inviabilidade (*g*(*s*)), que avalia o não atendimento a requisitos essenciais, as chamadas restrições fortes, além de uma componente de qualidade (*h*(*s*)), que avalia o não atendimento de requisitos não essenciais, as chamadas restrições fracas. Deve-se buscar sempre a minimização da função de custo de uma solução (*s*), utilizando-se a expressão: $f(s) = g(s) + h(s)$(CISCON et al, 2005).

2.4 COMPLEXIDADE

Os problemas que envolvem a organização de horários escolares apresentam diferentes níveis de complexidade, sendo essencial conhecê-los e compreendê-los para evitar que atrapalhem o desenvolvimento dos horários, levando à necessidade de revê-los futuramente, por não se adequarem à escola

(SIMÃO, 2013).

Quanto a sua complexidade, os problemas podem ser distribuídos como problemas passíveis de solução em tempo polinomial (*P*), problemas passíveis de verificação em tempo polinomial (*NP*), problemas que podem ser reduzidos em tempo polinomial, ainda que não estejam na classe *NP* (*NP difícil*), intersecção entre *NP* e *NP-difícil* (*NP completo*) (SIMÃO, 2013).

Para o desenvolvimento do presente trabalho, com foco na elaboração de horários escolares organizados, ágeis e eficientes, o problema relatado trata-se de problema NP completo, ou seja, interseção entre NP e NP-difícil.

3 ALGORITMO

Luger (2002) destaca que quando a busca que se pretende fazer necessita de uma precisão de alto grau, com base em dados incertos, mas com necessidade de soluções adequadas, a inteligência artificial torna-se uma das mais válidas ferramentas, dentro da qual os algoritmos atuam de forma muito relevante.

Algoritmos são sequências de ações executáveis com o intuito de encontrar a melhor solução para um problema, assim, compreender os algoritmos é essencial quando um problema é apresentado e cabe sua compreensão e busca por opções de solução.

3.1 ALGORITMO DE BUSCA

Segundo Gomes e Araújo (2010), o algoritmo de busca é um dos procedimentos mais utilizados na teoria de grafos. Ele tem por objetivo percorrer por toda a aresta e vértices de um grafo.

O algoritmo de busca toma um problema como entrada e, em retorno, oferece uma solução, sempre com base na resolução de mais de uma solução. É possível afirmar que uma das principais características do algoritmo é a aplicação sobre problemas de alta complexidade teórica, que não são resolvidos com técnicas de programação convencionais, principalmente as de natureza puramente numérica (RUSSEL; NORVING, 2004).

Pires (2006, p. 20) esclarece que “[...] algoritmos de busca local utilizam uma formulação de estados completos, ou seja, o estado inicial atribui um valor a cada variável e a função sucessor altera um valor de uma variável de cada vez”.

Segundo Russel e Norving (2004) o algoritmo de busca integra a Busca em Espaço de Estados, sendo importante esclarecer que o espaço de estados trata-se de um conjunto de estados acessíveis a partir do estado no qual se inicia a busca.

Medina (2015, p. 06) destaca que:

> Algoritmos de buscas funcionam da seguinte maneira: recebe um problema, faz uma busca em todo o espaço de busca, acha as melhores saídas, ou seja relaciona as ações mais eficientes para a resolução do problema, elimina as ações que não satisfazem, retorna a solução através de uma sequência de atos a serem executados.

Consequentemente, quanto maior é a complexidade de um problema, maior vai ser o tamanho do seu espaço de busca correspondente. Por isto, a utilização do mesmo algoritmo deve ser de forma cautelosa, pois dependendo da formulação, ou do seu desenvolvimento erroneamente, pode acarretar em grande perda de tempo na execução (GOMES; ARAÚJO, 2010).

Na sequência apresenta-se a figura 1, o algoritmo de busca para a verificação da utilização da carga horária de uma turma escolar de forma completa.

Figura 1 - Algoritmo de busca que verifica se a turma já completou a carga horária semanal

```
select isnull(sum(total_horas),0)
  into w_acuhoras
  from acumula_semana;
//Se completou a carga horaria semanal termina a geracao do horario
if w_acuhoras >= 1125 then // total de horas da semana
        leave lb_semana;
end if;
```

Fonte: Do autor (2015).

Compreende-se, assim, que um algoritmo permite uma busca dentro de um banco de dados, de forma a encontrar a melhor resposta para o problema que se apresenta. No caso dos horários escolares, o problema é alocar cada um dos professores dentro da carga horária necessária, sem associá-lo a duas turmas no mesmo horário, considerando todas as restrições envolvidas, como aulas geminadas, dias da semana, disponibilidade de cada professor, entre outros (CARDOSO; MARCELINO, 2015).

3.2 ALGORITMO DE SATISFAÇÃO DE RESTRIÇÕES

Este tipo de algoritmo utiliza uma técnica onde o problema é representado por um conjunto de variáveis a serem instanciadas, e um conjunto de restrições sobre estas variáveis. Na maior parte dos casos, a sequência de passos até atingir o estado final é irrelevante. O que se quer identificar é um valor para cada variável, de tal maneira que todas as restrições sejam respeitadas.

Cordenonsi; Aramburu e Almanca (2003) esclarecem que muitos problemas dentro da Inteligência Artificial (IA) podem ser descritos como problemas

de satisfação de restrições, para os quais existe o objetivo de encontrar uma solução que atenda as demandas de um conjunto de restrições. Os termos restrições (*constrains*) e propagação de restrições (*constraintpropagation*) são muito utilizados na Inteligência Artificial fazendo referência a diferentes tipos de técnicas.

De forma geral as restrições baseiam-se em problemas de combinação, por meio dos quais há uma restrição no espaço de busca, de forma a minimizar o gasto de memória e o tempo de processamento obtendo maior eficiência nos resultados obtidos. Com as restrições os caminhos que não produzem uma solução válida são eliminados, de modo que processamentos desnecessários não são realizados (CORDENONSI; ARAMBURU; ALMANCA, 2003).

Segundo Russell e Norvig (2004), um Problema de Satisfação de Restrição (PSR) é designado por um conjunto de variáveis e restrições, onde cada variável possui um domínio não vazio de valores possíveis. Cada restrição envolve um subconjunto das variáveis, especificando as combinações de valores permitidas para aquele subconjunto. Uma atribuição que não viola nenhuma restrição é considerada consistente ou válida, além disso, uma atribuição completa é aquela em que toda variável é mencionada. Analisando todo este contexto, a solução para um PSR é uma atribuição completa que satisfaça todas as restrições.

Segundo Machado et al (2015) o PSR pode receber uma formulação diferenciada como um problema de busca padrão da seguinte forma:

a) estado inicial: inicialmente a atribuição fica completamente vazia;
b) função sucessor: os valores inseridos nesta etapa, não podem ser conflitantes com os valores já inseridos anteriormente;
c) teste de objetivo: a atribuição corrente é completa;
d) custo de caminho: custo constante para todo o passo.

Diferentes linguagens em lógica com restrições foram criadas ao longo dos anos, visando aplicar problemas de combinação de forma eficiente e organizada. Este fator permite que sua aplicação na construção de horários escolares seja clara e eficiente. Para que um algoritmo seja efetivo na resolução dos problemas de horários escolares, ele deve apresentar uma função heurística, baseada em um horário comparado com todos os horários criados, assim como a apresentação de um conjunto de restrições definidas de forma específica pelos professores dentro de um quadro de horários (CORDENONSI; ARAMBURU; ALMANCA, 2003).

Pires (2006) esclarece que dentro dos algoritmos de satisfação de restrições a agregação da heurística permite que os conflitos sejam minimizados levando em conta as variáveis envolvidas com o processo de forma geral e as restrições que precisam ser atendidas para que a solução de problemas seja a mais apropriada possível.

A satisfação de restrições lida com os problemas definidos sobre domínios de valores discretos, do qual fazem parte os domínios finitos, sendo atualmente a mais usada na maioria das aplicações. A resolução de restrições possui todas as propriedades base da Programação com restrições (PR).

Machado et al (2015, p. 02) destacam que:

> Os algoritmos aplicados na resolução destes problemas permitem modelá-los por meio de estados que obedecem a uma representação padrão, estruturada e muito simples, tornando mais fácil resolver problemas extensos. Tais algoritmos auxiliam no desenvolvimento de ferramentas que manipulam uma série de restrições, com o objetivo de chegar a um determinado resultado, respeitando cada condição imposta.

No entanto, neste caso as restrições são definidas (na sua maior parte), sobre domínios infinitos ou mais complexos. Em vez dos métodos combinatórios para a satisfação de restrições, os algoritmos de resolução de restrições são baseados em técnicas matemáticas tais como a diferenciação automática, séries de Taylor, método de Newton, ou a programação linear.

A figura 2 demonstra a aplicação do algoritmo de satisfação de restrições com o intuito de resolver um problema específico, no caso, a construção de horários escolares.

Figura 2 - Algoritmo de Satisfação de Restrições implementado para o problema analisado

```
if w_professor is null then
   leave lb_horas;
end if;
insert into grade values (a_turma,w_dia,w_aula,w_idDisciplina,w_professor,w_turno);
//Se nao existir horas na disciplina na semana inseri , senao cai na questao de atualizar +45 min.
if not exists(select first 1
                                                    from acumula_semana
                                                where disciplina = w_idDisciplina) then
            insert into acumula_semana values (w_idDisciplina,45);
else
            update acumula_semana
               set total_horas = total_horas + 45
             where disciplina = w_idDisciplina;

end if ;
set w_aula = w_aula + 1;
if w_aula > 5 then
            leave lb_horas;
end if;
select isnull(sum(total_horas),0) into w_acuhoras
  from acumula_semana
    where disciplina = w_idDisciplina;

if w_acuhoras >= w_carga_horaria then
            leave lb_horas;
end if;
```

Fonte: Do autor (2015).

Segundo Russell e Norvig (2013) um PSR consiste em três elementos, X, D e C:

a) X - É um conjunto de variáveis, {X¹,….,Xn};

b) D - É um conjunto de domínios, {D¹,....,Dn} um para cada variável;

c) C – É um conjunto de restrições que especificam combinações de valores possíveis.

Os algoritmos de busca PSR aproveitam a estrutura de estados e utilizam heurísticas gerais em vez de heurísticas específicas de problemas para permitir a solução de problemas complexos. A ideia principal é eliminar grande parte do espaço de busca de uma só vez através da identificação de combinações de variável/valor que violam as restrições (RUSSELL; NORVIG, 2013).

> Um problema de satisfação de restrição possui esta definição por possuir um conjunto X= {x1, x2... xn} de variáveis, cada uma com seu domínio D= {d1, d2... dn} de valores possíveis e um conjunto R= {r1, r2... rn} de restrições que auxiliam nos valores que cada variável pode assumir. (MACHADO et al, 2015, p. 02).

3.2.1 Busca binária e linear

A busca binária demanda de elementos do vetor ordenados de forma crescente, de modo que não podem existir elementos duplicados no vetor. O usuário define as posições para que este algoritmo trabalhe no intuito de encontrar o elemento desejado. Esse algoritmo verifica se o elemento que se busca é maior ou

menor do que o elemento sugerido pelo usuário, sendo que os elementos do vetor abaixo ou acima da posição sugerida são desprezados.

Na sequência, apresenta-se a figura 3 que demonstra a estrutura de uma busca binária.

Figura 3 - Estrutura de uma busca binária

```
        writeln('Entre com o elemento a ser encontrado no vetor !');
        readln(e);
        i := 1;
        r := n;
        found := 1;

        while (i<= n) and (found = 1) do
        begin
          writeln;
          writeln('Entre com uma posicao entre ',i,' e ',r);
          readln(pos);
         if (pos<i ) or (pos> r) then
           found := 1
         else
          begin
          if(mat[pos] = e) or (i = r) then
          begin
            writeln;
            writeln('Elemento ',e,' encontrado na posicao ',pos);
            found:=0;
          end
          else
          begin
          if mat[pos] < e then
            i :=pos + 1
          else
            r := pos - 1
          end
          end;
        end;
end
```

Fonte: Do autor (2015).

A busca linear, por sua vez, faz a busca no vetor com o intuito de encontrar todas as ocorrências envolvidas com o elemento que se deseja obter, permitindo elementos duplicados no vetor. Mostra a posição na qual o elemento foi encontrado dentro do vetor. A variável C é um contador de verificação de possíveis ocorrências.

Figura 4 - Estrutura de uma busca linear

```
writeln('Entre com o elemento a ser encontrado no vetor !');
readln(e);
i := 1;
while (i<= n) do
begin
  if(mat[i] = e) then
   begin
     writeln('Elemento ',e,' encontrado na posicao ',i);
     c := c + 1;
   end;
   i :=i +1;
end;
```

Fonte: Do autor (2015).

Compreende-se, assim, que a busca binária permite a definição pelo usuário das posições de trabalho na busca da solução para o problema estabelecido.

3.2.2 Algoritmos *backtracking*

Marinho (2015) destaca que algoritmos *backtracking* são aqueles que seguem o padrão de busca em profundidade e integram um projeto de ampla utilização atualmente, quando a busca a ser realizada apresenta problemas complexos de busca combinatória. Em casos nos quais os problemas a serem solucionados devem ser analisados dentro de um conjunto de possibilidades, para as quais algumas restrições precisam ser aplicadas, os algoritmos de *backtracking* são de grande valia.

Os algoritmos de *backtracking* são também chamados de tentativa e erro ou busca com retrocesso, atuam por meio da decomposição dos processos em tarefas e subtarefas de número finito, exploradas exaustivamente (POTROS, 2015).

Russel e Norving (2013) afirmam que esses algoritmos de busca atuam de forma recursiva e em profundidade, atribuindo valores a determinado nodo da árvore de busca. Quando houver inconsistência na atribuição de valor, em função de uma restrição, a busca será aprofundada pela esquerda, retrocedendo e buscando o ramo mais próximo à direita. Ainda que não haja mais ramos, a pesquisa retornará a um nível da árvore para continuar a busca até que a solução possa ser apresentada.

Os autores afirmam que a expressão *backtracking* ou busca com retrocesso encampa o fato de que é realizada uma busca em profundidade, escolhendo valores para a variável apresentada, sendo que esses valores se alteram a cada retrocesso. A atribuição corrente é estendida com o intuito de gerar um sucesso, não realizar uma cópia do mesmo (RUSSEL; NORVING, 2013).

Caldas (2004) afirma que o *backtracking* trata-se de uma estratégia que permite o exame sistemático de uma lista, com o intuito de encontrar possíveis soluções. Elimina a verificação explícita de parte expressiva dos candidatos, desde que o problema respeite as restrições que maximizam ou minimizam a função de otimização.

Os seguintes passos são respeitados quando do uso desse tipo de algoritmo: definição de espaço de solução para o problema, espaço de solução com, pelo menos, uma solução ótima para o problema proposto; organização do espaço de solução para facilitar a pesquisa, em geral com a organização em árvore; e o processo de busca em profundidade (CALDAS, 2004).

Trata-se de uma estratégia aplicável a problemas de solução definida a partir de uma sequência de inúmeras decisões (*n*), modeladas por uma árvore representando todas as possíveis sequências de decisão.

> De fato, se existir mais de uma disponível para cada uma das *n* decisões, a busca exaustiva da árvore é exponencial. A eficiência dessa estratégia depende da possibilidade de limitar a busca, ou seja, podar a árvore, eliminando as sub-árvores que não levam a nenhuma solução. (CALDAS, 2004, p. 04).

Robaina et al (2014) esclarecem que o algoritmo de *backtracking* utiliza como estratégia a busca por soluções adicionando um a um os componentes necessários para que o problema seja resolvido, bem como avaliar parcialmente as soluções construídas. Essa avaliação parcial refere-se à possível violação das restrições apresentadas pelo problema.

> Caso não haja opção viável para continuar a resolução do problema o algoritmo recua/retrocede (backtrack) para trocar o último componente. O entendimento da estratégia proposta em algoritmos de *backtracking* pode ser melhor visualizada quando se associa o seu processamento à construção de uma árvore com as escolhas feitas e, no pior caso, o algoritmo *backtracking* terá que gerar todos os candidatos possíveis (ramos). Por outro lado, espera-se que o algoritmo *backtracking* seja capaz de fazer escolhas evitando o cálculo de todas as soluções possíveis. Estas escolhas são chamadas de podas.(ROBAINA et al, 2014, p. 06).

Esses algoritmos são usados com facilidade na implementação de um problema complexo, além do fato de que as linguagens da área de programação de lógica costumam apresentar mecanismos que dão suporte a esse tipo de algoritmos. Por outro lado, esses algoritmos, sem a programação de restrições, executam uma busca exaustiva e tendem a apresentar uma explosão combinatória. Necessitam de maior memória Stack, de acordo com o problema que se apresenta (MARINHO, 2015).

3.3 ALGORITMOS GENÉTICOS

Lucas (2002) esclarece que os Algoritmos Genéticos (AG) baseiam-se no princípio da seleção natural para abordar os problemas propostos de forma ampla, buscando sua otimização. Os algoritmos genéticos são robustos, genéricos e se adaptam facilmente a diferentes situações, de modo que podem ser aplicados em inúmeras áreas.

> Algoritmos Genéticos buscam a melhor solução para os problemas de otimização, utilizado um processo iterativo de busca da melhor solução para o seu problema. Onde a busca se dá a partir de uma população inicial, que combinando os melhores representantes desta população, obtém uma nova, que passa a substituir à anterior. A cada nova iteração é gerada uma nova população que apresenta novas e melhores soluções para o problema em questão, culminando com a sua convergência. (FILITTO, 2008, p. 137).

De forma simplificada, pode-se dizer que os algoritmos genéticos criam uma população de respostas possíveis em cada problema apresentado (inicialização), para então submeter essas respostas ao processo de evolução que encampa a avaliação, seleção, cruzamento, mutação, atualização e finalização (LUCAS, 2002).

Goldberg (1989) destaca que os algoritmos genéticos permitem a otimização das buscas com base em mecanismos de evolução semelhantes aos dos

seres vivos, visando sempre a sobrevivência do mais apto para a função ao qual será aplicado.

> Definindo de outra maneira, pode-se dizer que algoritmos genéticos são algoritmos de busca baseados nos mecanismos de seleção natural e genética. Eles combinam a sobrevivência entre os melhores indivíduos com uma forma estruturada de troca de informação genética entre dois indivíduos para formar uma estrutura heurística de busca (PINHO; MONTEVECHI; MARTINS, 2007, p. 320).

Maciel (2004) destaca que os AG permitem uma busca de todas as combinações, de modo a encontrar a combinação que representa a melhor solução para o problema que se estabelece, por meio de processos naturais de sobrevivência e reprodução das populações.

Hamawaki (2005) destaca que o AG precisa conter uma representação genética para as potenciais soluções, uma população inicial cuja geração se dá de forma aleatória, uma função de avaliação que opera como a pressão ambiental, classificando as soluções de acordo com sua adaptação ao meio, os operadores genéticos, os valores para os parâmetros genéticos.

Luger (2002) destaca que os algoritmos genéticos são de grande valor dentro da inteligência artificial, tendo em mente que permitem uma busca por pontos de aptidão elevada.

Leite, Carneiro e Carvalho (2006) esclarecem que os AG são algoritmos de busca inspirados na Teoria da Seleção Natural, atuando sobre a população de indivíduos com base em suas características genéticas, com maiores chances de sobrevivência, além de produzirem indivíduos cada vez mais aptos, levando ao desaparecimento dos indivíduos menos aptos.

Algoritmos genéticos procuram soluções diante de problemas de otimização, do mesmo modo que ocorre no processo de evolução natural, sempre com base em uma população inicial. Por meio do processo de evolução dessa população, cria-se uma nova população que pode oferecer as soluções mais adequadas ao problema. Este processo consiste na seleção dos indivíduos mais aptos para a correta solução do problema por meio do cruzamento dos indivíduos “mais aptos para resolver o problema, efetuamos o cruzamento destes indivíduos para gerar a nova população e aplicamos a mutação na mesma. Efetuamos este processo até que se obtenha a solução desejada”. (FILITTO, 2008, p. 141).

