UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA MULTIDISCIPLINAR DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL DE CONHECIMENTO

Euzebio Lourenço da Silva

Um Sistema Evolutivo para Alocação de Serviços de acordo com o Perfil dos Funcionários

Maceió
2013

Euzebio Lourenço da Silva

Um Sistema Evolutivo para Alocação de Serviços de acordo com o Perfil dos Funcionários

Dissertação submetida ao Mestrado Multidisciplinar em Modelagem Computacional de Conhecimento como requisito parcial para a obtenção do título de Mestre em Modelagem Computacional de Conhecimento.

Orientadora: Prof.ª Dr.ª Roberta Vilhena Vieira Lopes
Co-orientador: Prof. Ms. Elvys Alves Soares

Maceió
20\aaaaa13

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecário Responsável: Valter dos Santos Andrade

S586s Silva, Euzébio Lourenço da.
Um sistema evolutivo para alocação de serviços de acordo com o perfil dos funcionários / Euzébio Lourenço da Silva. – 2013.
72 f. : il.

Orientadora: Roberta Vilhena Vieira Lopes.
Coorientador: Elvys Alves Soares.
Dissertação (Mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Programa de Pós-Graduação em Modelagem Computacional de Conhecimento. Maceió, 2013.

Bibliografia: f. 68-72.

1. Algoritmos genéticos. 2. Alocação de serviços. 3. Serviços - Otimização. I. Título.

CDU: 004.421

Euzebio Lourenço da Silva

Um Sistema Evolutivo para Alocação de Serviços de acordo com o Perfil dos Funcionários

Dissertação submetida à banca examinadora do Programa de Pós-Graduação Multidisciplinar em Modelagem Computacional de Conhecimento, da Universidade Federal de Alagoas e aprovada no dia _____o mês de dezembro do ano de 2013.

Banca Examinadora:

Prof.ª Dr.ª Roberta Vilhena Vieira Lopes
Orientadora – Instituto de Computação/UFAL

Prof. Ms. Elvys Alves Soares
Co-orientadora – Instituto de Computação/UFAL

Profa. Dra. Mônica Ximenes Carneiro da Cunha
Examinador a - Informática /IFAL

Prof. Dr. Marcus de melo Braga
Examinador - Instituto de Computação/UFAL

AGRADECIMENTOS

Antes de tudo a minha profunda gratidão ao Deus, pelos livramentos, pelas ricas e poderosas bençãos e oportunidades a mim concedidas durante toda minha existência juntamente com toda minha família.

A minha orientadora Professora Reberta, sem sua paciência, competência, compreensão, compromisso e dedicação não teria sido possível realizar este trabalho.

Minha família, meu pai e minha mãe a quem dedico este trabalho e meus irmãos e irmãs, que tanto amo, pelo apoio e aconchego, minha esposa Ana Lúcia pela paciência de suportar-me nos momentos mais difíceis da minha vida, meus filhos Isaac, Miriam, Euzébio Junior, Gabriel e Lukas, todos são razão do meu viver, que sempre me motivaram, acompanharam, e sempre estiveram e sempre estão ao meu lado em qualquer situação e especialmente o Lukinha e o Gabriel que voluntariamente quando vinham e ainda veem diariamente, quando chego ao portão de casa, correndo ao meu encontro com seus calorosos bracinhos abertos para abraçar- me apertadamente e com seus lindos e precisos sorrisos a proporcionar-me forças para continuar na caminhada quando quase não tinha mais forças.

A Universidade Federal de Alagoas, que abriu as portas para mim realizar mais este sonho, nas pessoas dos meus gestores, aos meus professores do Curso pelos ensinamentos e contribuição de maneira singular no alcance deste objetivo, a banca examinadora por examinar, avaliar e contribui sensivelmente para a grandiosidade deste trabalho, aos(as) minhas(meus) amigos(as) Allan, Andreá, Jobson, Fabrícia, Fabiano, Neto, Marcus, Leo,

Henrique, José Neto, Deniz Firemam, e demais que direto e indiretamente estiveram sempre ao meu lado, na ajuda mutua e encorajando nos momentos difíceis e desafiadores na busca do conhecimento.

Albert Einstein disse:

> “Jamais considere seus estudos como uma Obrigação, mas como uma Oportunidade Invejável para aprender a conhecer a Influência libertadora da Beleza do reino do Espírito, para seu próprio prazer pessoal e para proveito da Comunidade à qual seu Futuro o Trabalho pertencer.”

RESUMO

O problema de alocação de serviços de acordo com o perfil dos funcionários pertencentes a uma determinada repartição de uma empresa, parece uma tarefa trivial, mas se analisada com mais cautela apresenta objetivos e restrições conflitantes, o que a torna bastante complexa. Para realizar esta alocação deve-se considerar: a limitação do número de funcionários da repartição, a heterogeneidade de formação e experiência dos funcionários, a existência de material para execução da tarefa, o tempo de execução do serviço, o tempo ocioso dos funcionários, etc. Existem no mercado alguns sistemas utilizados na industria, que se propõem a realizar a tarefa de alocação de serviços, utilizando abordagens computacionais tanto convencionais (grafos e programação matemática), como inteligentes (algoritmos de busca, redes neurais e algoritmo genético). Porém independente da abordagem adotada tais sistemas trabalham a alocação do serviço, não levando em consideração as características intrínsecas a unidade alocada para cada serviço, de modo a otimizar o trabalho a ser realizado. Esta dissertação apresenta a especificação de um sistema baseado em Computação Evolutiva, mais especificamente em uma variação do Algoritmo Genético de Holland, para auxiliar o setor de manutenção da Coordenação de Gestão de Tecnologia de Informação, podendo ser expandido para toda Universidade, na tarefa de designar um profissional capacitado ao tipo de problema apresentado por um computador pertencente ao acervo computacional do Instituto de Ciências Humanas, Comunicação e Artes da Universidade Federal de Alagoas.

Palavras chave: Algoritmo Genético. Alocação e Otimização dos Serviços.

ABSTRACT

The problem of service allocation according to the profile of employees that belonging to a particular division of a company seems a trivial task, but if analyzed more cautiously it presents some conflicting objectives and constraints, which makes it very complex. To accomplish this allocation should be considered: the maximum number of employees in the division, the staff heterogeneity of training and experience, if there is enough material to accomplish the job, the time needed do finish the service, the employee's idle time, etc. There are on the market some systems used by the industry, they propose to allocate services using both computational approaches the conventional ones (using graphs and mathematical programming) an the intelligent ones (search algorithms, neural networks and genetic algorithms). But regardless of the adopted approach this systems do not take into account the allocated unit intrinsic characteristics for each service, to optimize the realized work. This paper presents the specification of a system based on evolutionary computation, more specifically on a variation of Holland's Genetic Algorithm, to assist the maintenance section of the Coordenação de Gestão e Tecnologia da Universidade Federal de Alagoas, and can be expanded to all the university, in the task to designate a skilled professional to fix a problem presented by a computer owned by the Instituto de Ciências Humanas, Comunicação e Artes da Universidade Federal de Alagoas.

Keywords: Genetic Algorthm. Job Scheduling.

Lista de Figuras

Lista de Tabelas

Sumário

1- INTRODUÇÃO

1.1 - UMA DESCRIÇÃO INFORMAL DO PROBLEMA DE ALOCAÇÃO DE SERVIÇOS DE ACORDO COM O PERFIL DO FUNCIONÁRIOS

O problema de alocação de serviços para um conjunto de funcionários, que na literatura técnica é usualmente tratado como *job scheduling*, é um problema de otimização que existe em várias áreas, como por exemplo o caso de alocação de unidades de processamento diferentes para executar as tarefas de um programa em um ambiente de programação paralela, o caso de alocação de tarefas para um conjunto de processadores em um ambiente de programação paralela. Neste trabalho, a situação de *job scheduling* considerada é a funcionalidade do setor de manutenção de computadores do Instituto de Ciências Humanas, Comunicação e Artes da Universidade Federal de Alagoas, que tem como finalidade realizar atividades de instalação, atualização e configuração tanto de *software* como de *hardware*, além do conserto de *hardware* quando necessário, podendo estes *softwares* e *hardwares* serem dos mais diversos tipos, naturezas e fabricantes.

