1 REFERENCIAL TEÓRICO

1.1 TimeTable

Falar sobre PAS 1- Problema de Programação de Horários em Escolas (School Timetabling Problem) ou Problema Professor-Turma (Class-Teacher): 2 - O Problema de Alocação de Horários (Course Timetabling Problem) ou Programação de Horários em Universidade (University Timetabling)

O Problema de Programação de Horários pode ser, por sua vez, classificados em dois tipos, conforme Souza (2000), dependendo de necessitarem otimizar ou não uma função objetivo: 1-Problemas de otimização: neste tipo encontram-se os problemas nos quais, dentre todos os quadros de horários viáveis (satisfazem certo conjunto de restrições essenciais), deseja-se encontrar um quadro, chamado ótimo que minimize uma função objetivo, a qual incorpora as restrições ditas não-essenciais. 2-Problemas de viabilidade: esta classificação refere-se a todos os problemas nos quais se requer encontrar um quadro de horário viável, ou seja, um quadro de horário que satisfaça a todas as restrições impostas.

3- Problema de programação de horários de exames (Examination time-timebling problem).

Conforme Souza (2000) cita Even (Even et. Al. 1976), o problema de geração de horários é NP-Completo em sua versão de decisão. A intratabilidade pode ser mostrada pela redução do problema de coloração de grafos em um problema de geração de horários.

Conforme Souza (2000) a maioria das técnicas antigas usadas na automação do Problema de Geração de Horários era baseada no uso de heurísticas construtivas que, de um modo geral, adotavam um preenchimento gradual do quadro de horários. Ainda conforme Souza (2000), no segundo momento, os pesquisadores começaram a usar métodos gerais para resolver o problema, tais como coloração de grafos, programação inteira e fluxo em grafo. Mais recentemente, apareceram soluções baseadas em novas técnicas de pesquisa, como Recozimento Simulado, Algoritmos Genéticos, Busca Tabu, Satisfação de Restrições e inclusive a combinação destas técnicas.

Conforme Silva (2005) cita Bardadym (1996), o Problema de Alocação de Salas pode ser tratado como um Problema de Geração de Horários, mais precisamente como um Problema de Programação de Cursos Universitários (Course Timetabling), ou como um problema derivado deste (Classroom Assigment). Na variante Classroom Assigment considera-se que as aulas dos cursos já têm seus horários de início e de término definidos sendo que o problema se resume a alocar as aulas às salas respeitando os horários destas aulas e outras restrições. O PAS é um exemplo de problema de Otimização Combinatória e por se tratar de um problema NP-Difícil tem sido tratado, atualmente, por técnicas heurísticas e meta- heurísticas.

Complexidade do problema

O problema de programação de horários é NP-Difícil, conforme Even et al. (1976). Cooper Kingston (1996) demonstram para uma série de problemas de horários que aparecem comumente na prática estarem todos na classe NP-Difícil. Entretanto, algumas instâncias do problema podem ser resolvidas em tempo polinomial. Este é o caso do problema de programação de horários em escolas, em sua versão básica, quando turmas e professores estão sempre disponíveis. A maioria dos casos resolvíveis é, no entanto, muito especial e não incluem as restrições mais comuns. Em vista disso, justifica-se a abordagem do problema de programação de horários por métodos heurísticos, os quais, não garantem a existência de uma solução viável, mesmo que essa exista, e tampouco sua otimalidade. Para o problema de geração de horários,

conceitua-se como solução viável aquele horário que pode ser utilizado pela escola, não necessariamente sendo o mais adequado possível. Isto é, existem restrições de inviabilidade e restrições de qualidade, sendo que a satisfação apenas das primeiras restrições já qualifica um horário como viável.

1.2 Definição de Heurística

Falar sobre euristica

Como dito anteriormente em Problemas de Otimização Combinatória cujo universo de dados é grande, existe um número muito extenso de combinações, tornando inviável a análise de todas possíveis soluções, visto que o tempo computacional para uma análise completa seria demasiadamente longo. Neste sentido, têm-se as heurísticas, também conhecidas como algoritmos heurísticos, que são métodos que compõem uma gama ralativamente nova de soluções para Problemas de Otimização Combinatória.