Na avaliação são avaliadas as aptidões das soluções (indivíduos da população) para ver de que forma respondem ao problema proposto.

“Um elemento essencial de qualquer algoritmo genético é a existência de uma função de avaliação, que permita a determinação do valor de adaptabilidade de cada organismo da população”. (MOORI; KIMURA; ASAKURA, 2010, p. 178).

Na seleção os indivíduos são selecionados para sua reprodução. A probabilidade de seleção de uma seleção é proporcional à sua aptidão frente ao problema (LUCAS, 2002).

Na seleção cada indivíduo ocupa uma posição, de acordo com sua aptidão, sendo que os indivíduos de alta aptidão ocuparão as posições mais altas, quando os indivíduos de baixa aptidão ocuparão as posições mais baixas.

> Com isto os indivíduos que possuem uma alta aptidão ocuparão uma porção maior do que os indivíduos que possuem uma aptidão menor. Esta roleta é girada várias vezes, onde a quantidade de giros varia conforme o tamanho da população. Em cada giro da roleta é selecionado um indivíduo que participará do processo de geração da nova população (FILITTO, 2008, p. 139).

No cruzamento ocorre o cruzamento das características das soluções escolhidas, de modo que novos indivíduos sejam gerados (LUCAS, 2002).

O cruzamento visa combinar os cromossomos dos pais, gerando os cromossomos dos filhos. Diferentes são os operadores de cruzamento, alguns mais genéticos e outros mais adequados a um tipo de codificação de cromossomos (FILITTO, 2008).

Na mutação as características dos indivíduos obtidos pela reprodução são alteradas, aumentando-se a variedade da população. Filitto (2008, p. 141) destaca que o operador de mutação insere pequenas mudanças aleatórias nos cromossomos filhos. Os operadores de cruzamento apresentam vários tipos de operadores de mutação.

Para Goldberg (1989) a mutação permite uma renovação dos resultados, considerando-se que são obtidos indivíduos com características novas, alteradas e peculiares.

Na atualização os indivíduos criados são inseridos na população. Quando se realiza a atualização da população antiga, ocorre sua substituição por uma nova população, que decorre do cruzamento de indivíduos selecionados da população anterior. De forma mais comum, cita-se as atualizações (x+y) e (x,y), conhecidas

também como estratégia soma e estratégia vírgula. A estratégia soma, também chamada de elitismo, se dá quando indivíduos da população anterior convivem com a nova população, formada por seus filhos. De forma geral apenas um pequena percentagem é selecionada para a próxima geração, já que há risco de convergência prematura do algoritmo genético. Na estratégia vírgula, por sua vez, a população anterior não convive com a próxima, perdendo-se algumas boas soluções encontradas (ROSA; LUZ, 2009, p. 07).

Na finalização são verificadas as condições de encerramento da evolução, de modo a compreender se foram atingidas. Caso isso não tenha ocorrido, retorna-se à etapa de avaliação, caso tenha ocorrido, a etapa de execução é encerrada (LUCAS, 2002).

> O operador de finalização é o responsável por determinar se a execução do Algoritmo Genético (evolução de população) será concluída ou não. Tal ação é realizada a partir da execução de testes baseados em uma condição de parada pré-estabelecida. Tal condição de parada pode variar desde a quantidade gerações desenvolvidas até o grau de proximidade dos valores de aptidão de cada cromossomo, de determinada população. (ROSA; LUZ, 2009, p. 07).

As principais características dos algoritmos genéticos são: a busca codificada, considerando-se que não operam sob o domínio do problema, mas com base em representações de seus elementos, impondo uma restrição ao seu uso, já que para a solução de um problema deve haver um conjunto de soluções viáveis, permitindo sua codificação em uma população de indivíduos (LUCAS, 2002).

Cita-se, também, a generalidade, já que esses algoritmos simulam a natureza no que se refere à sua adaptabilidade. Nesse sentido, o programador não precisa se preocupar com a forma de chegar a uma solução, mas como ela deve se parecer (LUCAS, 2002).

Outra característica dos algoritmos genéticos é o paralelismo explícito, que pode ser verificado levando-se em conta que cada indivíduo da população existe isoladamente e pode ser avaliado de forma independente (LUCAS, 2002).

Quanto ao funcionamento dos algoritmos genéticos, Rosa e Luz (2009, p. 02) esclarecem que:

> Inicialmente é gerada uma população formada por um conjunto aleatório de indivíduos, que podem ser vistos como possíveis soluções do problema.
> Durante o processo evolutivo, esta população é avaliada, sendo que para cada indivíduo é atribuída uma nota, ou índice, que reflete sua habilidade de adaptação a determinado ambiente.

> Uma porcentagem dos indivíduos mais adaptados é mantida, enquanto os outros são descartados.
> Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais por meio de cruzamentos (crossover), mutações ou recombinação genética gerando descendentes para a próxima geração.
> Este processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada. Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativos poderosos e robustos.

De acordo com Moori, Kimura e Masakura (2010, p. 177) os elementos mais importantes do AG são organismos e populações. Além disso, os autores afirmam que:

> Dado um problema específico, um algoritmo genético tenta encontrar sua solução através da criação e manipulação de um conjunto de estruturas (organismos), que representam soluções em potencial do problema. Este conjunto de organismos recebe o nome de população e, no AG padrão, cada organismo é representado como um cromossomo simples. O algoritmo genético representa cada possível solução “x” no espaço de busca como uma seqüência de símbolos “s” gerados a partir de um dado alfabeto finito A. No caso mais simples, usa-se o alfabeto binário A = {0,1}; no caso geral, tanto o método de representação quanto o alfabeto genético dependem de cada problema.

Lobo (2005) apresenta as principais definições relacionadas aos algoritmos genéticos. O cromossomo ou genótipo trata-se de uma cadeia de caracteres que representam uma informação ligada as variáveis do problema, sendo que cada cromossomo representa uma solução para o problema (LOBO, 2005).

O gen ou gene é a unidade básica do cromossomo, enfatizando-se que um cromossomo tem um número de gens que descrevem uma variável do problema. Esses gens podem ser binários, inteiros ou reais (LOBO, 2005).

A população trata-se do conjunto de cromossomos ou soluções. O fenótipo, por sua vez, é o cromossomo decodificado, enquanto a geração pode ser compreendida como o número da iteração que o AG executa para gerar uma nova população (LOBO, 2005).

As operações genéticas são operações que o algoritmo genético realiza sobre os cromossomos. Por seu turno, os espaços de busca ou regiões viáveis são o conjunto, espaço ou região que encampam as soluções possíveis ou viáveis para o problema que deverá ser otimizado (LOBO, 2005).

A função objetivo ou de aptidão é construída por parâmetros envolvidos no problema e permite o cálculo da aptidão de cada indivíduo, fornecendo o valor para aplicação no cálculo e probabilidades. A aptidão bruta é a saída gerada pela

função de aptidão para um indivíduo da população, enquanto a aptidão máxima refere-se ao melhor indivíduo da população (LOBO, 2005).

Após proceder da análise e seleção do algoritmo mais adequado para cada finalidade, é preciso conhecer e selecionar a métrica de comparação a ser aplicada, considerando-se que sempre que se procede de comparações entre os algoritmos, torna-se possível verificar quais apresentam os resultados que melhor atendem as demandas do trabalho desenvolvido e, assim, suas chances de sucesso tornam-se maiores.

4 MÉTRICAS DE COMPARAÇÃO

Ao longo dos anos, a indústria de softwares vem identificando que a qualidade e confiabilidade do produto desenvolvido são essenciais para sua aceitação no mercado, reduzindo-se os custos associados a ele e aumentando a satisfação dos usuários, com agilidade e eficiência dos processos por eles conduzidos (MEIRELLES, 2008).

Calazans e Alvarenga (2014) destacam que alguns ou vários processos sempre precisam ser avaliados quando se desenvolve um software, visando identificar possíveis falhas, desenvolver novas possibilidades e permitir uma utilização ainda melhor no futuro. Para isso, as medições e métricas são indispensáveis, pois permitem uma análise quantitativa do software desenhado.

Segundo Meirelles (2008) as métricas de avaliação têm como objetivo identificar e medir os principais parâmetros que afetam os softwares desenvolvidos, já que quando são criados de forma falha os resultados esperados para sua utilização não serão alcançados ou isso ocorrerá de forma parcial, o que não é o objetivo de nenhum produto. Todo software apresenta parâmetros essenciais e, assim, a métrica de comparação permite que se proceda da identificação, medição e controle desses parâmetros de forma muito mais efetiva.

Alguns critérios devem ser considerados para que se obtenha uma métrica ágil e adequada ao uso para o qual é proposta, quais sejam: reforço de princípios ágeis com foco em toda a equipe e com base em tendências, não em números; integrar um conjunto restrito de métricas e diagnósticos; possibilidade de coleta facilitada, esclarecendo fatos e fatores que interferem na manipulação e melhoria dos processos (CALAZANS; ALVARENGA, 2014).

Zadra, Carvalho e Cardoso (2015) esclarecem que métricas de software são medidas quantitativas de qualidade de seus atributos, de modo que ainda antes de sua criação seja possível analisar a qualidade que apresentará. Por meio das métricas são obtidos dados que permitem criar requisitos eficazes e modelos de projeto, códigos sólidos e testes completos. Insta destacar que as métricas permitem uma avaliação completa e detalhada do projeto de software que se pretende desenvolver e, assim, torna-se essencial em um cenário no qual cada vez mais os recursos de informática são tão utilizados.

Calazans e Alvarenga (2014) esclarecem que a criação de um software se baseia em fatores diversos, como os processos e as pessoas envolvidas. Em um mercado dinâmico com demandas cada vez maiores por qualidade a preços acessíveis, esses programas precisam apresentar cada vez mais qualidade e atender cada vez mais rápido a demanda dos clientes para que sejam considerados adequados ao mercado atual.

A aplicação das métricas leva à percepção da eficácia do processo de software, bem como os projetos executados com base nesse processo e, assim, possibilitam a obtenção dos indicativos essenciais sobre o processo, levando à otimização dos custos de produção. Essa economia se dá em função do fato de que com as métricas, as chances de desenvolvimento de um software que não atende as demandas de uso são reduzidas ou eliminadas (ZADRA; CARVALHO; CARDOSO, 2015).

Mensurar um software é atividade que ocorre no processo de desenvolvimento do software, buscando identificar seus atributos e, diante desse conhecimento, melhorar suas características para que atinja um patamar de excelência e efetividade durante sua utilização (PRESSMAN, 2006).

Os principais objetivos das métricas podem ser destacados como o aumento da exatidão das previsões a respeito de custos e prazos dos projetos, redução de defeitos e demais problemas que podem surgir e comprometer o produto, dedução de custos e prazos relacionados ao projeto, aumento do retorno do investimento da organização em processos, tecnologia e treinamento, bem como identificar necessidade de novos investimentos no que se refere aos processos, tecnologia e treinamento (COSTA et al, 2015, p.02).

Ruck, Oliveira e Koslovski (2014) destacam que a compreensão do produto, bem como das necessidade de seu futuro usuário, permite que o desenvolvedor mantenha um foco muito mais específico durante seu trabalho, eliminando do programa todos os recursos que não serão necessários e, com isso, ganha tempo e reduz os gastos envolvidos em todas as etapas de criação do software.

> As atividades de desenvolvimento de software, quando enfocadas com objetivos profissionais, trazem à tona uma série de dificuldades que merecem particular atenção por parte do desenvolvedor. Dentre essas dificuldades estão aquelas relacionadas a estimativas de cronograma e custo de desenvolvimento. Entretanto, antes de se fazerem futuras

> estimativas, é preciso efetuar medições dos parâmetros obtidos no presente e verificar os parâmetros do passado. (MIRANDA, 2002, p. 37).

Compreende-se, assim, que a aplicação das métricas é essencial para que se desenvolva um software de qualidade avançada, com fácil utilização, que atenda às demandas de seus usuários e que possa ser melhorado de forma contínua (MIRANDA, 2002).

4.1 MEDIDAS DIRETAS

As medidas de software podem ser conduzidas de forma direta ou indireta. As medidas diretas encampam fatores relacionados aos custos de produção, o trabalho necessário para o desenvolvimento do software, seu tamanho em linhas de código, entre outros dados que precisam ser estimados ainda antes de sua produção, já que após o investimento para seu desenvolvimento e produção, encontrar falhas e lacunas, torna-se um processo dispendioso e que coloca em risco a credibilidade do desenvolvedor e outros trabalhos que venha a desenvolver (ZADRA; CARVALHO; CARDOSO, 2015).

As métricas de comparação consideradas as mais efetivas são aquelas que apresentam simplicidade, com significado claro, objetividade, fácil obtenção, com baixo custo de obtenção da métrica, são válidas, ou seja, medem efetivamente o que estão propostas a mediar, são robustas, pois não sofrem grandes desvios com pequenas mudanças no processo e no software, são lineares, já que permitem formas de mapeamento por meio da manipulação dos números obtidos (MEIRELLES, 2008).

4.2 MEDIDAS INDIRETAS

As medidas indiretas referem-se aos fatores que englobam a funcionalidade, qualidade e complexidade do software, mas não há uma maneira facilitada de medir essas características, assim, são medidas citadas como indiretas. Um produto perfeito logo em sua primeira versão é muito difícil de ser alcançado, porém, com a aplicação das medições adequadas, é possível desenvolver um software de qualidade, para o qual as variáveis mais relevantes tenham sido consideradas antes de ser distribuídos aos seus usuários. Os próprios profissionais

da área são beneficiados por meio das medições, já que passam a conhecer seu produto de forma ainda mais detalhada (ZADRA; CARVALHO; CARDOSO, 2015).

4.3 TIPOS DE MÉTRICAS

É essencial que se recorde que as métricas de comparação de software podem ser de diferentes tipos, de acordo com sua aplicação, com os resultados esperados, com as características do produto, entre tantos outros fatores. Nesse sentido, parte-se para a análise dos tipos de métrica de comparação que podem ser selecionados e utilizados na avaliação de um software.

4.3.1 Métricas orientadas ao tamanho

As métricas orientadas ao tamanho tomam como base a quantidade de linhas de códigos escritas durante o desenvolvimento do software e é possível representá-las por meio de uma tabela, armazenando os dados de um projeto para aplicação em novos projetos no futuro (ZADRA; CARVALHO; CARDOSO, 2015).

Essas métricas permitem medições mais detalhadas ao invés de considerar apenas o número de linhas de código escritas, como a quantidade de erros por KLOC, defeitos por KLOC, documentação por KLOC, entre outras. Apesar de sua relevância, essas métricas não apresentam aceitação universal, principalmente em função das diferenças entre a quantidade de linhas de código necessárias e diferentes linguagens de programação (ZADRA; CARVALHO; CARDOSO, 2015).

Monteiro (2005) esclarece que o tamanho de um software está ligado ao trabalho que será empregado em seu projeto, de forma que cada projeto poderá ser medido conforme seu tamanho.

4.3.2 Métricas orientadas a função

As métricas orientadas a função medem a quantidade de funções de um software por meio de dados históricos, seus custos e esforços do projeto de codificação. Os pontos de função são derivados de uma relação empírica apoiada em medidas calculáveis (diretas) sobre o domínio das informações do software e

avaliações qualitativas de sua complexidade (ZADRA; CARVALHO; CARDOSO, 2015).

Esses pontos são definidos por número de Entradas Externas (EE), Número de Saídas Externas (EOs), Número de Consultas Externas (EQs), Número de Arquivos Lógicos Internos (ILFs) e Número de Arquivos de Interface Externos (EIFs). Os dados são coletados, assim como as respostas de um questionário de complexidade, para a elaboração de uma fórmula base para o cálculo do ponto de função (FP) (ZADRA; CARVALHO; CARDOSO, 2015).

> A Análise de Pontos por Função (APF) consiste em uma importante técnica para a medição de projetos de software. Seu papel é estabelecer uma medida através dos Pontos de Função, observando as funcionalidades que o software possui através do ponto de vista do usuário. É importante destacar que a utilização do APF é totalmente independente de linguagens de programação e de tecnologias utilizadas em projetos. O objetivo dessa análise relaciona-se justamente com a medida de projetos desvinculando-os das tecnologias utilizadas. Outro ponto a ser destacado é a medição da funcionalidade requisitada pelo cliente, o que auxilia na tomada de decisões, relacionadas com custo, prazo, quantidade de recursos alocados, melhoria no cronograma e etc. (COSTA *et al*, 2015, p. 04).

Miranda (2002, p. 37) destaca que “essa métrica foi originalmente projetada para sistemas de aplicações comerciais e, por isso, necessitou evoluir devido ao rápido aumento da complexidade algorítmica das aplicações”.

4.3.3 Métricas de qualidade do produto

As métricas de produto são aquelas que medem a complexidade e o tamanho final de um programa, sua qualidade, corretude, eficiência, usabilidade, grau de reutilização, confiabilidade, facilidade de manutenção, entre outros (MEIRELLES, 2008).

Machado e Souza (2015) esclarecem que avaliar um produto de software integra os processos do ciclo de vida de desenvolvimento de software. Nesse sentido, a qualidade de um produto de software pode ser avaliada conforme seus atributos internos ou externos, ou ainda, pelos atributos de sua qualidade durante o uso, visando que ele alcance os efeitos desejados para a finalidade que foi projetado.

> Escalas de medidas para as métricas usadas em requisitos de qualidade podem ser divididas entre as categorias correspondentes para os diferentes graus de satisfação dos requisitos. Como exemplo, a escala poderia ser dividida em duas categorias: satisfatório e insatisfatório, ou em quatro

> categorias: excedeu os requisitos, atingiu, requisitos suficientemente aceitos e inaceitável. Basta apenas que as categorias sejam especificadas de forma que o usuário e o desenvolvedor possam evitar o excesso de custo e planejamento desnecessário. (MACHADO; SOUZA, 2015, p. 3).

A qualidade encampa as necessidades do usuário do software, pois se ele não atender essas necessidades essenciais, é pouco provável que será considerado como qualitativo.

4.3.4 Métricas de processo

As métricas de processo referem-se a todo o procedimento de concepção e desenvolvimento de um software, realizando a medida do processo de desenvolvimento, tipo de metodologia aplicada, tempo de desenvolvimento, entre outros (MEIRELLES, 2008).

As métricas de processo auxiliam na avaliação da qualidade dos produtos de software desenvolvido. As métricas de processo são de grande valia, considerando-se que permitem a avaliação das etapas desenvolvidas e, assim, compreender com maior facilidade os resultados finais que serão alcançados (TRODO, 2009).

Para Vieira (2015) as métricas de processo encampam desde a concepção até o desenvolvimento do software, analisando o processo de desenvolvimento de forma geral, com ênfase na metodologia usada, tempo de desenvolvimento, entre outros fatores.

4.3.5 Métricas objetivas e subjetivas

As métricas objetivas são obtidas por meio de regras específicas e clara, permitindo comparações posteriores eficientes e consistentes. Os valores obtidos pelas métricas objetivas deveriam ser sempre os mesmos, sendo indiferente o momento, as condições ou o indivíduo que determinará essas métricas. A obtenção dessas métricas pode ser automatizada, como ocorre nos casos de linhas de códigos. Em geral, as métricas objetivas tratam características do código-fonte (MEIRELLES, 2008).

Por sua vez, as métricas subjetivas partem de valores, todavia, dependem de um julgamento que também atua como dado de entrada para o levantamento que

será feito, como nos modelos de estimativas de custos que dependem da classificação do tipo de software (MEIRELLES, 2008).

4.3.6 Métricas primitivas e compostas

As métricas primitivas podem ser observadas de forma direta em uma medida única, como o número de linhas de código, erros indicados em um teste de unidade, ou o tempo total para o desenvolvimento de um projeto. No caso das métricas compostas, elas são combinações de uma ou mais medidas como o número de erros encontrados a cada mil linhas de código, ou número de linhas de teste por linha de código (MEIRELLES, 2008).

Para Vieira (2015), as métricas primitivas podem ser diretamente observadas em uma única medida, como número de linhas de código, erros indicados em um teste ou tempo total para o desenvolvimento do software. As métricas compostas, por sua vez, são combinações de uma ou mais medidas como o número de erros a cada mil linhas do código, ou o número de linha de teste por linha de código.