Dependendo do serviço de reparo solicitado, um determinado perfil de funcionário é requerido e dependendo da natureza do serviço, diferentes perfis são requeridos. A prioridade de atendimento da solicitação também pode variar em função da natureza do serviço. Por exemplo: Um erro no processo de inicialização do sistema operacional \textit{windows}, possui prioridade de atendimento diferente do erro de falha do drive de leitura da unidade de DVD de um computador. Isso porque a não inicialização do sistema operacional impede que o computador seja utilizado para realizar qualquer tarefa, e a ausência do drive de leitura de DVD não impede que o computador seja usado para realizar outras tarefas.

O serviço ainda pode ser realizado parcialmente, ou porque o setor de manutenção não dispõe no momento da realização do reparo, de um perfil de funcionário disponível no momento que atenda completamente a solicitação feita, ou porque o serviço solicitado ultrapassa a carga horária diária do serviço a ser executado. Independente do motivo para uma

realização parcial do serviço, o setor de manutenção deve considerar que o reparo completo deve ser concluído no menor tempo possível, de preferência sem a remoção do equipamento para a sala do setor de manutenção de computadores.

Quando o setor de manutenção possui os recursos profissionais correspondentes à solicitação do acervo tecnológico do Instituto de Ciências Humanas, Comunicação e Artes, então a solicitação vai para a lista de serviços a atender. Caso contrário, o responsável pelo setor de Manutenção de Computadores deve enviar um comunicado para o solicitante informando a impossibilidade de atender sua demanda. Cada solicitação recebe um valor de prioridade que pode alterar sua posição na lista de serviços a atender e prazo de entrega, em função da disponibilidade do setor de manutenção. Por fim, deve-se observar que cada funcionário só pode ser alocado a um serviço por vez.

O restante deste texto apresenta detalhadamente a especificação e implementação de uma abordagem computacional para o problema de alocação de funcionários para serviços em função da correspondência entre a capacidade do funcionário e as tarefas a serem realizadas para completar o serviço. Tal abordagem poderá muito bem ser adotada para outros problemas de alocação onde deve ser considerado a referida correspondência.

1.2 - MOTIVAÇÃO

O mercado de *softwares* tem disponibilizado um conjunto de sistemas para alocação de serviços, que se propõem a encontrar uma alocação de recursos ótima. Geralmente esses sistemas realizam a alocação de serviço sem considerar todas as variáveis envolvidas em um problema real de alocação de serviços, devido a grande dificuldade de inserir estas variáveis no algoritmo e encontrar uma solução de otimização para a alocação.

O uso de métodos exatos, baseados em computação convencional, para resolução de problemas de alocação de serviços, na maioria das aplicações é inviável devido a impossibilidade de se formular uma função objetivo analítica e/ou da existência de restrições subjetivas. Isso porque os problemas de alocação pertencem uma classe de problemas NP-

HARD, que são de difícil solução por métodos exatos.

Um problema NP-hard é um problema que não têm uma fórmula geral de solução. Por exemplo, a alocação de professores dos cursos do Instituto de Computação para as disciplinas a serem ofertadas para seus cursos de graduação e pós-graduação, depende da formação e carga horária de cada docente; das disciplinas a serem ofertadas levando em consideração o turno (matutino ou vespertino); das salas de aulas disponíveis por curso; da existência de alunos habilitados a cursar as disciplinas; da prioridade das disciplinas para a conclusão dos cursos pelos alunos; etc.

Nestes casos, poderia ser verificada a viabilidade de uma modelagem baseada em algoritmos genéticos (AG). Uma vez que estes algoritmos foram propostos por Holland para tratar problemas adaptativos complexos, sendo estes sistemas um subclasse da classe dos problemas NP-Hard, a qual pertence o problema estudado. Um sistema adaptativo complexo é um sistema cujo resultado depende da interação não linear de vários agentes adaptativos. Como razões adicionais para a utilização da abordagem computacional evolutiva para o problema tratado nesta dissertação pode-se citar que:

- Os algoritmos genéticos são eficiente para a totalidade dos problemas que buscam um resultado capaz de satisfazer um conjunto predefinido de exigências;
- O comportamento dos operadores genéticos garantem que o algoritmos genéticos explore um amplo subespaço do espaço de resultados;
- Os algoritmos genéticos já foram usado com sucesso na resolução de problemas similares, como na dissertação de Feitoda (2011), cujo título é *Um Modelo Computacional de Combinação Social Aplicado ao Processo de Planejamento de Orientadores em Ambientes Virtuais de Aprendizagem*;
- A busca no espaço de resultados realizada pelos algoritmos genéticos é uma busca cega, logo este algoritmo não precisar de um modelo analítico nem de cálculo de derivadas para encontrar um resultado satisfatório; e
- Os algoritmos genéticos são extremamente simples de ser implementados.

1.3 - OBJETIVOS

1.3.1 - Objetivo Geral

Implementar um sistema evolutivo de alocação de serviços baseado na correspondência entre as características do serviço a ser realizado e as habilidades profissionais de funcionários, visando otimizar o tempo de atendimento dos serviços solicitados.

1.3.2 - Objetivos Específicos

- Definir as variáveis a serem consideradas do perfil de funcionário para a alocação de serviços.
- Decompor os serviços oferecidos em partes capes de serem alocadas em um turno.
- Estabelecer a representação do cromossomo que será implementada pelo sistema.
- Descrever matematicamente a política de atendimento de serviços, das prioridades e das restrições impostas pelos ambiente no qual o sistema está inserido.

1.4 - ORGANIZAÇÃO DA DISSERTAÇÃO

A narrativa desta dissertação foi organizadas em cinco capítulos, incluindo este capítulo introdutório.

No capítulo dois será apresentada uma breve explanação da fundamentação teórica sobre o problema de alocação de serviços, a abordagem de computação evolutiva denominada de algoritmo genético, e um levantamento de sistemas de alocação de serviços utilizando a abordagem computacional do algoritmo genético.

No capítulo três será descrita a metodologia empregada, a definição do cromossomo, a função *fitness* e as primeiras versões implementadas na tentativa de resolver a problemática. Ainda nesse capítulo, tenta-se apresentar as dificuldades encontradas que levaram a evolução da versão do algoritmo, o tratamento com dados reais, o aparecimento de itens clones e

culminando com a incorporação de um algoritmo de aprendizagem.

No quarto capítulo, tem-se os testes efetuados, a apresentação dos resultados e suas interpretações. Faz-se também a validação destes resultados e o comparativo entre a versão implementada utilizando o AG e a versão manual utilizando um técnico especialista.

Encerra-se o trabalho, apresentando no capítulo cinco as conclusões e sugestões para trabalhos futuros.