O termo heurística é derivado do grego heuriskein, que significa descobrir ou achar. Mas o significado da palavra em pesquisa operacional 1 vai um pouco além da raiz etimológica. De um modo geral, o sentido dado ao termo heurística, refere-se a um método de busca de soluções em que não existe qualquer garantia de sucesso.

De acordo com P APADIMITRIOU S TEIGLITZ (1982), as heurísticas são quaisquer métodos de aproximação 2 sem uma garantia formal de seu desempenho. Sendo necessárias para implementação de problemas NP Difícil , caso deseje-se resolver tais problemas em um tempo computacional razoável (E VANS M INIEKA , 1978). Para N ORONHA (2000), o sucesso de uma heurística depende de sua capacidade de: (**??**)

1.2.1 Algoritmos genéticos

citar os brother do algoritimo genetico

1.2.2 Busca Tabu

(**??**)

A metaheurística BT foi inicialmente desenvolvida por Glover (1986) como uma proposta de solução para problemas de programação inteira. A partir de então, o autor formalizou esta técnica e publicou uma série de trabalhos contendo diversas aplicações da mesma. A metaheurística BT utiliza uma lista contendo o histórico da evolução do processo de busca, de modo a evitar ciclagem; incorpora uma estratégia de balanceamento entre os movimentos aceitos, rejeitados e aspirados; e adota procedimentos de diversificação e intensificação para o processo de busca. A cada iteração, a solução atual (S) muda para outra que seja sua vizinha no espaço de busca (S'). Partindo de uma solução inicial S , um algoritmo BT explora, a cada iteração, um subconjunto V da vizinhança) (S N da solução corrente S. O membro S' de V com melhor valor nessa região segundo a função f(.) torna-se a nova solução corrente mesmo que S' seja pior que S isto é, que f(S') ¿ f(S) para um problema de minimização (SOUZA, 2000). A proibição de determinados movimentos tem a intenção de impedir que a solução retorne ao ponto de mínimo local nas T iterações seguintes. O não veto de determinados movimentos pode fazer com que o algoritmo entre em loop. Um artifício criado com o intuito de não "autorizar"

a ocorrência destes movimentos é a Lista Tabu. Esta possui uma lista de tamanho t contendo as soluções visitadas durante as ultimas T iterações seqüenciadas na forma Fifo (first in first out). Souza (2000) e White et al. (2004) ressaltam a existência de um mecanismo, relacionado com a Lista Tabu, que anula o status tabu de um movimento, denominado função de aspiração. Se um movimento pode proporcionar uma melhora considerável da função objetivo, então o status tabu é abandonado e a solução resultante é aceita como potencial vizinho.

1.2.3 Algoritmo da colônia de formigas

1.3 Trabalhos Relacionados

trabalho do marinho que usa tabu.
trabalho da silvia que usa Recozimento Simulado (Simulated Annealing)
trabalho da leonardo que usa Algoritmos Genéticos (AG)
falar porque o trabalho do cara se assemelha ao meu trabalho.

2 METODOLOGIA A resolução deste trabalho

Algoritmos e porque levantamento de requistiso arquitetura

Dcumentação com o tadeu

Implementação

passo a passo

criou-se o crude a interface e testo o algoritimo de alocação se der tempo teste de unidade.

2.1 Ferramentas Utilizadas

Breve descrição e falar das ferramentas utilizadas

Bibliotecas

IDE PGADMIN

falar pq tomei essas decisões

JavaScript é uma linguagem de programação interpretada2 . Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido. É atualmente a principal linguagem para programação client-side em navegadores web. Foi concebida para ser uma linguagem script com orientação a objetos baseada em protótipos, tipagem fraca e dinâmica e funções de primeira classe. Possui suporte à programação funcional e apresenta recursos como fechamentos e funções de alta ordem comumente indisponíveis em linguagens populares como Java e C++. É baseada em ECMAScript padronizada pela Ecma international nas especificações ECMA-2623 e ISO/IEC 16262.(**??**)

Java é uma linguagem de programação orientada a objeto desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é executado por uma máquina virtual. A linguagem de programação Java é a linguagem convencional da Plataforma Java, mas não sua única linguagem.(**??**)

The Play! Framework is a modern Java (and Scala) web application open-source framework that provides a clean alternative to bloated Enterprise Java stacks. Play has two version Play 1.x (Java Scala) and Play 2.x (Java Scala). Play is a high-productivity Java and Scala web application framework that integrates the components and APIs you need for modern web application development. Play is based on a lightweight, stateless, web-friendly architecture and features predictable and minimal resource consumption (CPU, memory, threads) for highly-scalable applications thanks to its reactive model, based on Iteratee IO.