4.3.7 Métricas de Halstead

Por métricas de Halstead cita-se um conjunto de métricas baseadas na teoria da informação, uma das primeiras técnicas utilizadas como base teórica comum. São aplicáveis a diversos aspectos dos softwares, diferente do que ocorre com muitas outras, que abordam um aspecto particular, além de serem usadas para a avaliação do esforço global de desenvolvimento do software (MEIRELLES, 2008).

4.4 AVALIAÇÃO DA QUALIDADE SEGUNDO A NORMA ISO/IEC 2026

O termo “qualidade” é de origem latina, derivando-se da palavra qualita. Trata-se de um vocábulo genérico, com definição difícil. Mesmo dessa forma, é um dos termos mais disseminados nas sociedades, tendo em vista representar uma série de coisas que são relativamente diferentes (TOLEDO et al, 2013).

De forma geral, a palavra de qualidade refere-se às características ou traços que podem ter um objeto, uma pessoa ou situação. É um atributo que pode

ser inerente a quem ou algo que a possui ou pode ser adquirida ao longo do tempo (PALADINI, 2011).

Quando se trata de um produto, a qualidade assume características como zero defeito, conformidade, garantia de procedência, adequação ao uso, atendimento às especificações e satisfação dos usuários (ARROTEIA; ZUCCARI; TOMAZ, 2015).

Para padronizar o conceito de qualidade, no ano de 1987, foi criada em Genebra, na Suíça, a ISSO - *International Organization for Standardization* – ISO (no Brasil, chamada de Organização Internacional para a Padronização). A ISO criou o primeiro modelo em nível internacional de sistema ou gestão da qualidade voltadas às empresas e produtos (ARROTEIA; ZUCCARI; TOMAZ, 2015).

No ano de 1991, a ISO, em conjunto com a IEC - International Engineering Consortium, publicou um modelo qualidade para a avaliação de produtos de software (ISO 9126:1991) (MIGUEL; MAURICIO; RODRÍGUEZ, 2014).

A ISO/IEC 9126 propõe um conjunto subfuncionalidades de características para avaliar atributos de um produto de software em relação a sua qualidade (MIGUEL; MAURICIO; RODRÍGUEZ, 2014).

Essas entidades trabalham em parceria com outras organizações internacionais, governamentais e privadas, formando comitês técnicos, através do qual desenvolveram revisões da norma ISO/IEC 9126 desde sua criação (SURYN et al, 2003).

Uma das características da ISO/IEC 9126 é padronizar a terminologia em relação à qualidade do software (MIGUEL; MAURICIO; RODRÍGUEZ, 2014).

A ISO/IEC 9126 refere-se uma norma que é integrada por um conjunto de atributos ou características que devem ser verificadas em um software, de forma que ele seja considerado como um "software de qualidade" (SURYN et al, 2003).

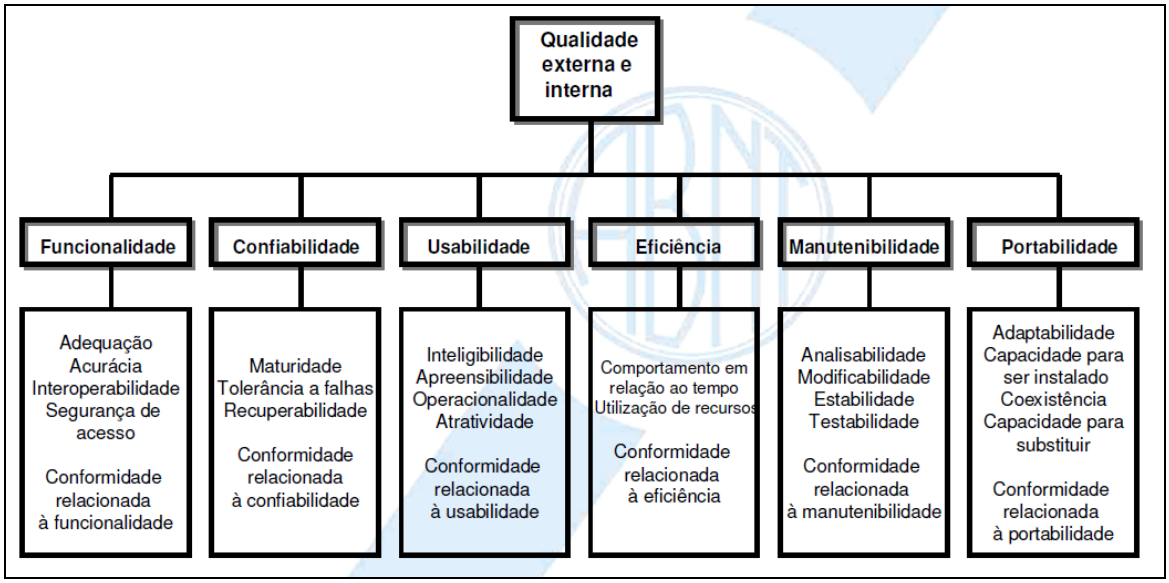
A ISO/IEC 9126 faz distinção entre qualidade interna e qualidade externa.

A qualidade interna de software é medida objetivamente por fatores mensuráveis durante o desenvolvimento. A qualidade externa, por sua vez, tem como objetivo medir a qualidade do software levando em consideração o comportamento do software em um sistema do qual ele faz parte (ABNT, 2003).

Na primeira parte do modelo, são especificadas seis características para avaliação da qualidade interna e externa: funcionalidade, confiabilidade, usabilidade,

eficiência, manutenibilidade (facilidade de manutenção) e portabilidade, conforme se verifica na Figura 5.

Figura 5 - Modelo de qualidade para qualidade externa e interna



Fonte: ABNT (2003)

Tais características ou categorias, por sua vez, desdobram-se em subcategorias, que durante o processo de avaliação suas respectivas perguntas-chave devem ser respondidas, conforme representado no Quadro 1.

Quadro 1 - Características da qualidade do software segundo a ISO/IEC 9126/1

Funcionalidade	**Adequação**: Propõe-se a fazer o que é apropriado? **Acurácia**: Gera resultados corretos ou conforme acordados? **Interoperabilidade**: É capaz de interagir com os sistemas especificados? **Segurança de acesso**: Evita acesso não autorizado, acidental ou deliberado a programas e dados? **Conformidade**: Está de acordo com normas e convenções previstas em leis e descrições similares?
Confiabilidade	**Maturidade**: Com que frequência apresenta falhas? **Tolerância a falhas**: Ocorrendo falhas como ele reage? **Recuperabilidade**: É capaz de recuperar dados após uma falha?
Usabilidade	**Inteligibilidade**: É fácil entender os conceitos utilizados? **Apreensibilidade**: É fácil aprender a usar? **Operacionalidade**: É capaz de operar e controlar a operação?
Eficiência	**Comportamento em relação ao tempo**: Qual é o tempo de resposta e de processamento? **Comportamento em relação aos recursos**: Quanto recurso utiliza?
Manutenibilidade	**Analisabilidade**: É fácil encontrar uma falha quando ocorre? **Modificabilidade**: É fácil modificar e remover defeitos? **Estabilidade**: Há grandes riscos de *bugs* quando se faz alterações? **Testabilidade**: É fácil testar quando se faz alterações?
Portabilidade	**Adaptabilidade**: É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta finalidade no software considerado? **Capacidade para ser instalado**: É fácil instalar em outros ambientes? **Capacidade para substituir**: É fácil substituir por outro software?

Fonte: Flôres (2014)

Estas métricas oferecem a possibilidade de medir a qualidade dos artefatos intermediários e de prever a qualidade do produto final. Isto permite que sejam identificados problemas de qualidade e se inicie a ação corretiva assim que possível no ciclo de vida do desenvolvimento (SURYN et al, 2003)..

4.5 ESCALAS DE MEDIÇÃO

As escalas de medição podem ser nominais, ordinais, de intervalo e racionais. As escalas nominais abrangem a existência de um nome ou valor para cada atributo, porém, a ordem desses valores não interfere em sua interpretação (MEIRELLES, 2008).

As escalas ordinais apresentam resultados em determinada ordem, ou seja, ascendente ou descendente, porém, com uma distância entre os pontos que não possui significado (MEIRELLES, 2008).

As escalas de intervalo preservam a importância da ordem dos resultados e apresentam informações sobre os tamanhos dos intervalos entre cada um dos pontos da escala, destacando-se que as relações entre valores não necessariamente válidas (MEIRELLES, 2008).

As escalas racionais se assemelham a escala de intervalo, todavia, também representam as proporções entre entidades e possuem um zero absoluto (MEIRELLES, 2008).

5 TRABALHO CORRELATOS

Atualmente existem inúmeras maneiras para geração de horário escolar, na qual faz da utilização de vários algoritmos e tecnologias diferentes. Abaixo, são descritos trabalhos e pesquisas relacionados ao presente trabalho propostos.

5.1 DEFINIÇÃO E IMPLEMENTAÇÃO DE UMA FUNÇÃO DE AVALIAÇÃO PARA UM SISTEMA DE GERAÇÃO DE GRADE HORÁRIA QUE UTILIZA COMO MÉTODO DE BUSCA ALGORITMO GENÉTICO

A monografia foi escrita por Rodrigo Meurer Melo, no curso de Ciência da Computação na Universidade do Extremo Sul Catarinense (UNESC) no ano de 2013.

Nesta monografia descreve o estudo elaborado acerca de um sistema para geração de grade horária para Instituições de ensino por meio de algoritmos genéticos. Os estudos desenvolvidos tiveram como ponto de partida o “Sistema para Geração de Grade Horária Para Instituições de Ensino” desenvolvido na Universidade do Planalto Catarinense (UNIPLAC) em 2001. O problema de geração de grade horária, ou timetabling consiste na alocação de recursos, combinados ou não, necessários à execução de uma série de tarefas tendo normalmente como base o fator tempo. As grades geradas pelo sistema da UNIPLAC apresentavam coincidência de horário e não atendiam completamente ao critério de disponibilidade dos professores em lecionar.

O mesmo ocorria devido a uma definição incompleta da função de avaliação, que corresponde em um algoritmo genético à função que atribui o grau de adaptação de cada cromossomo dentro de uma população. Cada cromossomo da população representa uma possível solução do problema e, no problema em questão, quanto menos restrições apresentar um cromossomo melhor será o seu grau de adaptação e conseqüentemente, maior as suas chances de ser a solução do problema. Para eliminação das coincidências de horários do sistema da UNIPLAC foi implementada uma função de avaliação mais refinada e uma função de teste que seleciona com precisão a melhor solução gerada pelo algoritmo. Os resultados obtidos corresponderam aos objetivos propostos, eliminando em 100% dos casos as

coincidências de horários e atendendo a todas as restrições impostas pelo problema. O nome do sistema desenvolvido foi chamado de URANO (MELO, 2003).

5.2 GERAÇÃO DA GRADE HORÁRIA DO CURSO DE ENGENHARIA DE PRODUÇÃO DA UFPR ATRAVÉS DE PROGRAMAÇÃO LINEAR BINÁRIA

Este artigo foi escrito por Pedro Rochavetz de Lara Andrade, Cassius Tadeu Scarpin e Maria Teresinha Arns Steiner no curso de Engenharia de Produção da Universidade Federal do Paraná no ano de 2012.

Neste artigo os autores apresentam os resultados de uma aplicação sobre um modelo matemático de Programação Linear Binária para a geração da grade escolar. Como um auxílio, utiliza o software LINGO 12.0 para a solução do modelo matemático, e posterior montagem da grade horária.

Na conclusão do trabalho, foi verificado que o modelo gera a grade horária satisfatoriamente, atendendo as restrições impostas e considerando também os pesos estabelecidos para mensurar a preferência dos professores, alunos e da instituição, minimizando as escolhas diferentes das preferências (ANDRADE; SCARPIN; STEINER, 2012).

5.3 AMBIENTE DE OTIMIZAÇÃO NA WEB: UMA APLICAÇÃO EM TIMETABLING

Este artigo foi escrito por José Auriço Oliveira e Plácido Rogério Pinheiro e publicado no 35º Simpósio Brasileiro de Pesquisa Operacional no ano de 2003.Tendo como objetivo principal conceber um ambiente de otimização na WEB, que possibilite a resolução de problemas utilizando técnicas de pesquisa operacional com a agregação de novos modelos e resolvedores. Com o intuito de pesquisar trabalhos abrangendo a WEB e a utilização de técnicas de pesquisa operacional, foi realizado um estudo na literatura, gerando como produto um modelo de arquitetura para o ambiente de otimização.

A implementação do ambiente foi aplicada no problema da alocação de professores a horários nas Escolas de Ensino Médio do Governo do Estado do Ceará. Para a solução do problema foi definido um modelo utilizando Programação Linear Inteira, possibilitando a geração e resolução com uma solução viável. Os resultados obtidos pela implementação no Estado do Ceará permitirão otimizar o

processo de construção das tabelas de horários nas escolas estaduais, bem como a reutilização do ambiente desenvolvido na plataforma INTERNET em outros problemas do Serviço Público (OLIVEIRA; PINHEIRO, 2003).

O estudo demonstra que a Internet poderá ser um importante componente nos ambiente de otimização a serem desenvolvidos, e que a utilização de uma arquitetura distribuída contribui para uma maior flexibilidade na utilização de várias técnicas de pesquisa operacional para a solução de problemas (OLIVEIRA; PINHEIRO, 2003).

5.4 UM ALGORITMO HÍBRIDO BASEADO EM ALGORITMOS MEMÉTICOS E RECONEXÃO POR CAMINHOS PARA RESOLUÇÃO DO PROBLEMA DE HORÁRIO ESCOLAR

Este artigo foi descrito por Alessandra Martins Coelho e Sérgio Ricardo de Souza, utilizando os dados referentes aos professores do Centro Federal de Educação Tecnológica de Rio Pomba-MG nos anos de 2005 e 2006.Propõe utilizar uma metodologia híbrida entre algoritmos meméticos e a técnica de reconexão por caminhos, para solucionar o Problema de Horário Escolar.

A metodologia proposta tem como objetivo satisfazer o maior número de professores quanto à quantidade de dias em que suas aulas são distribuídas; minimizar o número de janelas e quebras de aula; além de atender a um certo número de aulas duplas. Os resultados encontrados pelo algoritmo híbrido são comparados com os do método *iterated local search* (COELHO; SOUZA, 2006).

No final o trabalho apresenta uma aplicação de um algoritmo evolutivo híbrido para o Problema de Horário Escolar. Considerou-se, para análise, os dados referentes aos professores SPOLM 2006 Rio de Janeiro, Brasil, 15 e 16 de agosto de 2006 do Centro Federal de Educação Tecnológica de Rio Pomba nos anos de 2005 e 2006, para a programação de dois turnos em simultâneo.

Procurou-se, neste estudo, satisfazer ao maior número de professores quanto à quantidade de dias em que suas aulas são distribuídas, minimização do número de janelas e quebras de aula, bem como atender certo número de aulas duplas. A qualidade da solução final está relacionada com uma boa estrutura de vizinhança; com a maneira de se avaliar os requisitos não atendidos e o tempo computacional disponibilizado para a execução. No caso dos testes realizados, o

algoritmo híbrido proposto mostrou-se bastante eficaz, conseguindo encontrar, em todos os testes, quadros de horários de alta qualidade. O primeiro teste, referente aos cursos de Informática e Meio Ambiente, no qual todas as aulas são duplas, o algoritmo consegue encontrar sua melhor solução, na construção da população inicial (COELHO; SOUZA, 2006).

O tempo de execução cresce consideravelmente à medida que se aumentam as componentes do problema. O esforço computacional do algoritmo híbrido, causado pela utilização da reconexão por caminhos no operador de recombinação e pelo operador local, é compensado pela alta qualidade da solução final. A utilização de reconexão por caminhos, na fase de reprodução, garante a viabilidade dos quadros de horários filhos e a qualidade dos mesmos, no que se refere ao atendimento da satisfação dos professores (COELHO; SOUZA, 2006).

5.5 GERAÇÃO AUTOMÁTICA DE HORÁRIOS ESCOLARES UTILIZANDO ALGORITMOS DE SATISFAÇÃO DE RESTRIÇÕES

Este artigo foi desenvolvido por Claudenir de F. Machado, Patrick P. Silva, Ricardo V. Saggioro, Márcio Cardim e Elvio Gilberto da Silva, utilizando os dados da Universidade Sagrado Coração, sendo uma instituição de ensino superior, localizada na cidade de Bauru, no estado de São Paulo. Atualmente a Universidade dispõe de aproximadamente 270 professores distribuídos nos 16 cursos na área de Exatas, 9 cursos na área de Saúde, 18 cursos na área de Humanas e 7 cursos de curta duração. Diante destes dados é possível ter uma dimensão da quantidade de disciplinas vinculadas a cada curso e o nível de complexidade para gerar um horário.

Na metodologia proposta do artigo tem por finalidade desenvolver um programa para geração automática de um horário escolar, utilizando o algoritmo de satisfação de restrições, afim de que gere o horário escolar respeitando as condições de restrições dos professores. O propósito deste trabalho também foi mostrar que o algoritmo de satisfação de restrições é eficaz no ato da geração de horários escolares.

Foi possível concluir que o Algoritmo processa as informações de forma mais rápida e eficiente, colaborando com a redução de tempo. O trabalho atingiu o resultado esperado, demonstrando que para gerar um horário escolar é necessário respeitar a uma série de restrições cujo Algoritmo de Satisfação de Restrições

possui todo um conceito que auxilia na manipulação destas inter-relações professores, disponibilidade e disciplinas (MACHADO et al, 2015).

5.6 UMA SOLUÇÃO DO PROBLEMA DE HORÁRIO ESCOLAR VIA ALGORITMO GENÉTICO PARALELO

Este trabalho foi desenvolvido pelo Eduardo Luiz Miranda Lobo, aplicado na Universidade Presidente Antônio Carlos – UNIPAC, Situada em conselheiro Lafaiate – MG.

Como não há, na literatura, pelo menos do conhecimento do autor dessa dissertação, um estudo sobre o quadro de horários escolar segundo os regimentos pedagógicos e administrativos da instituição objeto de estudo, fica difícil fazer uma comparação com trabalhos anteriores. No entanto, com base na confecção manual dos quadros de horários que já ocorreram na instituição objeto de análise, o algoritmo mostrou eficiência com relação a tempo e qualidade de confecção em todos os casos de teste. Seu comportamento demonstra que, à medida que o tamanho da população é ampliado, há melhorias em seus resultados e, obviamente, um custo computacional maior é acarretado. No entanto, isto já é uma característica peculiar de algoritmos genéticos e a abordagem do paralelismo computacional é justamente para atenuar custos que tornassem inviáveis a aplicação desta técnica para problemas desta categoria (LOBO, 2005).

A abordagem da computação ‘paralela mostrou-se extremante eficaz, dada a questão da rapidez com que as convergências ocorrem, devido, principalmente, à troca de cromossomos entre as populações, o que fortalece a heterogeneidade das mesmas, assim como a obtenção de indivíduos cada vez mais adaptados. No entanto, o paralelismo computacional invoca uma série de sintonias que envolvem ambiente operacional, padrões de hardware, linguagens de programação, padrões de implementação, o que torna o ambiente de desenvolvimento altamente propício a erros (LOBO, 2005).

6 TRABALHO DESENVOLVIDO

Neste capítulo, apresenta-se a descrição das etapas adotadas para a realização do estudo, no qual são descritos os procedimentos metodológicos, as formas para a análise comparativa e as etapas envolvidas na construção do protótipo.

6.1 METODOLOGIA

Na primeira etapa de desenvolvimento do presente trabalho, procedeu-se uma pesquisa bibliográfica-exploratória, tendo como intuito maior a identificação e seleção de trabalhos sobre o tema de estudo, visando alcançar uma organização adequada, coerência necessária e confiabilidade dos dados apresentados pelo estudo.

No que se refere à técnica de estudo, foi realizada uma pesquisa de base tecnológica, que conforme Ramos (2013), é um tipo de estudo aplicado, que objetiva alcançar uma inovação em determinado processo ou produto, perante a uma necessidade ou demanda preestabelecida.