2 - JOB SCHEDULING

2.1 - PROBLEMA DE *JOB SCHEDULING*

A classe de problemas mais geral para o problema tratado nesta dissertação é *Job Scheduling*. Os objetos dessa classe são problemas que “envolvem a ordenação de alocação de recursos com a finalidade de executar uma série de tarefas” (BAKER,1974). Os problemas de *Job Scheduling* são de difícil solução e têm sido considerado um grande desafio para a computação, não existindo até a presente data pelo menos até onde se pode averiguar um algoritmo eficiente que garanta a obtenção de um resultado ótimo. Pode ser descrito da seguinte forma: considere uma ambiente de produção com um número finito de *jobs* (serviços) e unidades de produção (máquinas ou homens); cada *job* é definido por uma sequência de operações que devem ser executadas em uma dada ordem preestabelecida. Cada uma destas operações é realizada por uma unidade de produção durante um determinado período de tempo. É também assumido que cada unidade de produção pode realizar apenas uma operação ao mesmo tempo. No caso mais geral, pode haver mais de uma unidade de produção do mesmo tipo formando centros de unidades de produção (PINEDO,2002).

O resultado de um problema de *Job Scheduling* é alocar as unidades de produção aos serviços de tal maneira que um critério de satisfação seja atendido. Na prática os critérios adotados baseiam-se em uma combinação de medidas de custo e de eficiência, tais como: cumprir com o prazo de conclusão de um serviço; diminuir o atraso relativo ao prazos de conclusão de um serviço; reduzir os trabalhos em curso; tornar mais eficiente a utilização dos recursos e reduzir os tempos de fluxo (ROLDÃO,1995).

Os problemas de *job scheduling* podem ser classificados em função das unidades de produção, do seguinte modo (BLAZEWICZ,2001):

- Paralelos: quando todas as unidade de produção desempenham as mesmas funções, sendo diferenciadas apenas pelas suas velocidades de processamento:
    - Idênticos - Todas têm igual velocidade de produção (ALCOFORADO,2002);

- Uniformes - as unidade de produção tem as mesmas velocidades de processamento e sua velocidade de processamento não varia em função da operação a ser executada; e
- Não relacionados - as unidade de produção tem as mesmas velocidades de processamento e sua velocidade de processamento varia em função da operação a ser executada (MULLER,2005).

- Dedicados: quando as unidades de produção são especializadas na execução de certas operações.

2.2 - FORMALIZAÇÃO DO PROBLEMA DE *JOB SCHEDULING*

De modo geral um problema da classe de *Job Scheduling* é definido da seguinte maneira (adaptado de Sadeh (1994) e de Kan (1976):

- Dado um conjunto de v unidades de produção $M = \{m_1, m_2, \ldots, m_v\}$
- Dado um conjunto de q recursos $U = \{u_1, u_2, \ldots, u_q\}$
- Dado um conjunto de f serviços $S = \{s_1, s_2, \ldots, s_f\}$
- Dado um conjunto de w operações associadas a cada elemento do conjunto S, $O^{s_i} = \{o_1^{s_i}, o_2^{s_i}, \ldots, o_w^{s_i}\}$ com $1 \leq i \leq f$
- Encontre a sequência de ternos $(m_x, u_j, o_z^{s_i})$ com $1 \leq x \leq v$, $1 \leq j \leq q$, $1 \leq z \leq w$ e $1 \leq i \leq f$ que satisfaça os objetivos de produção.

Para expressar os objetivos de um problema de *Job Scheduling* devem ser considerados alguns parâmetros adicionais, para avaliar as possíveis funções de satisfação relacionadas a este problema, tais como:

- tempo de inicio do serviço, $t^s_{início}$;
- tempo de conclusão do serviço, t^s_{fim};
- tempo de execução do serviço, $t^s_{ececução}$, sendo que $t^s_{fim} \geq t^s_{início} + t^s_{ececução}$;

- peso atribuído a cada serviço θ^s ;
- tempo de inicio da operação, $t^o_{início}$;
- tempo de conclusão da operação, t^o_{fim} ; e
- tempo de execução da operação em cada unidade de produção, $t^{(o,m)}_{ececução}$, sendo que $t^o_{fim} \geq t^o_{início} + t^{(o,m)}_{ececução}$.

2.3 - CLASSIFICAÇÃO DO PROBLEMA DE *JOB SCHEDULING*

Devido a cardinalidade da classe dos problemas de *Job Scheduling* ser muito grande, a proposição de uma solução ótima para esta classe torna-se muito difícil. Como alternativa propôs-se a divisão desta classe em subclasses em função de três propriedades denominadas de:

- Complexidade das unidades de produção: definida em função do número de passos de processamento associados a cada tarefa de produção.
- Critérios de *scheduling*: definida em função das medidas pelas quais serão avaliados os resultados gerados.
- Natureza estática, dinâmica ou estocástica do problema: definida em função da disponibilidade ao longo do tempo dos valores precisos, incompletos ou incertos para os parâmetros do problema.

2.3.1 - Complexidade do Processamento

Os ambientes de produção podem ser classificados quanto ao grau de continuidade do processo de execução de um serviço, originando problemas de alocação com características que diferem. Se as unidades de produção são dedicados pode ocorrer:

- Ambiente contínuo (*flow-shop*): quando o número de operações por unidade de produção é igual ao número de unidades de produção, sendo a ordem das operações que compõe um serviço sempre a mesma (BERTEL, 2004).

- Ambiente aberto (*open-shop*): quando o número de operações por unidade de produção é igual ao número de unidades de produção, sendo a ordem das operações que compõe um serviço determinada no momento da sua execução (GONZALES,1976).
- Ambiente intermitente (*job-shop*): quando o número de operações por unidade de produção e a ordem das operações que compõe um serviço são determinadas no momento da sua execução (YAMADA,1997).

Segundo Blazewicz (2001) o problema de *Job Scheduling* de um ambiente de produção intermitente corresponde a subclasse de problema de *Job Scheduling* mais genérico e mais complexo, sendo o que mais desafia os métodos clássicos de otimização combinatorial, motivo pelo qual os métodos heurísticos de otimização combinatorial são os que têm obtido maior sucesso.

2.3.2 - Critérios de *Job Scheduling*

Um critério é uma medida para avaliar um resultado para um determinado problema. O valor atribuído por um critério ao resultado encontrado por uma determinada solução para o problema de *Job Scheduling* permite qualificar as soluções existentes, de tal forma que a solução responsável por gerar o resultado com maior valor segundo um determinado critério é identificada como a melhor.

Em (KAN,1976), encontra-se critérios para avaliar as soluções existentes de *Job Scheduling* baseados:

- nos tempos de termino de um serviço;
- no tempo de atraso na conclusão de um serviço;
- no custo das unidades de produção; e

na utilização das unidades de produção.2.3.3 - Natureza Estática, Dinâmica ou Estocástica de *Job Scheduling*

Um aspecto do problema de *Job Scheduling* que pode ser importante na prática é o da

natureza do problema. Segundo este aspecto, um problema de *Job Scheduling* pode ser classificado em (LAWLER, 1989):

- problema de *Job Scheduling* de natureza estática: se os valores dos parâmetros que definem o problema são todos conhecidos de antemão e não sofrem variações ao longo do tempo;
- problema de *Job Scheduling* de natureza dinâmica: se os valores dos parâmetros que definem o problema são todos conhecidos de antemão, mas podem sofrem variações ao longo do tempo; e

problema de *Job Scheduling* de natureza estocástica: se os valores dos parâmetros que definem o problema são todos considerados aleatórias e independentes, as quais obedecem certas distribuições probabilísticas.

2.4 - MÉTODOS DE SOLUÇÃO DE *JOB SCHEDULING*

Um método para encontrar o resultado de um problema de *Job Scheduling* é descrever a sequência de passos que devem ser executados por uma entidade cognitiva para encontrar as alocações de serviços e recursos das unidades de produção de um ambiente de produção que correspondam aos maiores valores dos critérios a otimizar. Quando o resultado obtido segundo um determinado método para o problema de *Job Scheduling* for sempre o mesmo, diz-se que este método é determinístico, caso contrário diz-se que este método é não determinístico.