(**??**)

AngularJS is an open-source JavaScript framework, maintained by Google, that assists with running single-page applications. Its goal is to augment browser-based applications with model–view–controller (MVC) capability, in an effort to make both development and testing easier. The library reads in HTML that contains additional custom tag attributes; it then obeys the directives in those custom attributes, and binds input or output parts of the page to a model represented by standard JavaScript variables. The values of those JavaScript variables can be manually set, or retrieved from static or dynamic JSON resources.

(**??**)

3 SISTEMA DESENVOLVIDO

3.1 Modelo Tratado

3.2 Proposta de Solução

Será desenvolvido um sistema que otimiza a alocação das salas em até 90% facilizando a vida do gerente.Por se tratar de um problema especifico fica dificil encontrar tecnoloagias disponiveis para a resolução do problema sendo assim necessario o atendimento de um sistema que atenda todas as necessisdades exigidas.

3.3 O Sistema Desenvolvido

Descrição sobre o Sistema

3.3.1 Linguagens e Ferramentas Utilizadas

Descrição sobre o que o que será abordada nesta subseção

3.3.1.1 Sistema Gerenciador de Banco de Dados

Achar alguma referencia sobre o postgresql
(**??**) Postrgreesql

3.3.1.2 Ambiente de Desenvolvimento

IDE eclipse, sublimeText, Google Chrome, programa DIA para o desenvolvimento dos diagramas(**??**)

3.3.2 Modegem do Sistema

Antes de tudo foi necessaria a modelagem do sistema, para que todos os requisitos fossem atendidos de acordo com a necessidade.
Achar alguma referencia sobre metodologias de modelagem de dados UML
(**??**)
Para a analise deste sistema foram desenvolvidos os seguintes diagramas:
(**??**)
Diagramas de Caso de Uso
Diagramas de classes
Diagramas de Seqüência
Diagrama de Atividades
Diagrama de Estados

3.3.2.1 Diagrama de Caso de Uso: Sistema

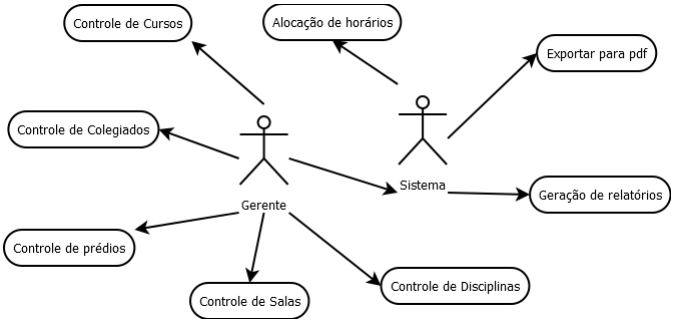
Criar o caso de uso que diz respeito a todas as funcionalidades que o sistema tem de cadastro e manutenção.

(**??**)

Caso de uso do sistema

(**??**)

Figura 1 – Diagrama de caso de uso



Fonte: Autor

3.3.2.2 Diagrama de Atividade: Alocação

Descrever a rotina de atividades da alocação do sistema

3.3.2.3 Diagrama de Classe das controllers

Achar alguma referencia de diagrama de classe.
Imagem do diagrama de classe das controllers