A pesquisa tecnológica desenvolvida teve por objetivo obter uma análise comparativa de desempenho entre dois algoritmos, para solucionar o problema de formulação de horário escolar.

Para tanto, utilizou-se para análise comparativa, o *software* Cronos, que se trata de software web desenvolvido por ex-alunos da Universidade Federal de Lavras – UFLA, fruto de um trabalho da disciplina de Inteligência Artificial do curso de Ciência da Computação da mencionada instituição no ano de 2005 e lançado em 2009 comercialmente. Tal sistema é baseado em um algoritmo que gera horários escolares de acordo com as necessidades e restrições das instituições de ensino.

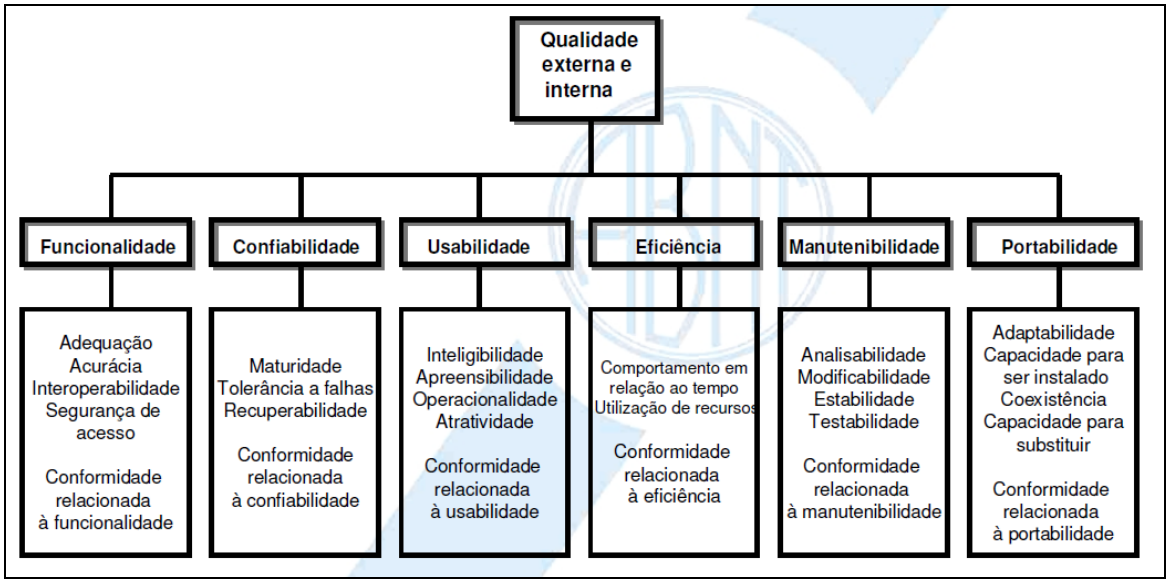
Desse modo, por meio do *software* Cronos é possível montar um quadro de horários para os professores e turmas de maneira automatizada, considerando as principais restrições existentes na elaboração do mesmo.

Através de contato via e-mail com a equipe de desenvolvimento, foi possível compreender que o *software* utiliza-se do algoritmo genético como base.

Para a análise comparativa, procedeu-se a formulação de um programa computadorizado que utiliza o algoritmo *backtracking* como base.

Visando-se analisar ambos algoritmos utilizou-se a métrica de qualidade do produto, segundo a ISO/IEC 9126. Essa série subdivide-se em três normas e fornece características (parte 1) e métricas (parte 2 e 3) para avaliação do software. Neste trabalho foram utilizadas as métricas da parte 1, cuja distribuição ocorre em seis características principais, com cada uma delas divididas em sub-características, conforme se pode verificar na Figura 6:

Figura 6 - Modelo de avaliação utilizado



Fonte: ABNT (2003)

Utilizando-se estes critérios, foi realizada a comparação das categorias e suas respectivas subcategorias em relação aos dois algoritmos do estudo. Em seguida, foi feita a conclusão do estudo.

As fases da construção do protótipo são apresentadas na sequência.

6.2 PROTÓTIPO GHE – GERAÇÃO DE HORÁRIO ESCOLAR

Utilizou-se no desenvolvimento do protótipo a linguagem Java 8, no ambiente de desenvolvimento NetBeans, versão 8.0.2. O SGBD utilizado foi o Sybase 9.0 e a versão do Hibernate foi a 4.3. Na Figura 7, demonstra-se o arquivo xml da classe de persistência.

Figura 7 - Arquivo xml de persistência do Hibernate

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
                                 http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="escolarPU">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <!-- <jta-data-source>java:/MyApp</jta-data-source> -->
    <class>br.com.unesc.principal.modelos.DisciplinaSerie</class>
    <class>br.com.unesc.principal.modelos.Disciplina</class>
    <class>br.com.unesc.principal.modelos.Professor</class>
    <class>br.com.unesc.principal.modelos.Turma</class>
    <class>br.com.unesc.principal.modelos.Serie</class>
    <class>br.com.unesc.principal.modelos.ProfessorDisciplina</class>
    <properties>
      <property name="hibernate.dialect" value="org.hibernate.dialect.SybaseAnywhereDialect"/>
      <property name="hibernate.connection.driver_class" value="com.sybase.jdbc3.jdbc.SybDriver"/>
      <property name="hibernate.connection.username" value="admdba"/>
      <property name="hibernate.connection.password" value="sql"/>
      <property name="hibernate.connection.url" value="jdbc:sybase:Tds:localhost:2638/TCCTagner"/>
      <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
      <!--  atualiza o banco, gera as tabelas se for preciso -->
      <!--     <property name="hibernate.hbm2ddl.auto" value="update"/> -->
    </properties>
  </persistence-unit>
</persistence>
```

Fonte: Do autor

Com o decorrer do desenvolvimento, percebeu-se a necessidade de utilizar um *software* para abrir ou gerenciar o documento gerado pelo protótipo. Assim optou-se gerar em extensão .pdf, no qual pode ser aberto pela ferramenta disponibilizada gratuitamente na internet, *Adobe Reader* para qualquer versão.

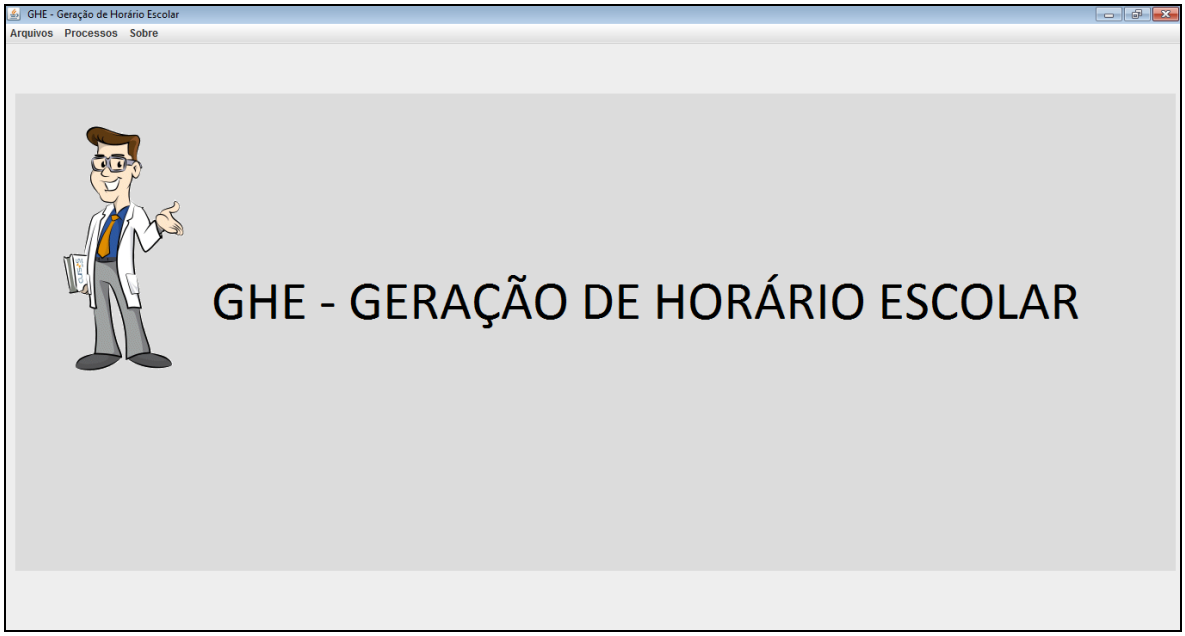
Para realizar a comunicação entre o banco de dados e aplicação, utilizou-se da biblioteca do Sybase, sendo ela: **com.sybase.jdbc3.jdbc.SybDriver**. Para geração de relatório foi utilizado à biblioteca itextpdf-5.5.1. Com a utilização da API *OpenSource iText,* tenho a possibilidade de manipular e criar documentos em formato PDF, além de XML e RTF. Devido seu código ser aberto, essa API (biblioteca) tem distribuição sob as licenças LGPL e MPL, proporcionando assim seu uso em diversos sistemas.

A interface do protótipo foi projetada e modelada através da própria IDE *Netbeans,* foram utilizados os componentes do próprio e também foi importado outra biblioteca, sendo ela *SwingX,* que é uma API, que originou-se de um subprojeto do *SwingLabs*, que conta com suporte da *Sun Microsystems*.

Com a integração da API ao *Netbeans* pode-se obter outros componentes para o desenvolvimento da interface. Foram utilizados componentes como: *JTextField* ,*JButton*, *JTable*,*JPanel,*dentre outros.

Na Figura 8, a seguir, apresenta-se a interface principal do GHE – Geração de Horário Escolar.

Figura 8 - Tela principal do protótipo



Fonte: Do autor (2015)

No desenvolvimento da interface principal, que seria o primeiro contato do usuário com a aplicação, procurou-se obter algo que pudesse ser de fácil compreensão e utilização.

6.2.1 Funcionamento do GHE

Para a utilização do protótipo se faz necessário ter conhecimentos das seguintes funcionalidades:

Cadastro de Professores: no cadastro de professores nos deparamos com uma tela simples e objetiva, onde deve ser informado o código sequencial do cadastro e o nome.

Cadastro de Disciplina: no cadastro de disciplinas podemos encontrar os dados que uma disciplina pode ter que são: código sequencial do cadastro, nome, grau de dificuldade (1 – difícil, 2 – intermediária ou fácil), carga horária em minutos e uma opção de adicionar, onde na mesma opção eu associo os professores que irão lecionar a disciplina.

Vale ressaltar também, que o grau de dificuldade e a carga horária semanal, é um dos campos principais do cadastro, pois a mesma informação é utilizada diversas vezes no algoritmo da geração.

Cadastro de Série: no cadastro de série podemos encontrar os dados que uma série necessita, tais como: código sequencial do cadastro, nome, série, classif. (Ensino Fundamental ou Ensino Médio). Também é importante frisar que temos a opção de adicionar N disciplinas para uma mesma série.

Cadastro de Turma: no cadastro de turma funciona exatamente para dizer que uma série pode ter mais de uma turma.

Ex: 3º série A, 3º série B, etc.. Neste cadastro temos a possibilidade de informar o código sequencial do cadastro, nome da turma, série que a turma esta cadastrada e o turno da mesma.

Geração do Horário: o processo de gerar o horário escolar é o ponto principal do protótipo, pois é no mesmo onde é aplicado o PR e desenvolvido via banco sua geração, aplicando o algoritmo de busca e satisfação de restrições.

A geração do horário escolar é feita por turma, ou seja, ao informar a turma tenho em mãos qual a série que ela pertence, as disciplinas daquela série e sucessivamente o professor que poderá lecionar a turma.

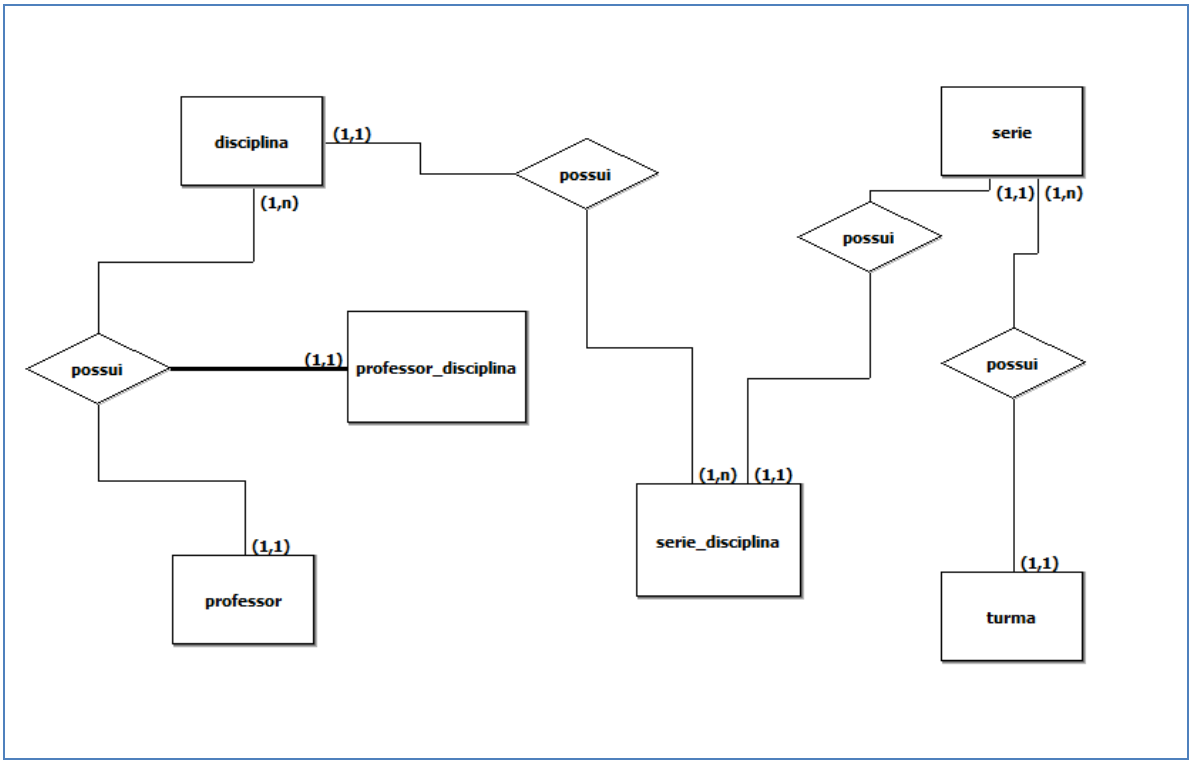
Como regras gerais para todos os cadastros podem ser efetuados alterações, seleções, criações, exclusões e manipulações dos registros, pelas opções: Novo, Selecionar, Excluir e Gravar.

O principal processo do protótipo refere-se a possibilidade de gerar o horário escolar. No *menu* gerar horário escolar existe a possibilidade de informar a turma a ser gerada. Se informar uma turma que não foi cadastrada no sistema, será apresentada uma mensagem de que a turma não foi cadastrada.

6.2.2 Modelagem de bancos de dados GHE

Para o protótipo ser desenvolvido buscou-se obter uma base de dados que pudesse de uma forma simples, elaborar a geração e atender todas as restrições necessárias dos alunos e professores. A modelagem do banco de dados foi feita através do processo de levantamento de requisitos. Com os requisitos levantados, foi possível assim elaborar o modelo entidade relacionamento, conforme Figura 9.

Figura 9 - Modelo entidade relacionamento



Fonte: Do autor (2015).

Segundo Silberschatz, Korth e Sudarshn (2006) o modelo entidade relacionamento é baseado em coleções de objetos do mundo real, onde se pode atribuir várias entidades. Essas entidades têm há possiblidade de inserir muitos atributos. Quando se associa várias entidades, cria-se uma relação entre elas.

Através da utilização do *software* brModelo, foi possível concluir o processo de modelagem do banco de dados.

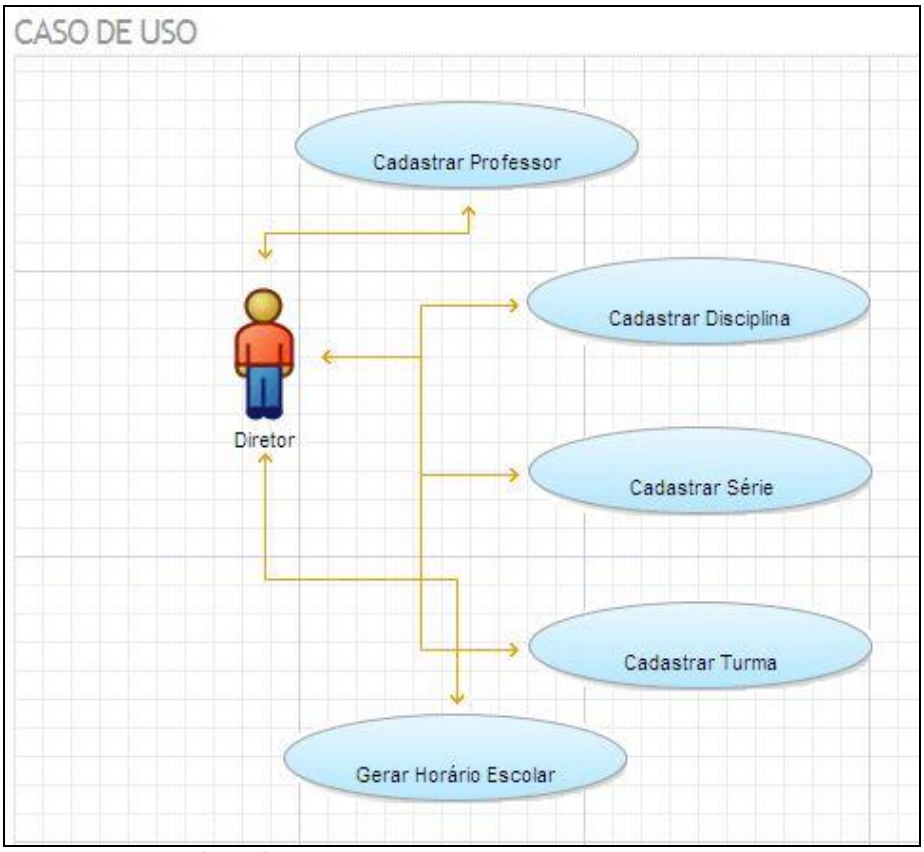
6.2.3 Caso de uso GHE

Com o desenvolvimento da base de banco de dados partiu-se para o desenvolvimento do protótipo em si. Através da ferramenta *Diagrame*, foi elaborado o diagrama de caso de uso.

Casos de uso lidam apenas requisitos funcionais de um sistema. Outros requisitos, como regras de negócios, qualidade do serviço requisitos e restrições de implementação devem ser representados separadamente. Arquitetura e detalhes internos também devem ser descritos separadamente (MICROSOFT, 2013 - tradução nossa).

Na Figura 10, é possível observar os métodos necessários para o desenvolvimento do protótipo.

Figura 10 - Diagrama de Caso de Uso



Fonte: Do autor (2015).

Pode-se afirmar, assim, que as ações primordiais são o cadastramento do professor, da disciplina, da série e da turma, para a posterior geração do horário escolar.

6.2.4 Descrevendo e exemplificando a utilização do algoritmo *Backtracking*

A utilização do algoritmo *backtracking* no protótipo desenvolvido se fez necessário por utilizar de mecanismos que eliminam os caminhos que não levam as respostas válidas e retroceder em busca de uma solução viável para o problema, assim apresenta resultados que satisfaçam as restrições aplicadas a uma formulação ou geração de um horário escolar.

6.2.4.1 Variáveis de entrada do algoritmo *backtraking*

As variáveis de entrada fazem parte do cadastro inicial de informações para a posterior geração dos horários escolares. Elas encampam o número de professores, turmas, disciplinas, número de períodos de tempo disponíveis e carga horária da disciplina.

6.2.4.2 Restrições estabelecidas para o algoritmo *backtraking*

Para o adequado desenvolvimento do presente estudo, algumas restrições foram estabelecidas, de modo que não poderiam ser desrespeitadas na formulação de horários escolares.

As restrições fracas determinadas foram:

a) garantir que aulas de maior grau de dificuldade sejam intercaladas com aulas de menor grau de dificuldade;
b) garantir que apenas duas aulas da mesma disciplina ocorram no mesmo período para a mesma turma;
c) a escolha do professor é feita conforme o professor que nunca deu aula em uma turma;
d) caso o professor já lecionou, verifica ou seleciona o professor que menos deu aula nas turmas.

Nas Figuras 11 a 13, pode-se verificar como foram aplicadas as restrições fracas no algoritmo backtracking.

Figura 11 – Restrição fraca: Tenta intercalar disciplina difíceis com as fáceis