2.4.1 - Métodos Determinísticos

A forma natural de resolver um problema de otimização combinatorial é escrever todas as possíveis combinações, soluções do problema, para em seguida avaliá-las segundo um critério especifico, e por fim escolher a combinação que obteve o melhor valor para o critério considerado (KAN,1976). No entanto, esta forma de solução de problema não permitirá encontrar o resultado para qualquer conjunto de elementos a serem combinados, por o número de combinações possíveis crescer exponencialmente com a cardinalidade do

conjunto de elementos, fazendo com que o tempo para encontrar o resultado ótimo seja inviável.

Segundo (KAN,1976) para um problema geral com v serviços e q unidades de produção terá $v!^q$ combinações possíveis. Mesmo para valores relativamente pequenos de v e q o número de combinações a serem avaliadas para encontrar o resultado ótimo de um problema de *Job Scheduling* é elevado. Por exemplo, considere que $v=5$ e $q=5$ o número de combinações possíveis é $v!^q=24.883.200.000$. Um sistema que possa gerar e avaliar 100.000 resultados por segundo precisaria de pelo menos três dias para determinar o resultado ótimo. Deste modo, a enumeração completa parece ser um método cuja aplicabilidade fica restrita a casos especiais de problemas de *Job Scheduling*, mais especificamente aos que têm número de serviços e unidades de produção pequenos, os quais representam casos sem interesse prático.

Um outro método de resolver o problema do *Job Scheduling* é denominado de "ramifica e limita", sendo o método exato mais usado para resolver problemas de optimização combinatória. O método "ramifica e limita" segue o princípio de construção de forma indutiva, isto é, considera todos os resultados possíveis avaliando a colocação de cada novo termo na combinação a ser construída. O método consiste em considerar subconjuntos sucessivamente menores do conjunto de resultados possíveis até obter conjuntos com uma única solução. Há três aspectos básicos neste método (LAND,1960) (EASTMAN,1959):

- O procedimento de ramificação realiza o particionamento de um problema grande em dois ou mais subproblemas, que quando combinados formam o problema grande inicial.
- O procedimento de limitação calcula o limite inferior do resultado ótima para cada subproblema encontrado pelo procedimento de ramificação.
- A qualidade do resultado encontrado depende dos subproblemas que são escolhidos para construí-lo em cada execução do procedimento de ramificação.

Uma variação do método "ramifica e limita" que esta sendo bastante utilizada é o

método de programação dinâmica, proposto por Bellman nos anos 1950 (BELLMAN,1957)(BELLMAN,1962). Segundo este método em cada execução do procedimento de ramificação uma decisão deve ser tomada, a qual tem impacto direto nas próximas execuções deste mesmo método. O princípio de otimização de Bellman estabelece que começando em qualquer estágio do processo de construção de um resultado o critério a ser otimizado nos estágios subsequentes é independente da sua aplicada nos estágios anteriores. Como resultado obtêm-se uma equação recursiva para descrevem os critérios de otimização de um dado estágio em função do valor prévio desses critérios.

Como pode ser percebida, a complexidade do problema de *Job Scheduling* é um fator muito importante na escolha do método de resolução a ser adotado. No entanto, dependendo do caso particular de problema a ser tratado, poderão existir métodos de resolução que encontrem o resultado do problema em tempo polinomial em função da dimensão do problema (BLAZEWICZ,2000). Então, o método de resolução para um determinado problema de *Job Scheduling* deverá começar por uma análise da sua complexidade, e o resultado final desta pode ser um das seguintes situações:

- Sim, o problema pode ser resolvido em tempo limitado por uma função polinomial da dimensão do problema. A utilidade do algoritmo apropriado para o caso de problema dependerá do grau da sua função de complexidade no pior caso, ou no caso médio, e da sua aplicação particular.
- Não, o problema é NP-difícil[1] (GAREY, 1979). Nesta situação não se tem garantia de encontrar a solução ótima para o problema mas sim uma solução satisfatória. Os métodos de resolução passam por:
    - Usar um método de aproximação, introduzindo simplificações no problema original pelo relaxamento de algumas restrições nele impostas. Isto equivale a resolver uma versão simplificada do problema real, cuja solução pode ser uma boa

1

o número de soluções possíveis aumenta numa razão exponencial em relação à dimensão ds números de parâmetros envolvidos.

aproximação da solução do problema original.

- Usar métodos aproximados que fazem uso de heurísticas. Estes tendem a encontrar uma alocação ótima mas sem sucesso garantido. Enquadram-se aqui o algoritmo de programação com fila de prioridade, aplicado ao sequenciamento das operações nas máquinas, ou um dos métodos heurísticos, descritos mais abaixo.
- Usar um método determinístico quando o problema pode ser resolvido em tempo limitado por uma função pseudo polinomial, cuja função de complexidade no pior caso é limitada por um polinômio do maior valor presente na instância do problema.

Embora os trabalhos teóricos orientados para métodos exatos, tenham trazido vários avanços ao longo dos anos para o problema do *Job Scheduling*, ele não é completamente bem sucedido em aplicações realistas. A complexidade revela-se um obstáculo aos métodos de resolução de problemas de optimização combinatória, salvo em casos em que se introduzam simplificações. Este aspecto tem sido referido na literatura clássica e moderna como *Job Scheduling*, por vários investigadores. Segundo Dorn (1994), há três tipos de problemas com os modelos exatos referidos acima:

- Os algoritmos são demasiado complexos para aplicações do mundo real.
- Os modelos exigem o conhecimento exato de durações e restrições técnicas.
- O esforço necessário para formalizar um novo problema é considerável.

2.4.2 - Métodos Não-Determinísticos

O desenvolvimento de métodos para resolução de problemas de *Job Scheduling* do mundo real, com todas as suas exigências fez com que a comunidade científica se voltasse para o desenvolvimento de métodos baseados em aproximação e heurísticas. Uma das características desses métodos é que o resultado encontrado em cada execução do mesmo método em uma mesma instanciação real do problema de *Job Scheduling* pode ser diferente (Araújo,2011).

Os métodos heurísticos para a resolução do problema de *Job Scheduling* podem ser

gerais ou específicos. Um método heurístico geral pode ser usado em uma grande variedade de problemas de *Job Scheduling*, podendo mesmo ser usado em combinação com outros métodos para a resolução desses problemas (REEVES,1993). Uma característica comum dos métodos heurísticos gerais é a de convergirem para um resultado ótimo local no espaço de busca do problema, tais como os métodos heurísticos: *simulated annealing*, *tabu search* e algoritmos genéticos. Entretanto os métodos heurísticos específicos são muito dependentes do tipo de problema de *Job Scheduling* a ser tratado, tal que só podem ser usadas para um problema de *Job Scheduling* particular, como o método heurístico fila de prioridade.

Nas próximas seções serão descritos alguns métodos heurísticos para solução do problema de *Job Scheduling* de alta complexidade.

2.4.2.1 - Fila de Prioridade

Uma método genérico frequentemente usado, consiste no que se designa o método com fila de prioridade (BLAZEWICZ,2000), também conhecido o método com regras heurísticas (ROLDAO,1995). Neste método é atribuído para cada unidade de produção, há uma fila de operações por realizar e, em cada passo, é escolhida para ser processado a primeira operação da fila.