3.3.2.4 Diagrama de Classe das models

Imagem do diagrama de classe das models

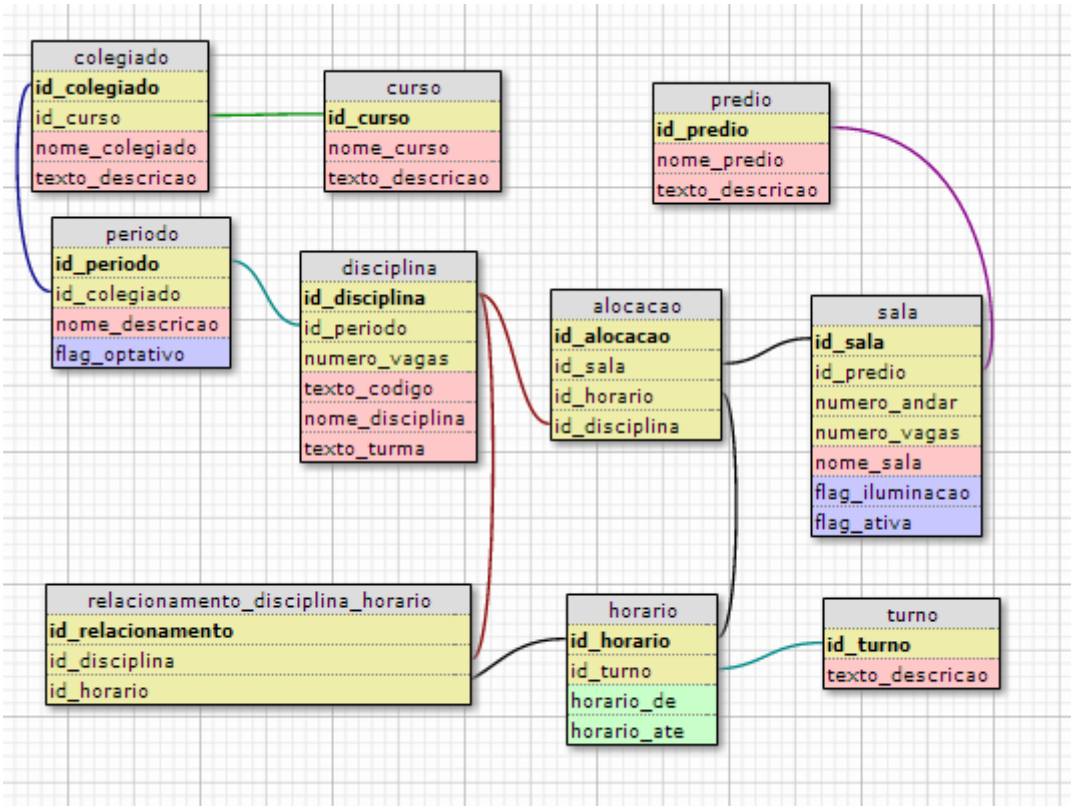
3.3.2.5 Diagrama de Classe das views

Imagem do diagrama de classe das views

3.3.2.6 Modelagem de Dados

Achar alguma referencia de modelagem de dados
Inserir imagem do modelo

Figura 2 – Modelagem Banco de Dados



Fonte: Autor

3.3.3 Interface

Achar alguma referencia sobre interface
Twitter bootstrap.

3.3.4 Funcionalidades

O sistema consiste nas seguintes funcionalidades.
1. Controle de cursos
2. Controle de colegiados
3. Controle de disciplinas
4. Controle de salas
5. Controle de prédios
6. Alocação de horários
7. Geração de relatórios
8. Exportar para pdf

3.3.5 Dados de Entrada

Como serão inseriadas as informações, e quais são os dados de entrada
Informados pelo gerente.

3.3.6 Alocação

Processa os dados de alocação

3.3.7 Relatórios

Geração dos relatorios determinados na analise do sistema, todos os relatorios podem ser exportados para pdf

3.3.8 Considerações Finais do Capítulo

Considerações finais do capitulo

4 RESULTADOS OBTIDOS Despois do sistema implementado

Entrada processamento e saida
ajuste do algoritomo de alocação

5 CONSIDERAÇÕES FINAIS

*Relembra ro problema e comprar como o resultadoo bjetido.

Discussão dos resultados obtidos na pesquisa, onde se verificam as observações pessoais do autor. Poderá também apresentar sugestões de novas linhas de estudo. A conclusão deve estar de acordo com os objetivos do trabalho. A conclusão não deve apresentar citações ou interpretações de outros autores.

5.1 Trabalhos futuros

Criar um DW para geração dos relatorios de acordo com a dimensão escolhida.

Utilização de outros algoritimos para a resolução do problema ex. algoritimos evolutivos formiga entre outros.

Pegar o feed back do usuario para melhoria na interface, e do algoritimo.

REFERÊNCIAS

[titletoc,toc,page]appendix