```
select count(distinct(turma)) into w_totalTurmasGrade
            from grade, turma where grade.turno = w_turno and
            grade.turma = turma.id and
                 turma.id_serie = w_idSerie;

           set w_ordem = 0;
           set w_grau = 1;
           lb_ordena: loop
              set w_ordem = w_ordem + 1;
                 open curOrdem with hold;
                      lb_ordena2: loop
                                  fetch next curOrdem into w_idDisciplina;
                                  if sqlstate = 2000 then
                                             leave lb_ordena2;
                                  end if;
                                  insert into ordena_disciplinas values (w_idDisciplina,w_ordem);

                                  leave lb_ordena2;
                      end loop;
   close    curOrdem;
                     //Restrição fraca >> Tenta intercalar disciplina de grau 1 com grau 2.(Dificil com facil)
                      if w_grau = 1 then
                         set w_grau = 2;
                      else
                         set w_grau = 1;
                      end if;

                      if (select count() from serie_disciplina where id_serie = w_idSerie) =
                         (select count() from ordena_disciplinas) then
                          leave lb_ordena;
                      end if;
```

Fonte: Do autor (2015).

Figura 12 – Restrição fraca: Função que define a quantidade que o professor pode lecionar em uma turma no dia

```
//Restrição fraca >> Define qual a quantidade de aulas que um professor pode dar em um dia
set w_tentativas = w_tentativas + 1;
if isnull(dbf_getTotalAulaTurmaDia(a_turma,w_dia,w_idDisciplina),0) >= 2 and w_tentativas < 20 then
    leave lb_horas;
end if;
```

Fonte: Do autor (2015).

Figura 13 – Restrição fraca: Irá selecionar um professor que menos lecionou na disciplina correspondente

```
//Restrição fraca >> Irá pegar um professor que menos lecionou da disciplina correspondente
 select first id_professor as professor,
                     isnull((select count()
                                 from grade
                                where grade.professor = id_professor
                             group by grade.professor
                             order by 1 asc),0) as tot
                into w_professor,w_teste
                from professor_disciplina
               where id_disciplina = w_disciplina
            order by  2 ,1;
```

Fonte: Do autor (2015).

As restrições fortes determinadas foram:

a) garantir que o número de 5 aulas (45 min/cada) por período seja respeitado, inclusive evitando-se as janelas;
b) garantir que a carga horária semanal de uma turma seja de 1.125 minutos, no total;
c) garantir que um professor será destacado para atender apenas uma turma em um mesmo horário;
d) garantir que cada disciplina tenha o número necessário de aulas de determinada matéria.

Nas figuras 14 a 16, pode-se verificar como foram aplicadas as restrições fortes no algoritmo *backtracking*:

Figura 14 – Restrição forte: Verificação que garante que uma aula será de 45 min.

```
//verificação para acumular uma aula de 45 min
if not exists(select first 1
                from acumula_semana
                where disciplina = a_disciplina) then

            insert into acumula_semana values (a_disciplina,45);
else

            update acumula_semana
               set total_horas = total_horas + 45
             where disciplina = a_disciplina;

end if ;
```

Fonte: Do autor (2015).

Figura 15 - Restrição forte: Verificação que garante que uma semana terá 25 aulas de 45 min.

```
//Se completou a carga horaria semanal termina a geracao do horario
if dbf_getTotalAcumuladoHorasSemana() >= 1125 then // total de minutos na semana
            leave lb_semana;
end if;
if w_tentativas > 25 then
            leave lb_semana;
end if;
```

Fonte: Do autor (2015).

Figura 16 – Restrição forte: Evita o choque de horário do professor

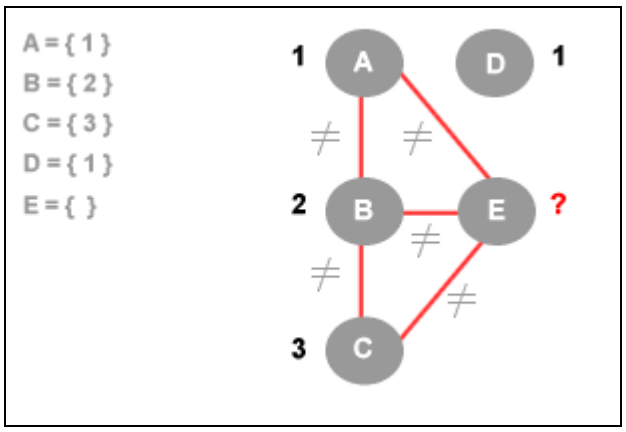
```
//Restrição forte >> Se o professor da disciplina XX, esta lecionando neste mesmo dia e aula, não irá aceitar
if dbf_getProfessorLecionando(w_professor,w_dia,w_aula,a_turma,w_turno) = 1 then
    leave lb_horas;
end if;
```

Fonte: Do autor (2015).

6.2.4.3 Funcionamento do algoritmo *backtraking*

Para resolver um problema o algoritmo backtracking deve-se realizar as operações de atribuição de valores a variáveis, até que ele encontra a combinação de valores, capazes de atender todas as restrições binárias do problema. Suponde-se que se tenha seguido um processo de atribuição sequencial que leva à seguinte situação, conforme Figura 17.

Figura 17 – Situação simulada



Fonte: Massimo (2013)

Com base na atribuição sequencial das variáveis relacionadas a um problema a ser solucionado, tem-se a atribuição dos valores do protótipo desenvolvido, representados na Figura 18.

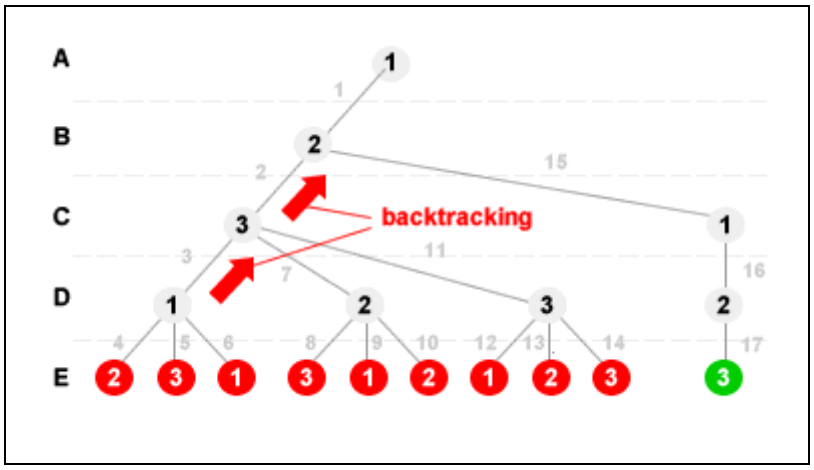
Figura 18 – Atribuição dos valores do protótipo desenvolvido

```
declare curTurma dynamic scroll cursor for
    select nome,
           id_serie,
           turno
      from turma
     where id = a_turma;

eclare curOrdem dynamic scroll cursor for
    select disciplina.id
      from serie_disciplina key join disciplina
     where id_serie = w_idSerie and
           grau_dificuldade = w_grau and
           (select count()
              from ordena_disciplinas
             where ordena_disciplinas.disciplina = serie_disciplina.id_disciplina) = 0
  order by carga_horaria_semanal desc, id_disciplina asc;

declare curDisciplina dynamic scroll cursor for
    select disciplina, nome, carga_horaria_semanal, if w_totalTurmasGrade < 2 then 1 else carga_horaria_semanal endif as ordem2
      from ordena_disciplinas,
           disciplina
     where ordena_disciplinas.disciplina = disciplina.id and
           carga_horaria_semanal > isnull((select sum(total_horas)
                                             from acumula_semana
                                            where acumula_semana.disciplina = disciplina.id),0)
  order by ordem,ordem2,disciplina;

      declare w_nome char(100);
      declare w_idSerie integer;
      declare w_turno integer;
      declare w_carga_horaria integer;
      declare w_idDisciplina integer;
```

Fonte: Massimo (2013)

Não é possível fazer qualquer atribuição para a variável E porque cada domínio valor {1, 2, 3} viola uma restrição com relação às variáveis A, B, C. O processo de atribuição falhou. Um algoritmo de busca retrocesso iria saltar de volta para a última variável atribuída (D) para tentar mudar o seu valor. No entanto, a transação é inútil uma vez que o valor da variável D não afeta as restrições das outras variáveis (A, B, C, E). Por exemplo, pode-se atribuir à variável D o valor 2 ou 3, mas , em qualquer caso, a atribuição para a variável E violaria uma restrição do problema. Depois de observar o fracasso das atribuições da variável D, o algoritmo seria forçado a mais retrocesso para a variável C. Usando o algoritmo de busca retrocesso, faz 17 tentativas e 2 retrocesso antes de chegar a uma solução (Figura 19).

Figura 19– Processo de retroceder



Fonte: Massimo (2013)

Com base no processo referido, a seguir destaca-se a forma pela qual foi realizado o mesmo processo no protótipo desenvolvido, conforme Figura 20.

Figura 20 - Processo que retrocede na geração do horário GHE

```
//Retrocede na geração do horário, até achar um professor que poderá lecionar naquele dia e aula
if admdba.dbf_getProfessorLecionando(w_professor,w_dia_semana,w_aula_vaga,a_turma,a_turno) = 0 and
   admdba.dbf_getProfessorLecionando(w_professor_update,w_dia,w_aula,a_turma,a_turno) = 0 and
   (select count()
      from grade
     where turma = a_turma and
           professor = w_professor_update and
           dia_semana = w_dia) <= 3 and
   w_professor_update <> w_professor then

    if dbf_getTotalAcumuladoHorasSemana(*) >= 1125 then // total de horas da semana
      leave lb_grade;
    end if;

    update analise_dados
       set numero_reajuste = isnull(numero_reajuste,0) + 1
     where a_turma = turma;

   insert into grade values(a_turma,w_dia_semana,w_aula_vaga,w_disciplina,w_professor,a_turno);

   update grade
      set disciplina = w_disciplina_faltante,
          professor = w_professor_update
    where turma = a_turma and
          dia_semana = w_dia and
          aula = w_aula;