A precisão deste método de aproximação depende da ordem pela qual as operações a realizar sejam colocadas na fila em uma posição de acordo com a regra de prioridade do problema. Na literatura existem várias regras de prioridade tais como:

- **EDD** (***E**arliest **D**ue **D**ate*): prioridade maior à operação com menor data limite (AZIZOGLU, 1998).
- **STR** (***S**lack **T**ime **R**emaining*) prioridade maior à operação com menor folga. A folga é a diferença entre o tempo restante antes da data limite e o tempo de processamento restante.
- **SPT** (***S**hortest **P**rocessing **T**ime*): prioridade à operação com o tempo de processamento mais curto (DOMINGOS,2008).
- **LPT** (***L**argest **P**rocessing **T**ime*): prioridade à operação com data de entrega mais

próxima (LEE,1991).

- **FIFO** (***F**irst **I**n **F**irst **O**ut*): prioridade maior à operação que primeiro fica disponível para processamento.
- **LIFO** (**Last In First Ou**}): prioridade maior à última operação que ficou disponível para processamento.
- **RANDOM**: prioridade atribuída a uma operação escolhida de forma aleatória.
- **CR** (***C**ritical **R**atio*): prioridade à operação com menor razão crítica. A razão crítica é o quociente entre o tempo restante antes da data limite e o tempo de processamento.
- **NOP** (**N**úmero de **Op**erações): prioridade à operação cujo processo tem menor número de passos para se executar.

Experiências feitas em larga escala, segundo Kan (1976) relatadas em (CONWAY,1967), indicam que as regras de prioridade funcionam bem na obtenção de alocações *no-delay*, havendo superioridade das regras **SPT** e **RANDOM**. Segundo Roldão (1995) a regra **SPT** funciona bem em ambientes de produção congestionados e tem um desempenho ótimo quando não há atrasos; a regra **NOP** é eficiente quando o número de operações é elevado; as regras **EDD**, **STR** e **CR** só são boas quando os níveis de carga são baixos; as regras **EDD** e **STR** são pouco aplicáveis quando há tempos de espera curtos entre operações e há pouca folga; a conjugação das regras **SPT** e **NOP** origina frequentemente bons resultados.

2.4.2.2 - *Simulated Annealing*

O método de *simulated annealing* baseia-se na noção de vizinhança de um resultado, no espaço de resultados possíveis do problema e pode ser visto como uma variante da técnica heurística de procura local, em que um subconjunto dos resultados possíveis é explorado indo repetidamente de um resultado para um resultado vizinho. O resultado encontrado com este método é considerado satisfatoriamente bom. Motivo pelo qual foram gerados dois outros métodos correspondentes a casos particulares do método de *simulated annealing* denominados de:

- *hill-climbing*: quando o objetivo é maximizar os critérios impostos pelo ambiente de produção; e
- *hill-descending*: quando o objetivo é minimizar os critérios impostos pelo ambiente de produção.

O método de *simulated annealing* (KIRKPATRICK,1983) faz uso de uma analogia; a teoria da Termodinâmica Estatística, com respeito ao fenômeno da mudança do estado de energia de um sólido quando é submetido a um processo de arrefecimento até convergir para um estado final.

2.4.2.3 - Metropolis

O sucesso obtido pelo método de *simulated annealing* fez com que Kirkpatrick (1983) propusesse usar o método Metropolis, que já havia sido usado para simular o fenômeno de mudança de energia de um material sólido, para procurar os resultados de um problema de *Job Scheduling*.

No método Metropolis em cada iteração dá-se um valor fixo a temperatura do material sólido, começando-se com um dada temperatura inicial. Em seguida é gerada uma perturbação no estado do material e calcula-se a alteração de energia correspondente a esta pertubação. Se a energia diminuiu, o sistema muda para este novo estado; caso contrário o novo estado é aceito mediante o valor de uma dada função probabilística. Isto é repetido um número determinado de vezes, dentro da mesma iteração, tendo como consequência a diminuição da temperatura do material. O ciclo deve ser repetido até que o material sólido congele, atingindo assim o estado de equilíbrio do sistema.

A função probabilística deve ser definida de tal forma que a probabilidade diminua proporcionalmente ao aumento da energia e à medida que a temperatura diminua, refletindo a tendência natural de o material sólido preferir estados de energia mínima, e de essa preferência ser cada vez mais forte à medida que a temperatura diminua.

Na analogia, ao problema de *Job Scheduling*, um estado sólido corresponde a um resultado do problema; o nível de energia do sólido corresponde ao valor da função de

avaliação para um resultado; e a temperatura corresponde a um parâmetro de controle que varia em cada iteração, servindo basicamente de contador de iterações. O estado final corresponde ao resultado encontrado com melhor avaliação.

O método começa com um resultado sub-ótimo do problema para um certo valor do parâmetro de controle. Em cada iteração este parâmetro mantém-se constante. Um resultado vizinho do resultado atual é escolhido como o novo resultado atual do problema, se a sua temperatura for menor. Caso contrário o resultado escolhido pode mesmo assim ser aceito, se um valor aleatoriamente gerado for menor que o valor da função de probabilidade. Isto é repetido um número determinado de vezes dentro da mesma iteração. Em seguida, o parâmetro de controle é decrementado de uma unidade, repetindo-se todo o ciclo até que o parâmetro de controle seja zerado.

2.4.2.4 - *Tabu Search*

Semelhante do método de *simulated annealing* o método de *tabu search* faz uso da noção de vizinhança e de uma analogia com um fenômeno natural que é, no caso, o uso de uma forma de memória.

O princípio básico do método de *tabu search* (GLOVER,1986) (HANSEN,1986) é o de uma procura heurística do tipo *hill-descending*, guiada para explorar o espaço de resultados, de modo a evitar os resultados ótimos locais que já tenham sido anteriormente encontrados. Para isto é usada uma estrutura de memória designada por lista tabu, que contém em cada momento um número de resultados proibidos encontrados em um certo número de iterações anteriores.

Alternativamente, a lista tabu pode memorizar, não os próprios resultados proibidos, mas os movimentos no espaço de resultados que conduzem a eles. Esta forma alternativa pode economizar memória para armazenamento da lista tabu e economizar tempo de verificação se um dado resultado é um ótimo local presente na lista tabu, mas poderá revelar-se demasiadamente restritivo proibindo resultados ótimos que encontram-se muito próximos aos ótimos locais da lista tabu.

No método de *tabu search* em cada iteração passa-se do resultado atual para um resultado na sua vizinhança que não esteja na lista tabu e que aumente ou mantenha o valor da função de avaliação. Existe um critério de admissibilidade que determina se um movimento é aceitável mesmo estando registrado na lista tabu. Podem existir, no contexto deste método, estratégias de intensificação, para concentrar a procura em certas regiões do espaço de resultados mais promissoras ou de diversificação.

Outras características são o uso de listas de candidatos, usadas para reduzir o número de movimentos investigados em cada iteração. Quando a vizinhança ou o custo computacional de avaliar cada resultado são muito grandes, tem-se duas opções:

- usar os resultados bons para guiar um procura do resultado do problema;
- calcular a frequência de aparecimento de certos elementos nos resultados bons para sugerir uma diversificação do espaço de resultados considerado.

Este método requer o uso intensivo da memória e uma implementação muito trabalhosa mas oferece as vantagens de poder produzir bons resultados em tempos relativamente pequenos se for bem aplicado (RIBEIRO, 1996) (GLOVER , 1993). Armentano (2013) descreve resumidamente várias aplicações com sucesso do *tabu search* a variados problemas de *Job Scheduling*.

2.4.2.5 - Algoritmos Genéticos

Este método faz uso de uma analogia com o fenômeno natural da evolução dos organismos vivos. A ideia base dos Algoritmos Genéticos (HOLLAND,1975) (GOLDBERG,1989) (REEVES,1993b) é a de simular o desenvolvimento de populações de indivíduos de uma espécie com características cada vez melhores, face a uma seleção natural. Estes algoritmos consideram que o espaço de resultados do problema é uma população de indivíduos; que as regras para a identificação do resultado ótimo devem ser combinadas em um sistema de equações denominado de função de adaptação; que os subespaços de interesse contendo o resultado do problema são determinados pela função de seleção e que a busca do resultado do problema dentro dos subespaços de interesse são realizadas pelos operadores

genéticos (cruzamento e mutação). Maiores detalhes sobre este método de resolução de problemas de *Job Scheduling* podem ser encontrados em (ANANDARAMAN,2011) (MATTFELD,2004).

2.5 - CONSIDERAÇÕES FINAIS

O presente capítulo apresentou de forma sucinta a classe de problema *Job Scheduling*, sua classificação e alguns métodos de resolução para essa classe de problema.

Os métodos determinísticos não são considerados bons em problemas reais devido a inviabilidade de serem executados em tempo polinomial. E os métodos não determinísticos embora apresentem algumas vantagens em relação aos outros métodos, não têm o compromisso de encontrar o resultado ótimo global para um problema de *Job Scheduling*.

No próximo capítulo será descrito com mais detalhes o método de resolução para *Job Scheduling* denominado de algoritmo genético, o qual foi utilizado como abordagem computacional da presente dissertação.

3 - ALGORITMO GENÉTICO HOLLAND

3.1 - CONTEXTO HISTÓRICO

A primeira tentativa de representação, por meio de um modelo matemático, da teoria da Evolução das Espécies de Charles Darwin, surgiu com o livro *The Genetic Theory of Natural Selection*, escrito pelo biólogo evolucionista Fisher (1930). Segundo a teoria de Darwin os indivíduos de uma espécie que apresentam as características mais favoráveis ao ambiente em que habita, denominados de indivíduos adaptados, têm maior chance de sobreviver que os indivíduos que não apresentam estas características. Isto porque os indivíduos adaptados são preteridos pelos indivíduos da sua espécie para procriar, gerando assim um maior número de descendentes potencialmente adaptados ao meio ambiente, uma vez que as características apresentadas por um indivíduo são provenientes da combinação aleatória das características herdadas dos seus pais.

A explicação de Darwin para o desenvolvimento do pescoço da gira pode servir para ilustrar sua teoria. Ela tem início em uma população de girafas que apresentavam pescoço de comprimentos variados. Nesta população aquelas girafas com pescoço mais longo tinham maior facilidade de conseguiam alimento. As girafas com pescoço curto, no entanto, eram naturalmente eliminadas. As girafas com pescoço longo transmitiam essa sua característica aos seus descendentes, que por sua vez a transmitiam aos seus descendentes e assim sucessivamente. Assim, depois de um longo período de tempo, todas as girafas da população apresentavam pescoço mais comprido do que as girafas da população inicial, este processo se repetiu até que a característica pescoço longo não fosse mais uma característica importante na adaptação das girafas. Darwin, no entanto, não consegui explicar a causa da existência de diferentes comprimentos de pescoço na população inicial de girafas, pois o DNA era desconhecido na época, e a operação de mutação totalmente ignorada.

A adoção de um modelo matemático baseada na teoria de Darwin para especificar um problema de otimização ocorreu em 1957 foi apresentado pela primeira vez um esquema

de operações evolucionárias por Box (1975). Partindo da hipótese de que uma forma de evitar que a população trabalhada para um dado problema pelos modelos matemáticos baseados no processo de evolução das espécies convergissem para um resultado ótimo local seria inserir um elemento perturbador ao modelo. Logo depois no começo da década de 1960, Bledsoe e Bremmerman começaram a trabalhar com: os alfabetos usados na construção dos cromossomos e com a adequação dos operadores genéticos aos novos alfabetos propostos.

Em meados da década de 60, John Holland dedicou-se ao estudo de processos naturais adaptáveis, propondo assim os algoritmos genéticos. Ele desenvolveu vários modelos de algoritmos genéticos em conjunto com seus alunos e colegas da Universidade de Michigan nos anos 60 e 70, tendo como objetivos:

- estudar formalmente o fenômeno da adaptação como ocorre na natureza e
- desenvolver modelos em que os mecanismos da adaptação natural pudessem ser importados para os sistemas computacionais.

Como resultado do seu trabalho, em 1975 Holland editou *Adaptation in Natural and Artificial Systems* e em 1989 Goldberg editou *Genetic Algorithms in Search, Optimization and Machine Learning*, sendo atualmente considerado trabalho mais importantes sobre algoritmos genéticos.

Algoritmos genéticos fazem parte da classe dos algoritmos evolucionários, podendo ter uma arquitetura sequencial ou paralela. A implementação paralelizada total ou parcial dos algoritmos genéticos tem sido usada como alternativa para o aumento de seu desempenho (VIEIRA, 2003). Não se pretende detalhar as formas de paralelização dos algoritmos genéticos neste trabalho.

A busca realizada por um algoritmo genético dentro do espaço de busca não é aleatória, pois o algoritmo tem a informação da adaptação dos cromossomos da população atual para determinar os próximos subespaços de busca a serem explorados. Os ótimos locais e as descontinuidades existentes no domínio do problema são superados pelo operador genético de mutação, responsável por gerar pertubações nos cromossomos da população. Na próxima seção será descrito o algoritmo genético de Holland em mais detalhes.

3.2 - O ALGORITMO GENÉTICO DE HOLLAND

As abordagens de computação evolutiva procuram utilizar o princípio evolucionário encontrado na natureza para encontrar resultados para problemas computacionais, tomando como base para isto o modelo natural da sobrevivência do mais apto, compreende as abordagens de:

- algoritmo genético: foi proposto por Holland (1995) com o objetivo de abordar fenômenos gerados por sistemas adaptativos complexos, ou seja, fenômenos cujos resultados dependem das interações não lineares entre vários agentes adaptativos.
- programação evolutiva: foi concebida por Fogel (1994) como uma estratégia de otimização estocástica que enfatiza o relacionamento comportamental entre os indivíduos progenitores e sua descendência. Deve-se destacar que a programação evolutiva só trabalha com o operador genético de mutação baseada em uma distribuição estatística, a qual pondera as variações menores no comportamento dos indivíduos descendentes como altamente prováveis e variações de maior como improváveis,
- estratégias evolutivas: foi introduzida por Rechernberg (1965) como um método capaz de otimizar parâmetros com valores reais. Por este motivo, o indivíduo manipulado por um algoritmo de estratégia evolutiva é representado por um vetor de tamanho fixo de valores reais. Na estratégia evolutiva o operador que desempenha o papel principal para a evolução da população é a mutação, ficando a operação de cruzamento relegada a um segundo plano,
- programação genética: foi apresentada por Koza (1994) como um sistema que tem competência de encontrar uma função para descreve um dado fenômeno a partir de um conjunto de fatos conhecidos e de operações válidas no contexto do fenômeno referido. A representação genotípica dos indivíduos manipulado por este abordagem computacional é uma função que quando calculada sobre os fatos conhecidos oferecem diferentes soluções para o mesmo fenômeno, ao contrário dos indivíduos manipuladas pelo algoritmo genético e pela programação evolutiva cujos indivíduos

manipulados codificam possíveis resultados para um problema e

- sistemas de classificação: foi idealizado por Holland (1995) como um sistema capaz de perceber e classificar os acontecimentos de um ambiente e reagir a eles de maneira apropriadamente. A arquitetura típica de um sistema classificador é constituída de cinco componentes: uma lista de mensagem que realiza o papel de uma base de fatos ou memória de trabalho de um sistema baseado em regras; uma lista de indivíduos que são versões extremamente simplificadas de regras de produção (**se** *condições* **então** *ações*) com número de *condições* fixo, um *interface* de entrada que informam ao sistema o que está ocorrendo no ambiente e uma **interface** de saída que permite ao sistema agir sobre o ambiente. Em um sistema classificador a avaliação de cada indivíduo pode variar dinamicamente em função da experiência adquirida e pela análise dos estímulos positivos ou negativos provenientes do ambiente. O sistema classificador considera que as operações de cruzamento e mutação possuem igual importância para a evolução dos indivíduos da população.