   //Se nao existir horas na disciplina na semana inseri , se nao cai na questao de atualizar +45 min.
   call AcumulaSemana(w_disciplina_faltante);
   commit;
end if
```

Fonte: Massimo (2013)

Na sequência, são destacados os aspectos referentes aos testes de

implementação realizados no protótipo desenvolvido.

6.2.5 Testes de implementação no protótipo desenvolvido

Na etapa de realização dos testes foi utilizado o seguinte sistema: computador desktop com sistema operacional Windows 7, 4 GB de memória RAM, processador AMD Phenom II N830 Triple-Core Processador, 2.10 GHz.

O quadro de testes foi definido da seguinte forma:

a) cadastrar treze professores que lecionam nas diferentes disciplinas;
b) cadastrar as doze disciplinas a serem ministradas para as diferentes turmas do turno matutino, com cargas horárias específicas;
c) associar os professores a disciplina correspondente;
d) cadastrar quatro séries, todas elas do ensino fundamental (6º, 7º, 8º e 9º ano);
e) associar as disciplinas com as respectivas séries;
f) cadastrar quatro turmas;
g) informar a turma para a qual cada série pertence.

Com base nisso, a seguir, são apresentados os resultados obtidos.

6.2.6 Resultados obtidos pelo GHE

Após a organização e condução dos testes de implementação do protótipo de GHE, diferentes foram os resultados obtidos pelo trabalho. Foi realizada1 (uma) execução do processo de geração escolar englobando as 4 (quatro) turmas com as respectivas séries, professores e disciplinas cadastradas.

Na sequência, foi realizada outra execução do processo de geração escolar englobando seis turmas (1º ano A, 1º 2º ano A, 3º ano A, 4º ano A, 5º ano A, 6º ano A), com as respectivas séries, professores e disciplinas cadastradas.

No primeiro teste, o tempo do processo com o algoritmo *backtracking* foi de 0.6720 segundos (s). Não houve choque de professores por horário, ou seja, nenhum professor foi destacado para atuar em mais de uma turma no mesmo horário.

O número de dias em que houve excesso de aulas da mesma disciplina foi dezessete, ou seja, em dezessete situações na mesma semana o número de aulas da mesma disciplina ultrapassou duas horas/aula.

O número de reajustes foi de sete, em outras palavras, o sistema teve que retroceder sete vezes até encontrar o horário adequado.

O número de janelas entre as aulas dos professores foi de vinte e oito, ou seja, os treze professores tiveram vinte e oito intervalos entre suas aulas.

Para a melhor compreensão dos resultados obtidos, são apresentadas as figuras de 21 a 24.

Figura 21 - Horário turma 1-A

Matutino

Turma 1-A

Horário	Segunda	Terça	Quarta	Quinta	Sexta
Aula 1	MAT (LENIR)	FISICA (JOSE ADRIANO)	GEO (LECIA VIEIRA)	CIEN (MARILEIA)	MAT (LENIR)
Aula 2	MAT (LENIR)	FISICA (JOSE ADRIANO)	GEO (LECIA VIEIRA)	L P (DEUSELI ALV)	MAT (LENIR)
Aula 3	CIEN (MARILEIA)	HIST (LUCELINA)	ED FISICA (LUCY)	L P (DEUSELI ALV)	L P (DEUSELI ALV)
Aula 4	CIEN (MARILEIA)	HIST (LUCELINA)	MAT (LENIR)	HIST (LUCELINA)	L P (DEUSELI ALV)
Aula 5	L P (DEUSELI ALV)	INGLES (VANIA)	MAT (LENIR)	GEO (LECIA VIEIRA)	L P (DEUSELI ALV)

Fonte: Do autor (2015).

Figura 22 - Horário turma 2-A

Matutino

Turma 2-A

Horário	Segunda	Terça	Quarta	Quinta	Sexta
Aula 1	L P (DEUSELI ALV)	HIST (ROBERTA)	ED FISICA (LUCY)	L P (DEUSELI ALV)	CIEN (MARILEIA)
Aula 2	L P (DEUSELI ALV)	HIST (ROBERTA)	MAT (DENILSON)	FISICA (JOSE ADRIANO)	L P (DEUSELI ALV)
Aula 3	L P (DEUSELI ALV)	INGLES (ITALO ANTONIO)	MAT (DENILSON)	HIST (ROBERTA)	MAT (DENILSON)
Aula 4	L P (DEUSELI ALV)	GEO (LECIA VIEIRA)	CIEN (MARILEIA)	GEO (LECIA VIEIRA)	MAT (DENILSON)
Aula 5	FISICA (JOSE ADRIANO)	GEO (LECIA VIEIRA)	CIEN (MARILEIA)	MAT (DENILSON)	MAT (DENILSON)

Fonte: Do autor (2015).

Figura 5 - Horário turma 3-A

Matutino

Turma 3-A

Horário	Segunda	Terça	Quarta	Quinta	Sexta
Aula 1	CIEN (MARILEIA)	L P (DEUSELI ALV)	L P (DEUSELI ALV)	FISICA (JOSE ADRIANO)	MAT (WILDNEY)
Aula 2	CIEN (MARILEIA)	INGLES (VANIA)	L P (DEUSELI ALV)	HIST (ROBERTA)	FISICA (JOSE ADRIANO)
Aula 3	MAT (WILDNEY)	GEO (LECIA VIEIRA)	CIEN (MARILEIA)	GEO (LECIA VIEIRA)	HIST (ROBERTA)
Aula 4	MAT (WILDNEY)	ED FISICA (LUCY)	MAT (WILDNEY)	L P (DEUSELI ALV)	GEO (LECIA VIEIRA)
Aula 5	HIST (ROBERTA)	ARTES (WALDINEI)	MAT (WILDNEY)	L P (DEUSELI ALV)	INGLES (VANIA)

Fonte: Do autor (2015).

Figura 6 - Horário turma 4-A

Matutino

Turma 4-A

Horário	Segunda	Terça	Quarta	Quinta	Sexta
Aula 1	MAT (WILDNEY)	GEO (LECIA VIEIRA)	CIEN (MARILEIA)	INGLES (VANIA)	HIST (ROBERTA)
Aula 2	MAT (WILDNEY)	L P (DEUSELI ALV)	CIEN (MARILEIA)	GEO (LECIA VIEIRA)	GEO (LECIA VIEIRA)
Aula 3	FISICA (JOSE ADRIANO)	L P (DEUSELI ALV)	L P (DEUSELI ALV)	CIEN (MARILEIA)	ED FISICA (LUCY)
Aula 4	FISICA (JOSE ADRIANO)	L P (DEUSELI ALV)	HIST (ROBERTA)	MAT (WILDNEY)	ARTES (WALDINEI)
Aula 5	INGLES (VANIA)	L P (DEUSELI ALV)	HIST (ROBERTA)	MAT (WILDNEY)	MAT (WILDNEY)

Fonte: Do autor (2015).

Todos os testes efetuados obtiveram êxito e foi possível identificar que a execução dos mesmos foi rápida, menos de um segundo quando da utilização do algoritmo de *backtracking*.

A seguir, são apresentados os resultados que foram observados com a realização dos testes no software Cronos.

6.2.7 Resultados obtidos pelo software Cronos

Na sequência, procedeu-se de uma comparação com outro teste baseado em uma base de dados do software Cronos que possui a utilização do algoritmo genético. A implementação do algoritmo se deu em JAVA™, enquanto a máquina utilizada foi em um computador desktop com sistema operacional Windows 7, 4 GB de memória RAM, processador Dell Core i7.

O tempo do processo com o algoritmo genético foi de 0.7434 segundos (s). Nesse teste também não houve choque de professores por horário.

O número de dias em que houve excesso de aulas da mesma disciplina foi doze situações na mesma semana o número de aulas da mesma disciplina ultrapassou duas horas/aula.

O número de janelas entre as aulas dos professores também foi de vinte e oito.

Para a melhor compreensão dos resultados obtidos nesse teste com algoritmo genético, são apresentadas as figuras de 25 a 28.

Figura 7 - Horário turma 6-A

Turma: 6ºAno A				
Segunda-feira	**Terca-feira**	**Quarta-feira**	**Quinta-feira**	**Sexta-feira**
ED. FISICA/Lucy	L P/Deuseli Alv	INGLÊS/Vânia	MAT/Lenir	L P/Deuseli Alv
L P/Deuseli Alv	GEO/Lecia Vieir	ED. F/José Adrian	GEO/Lecia Vieir	ED. F/José Adrian
MAT/Lenir	MAT/Lenir	MAT/Lenir	HIST/Lucelina	GEO/Lecia Vieir
MAT/Lenir	L P/Deuseli Alv	CIEN/Marileia	L P/Deuseli Alv	CIEN/Marileia
HIST/Lucelina	MAT/Lenir	L P/Deuseli Alv	CIEN/Marileia	HIST/Lucelina

Fonte: Geração da Cronos (2015).

Figura 8 - Horário turma 7-A

Turma: 7ºAno A				
Segunda-feira	**Terca-feira**	**Quarta-feira**	**Quinta-feira**	**Sexta-feira**
L P/Deuseli Alv	CIEN/Marileia	GEO/Lecia Vieir	HIST/Lucelina	MAT/Denilson
GEO/Lecia Vieir	MAT/Denilson	L P/Deuseli Alv	CIEN/Marileia	L P/Deuseli Alv
HIST/Lucelina	L P/Deuseli Alv	ED. F/José Adrian	L P/Deuseli Alv	ED. F/José Adrian
NGLÊS/ItaloAntôni	MAT/Denilson	L P/Deuseli Alv	MAT/Denilson	MAT/Denilson
ED. FISICA/Lucy	GEO/Lecia Vieir	HIST/Lucelina	MAT/Denilson	CIEN/Marileia

Fonte: Geração da Cronos(2015).

Figura 9 - Horário turma 8-A

Turma: 8ºAno A				
Segunda-feira	**Terca-feira**	**Quarta-feira**	**Quinta-feira**	**Sexta-feira**
HIST/Lucelina	MAT/Wildney	ED. F/José Adrian	L P/Deuseli Alv	ED. F/José Adrian
MAT/Wildney	CIEN/Marileia	HIST/Lucelina	MAT/Wildney	MAT/Wildney
L P/Deuseli Alv	GEO/Lecia Vieir	L P/Deuseli Alv	CIEN/Marileia	CIEN/Marileia
ED. FISICA/Lucy	MAT/Wildney	INGLÊS/Vânia	GEO/Lecia Vieir	HIST/Lucelina
L P/Deuseli Alv	L P/Deuseli Alv	GEO/Lecia Vieir	INGLÊS/Vânia	ARTE/Waldinei

Fonte: Geração da Cronos (2015).

Figura 10 - Horário turma 9-A

Turma: 9ºAno A				
Segunda-feira	**Terca-feira**	**Quarta-feira**	**Quinta-feira**	**Sexta-feira**
MAT/Wildney	MAT/Wildney	L P/Deuseli Alv	HIST/Roberta	ARTE/Waldinei
ED. FISICA/Lucy	L P/Deuseli Alv	INGLÊS/Vânia	L P/Deuseli Alv	GEO/Lecia Vieir
MAT/Wildney	CIEN/Marileia	CIEN/Marileia	INGLÊS/Vânia	HIST/Roberta
L P/Deuseli Alv	GEO/Lecia Vieir	ED. F/José Adrian	CIEN/Marileia	ED. F/José Adrian
MAT/Wildney	MAT/Wildney	HIST/Roberta	GEO/Lecia Vieir	L P/Deuseli Alv

Fonte: Geração da Cronos (2015).

A seguir, apresenta-se a comparação entre os dois algoritmos.

6.3 RESULTADOS OBTIDOS

Os tópicos seguintes, buscam demonstrar os resultados obtidos na análise comparativo dos algoritmos em estudo.

6.3.1 Análise comparativa entre os dois algoritmos

Realizar essa comparação entre os protótipos trata-se de ação de grande valia, considerando-se que surge a possibilidade de identificar a aplicação que melhor atende as demandas do seu usuário (MEIRELLES, 2008).

A avaliação de alguns ou diversos processos dentro do desenvolvimento de softwares é relevante, como forma de apontar para os pontos fortes e fracos do mesmo e, assim, permitir que alterações sejam realizadas e resultados mais positivos sejam alcançados (CALAZANS; ALVARENGA, 2014).

Após os testes com o protótipo e a apresentação dos resultados, procedeu-se de uma comparação entre os dois protótipos, sendo um deles com algoritmos *backtracking* e outro com algoritmo genético.

A métrica ou modelo de avaliação da qualidade ou dos critérios qualitativos adotada foi a ISO/IEC 9126 (NBR 13596), que fornece um modelo de propósito geral, o qual define seis amplas categorias de características de qualidade de software que são, por sua vez, subdivididas em subcaracterísticas.

Para a maior clareza e organização da comparação, apresenta-se os quadros a seguir, com o resumo comparativo dos dados dos dois protótipos considerados na avaliação em cada categoria e suas respectivas subcategorias.

6.3.1.1 Funcionalidade

A funcionalidade é o conjunto de funções satisfazem as necessidades explícitas e implícitas para a finalidade a que se destina o produto. Nesta categoria, incluem-se as seguintes subcategorias: adequação, acurácia, interoperabilidade, segurança do acesso, e conformidade. No Quadro 2, é representado o comparativo entre ambos os protótipos analisados segundo este critério.

Quadro 2 - Comparativo entre os protótipos conforme o critério da funcionalidade

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Adequação: Propõe-se a fazer o que é apropriado?	Sim	Sim
Acurácia: Gera resultados corretos ou conforme acordados?	Sim	Sim
Interoperabilidade: É capaz de interagir com os sistemas especificados?	Sim	Sim
Segurança de acesso: Evita acesso não autorizado, acidental ou deliberado a programas e dados?	Não	Sim
Conformidade: Está de acordo com normas e convenções previstas em leis e descrições similares?	Não	Sim

Fonte: Do autor (2015).

A análise dos protótipos segundo a categoria da funcionalidade em suas respectivas quatro subcategorias ou métricas permite evidenciar que o algoritmo genético atende todas as perguntas-chaves da ISO/IEC 9126. Já algoritmo de *backtracking*, por sua vez, encontra-se em conformidade em praticamente todos os requisitos da categoria, exceto nas subcategorias relacionadas à segurança do acesso e conformidade. No critério segurança de acesso, frente ao fato da implementação do algoritmo ter sido em um banco de dados poderá haver *hackers* sequestrando informações, enquanto o genético fornece maior padrão de segurança

em relação aos seus dados por rodar na Web. Já no critério relacionado à conformidade, o algoritmo *backtraking* não está conforme segundo as leis por se tratar de um protótipo e ser utilizado somente para a análise comparativa.

6.3.1.2 Confiabilidade

A confiabilidade é o desempenho mantido ao longo do tempo e em condições estabelecidas. Nesta categoria, incluem-se as seguintes subcategorias: adequação, acurácia, Interoperabilidade, segurança do acesso, e conformidade.

Quadro 3- Comparativo entre os protótipos conforme o critério da confiabilidade

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Maturidade: Com que frequência apresenta falhas?	Raramente	Raramente
Tolerância a falhas: Ocorrendo falhas como ele reage?	Não gera o produto (horário)	Não gera o produto (horário)
Recuperabilidade: É capaz de recuperar dados após uma falha?	Não – Nesse caso, todos os dados devem ser realimentados	Sim

Fonte: Do autor (2015).

Conforme se verifica no quadro 3, que representa a avaliação dos protótipos em relação ao critério de confiabilidade, há atendimento das subcategorias relacionadas à maturidade, tendo em vista que ambos raramente apresentam falhas. Quando essas ocorrem, no entanto, encontram-se relacionadas à não geração do produto, que, no caso, é o horário escolar. Ainda neste caso, no que se refere à recuperabilidade, ou seja, à subcategoria de recuperar dados após a falha, observa-se que este critério é atendido pelo algoritmo genético, não sendo possível, porém, no algoritmo de *backtracking*.

6.3.1.3 Usabilidade

A usabilidade refere-se à facilidade de se utilizar o software. Nesta categoria, incluem-se as seguintes subcategorias: inteligibilidade, apreensibilidade e operacionalidade.

Quadro 4 - Comparativo entre os protótipos conforme o critério da usabilidade

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Inteligibilidade: É fácil entender os conceitos utilizados?	Sim	Sim
Apreensibilidade: É fácil aprender a usar?	Sim	Sim
Operacionalidade: É capaz de operar e controlar a operação?	Não	Sim

Fonte: Do autor (2015).

Os resultados da avaliação relacionada à categoria usabilidade, representadas no quadro 4, permitem evidenciar que nas subcategorias inteligibilidade e apreensibilidade, ambos apresentam conformidade. A subcategoria operacionalidade, no entanto, é conforme no algoritmo genético, não sendo observada no algoritmo de *backtracking*, pois em se tratando de um protótipo foram atribuídas as restrições fixas, não podendo, desse modo, informar as restrições manualmente ou cadastradas.

6.3.1.4 Eficiência

A eficiência refere-se aos recursos e os tempos utilizados e se esses são compatíveis com o nível de desempenho requerido para o produto. Nesta categoria, incluem-se as seguintes subcategorias: comportamento em relação ao tempo e comportamento em relação aos recursos.

Quadro 5 - Comparativo entre os protótipos conforme o critério da eficiência

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Comportamento em relação ao tempo: Qual é o tempo de resposta e de processamento?	**1ª amostra** 0.6720 s **2ª amostra** 1.1430 s	**1ª amostra** 0.7434 s **2ª amostra** 1.0453s
Comportamento em relação aos recursos: Quanto recurso utiliza?	Utiliza vários recursos voltado para um desenvolvimento desktop. Ex:JAVA 8, JPA.	Utiliza vários recursos voltado para um desenvolvimento WEB.Ex:JavaScrip,AngularJS.

Fonte: Do autor (2015).

Conforme se observa no Quadro 5, relacionado ao critério ou categoria da eficiência, na subcategoria relacionada ao comportamento em relação ao tempo, percebe-se que no primeiro teste, o tempo foi menor quando da utilização do algoritmo *backtracking*. O excesso de aulas, por sua vez, foi menor no protótipo utilizando o algoritmo genético.

Os reajustes foram percebidos apenas no algoritmo de *backtracking*, já que esta é sua principal característica, retroceder, reajustar e encontrar as melhores soluções. Por outro lado, o algoritmo genético não retrocede, apenas elimina os resultados menos eficientes.

Além disso, o número de janelas por professor foi o mesmo em ambos os protótipos no primeiro teste, porém menor para o algoritmo de *backtracking* no segundo teste.

É preciso destacar que ambos os protótipos apresentam pontos positivos e negativos, porém, analisando-se apenas o tempo gasto para que os resultados esperados sejam alcançados, insta destacar que o protótipo que utiliza o algoritmo de *backtracking* apresentou maior agilidade na geração de resultados no primeiro teste enquanto o protótipo genético apresentou maior agilidade no segundo teste.

Por outro lado, na subcategoria relacionada ao comportamento em relação aos recursos, a avaliação permite evidenciar que enquanto o algoritmo de *backtracking* utiliza vários recursos voltados para um desenvolvimento desktop, como por exemplo o JAVA 8 e JPA, o protótipo genético utiliza vários recursos voltado para um desenvolvimento WEB, citando-se como exemplo o JavaScrip e Angular JS.

6.3.1.5 Manutenibilidade

A manutenibilidade refere-se facilidade para correções, atualizações e alterações. Nesta categoria, incluem-se as seguintes subcategorias: analisabilidade, modificabilidade, estabilidade e testabilidade.

Quadro 6- Comparativo entre os protótipos conforme o critério da manutenibilidade

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Analisabilidade: É fácil encontrar uma falha quando ocorre?	Sim	Sim
Modificabilidade: É fácil modificar e remover defeitos?	Sim	Sim
Estabilidade: Há grandes riscos de *bugs* quando se faz alterações?	Sim	Sim
Testabilidade: É fácil testar quando se faz alterações?	Sim	Sim

Fonte: Do autor (2015).

Submetendo-se os dois protótipos à avaliação segundo a categoria da manutenibilidade, observa-se que ambos estão conformes nas quatro subcategorias relacionadas a este critério.

6.3.1.6 Portabilidade

A portabilidade refere-se à possível utilização do produto em diversas plataformas com pequeno esforço de adaptação. Nesta categoria, incluem-se as seguintes subcategorias: adaptabilidade, capacidade para ser instalado, capacidade para substituir e conformidade.

Quadro 7- Comparativo entre os protótipos conforme o critério da portabilidade

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Adaptabilidade: É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta finalidade no software considerado?	Sim	Sim
Capacidade para ser instalado: É fácil instalar em outros ambientes?	Sim	Não
Capacidade para substituir: É fácil substituir por outro software?	Não	Não

Fonte: Do autor (2015).

Em relação ao critério da portabilidade, pode-se identificar que na subcategoria relacionada à adaptabilidade, ambos os protótipos estão em conformidade. No que se refere à capacidade para se instalado, enquanto o

algoritmo de backtracking pode ser instalado em outros ambientes, no algoritmo genético não há necessidade de instalação, tendo em vista tratar-se de um programa voltado à Web. Além disso, em ambos os casos também não há facilidade de substituição por outros softwares, tendo em vista que nos dois casos, há a necessidade substituir o algoritmo, que é a base principal do programa.

7 CONCLUSÃO

O desenvolvimento de horários escolares trata-se de um problema para muitas escolas no início do ano letivo, ou sempre que se torna necessário proceder de alguma alteração nos horários estabelecidos, considerando-se que quando isso ocorre de forma manual, além do tempo envolvido ser longo, algumas restrições podem não ser consideradas e, assim, surgirá à necessidade de uma nova adaptação desses horários.

Quanto mais turmas e professores uma escola apresenta, maiores as dificuldades para o desenvolvimento desses horários, considerando-se que as opções tornam-se mais numerosas e as restrições a elas aplicáveis seguem a mesma tendência.

Diante disso, optou-se por desenvolver um estudo sobre os algoritmos de busca e satisfação de restrições, de modo a permitir uma comparação entre os algoritmos *backtracking* e genéticos. Este objetivo foi alcançado através do desenvolvimento de um protótipo geração de horário escolar que utilizou o algoritmo *backtracking*, bem como a disponibilização de uma base de dados do *software* Cronos, sendo necessário destacar que o mesmo hoje é comercializado.

Por meio de contato com um dos desenvolvedores do *software*, foram disponibilizadas algumas gerações de horários escolares, conforme os parâmetros solicitados, prontas para as comparações.

Pensando-se nos objetivos específicos desta pesquisa, estabelecidos como: compreender o funcionamento do processo de organização dos algoritmos de busca de satisfação de restrições *backtracking* e genético; identificar as etapas no processo de organização e elaboração de um horário escolar; desenvolver um protótipo com base no algoritmo *backtracking*; analisar o desempenho do algoritmo genético e algoritmo *backtracking* implementado; esclarecer as métricas de comparação existentes, apontando a mais apropriada para o protótipo desenvolvido, pode-se afirmar que todos foram cumpridos, tanto pelos estudos teóricos desenvolvidos quanto pelos testes aplicados com os protótipos citados.

Por meio da aplicação dos critérios estabelecidos pela ISO/IEC 9126 (NBR 13596), que fornece um modelo de propósito geral englobando amplas categorias de características de qualidade, identificou-se que os protótipos utilizando-se o algoritmo de *backtracking* e genético apresentam fatores qualitativos

praticamente similares em todas as categorias analisadas, ressaltando-se que no primeiro teste, o protótipo *backtracking* foi mais eficiente na subcategoria relacionada ao tempo, enquanto no segundo teste foi o algoritmo genético.

Em outras palavras, no caso de eficiência do produto como sendo uma métrica de sua qualidade, destaca-se que o protótipo desenvolvido com base no algoritmo de *backtracking* demonstrou maior eficiência quando de sua utilização no quesito tempo, que foi menor do que o protótipo baseado em algoritmo genético.