Holland propôs dois algoritmos genéticos, denominados de R_1 e R_d (HOLLAND,1995). No algoritmo R_1, em cada iteração do algoritmo será gerado um novo indivíduo que substituirá um dos indivíduos da população atual. Enquanto que, no algoritmo R_d, em cada iteração do algoritmo serão gerados m descendentes, onde m é o tamanho da população atual, os quais irão substituir todos os indivíduos da população atual. Em seguida, Holland propôs o algoritmo R que simula o comportamento desses dois algoritmos dependendo dos valores dos parâmetros h e k. Os indivíduos da população são representados por cadeias binárias, denominadas de cromossomo.

As variáveis t, h, k, $P(t)$, P_1, P_2 e W do algoritmo R representam o número de gerações, o número de cromossomos selecionados para gerar descendentes, o número de descendentes que devem ser gerados, a população na geração t, o reservatório dos cromossomos que irão gerar novos cromossomos, o reservatório dos novos cromossomos gerados e um conjunto de operadores genéticos que serão aplicados aos cromossomos do

reservatório P_1 para gerar os cromossomos do reservatório P_2, os quais farão parte da próxima população $P(t+1)$, respectivamente. O número de gerações é incrementado em cada execução do laço *faça– enquanto* para simular a contagem do tempo genealógico. A população atual $P(t)$ é formada por um conjunto de m cromossomos, os quais representam alguns dos possíveis resultados para o sistema adaptativo complexo em análise.

A adaptação de um cromossomo é dada pela função objetiva δ do sistema adaptativo em análise. Se a função objetiva for de maximização, então a função de adaptação f do algoritmo genético será igual a δ, caso contrário, a função de adaptação f do algoritmo genético será igual a $-\delta$. Essa convenção se faz necessária porque o algoritmo genético, por definição, estará sempre em busca do cromossomo mais adaptado.

A condição de parada do algoritmo genético pode ser o menor valor de adaptação considerado satisfatório, um número de iterações determinado, ou qualquer combinação dessas duas condições.

Holland trabalhou com três operadores genéticos denominados de : cruzamento (de um ponto de corte), mutação (por complemento e por inversão) em seus algoritmos genéticos. O operador de cruzamento de um ponto de corte tem por objetivo combinar a cadeia de dois cromossomos da seguinte maneira: seja $a_1 a_2 \cdots a_m$ e $b_1 b_2 \cdots b_m$ dois cromossomos de $P(t)$, seleciona-se aleatoriamente um número x, pertencente ao conjunto $\{1,2,\cdots,m\}$ e então constrói-se dois novos cromossomos 11111 e 00000. A Figura 3.1 ilustra um cruzamento, sendo que $x=3$. Considere que

Figura 3.1 - Exemplo de Cruzamento

Fonte: Autor em 2013

O operador de mutação tem por objetivo alterar a cadeia de um cromossomo da

seguinte maneira: seja $a_1 a_2 \cdots a_m$ um cromossomo de P_t, seleciona-se aleatoriamente n números $x_1, x_2, \cdots x_n$ com $n<m$, pertencentes ao conjunto $\{1, 2, \cdots, m\}$ e então constrói-se um novo cromossomo $a_1 \cdots a_{x_1-1} \overline{a}_{x_1} a_{x_1+1} \cdots a_{x_2-1} \overline{a}_{x_2} a_{x_2+1} \cdots a_m$. Os caracteres com a barra em cima das posições x_i dessa cadeia representam o complemento binário do caractere que ocupava essa mesma posição no cromossomo $a_1 a_2 \cdots a_m$ de $P(t)$. A Figura 3.2 ilustra uma mutação considerando $n=2$, $x_1=2$ e $x_2=5$.

Figura 3.2 - Exemplo de Mutação

	$\downarrow x_1$			$\downarrow x_2$
1	1	1	1	1
		↓		↓
1	1	0	1	0

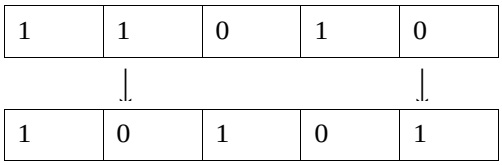
Fonte: Autor em 2013

O operador de inversão, tem por objetivo alterar a ordem dos caracteres de uma sub-cadeia de um cromossomo da seguinte maneira: seja $a_1 a_2 \cdots a_m$ um cromossomo de $P(t)$, seleciona-se aleatoriamente dois números x_1 e x_2, pertencentes ao conjunto $\{1, 2, \cdots, m\}$ e então constrói-se um novo cromossomo $a_1 \cdots a_{x_1-1} a_{x_2} a_{x_2-1} \cdots a_{x_1+1} a_{x_1} \cdots a_m$. A Figura 3.3 ilustra uma inversão considerando que $x_1=2$ e $x_2=5$.

A seguir será apresentado um exemplo de aplicação deste algoritmo a um problema adaptado de (VIEIRA 2003).

Figura 3.3 - Exemplo de Inversão

$\downarrow x_1$ $\downarrow x_2$



Fonte: Autor em 2013

3.3 - O COMPORTAMENTO DO ALGORITMO GENÉTICO DE HOLLAND

Segundo Ausubel (Ausubel,1980), há três tipos de aprendizagem significativa: a aprendizagem representacional, a aprendizagem conceitual e a aprendizagem proposicional. A aprendizagem representacional refere-se ao processo de nomeação de objetos, em que o aluno associa um dado nome a um determinado símbolos. Por exemplo, o aluno associa o nome "zero" ao numeral "0". Este tipo de aprendizagem constitui o primeiro tipo de aprendizagem a qual o aluno é submetido. Os símbolos são convenções que permitem ao aluno conhecer e organizar o seu mundo exterior e interior. Ausubel considera que este tipo de aprendizagem é a que mais se aproxima da aprendizagem mecânica.

Considerando que a população inicial de um algoritmo genético é escolhida aleatoriamente, logo o valor da função de adaptação dos cromossomos dessa população encontra-se distribuída no espaço de soluções aleatoriamente. À medida que o algoritmo é executado, os cromossomos da população vão se deslocam das regiões com baixo valor de adaptação para as regiões com valores de adaptação mais altos. Nas ultimas iterações do algoritmo, a maioria dos cromossomos encontram-se localizados em sub-espaços de resultados que contêm os resultados ótimo global ou local para o problema.

Figura 3.4 - Descrição do comportamento do Algoritmo Genético de Holland ao longo do tempo

início do procedimento R

$t \leftarrow 0$

gera $P(t)$

calcule o valor da adaptação dos cromossomos de $P(t)$

enquanto (condição de parada não for satisfeita) **faça**

$P_1 \leftarrow$ selecione h cromossomos de $P(t)$

$W \leftarrow$ selecione alguns operadores genéticos

$P_2 \leftarrow$ os k cromossomos descendentes de P_1 por aplicações sucessivas de W

$P(t+1) \leftarrow$ substitua os cromossomos de $P(t)$ pelos cromossomos de P_2

$t \leftarrow t+1$

calcula o valor da adaptação dos cromossomos de $P(t)$

fim do enquanto

fim do procedimento R

Fonte: Autor em 2013

A Figura 3.4 mostra esse movimento dos cromossomos em uma população das regiões do espaço de resultados de menor valor de adaptação em direção às regiões de maior valor de adaptação, até encontrar as regiões contendo os resultados ótimos. Essa migração pode ser dividida em dois momentos distintos:

- exploração: em que os cromossomos se movem para regiões ainda não testadas, ação gerada pelo operadores genéticos de mutação e inversão; e
- aproveitamento: em que os cromossomos se concentram nas redondezas de regiões contendo cromossomos com valor de adaptação alto, ação gerada pelo operador genético de cruzamento.