Dessa forma, acredita-se que ambos são viáveis, adequados e suficientemente qualitativos para a aplicação na organização de horários escolares com maior eficiência e menos tempo de espera pelos resultados.

Como sugestões de trabalho futuros, recomendação o envolvimento de outros usuários para avaliar a qualidade dos algoritmos, tais como professores, bem como avaliar os mesmos em relação a outros com as mesmas funções. Além disso, pode-se também sugerir o desenvolvimento de um protótipo *backtracking* para rodar na Web, de modo similar ao Cronos.

REFERÊNCIAS

ALENCAR, Wanderlei de Souza. ***Time TablingProblems*.**Disponível em <http//www.wanderley.br/dissertação/horusCapitulo01.pdf>. Acesso em 05 de Maio de 2015.

ALVES, Reginaldo Heidder de Jesus. **Metaheurísticas aplicadas ao problema de horário escolar**. Belo Horizonte, 2010. Disponível em: <http://www.files.scire.net.br/atrio/cefet-mg-ppgmmc_upl//THESIS/22/reginaldoheidderdejesusalves.pdf> Acesso em: 20 set. 2015.

AMARAL, Aurélio. 8 passos para elaborar a grade de aulas. **GESTÃO ESCOLAR,** Rio de Janeiro, Janeiro. .2013, 023.Disponível em: <http://gestaoescolar.abril.com.br/aprendizagem/planejamento-8-passos-elaborar-grade-aulas-729287.shtml >. Acesso em: 16 de nov. de 2014.

ANDRADE, Pedro Rochavetz de Lara; SCARPIN, Cassius Tadeu; STEINER, Maria Teresinha Arns. **Geração Da Grade Horária Do Curso De Engenharia De Produção Da Ufpr Através De Programação Linear Binária.** 2012. Rio de Janeiro: Brasil. Disponível em: < http://www.din.uem.br/sbpo/sbpo2012/pdf/arq0250.pdf >. Acesso em 22 de nov. de 2014.

ARAÚJO, Everton Coimbra de. **Algoritmos:** Fundamento e Prática. Florianópolis: Bookstore, 2003.

ARROTEIA, M. C. S.; ZUCCARI, P.; TOMAZ, W. L. Características e decisões de implantação da ISSO 9001:2008: estudo de caso múltiplo no centro-oeste paulista. **Revista de Administração, Contabilidade e Economia da FUNDACE**, v. 6, n. 1, p. 98-110, 2015.

CALAZANS, AngelicaToffanoSeidel; ALVARENGA, Márcia Silva de. Métricas para métodos ágeis de desenvolvimento: um estudo comparativo. **XI Simpósio de Excelência em Gestão de Tecnologia**. Out. 2014. Disponível em: < http://www.aedb.br/seget/arquivos/artigos14/20720158.pdf> Acesso em: 02 out. 2015.

CALDAS, Ruiter Braga. **Projeto e análise de algoritmos**. Belo Horizonte: UFMG, 2004. Disponível em: <http://homepages.dcc.ufmg.br/~nivio/cursos/pa04/tp2/tp22/tp22.pdf> Acesso em: 23 out. 2015.

CARDOSO Marta Pina; MARCELINO, Marcio Abud. Estudo para automação de horários escolares em uma instituição de ensino. **3º Simpósio Hipertexto e tecnologias na educação**. Anais eletrônicos. Disponível em: < https://www.ufpe.br/nehte/simposio/anais/Anais-Hipertexto-2010/Marta-Pina-Cardoso&Marcio-Abud-Marcelino.pdf> Acesso em: 26 set. 2015.

CISCON, Leonardo Aparecido et al. O problema de geração de horários: um foco na eliminação de janelas e aulas isoladas. **XXXVII SBPO – Simpósio Brasileiro de Pesquisa Operacional**. Set. 2005. Gramado – RS. Disponível em: <http://www.din.uem.br/sbpo/sbpo2005/pdf/arq0061.pdf> Acesso em: 17 set. 2015.

COELHO, Alessandra Martins; DE SOUZA, Sérgio Ricardo.**Um Algoritmo Híbrido Baseado Em Algoritmos Meméticos E Reconexão Por Caminhos Para Resolução Do Problema De Horário Escolar.** 2006. Rio de Janeiro: Brasil. Disponível em :<http://www.riopomba.ifsudestemg.edu.br/portal/sites/default/files/arq_paginas/Artigo6_AlessandraMartinsCoelho.pdf> . Acesso em 30 de nov. de 2014.

CORDENONSI, A. Z.; ARAMBURU, L. G. C.; ALMANCA, L. **Resolução do problema de quadro de horários através de um algoritmo de satisfação de restrições.** In: VIII Simpósio de Informática e III Mostra de Software Acadêmico, 2003. Rio Grande do Sul: Uruguaiana,2003. Disponível em: <http://www-usr.inf.ufsm.br/~andrezc/publicacoes/simposio_uruguaiana_2003.pdf> Acesso em: 20 set. 215.

COSTA, Artur et al. **Utilizando métricas para dimensionar um software**. Disponível em: <http://www.csi.uneb.br/engenharia_de_software/anexos/Artigo-MetricasdeSoftware.pdf> Acesso em: 05 out. 2015.

FERNANDES, C. et al. **Infected genes evolutionary algorithm for school timetabling.** WSES International Conference, 2002.

FERREIRA, José Carlos dos Santos; GLAZAR, Jean Eduardo. Definição de parâmetros na utilização de metaheurísticas para a programação de horários escolares. **Revista Educação e Tecnologia**. Ano 1, número 1, abr. set/2005. Disponível em: < http://www.faacz.com.br/revistaeletronica/links/edicoes/2005_01/edutec_jose_carlos_jean_2005_1.pdf> Acesso em: 19 set. 2015.

FILITTO, Danilo. Algoritmos genéticos: uma visão explanatória. **Saber Acadêmico**. Nº 6. Dez. 2008. Disponível em: <http://www.uniesp.edu.br/revista/revista6/pdf/13.pdf> Acesso em: 21 set. 2015.

FLÔRES, Alif Corrêa Flôres. Características dos testes associados à qualidade de *software.* SETIS- III Seminário de Tecnologia Inovação e Sustentabilidade. Disponível em: <www.setisjoinville.com/.../7d774657ef83fcf3daa8d9dccad8627f.pdf> Acesso em: 30 nov. 2015.

FREITAS, Cherze C. et al. **Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horário escolar**. 25 set. 2014. Disponível em: <http://www.researchgate.net/publication/265922481_Uma_Ferramenta_Baseada_em_Algoritmos_Genticos_para_a_Gerao_de_Tabela_de_Horrio_Escolar> Acesso em: 20 set. 2015.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine lear**ning. Addison: Wesley, 1989.

GOMES, Ana Fernanda; ARAÚJO, Graziela Santos de. **Estrutura de dados:** algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: ABDR, 2010.

HAMAWAKI,C.D.L. – **Geração Automática de Grade Horária Usando Algoritmos Genéticos: O Caso da Faculdade de Engenharia Elétrica da UFU**. 2005. 104 f. Dissertação – Pós-Graduação em Engenharia Elétrica, Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia.

ISO/IEC IS 9126-1. **Software Engineering - Product Quality – Part 1**: Quality Model. *International Organization for Standarization*, Geneva, Switzerland, 2001

KOTSKO, E. G. S. **Otimização na construção da grade horária escola – uma aplicação.** Curitiba, 2003. 66 f. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Setor de Ciências Exatas e Setor de Tecnologia, Universidade Federal do Paraná.

LEITE, Patricia Teixeira; CARNEIRO, Adriano A. F. M.; CARVALHO, André C. P. L. F.. Aplicação de algoritmos genéticos na determinação da operação ótima de sistemas hidrotérmicos de potência. **Sba Controle & Automação**. 2006, vol.17, n.1, pp. 81-88. Disponível em: <http://www.scielo.br/pdf/ca/v17n1/a08v17n1.pdf> Acesso em: 27 set. 2015.

LIMA JÚNIOR, Hélio Ferreira; CORRÊA, Marcos Vinícius. **Implementação de um sistema de alocação de professores e disciplinas em grades horárias utilizando algoritmos genéricos**. São Paulo: Universidade Anhembi Morumbi, 2010. Disponível em: <http://engenharia.anhembi.br/tcc-10/cco-07.pdf> Acesso em: 17 set. 2015.

LOBO, Eduardo Luiz Miranda. **Uma solução do problema de horário escolar via algoritmo genético paralelo.** 2005. Belo Horizonte: Minas Gerais. Disponível em: < http://www.mmc.cefetmg.br/info/downloads/D006-EduardoLuizMirandaLobo2005.pdf >. Acesso em: 10 de maio de 2015.

LUCAS, Diogo C. **Algoritmos genéticos**: uma introdução. Março 2002. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf> Acesso em: 20 set. 2015.

LUGER, G. F. **Inteligência artificial**: estruturas e estratégias para a solução de problemas complexos. 4. ed. São Paulo: Bookman, 2004.

MACHADO, Claudenir de F et al. **Geração automática de horários escolares utilizando Algoritmos de Satisfação de Restrições.** Disponível em <http://www.fatecgarca.edu.br/revista/volume4/artigos/6.pdf>. Acesso em:7 maio 2015.

MACHADO, Márcio; SOUZA, Sotério. **Métricas e qualidade de software**. Disponível em: <https://soterio.files.wordpress.com/2011/06/artigoqualidadesw.pdf> Acesso em: 27 nov. 2015.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Metodologia científica**. 5. ed. São Paulo: Atlas, 2007.

MARINHO, Leandro Balby. ***Backtracking***. Disponível em: <http://www.dsc.ufcg.edu.br/~lbmarinho/slides/atal_2011_2/*backtracking*_print.pdf> Acesso em: 30 set. 2015.

MARTINS, Jean Paulo. **O problema do agendamento semanal de aulas**. Goiânia: Universidade Federal de Goiás, 2010. Disponível em: < http://www.portal.inf.ufg.br/mestrado/sites/www.inf.ufg.br.mestrado/files/uploads/Dissertacoes/JeanPaulo.pdf> Acesso em: 18 set. 2015.

MASSIMO, Melucci. **Information retrieval**: metodi e modelli per i motori di ricerca. Milão: Franco Angeli, 2013.

MEDINA, Jonatan. **A utilização de algoritmos genéticos no desenvolvimento de jogos**. Disponível em: <periodicos.uems.br> Acesso em: 26 set. 2015.

MEIRELLES, Paulo Roberto Miranda.**Levantamento de métricas de avaliação de projetos de software livre**. São Paulo: USP, 2008. Disponível em: < http://ccsl.ime.usp.br/files/relatorioPauloMeirelles_final.pdf> Acesso em: 12 out. 2015.

MELO, Rodrigo Meurer. **Definição e implementação de uma função de avaliação para um sistema de geração de grade horária que utiliza como método de busca algoritmo genético.** Santa Catarina: 2003. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Extremo Sul Catarinense.

MICROSOFT. **Diagramas de caso de uso UML:** diretrizes. 2013.Disponível em: <>. Acesso em 22 de out. de 2014.

MIGUEL, José P. Miguel, MAURICIO, David Mauricio; RODRÍGUEZ, Glean. A Review of Software Quality Models for the Evaluation of Software Products. **International Journal of Software Engineering & Applications (IJSEA)**, vol.5, No.6, November 2014

MIRANDA, Everton Alves. Uma análise de métricas de software orientadas à função e sua aplicação ao desenvolvimento orientado a objetos. **Vértices**. Jan. 2002, ano 4, n 1. Disponível em: < http://essentiaeditora.iff.edu.br/index.php/vertices/article/view/1809-2667.20020007/138> Acesso em: 05 out. 2015.

MONTEIRO, T. C. Pontos de caso de uso técnicos (TUCP): uma extensão da UCP, 2005.

MOORI, Roberto Giro Moori; KIMURA, Herbert; ASAKURA, Oscar Kenjiro. Aplicação do algoritmo genético na gestão de suprimentos**. Revista de Administração e Inovação**, São Paulo, v. 7 , n. 2, p. 171-192, abr./jun. 2010. Disponível em: <http://www.revistarai.org/rai/article/view/328/296> Acesso em: 21 set. 2015.

MOURA, Arnaldo et al. Técnicas metaheuristicas aplicadas à construção de grades horárias escolares. **XXXVI SBPO -Simpósio Brasileiro de Pesquisa Operacional.** Nov. 2004. São João Del Tey – MG. Disponível em: <http://www.decom.ufop.br/marcone/Disciplinas/InteligenciaComputacional/Timetabling.pdf> Acesso em: 18 set. 2015.

OLIVEIRA, José Auriço; PINHEIRO, Plácido Rogério; **Ambiente de otimização na Web:Uma aplicação em TimeTabling.** 2003. Natal: Brasil. Disponível em: <www.din.uem.br/sbpo/sbpo2003/pdf/arq0174.pdf>. Acesso em 14 de nov. de 2014.

PALADINI, Edson Pacheco. **Avaliação estratégica da qualidade**. 2. ed. São Paulo : Atlas, 2011.

PINHO, Alexandre Ferreira de; MONTEVECHI, José Arnaldo Barra. MARTINS, Fernando Augusto Silva. Análise da aplicação de projeto de experimentos nos parâmetros dos algoritmos genéticos. **Sistemas e Gestão**, v. 2, n. 3, p. 314-325, setembro a dezembro de 2007. Disponível em: <http://www.revistasg.uff.br/index.php/sg/article/viewFile/SGV2N3A9/46> Acesso em: 19 set. 2015.

PIRES, Alexsandro Santos. **Implementação de um algoritmo heurístico para problemas de restrição**. Blumenau: Universidade Regional de Blumenau, 2006. Disponível em: <http://campeche.inf.furb.br/tccs/2006-II/2006-2alexsandrospiresvf.pdf> Acesso em: 25 set. 2015.

POTROS, Jonas. **Algoritmos tentativa e erro (*backtracking*)**. Disponível em: <http://www.riopomba.ifsudestemg.edu.br/dcc/dcc/materiais/1919377333_Aula%206%20-%20%20*Backtracking*.pdf> Acesso em: 01 out. 2015.

PRESSMAN, Roger S. **Engenharia de software**. 6. ed. São Paulo: McGraw-Hill, 2006.

ROBAINA, Diogo Tavares et al. Logística X Computação: uso de um algoritmo Brand-and-boud na redução de custos logísticos. **XI Simpósio de Excelência em Gestão e Tecnologia**. Out/2014. Disponível em: <http://www.aedb.br/seget/arquivos/artigos14/32820514.pdf> Acesso em: 23 out. 2015.

ROSA, Thatiane de Oliveira; LUZ, Hellen Souza. Conceitos Básicos de Algoritmos Genéticos: Teoria e Prática. In: XI Encontro de Estudantes de Informática do Tocantins, 2009, Palmas. **Anais do XI Encontro de Estudantes de Informática do Tocantins**. Palmas: Centro Universitário Luterano de Palmas, 2009. p. 27-37. Disponível em: <http://tinyurl.com/ylouf6> Acesso em: 20 set. 2015.

RUK, Denivy Briam; OLIVEIRA, Ramom de; KOSLOVSKI, Guilherme Piegas. Comparação de algoritmos para alocação de infraestruturas virtuais**. Revista Brasileira de Computação Aplicada**. Passo Fundo, v. 6, n. 2, p. 98-112, out. 2014.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Elsevier, 2004.

_____. **Inteligência Artificial**. 3º Edição. Rio de Janeiro: Elsevier, 2013.

SANTOS, Jalila Rios dos. **AST um modelo para automação de horários escolares**. Recife: Universidade Federal de Pernambuco, 2008. Disponível em: <http://repositorio.ufpe.br/xmlui/bitstream/handle/123456789/6958/arquivo1642_1.pdf?sequence=1&isAllowed=y> Acesso em: 19 set. 2015.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, **S. Sistema de banco de dados**. Rio de Janeiro: Elsevier, 2006. 781 p.

SIMÃO, Thiago Dias. **Utilização de algoritmos genéticos para otimização de soluções para o timetabling escolar**. Lavras: Universidade Federal de Lavras, 2013. Disponível em: < http://www.bsi.ufla.br/wp-content/uploads/2014/09/Thiago-Dias_UTILIZA%C3%87%C3%83O-DE-ALGORITMOS-GEN%C3%89TICOS-PARA-OTIMIZA%C3%87%C3%83O-DE-SOLU%C3%87%C3%95ES-PARA-O-TIMETABLING-ESCOLAR-THIAGO-DIAS-SIM%C3%83O_1310.pdf> Acesso em: 18 set. 2015.

SOUZA, Bruna Thabata Ribeiro de. **Utilização de algoritmo particleswarmoptimization para resolver o problema de timetabling na elaboração da grade de horários em um curso superior**. Palmas: Centro Universitário Luterano de Palmas, 2013. Disponível em: <file:///C:/Users/Claudia/Downloads/document52f8e59ea64c7.pdf> Acesso em: 18 set. 2015.

SURYN, Witold et al. Software Product Quality Practices Quality Measurement and Evaluation using TL9000 and ISO/IEC 9126. **Software Technology and Engineering Practice (STEP)**, Montreal, Canada, October 6-8, 2003.

TERRA, Ivone Piedade; RADAELLI, Joyce Lopes. Utilização dos métodos de otimização em problemas de timetabling. **PRINCIPIUM ONLINE: Iniciação Científica no Unileste-MG**, Coronel Fabriciano, v. 1, n. 1, p. 95-103, jul. 2007. Disponível em: <http://www.unilestemg.br/principiumonline/publicacoes/01/downloads/095_103_utilizacao_dos_metodos_de_otimizacao_em_problemas.pdf> Acesso em: 17 set. 2015.

TOLEDO, José Carlos de et al. Qualidade: gestão e métodos. Rio de Janeiro: LTC, 2013.

TORRES-OVALLE, Camilo et al. University course scheduling and classroom assignment. **Ing. Univ**. [online]. 2014, vol.18, n.1, pp. 59-75. Disponível em: <http://www.scielo.org.co/pdf/inun/v18n1/v18n1a04.pdf> Acesso em: 20 set. 2015.

TRODO, Lia Degrazia. **Uso de métricas nos testes de software**. Universidade Federal do Rio Grande do Sul. Porto Alegre, 2009. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/18574/000730980.pdf?...1> Acesso em: 28 nov. 2015.

VIEIRA; Andrws Aires. **Uma abordagem para a estimação prévia dos requisitos não funcionais em sistemas embarcados utilizando métricas de software**. Porto Alegre: UFRGS, 2015.

WREN, A. **Scheduling, timetabling and rostering – a special relationship?** . 1st Internacional Conference, Lecture Notes in Computer Science. Berlin, 1996, notas.

XAVIER, Bruno Missi. SILVA, Alcione Dias da. COSTA, Helder Gomes. Uma perspectiva dos problemas de alocação de horários sob a ótica da bibliometria. **Perspectivas on-line**. Campos dos Goytacazes. 4 (8) 1-14, 2014. Disponível em: < www.seer.perspectivasonline.com.br/index.php/exatas_e.../80/436> Acesso em: 16 set. 2015.

ZADRA, Augusto Nogueira; CARVALHO, Erivelton Oliveira; CARDOSO, Joécio Farley Santos. **Métricas de software**: Comparação entre Pontos de Função e Cocomo. Revista Pensar Tecnologia, v.4, n.2, jul. 2015.

APÊNDICE(S)

APÊNDICE A – ARTIGO CIENTÍFICO

Análise Comparativa de um Algoritmo de Busca e Satisfação de Restrições e Algoritmo Genético em um Sistema para Geração de Horário Escolar

Tágner Formanski Rosa[1], Luciano Antunes[2]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC) Av. Universitária, 1105 - Bairro Universitário – Criciúma – SC – Brasil

[2]Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC) Av. Universitária, 1105 - Bairro Universitário – Criciúma – SC – Brasil

tagner_@hotmail.com, luc@unesc.net

Abstract*. The objective of this study was to obtain a comparative analysis of performance between search algorithms and satisfaction of backtracking and genetic constraints on a school day prototype. For this, we used the Kronos software (genetic algorithm) and developed a prototype of the school timetable generation, using the backtracking algorithm. The metric or model assessment of quality was adopted ISO / IEC 9126 - Part 1 (NBR 13596), which provides a general purpose model. This standard defines six broad categories of software quality features are: functionality, reliability, usability, efficiency, maintainability and portability. These categories, in turn, are divided into subcharacteristics. Through the application of the criteria, it identified that the prototypes using the backtracking and genetic algorithm have almost similar qualitative factors in all categories analyzed, highlighting that in the first test, the backtracking prototype was more efficient in the subcategory related to the time while the second test was the genetic algorithm. Thus, it is believed that they are viable and sufficiently qualitative suitable for application to the organization school schedules with greater efficiency and less time waiting for results.*

Resumo. *O objetivo deste estudo foi obter uma análise comparativa de desempenho entre os algoritmos de busca e satisfação de restrições backtracking e genético em um protótipo de horário escolar. Para tanto, utilizou-se o software Cronos (algoritmo genético) e desenvolveu-se um protótipo de geração de horário escolar, utilizando o algoritmo backtracking. A métrica ou modelo de avaliação da qualidade adotada foi a ISO/IEC 9126 – Parte 1 (NBR 13596), que fornece um modelo de propósito geral. Esta norma define seis amplas categorias de características de qualidade de software que são: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Essas categorias,*

por sua vez, são divididas em subcaracterísticas. Por meio da aplicação dos critérios estabelecidos, identificou-se que os protótipos utilizando-se o algoritmo de backtracking e genético apresentam fatores qualitativos praticamente similares em todas as categorias analisadas, ressaltando-se que no primeiro teste, o protótipo backtracking foi mais eficiente na subcategoria relacionada ao tempo, enquanto no segundo teste foi o algoritmo genético. Dessa forma, acredita-se que ambos são viáveis, adequados e suficientemente qualitativos para a aplicação na organização de horários escolares com maior eficiência e menos tempo de espera pelos resultados.

1. Introdução

A formulação de quadros de horários é uma tarefa necessária e inevitável no cotidiano de qualquer instituição de ensino. De maneira geral, o processo consiste em gerar uma tabela associando professores, turmas e disciplinas em determinados horários do dia com suas respectivas restrições. Para que um horário seja considerado viável, é preciso respeitar as restrições de disponibilidade dos professores e atender suas preferências.

À medida que o número de turmas aumenta e mais variáveis são levadas em consideração, o processo se transforma em uma incógnita, ou até mesmo em um problema difícil de ser resolvido manualmente, devido ao seu caráter combinatório. Essa dificuldade motiva a criação de ferramentas computacionais que permitam gerar automaticamente quadros de horários de maior qualidade, satisfazendo as restrições necessárias para cada professor, e podendo esta tarefa de forma mais rápida do que o processo manual.

Problemas como este descrito acima, e também que envolvem Inteligência Artificial (IA), podem ser vistos como problemas de satisfação de restrições, na qual o objetivo é descobrir se existe algum estado de problema que satisfaça a um determinado conjunto de restrições.

Segundo Cordenonsi, Aramburu e Almanca (2003), as implementações que utilizam restrições, tratam problemas combinatoriais com o objetivo de restringir o espaço de busca, diminuindo o gasto de memória e de tempo de processamento, conseguindo com isso maior eficiência. O uso do algoritmo de satisfação de restrições foi proposto para diminuir o espaço de busca na execução dos programas. A execução utiliza um mecanismo de controle que elimina os caminhos inválidos, isto é, caminhos que provavelmente não levem ao resultado final, deste modo, minimiza-se o processamento desnecessário e o retrocesso.

O algoritmo de busca pode ser compreendido como aquele que toma em

consideração um problema, sendo este sua entrada, dando origem a uma solução, buscando sempre aquela que melhor atende as demandas estabelecidas. É aplicado sobre problemas complexos e cujo resolução não se dá pela aplicação de técnicas de programação convencionais, principalmente as de natureza puramente numérica (RUSSEL; NORVING, 2004).

O algoritmo de satisfação de restrições encampa um problema representado por variáveis e pelas restrições que sobre elas incidem. Para Cordenonsi, Aramburu e Almanca (2003) alguns problemas da Inteligência Artificial (IA) podem ser compreendidos como problemas de satisfação de restrições, visando encontrar uma solução que atenda as restrições existentes.

Por algoritmos de *backtracking* destacam-se aqueles que atuam com tentativa e erro, também a chamada busca com retrocesso, decompondo os processos em tarefas e subtarefas limitadas que são exploradas de forma exaustiva (POTROS, 2015).

Os algoritmos genéticos, por sua vez, buscam uma solução adequada ao problema estabelecido, partindo de uma população inicial da qual são selecionados os melhores representantes para que uma nova população passe a existir. Com isso, a cada substituição surgem soluções mais adequadas (FILITTO, 2008).

No presente trabalho procedeu-se uma comparação dos algoritmos de backtracking e genéticos, com vistas a compreender as diferenças e semelhanças entre ambos, além de identificar aspectos sobre a eficiência, usabilidade, qualidade, tempo de execução, e, diante dessas informações, verificar qual o algoritmo que melhor atende às demandas de construção do horário escolar, em uma situação genérica.

2. AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A NORMA ISO/IEC 2026

A ISO/IEC 9126 propõe um conjunto subfuncionalidades de características para avaliar atributos de um produto de software em relação a sua qualidade (MIGUEL; MAURICIO; RODRÍGUEZ, 2014).

Essas entidades trabalham em parceria com outras organizações internacionais, governamentais e privadas, formando comitês técnicos, através do qual desenvolveram revisões da norma ISO/IEC 9126 desde sua criação (SURYN et al, 2003).

Uma das características da ISO/IEC 9126 é padronizar a terminologia em relação à qualidade do software (MIGUEL; MAURICIO; RODRÍGUEZ, 2014).

A ISO/IEC 9126 refere-se uma norma que é integrada por um conjunto de

atributos ou características que devem ser verificadas em um software, de forma que ele seja considerado como um "software de qualidade" (SURYN et al, 2003).

A ISO/IEC 9126 faz distinção entre qualidade interna e qualidade externa. A qualidade interna de software é medida objetivamente por fatores mensuráveis durante o desenvolvimento. A qualidade externa, por sua vez, tem como objetivo medir a qualidade do software levando em consideração o comportamento do software em um sistema do qual ele faz parte (ABNT, 2003).

Na primeira parte do modelo, são especificadas seis características para avaliação da qualidade interna e externa: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, conforme se verifica na Figura 1.

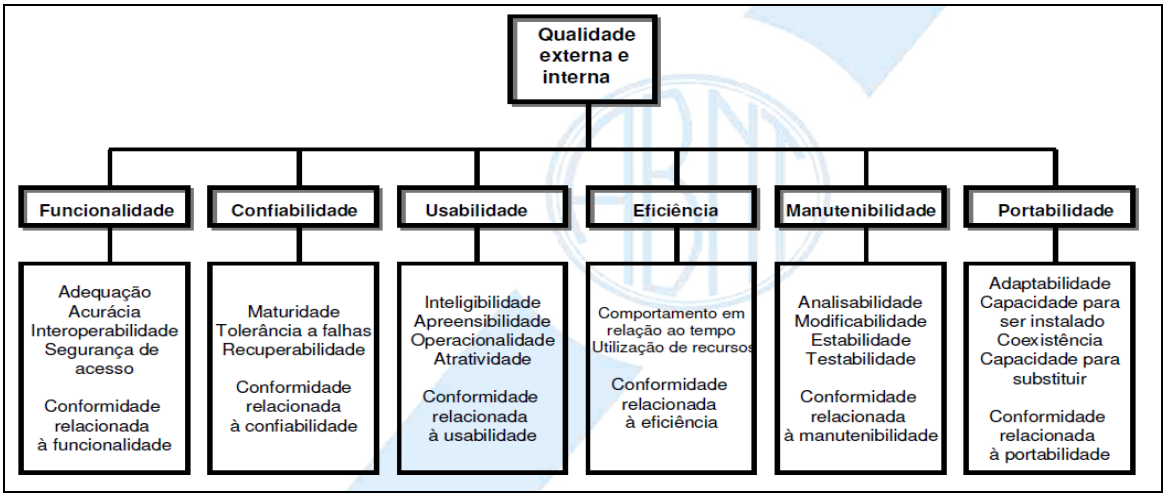


Figura 1 - Modelo de qualidade para qualidade externa e interna

Tais características ou categorias, por sua vez, desdobram-se em subcategorias, que durante o processo de avaliação suas respectivas perguntas-chave devem ser respondidas, conforme representado no Quadro 1.

Funcionalidade	**Adequação**: Propõe-se a fazer o que é apropriado? **Acurácia**: Gera resultados corretos ou conforme acordados? **Interoperabilidade**: É capaz de interagir com os sistemas especificados? **Segurança de acesso**: Evita acesso não autorizado, acidental ou deliberado a programas e dados? **Conformidade**: Está de acordo com normas e convenções previstas em leis e descrições similares?
Confiabilidade	**Maturidade**: Com que frequência apresenta falhas? **Tolerância a falhas**: Ocorrendo falhas como ele reage? **Recuperabilidade**: É capaz de recuperar dados após uma falha?
Usabilidade	**Inteligibilidade**: É fácil entender os conceitos utilizados? **Apreensibilidade**: É fácil aprender a usar? **Operacionalidade**: É capaz de operar e controlar a operação?
Eficiência	**Comportamento em relação ao tempo**: Qual é o tempo de resposta e de

	processamento? **Comportamento em relação aos recursos**: Quanto recurso utiliza?
Manutenibilidade	**Analisabilidade**: É fácil encontrar uma falha quando ocorre? **Modificabilidade**: É fácil modificar e remover defeitos? **Estabilidade**: Há grandes riscos de *bugs* quando se faz alterações? **Testabilidade**: É fácil testar quando se faz alterações?
Portabilidade	**Adaptabilidade**: É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta finalidade no software considerado? **Capacidade para ser instalado**: É fácil instalar em outros ambientes? **Capacidade para substituir**: É fácil substituir por outro software?

Quadro 1 - Características da qualidade do software segundo a ISO/IEC 9126/1

Estas métricas oferecem a possibilidade de medir a qualidade dos artefatos intermediários e de prever a qualidade do produto final. Isto permite que sejam identificados problemas de qualidade e se inicie a ação corretiva assim que possível no ciclo de vida do desenvolvimento (SURYN et al, 2003).

3. PROTÓTIPO GHE – GERAÇÃO DE HORÁRIO ESCOLAR

A pesquisa tecnológica desenvolvida teve por objetivo obter uma análise comparativa de desempenho entre dois algoritmos de busca de horário escolar.

Para tanto, utilizou-se para análise comparativa, o *software* Cronos, que se trata de software web desenvolvido por ex-alunos da Universidade Federal de Lavras – UFLA, fruto de um trabalho da disciplina de Inteligência Artificial do curso de Ciência da Computação da mencionada instituição no ano de 2005 e lançado em 2009 comercialmente. Tal sistema é baseado em um algoritmo que gera horários escolares de acordo com as necessidades e restrições das instituições de ensino.

Desse modo, por meio do *software* Cronos é possível montar um quadro de horários para os professores e turmas de maneira automatizada, considerando as principais restrições existentes na elaboração do mesmo. Através de contato via e-mail com a equipe de desenvolvimento, foi possível compreender que o *software* utiliza-se do algoritmo genético como base.

Para a análise comparativa, procedeu-se a formulação de um programa computadorizado que utiliza o algoritmo *backtracking* como base.

Visando-se analisar ambos algoritmos utilizou-se a métrica de qualidade do produto, segundo a ISO/IEC 9126. Essa série subdivide-se em três normas e fornece características (parte 1) e métricas (parte 2 e 3) para avaliação do software. Neste trabalho foram utilizadas as métricas da parte 1, cuja distribuição ocorre em seis características principais, com cada uma delas divididas em sub-características.

Utilizou-se no desenvolvimento do protótipo a linguagem Java 8, no ambiente de desenvolvimento NetBeans, versão 8.0.2. O SGBD utilizado foi o Sybase 9.0 e a versão do Hibernate foi a 4.3.

Com o decorrer do desenvolvimento, percebeu-se a necessidade de utilizar um *software* para abrir ou gerenciar o documento gerado pelo protótipo. Assim optou-se gerar em extensão .pdf, no qual pode ser aberto pela ferramenta disponibilizada gratuitamente na internet, *Adobe Reader* para qualquer versão.

Para realizar a comunicação entre o banco de dados e aplicação, utilizou-se da biblioteca do Sybase, sendo ela: com.sybase.jdbc3.jdbc.SybDriver. Para geração de relatório foi utilizado à biblioteca itextpdf-5.5.1. Com a utilização da API *OpenSource iText*, tenho a possibilidade de manipular e criar documentos em formato PDF, além de XML e RTF. Devido seu código ser aberto, essa API (biblioteca) tem distribuição sob as licenças LGPL e MPL, proporcionando assim seu uso em diversos sistemas.

A interface do protótipo foi projetada e modelada através da própria IDE *Netbeans*, foram utilizados os componentes do próprio e também foi importado outra biblioteca, sendo ela *SwingX*, que é uma API, que originou-se de um subprojeto do *SwingLabs*, que conta com suporte da *Sun Microsystems*.

Com a integração da API ao *Netbeans* pode-se obter outros componentes para o desenvolvimento da interface. Foram utilizados componentes como: *JTextField ,JButton*, *JTable,JPanel,*dentre outros.

Após a organização e condução dos testes de implementação do protótipo de GHE, diferentes foram os resultados obtidos pelo trabalho. Foi realizada1 (uma) execução do processo de geração escolar englobando as 4 (quatro) turmas com as respectivas séries, professores e disciplinas cadastradas.

Na sequência, procedeu-se de uma comparação com outro teste baseado em uma base de dados do software Cronos que possui a utilização do algoritmo genético. A implementação do algoritmo se deu em JAVA™, enquanto a máquina utilizada foi em um computador desktop com sistema operacional Windows 7, 4 GB de memória RAM, processador Dell Core i7.

4. RESULTADOS OBTIDOS

Após os testes com o protótipo e a apresentação dos resultados, procedeu-se de uma comparação entre os dois protótipos (algoritmos *backtracking* e algoritmo genético).

A métrica ou modelo de avaliação da qualidade ou dos critérios qualitativos

adotada foi a ISO/IEC 9126 (NBR 13596), que fornece um modelo de propósito geral, o qual define seis amplas categorias de características de qualidade de software que são, por sua vez, subdivididas em subcaracterísticas.

Para a maior clareza e organização da comparação, apresenta-se os quadros a seguir, com o resumo comparativo dos dados dos dois protótipos considerados na avaliação em cada categoria e suas respectivas subcategorias.

4.1 Funcionalidade

A funcionalidade é o conjunto de funções satisfazem as necessidades explícitas e implícitas para a finalidade a que se destina o produto. Nesta categoria, incluem-se as seguintes subcategorias: adequação, acurácia, interoperabilidade, segurança do acesso, e conformidade. No Quadro 2, é representado o comparativo entre ambos os protótipos analisados segundo este critério.

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Adequação: Propõe-se a fazer o que é apropriado?	Sim	Sim
Acurácia: Gera resultados corretos ou conforme acordados?	Sim	Sim
Interoperabilidade: É capaz de interagir com os sistemas especificados?	Sim	Sim
Segurança de acesso: Evita acesso não autorizado, acidental ou deliberado a programas e dados?	Não	Sim
Conformidade: Está de acordo com normas e convenções previstas em leis e descrições similares?	Não	Sim

Quadro 2 - Comparativo entre os protótipos conforme o critério da funcionalidade

A análise dos protótipos segundo a categoria da funcionalidade em suas respectivas quatro subcategorias ou métricas permite evidenciar que o algoritmo genético atende todas as perguntas-chaves da ISO/IEC 9126. Já algoritmo de *backtracking*, por sua vez, encontra-se em conformidade em praticamente todos os requisitos da categoria, exceto nas subcategorias relacionadas à segurança do acesso e conformidade. No critério segurança de acesso, frente ao fato da implementação do algoritmo ter sido em um banco de dados poderá haver *hackers* sequestrando informações, enquanto o genético fornece maior padrão de segurança em relação aos seus dados por rodar na Web. Já no critério relacionado à conformidade, o algoritmo *backtraking* não está conforme segundo as leis por se tratar de um protótipo e ser utilizado somente para a análise comparativa.

4.2 Confiabilidade

A confiabilidade é o desempenho mantido ao longo do tempo e em condições estabelecidas. Nesta categoria, incluem-se as seguintes subcategorias: adequação, acurácia, Interoperabilidade, segurança do acesso, e conformidade.

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Maturidade: Com que frequência apresenta falhas?	Raramente	Raramente
Tolerância a falhas: Ocorrendo falhas como ele reage?	Não gera o produto (horário)	Não gera o produto (horário)
Recuperabilidade: É capaz de recuperar dados após uma falha?	Não – Nesse caso, todos os dados devem ser realimentados	Sim

Quadro 3- Comparativo entre os protótipos conforme o critério da confiabilidade

Conforme se verifica no quadro 3, que representa a avaliação dos protótipos em relação ao critério de confiabilidade, há atendimento das subcategorias relacionadas à maturidade, tendo em vista que ambos raramente apresentam falhas. Quando essas ocorrem, no entanto, encontram-se relacionadas à não geração do produto, que, no caso, é o horário escolar. Ainda neste caso, no que se refere à recuperabilidade, ou seja, à subcategoria de recuperar dados após a falha, observa-se que este critério é atendido pelo algoritmo genético, não sendo possível, porém, no algoritmo de *backtracking*.

4.3 Usabilidade

A usabilidade refere-se à facilidade de se utilizar o software, incluem-se as subcategorias de inteligibilidade, apreensibilidade e operacionalidade.

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Inteligibilidade: É fácil entender os conceitos utilizados?	Sim	Sim
Apreensibilidade: É fácil aprender a usar?	Sim	Sim
Operacionalidade: É capaz de operar e controlar a operação?	Não	Sim

Quadro 4 - Comparativo entre os protótipos conforme o critério da usabilidade

Os resultados da avaliação relacionada à categoria usabilidade, representadas no quadro 4, permitem evidenciar que nas subcategorias inteligibilidade e apreensibilidade, ambos apresentam conformidade. A subcategoria operacionalidade, no entanto, é conforme no algoritmo genético, não sendo observada no algoritmo de *backtracking*, pois em se tratando de um protótipo foram atribuídas as restrições fixas, não podendo, desse modo, informar as

restrições manualmente ou cadastradas.

4.4 Eficiência

A eficiência refere-se aos recursos e os tempos utilizados e se esses são compatíveis com o nível de desempenho requerido para o produto. Nesta categoria, incluem-se as seguintes subcategorias: comportamento em relação ao tempo e comportamento em relação aos recursos.

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Comportamento em relação ao tempo: Qual é o tempo de resposta e de processamento?	**1ª amostra** 0.6720 s **2ª amostra** 1.1430 s	**1ª amostra** 0.7434 s **2ª amostra** 1.0453s
Comportamento em relação aos recursos: Quanto recurso utiliza?	Utiliza vários recursos voltado para um desenvolvimento desktop. Ex:JAVA 8, JPA.	Utiliza vários recursos voltado para um desenvolvimento WEB.Ex:JavaScrip,AngularJS.

Quadro 5 - Comparativo entre os protótipos conforme o critério da eficiência

Conforme se observa no Quadro 5, na subcategoria relacionada ao comportamente em relação ao tempo, percebe-se que no primeiro teste, o tempo foi menor quando da utilização do algoritmo *backtracking*. O excesso de aulas, por sua vez, foi menor no protótipo utilizando o algoritmo genético. Os reajustes foram percebidos apenas no algoritmo de *backtracking*, já que esta é sua principal característica, retroceder, reajustar e encontrar as melhores soluções. Por outro lado, o algoritmo genético não retrocede, apenas elimina os resultados menos eficientes.

Além disso, o número de janelas por professor foi o mesmo em ambos os protótipos no primeiro teste, porém menor para o algoritmo de *backtracking* no segundo teste.

É preciso destacar que ambos os protótipos apresentam pontos positivos e negativos, porém, analisando-se apenas o tempo gasto para que os resultados esperados sejam alcançados, insta destacar que o protótipo que utiliza o algoritmo de *backtracking* apresentou maior agilidade na geração de resultados no primeiro teste enquanto o protótipo genético apresentou maior agilidade no segundo teste. Por outro lado, na subcategoria relacionada ao comportamento em relação aos recursos, a avaliação permite evidenciar que enquanto o algoritmo de *backtracking* utiliza vários recursos voltados para um desenvolvimento desktop, como por exemplo o JAVA 8 e JPA, o protótipo genético utiliza vários recursos voltado para um desenvolvimento WEB, citando-se exemplificadamente JavaScrip e Angular JS.

4.5 Manutenibilidade

A manutenibilidade refere-se facilidade para correções, atualizações e alterações. Nesta categoria, incluem-se as seguintes subcategorias: analisabilidade, modificabilidade, estabilidade e testabilidade.

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Analisabilidade: É fácil encontrar uma falha quando ocorre?	Sim	Sim
Modificabilidade: É fácil modificar e remover defeitos?	Sim	Sim
Estabilidade: Há grandes riscos de *bugs* quando se faz alterações?	Sim	Sim
Testabilidade: É fácil testar quando se faz alterações?	Sim	Sim

Quadro 6- Comparativo entre os protótipos conforme o critério da manutenibilidade

Submetendo-se os dois protótipos à avaliação segundo a categoria da manutenibilidade, observa-se que ambos estão conformes nas quatro subcategorias relacionadas a este critério.

4.6 Portabilidade

A portabilidade refere-se à possível utilização do produto em diversas plataformas com pequeno esforço de adaptação. Nesta categoria, incluem-se as seguintes subcategorias: adaptabilidade, capacidade para ser instalado, capacidade para substituir e conformidade.

MÉTRICAS	***BACKTRACKING***	**GENÉTICO**
Adaptabilidade: É fácil adaptar a outros ambientes sem aplicar outras ações ou meios além dos fornecidos para esta finalidade no software considerado?	Sim	Sim
Capacidade para ser instalado: É fácil instalar em outros ambientes?	Sim	Não
Capacidade para substituir: É fácil substituir por outro software?	Não	Não

Quadro 7- Comparativo entre os protótipos conforme o critério da portabilidade

Em relação ao critério da portabilidade, pode-se identificar que na subcategoria relacionada à adaptabilidade, ambos os protótipos estão em conformidade. No que se refere à capacidade para se instalado, enquanto o algoritmo de backtracking pode ser instalado em outros ambientes, no algoritmo genético não há necessidade de instalação, tendo em vista

tratar-se de um programa voltado à Web. Além disso, em ambos os casos também não há facilidade de substituição por outros softwares, tendo em vista que nos dois casos, há a necessidade substituir o algoritmo, que é a base principal do programa.

5. CONCLUSÃO

Por meio da aplicação dos critérios estabelecidos pela ISO/IEC 9126 (NBR 13596), que fornece um modelo de propósito geral englobando amplas categorias de características de qualidade, identificou-se que os protótipos utilizando-se o algoritmo de *backtracking* e genético apresentam fatores qualitativos praticamente similares em todas as categorias analisadas, ressaltando-se que no primeiro teste, o protótipo *backtracking* foi mais eficiente na subcategoria relacionada ao tempo, enquanto no segundo teste foi o algoritmo genético.

Em outras palavras, no caso de eficiência do produto como sendo uma métrica de sua qualidade, destaca-se que o protótipo desenvolvido com base no algoritmo de *backtracking* demonstrou maior eficiência quando de sua utilização no quesito tempo, que foi menor do que o protótipo baseado em algoritmo genético.

Dessa forma, acredita-se que ambos são viáveis, adequados e suficientemente qualitativos para a aplicação na organização de horários escolares com maior eficiência e menos tempo de espera pelos resultados.

Como sugestões de trabalho futuros, recomendação o envolvimento de outros usuários para avaliar a qualidade dos algoritmos, tais como professores, bem como avaliar os mesmos em relação a outros com as mesmas funções. Além disso, pode-se também sugerir o desenvolvimento de um protótipo *backtracking* para rodar na Web, de modo similar ao Cronos.

References

ABNT. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Norma ISO/IEC 9126** – Parte I. 2003.

CORDENONSI, A. Z.; ARAMBURU, L. G. C.; ALMANCA, L. **Resolução do problema de quadro de horários através de um algoritmo de satisfação de restrições.** In: VIII Simpósio de Informática e III Mostra de Software Acadêmico, 2003. Rio Grande do Sul: Uruguaiana,2003. Disponível em: <http://www-usr.inf.ufsm.br/~andrezc/publicacoes/simposio_uruguaiana_2003.pdf> Acesso em: 20 set. 215.

FILITTO, Danilo. Algoritmos genéticos: uma visão explanatória. **Saber Acadêmico**. Nº 6. Dez. 2008. Disponível em: <http://www.uniesp.edu.br/revista/revista6/pdf/13.pdf> Acesso em: 21 set. 2015.

MIGUEL, José P. Miguel, MAURICIO, David Mauricio; RODRÍGUEZ, Glean. A Review of Software Quality Models for the Evaluation of Software Products. **International Journal of Software Engineering & Applications (IJSEA)**, vol.5, No.6, November 2014

POTROS, Jonas. **Algoritmos tentativa e erro (*backtracking*)**. Disponível em: <http://www.riopomba.ifsudestemg.edu.br/dcc/dcc/materiais/1919377333_Aula%206%20-%20%20*Backtracking*.pdf> Acesso em: 01 out. 2015.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Elsevier, 2004.

SURYN, Witold et al. Software Product Quality Practices Quality Measurement and Evaluation using TL9000 and ISO/IEC 9126. **Software Technology and Engineering Practice (STEP)**, Montreal, Canada, October 6-8, 2003.