A migração dos cromossomos geralmente segue uma relação de custo/benefício entre a exploração e o aproveitamento. O tempo de execução das operações de exploração e aproveitamento de um algoritmo genético irá depender do valor de probabilidade associado a seus operadores genéticos. Se o algoritmo genético ficar muito tempo na exploração ele irá gerar uma busca ineficiente, mas com maior probabilidade de encontrar o ótimo global. Por outro lado, se o algoritmo genético passar muito tempo na execução da operação de aproveitamento, ele pode ficar preso em uma região ótima local (EIBEN; SCHIPPERS,

1998).

3.4 – O ALGORITMO GENÉTICO DE HOLLAND APLICADO AO PROBLEMA CLASSIFICAÇÃO DE FARDAS

Agora que já foi descrito o algoritmo genético de Holland será mostrado um exemplo de como este algoritmo pode ser instanciado a um problema de otimização combinatorial, para que as possíveis lacunas no entendimento desta abordagem computacional sejam completadas antes de ser apresentado as limitações encontrados na literatura para a instanciação do algoritmo de Holland a diferentes tipos de problemas.

Considere o problema de classificar fardamentos proposto em Vieira (2003) de uma loja que vende uniformes escolares por escola e tamanho antes de arrumá-los nas prateleiras, onde as roupas fabricadas pela loja são das escolas: Aprender (1), Saber (2), Viver (3) e Ensinar (4); e os tamanhos são representados pelos tamanhos: Pequeno (P), Médio (M) e Grande (G), Extra Grande (E).

A solução consiste em uma lista contendo todas as combinações possíveis entre as escolas e tamanhos das fardas comercializados pela a loja, tal que o número das escolas e dos tamanhos esteja disposto em ordem crescente. O número de combinações possíveis entre as 4 escolas e os 4 tamanhos é 16, logo esta lista é formada por 16 pares $(escola_i, tamanho_j)$, tal que $escola_i \leq escola_{i+1}$ e $tamanho_j \leq tamanho_{j+1}$, com $i \in \{1,2,3,4\}$ e $j \in \{P, M, G, E\}$. A tabela abaixo ilustra todas as combinações possíveis para este problema.

Mas para que este resultado seja codificado em um cromossomo representado por um vetor binário de tamanho fixo m, como definido por Holland, será preciso primeiro codificar cada par $(escola_i, tamanho_j)$ em uma sequência binária. Como o número de pares é 16, então o tamanho desta sequência é 4, já que uma sequência binária de tamanho x pode expressar valores naturais pertencentes ao conjunto $\{0, 1, \ldots, 2^x - 1\}$. A Tabela 3.2 mostra a relação das escolas e dos tamanhos comercializados pela loja, em cadeias binárias de tamanho igual a 4.

Tabela 3.1 – Combinações escola versus tamanho

	Pequeno (P)	Médio (M)	Grande (G)	Extra Grande (E)
Aprender (1)	(1,P)	(1,M)	(1,G)	(1,E)
Saber (2)	(2,P)	(2,M)	(2,G)	(2,E)
Viver (3)	(3,P)	(3,M)	(3,G)	(3,E)
Ensinar (4)	(4,P)	(4,M)	(4,G)	(4,E)

Fonte: (Vieira, 2003)

Tabela 3.2 – Combinações escola versus tamanho em sequencias binárias

	Pequeno (P)	Médio (M)	Grande (G)	Extra Grande (E)
Aprender (1)	0000	0001	0010	0011
Saber (2)	0100	0101	0110	0111
Viver (3)	1000	1001	1010	1011
Ensinar (4)	1100	1101	1110	1111

Fonte: (Vieira,2003)

De posse da codificação do par $(escola_i, tamanho_j)$ em cadeias binárias de tamanho igual a 4, o cromossomo será representado por um vetor contendo todas as cadeias binárias correspondentes aos pares $(escola_i, tamanho_j)$ da tabela acima, ou seja, o cromossomo é uma cadeia $v_1, \square, v_{64}$, sendo cada sub-cadeia de quatro bits correspondente a codificação de um par $(escola_i, tamanho_j)$ relacionada na tabela descrita.

A função de adaptação para este problema deve considerar a ordem em que os pares estão dispostos, assim para calculá-la será necessário antes definir uma função que gere o índice do par $(escola_i, tamanho_j)$ correspondente a cadeia binária de tamanho 4 que o representa. Seja $decPar: S_B \rightarrow \square$, onde S_B é o conjunto das cadeias binárias de 4 bits e $\square$

é o conjunto dos números reais, tal que $decPar(b_1 b_2 b_3 b_4) = 2b_2 + b_1 + 1, 2b_3 + b_4 + 1$. A função de adaptação $adapt : C \rightarrow \square$, onde C é o conjunto das cadeias binárias de 64 bits, é definida por:

$$adapt(v_1, \square, v_{64}) = \sum_{i \in V} ordem(decPar(v_i, \square, v_{i+3}), decPar(v_{i+4}, \square, v_{i+7}))$$

onde:

$V = \{1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61\}$, e

$$ordem((i_1, j_1), (i_2, j_2)) = \begin{cases} 1 & \text{se } (j_1 \quad j_2) \\ 0 & \text{caso contrário} \end{cases}$$

Tendo, por exemplo, uma população inicial P_0 formada pelos cromossomos C_1 , C_2 , C_3 e C_4 , conforme segue:

Tabela 3.3 – População Inicial

C_1	0001011000111000111010010101101000001111011111001101001010110100
C_2	0001111000001101011010011111001000110101011110111000101011000100
C_3	1000101011000100000111100000110100110101011110111010100111110010
C_4	0111110010011110000001100001010011010101101000111000111100101011

Fonte: Autor em 2013

O cálculo da adaptação de cada cromossomo será obtido através da aplicação da função de adaptação definida, para o cromossomo:

$$adapt(C_1) = ordem(decPar(0001), decPar(0110)) + ordem(decPar(0110), decPar(0011))$$
$$+ ordem(decPar(0011), decPar(1000)) + ordem(decPar(1000), decPar(1110))$$
$$+ ordem(decPar(1110), decPar(1001)) + ordem(decPar(1001), decPar(0101))$$

$$+ ordem(decPar(0101), decPar(1010)) + ordem(decPar(1010), decPar(0000))$$
$$+ ordem(decPar(0000), decPar(1111)) + ordem(decPar(1111), decPar(0111))$$
$$+ ordem(decPar(0111), decPar(1100)) + ordem(decPar(1100), decPar(1101))$$
$$+ ordem(decPar(1101), decPar(0010)) + ordem(decPar(0010), decPar(1011))$$
$$+ ordem(decPar(1011), decPar(0100))$$

$$adapt(C_1) = ordem((1,2),(2,3)) + ordem((4,4),(4,5)) + ordem((5,6),(6,7))$$
$$+ ordem((7,8),(8,9)) + ordem((9,10),(10,11)) + ordem((11,12),(12,13))$$
$$+ ordem((13,14),(14,15)) + ordem((15,16),(16,1))$$
$$adapt(C_1) = 0+0+1+0+1+1+0+1+0+1+1+0+0+0+1+0 = 7$$

$$adapt(C_2) = ordem(decPar(0001), decPar(1110)) + ordem(decPar(1110), decPar(0000))$$
$$+ ordem(decPar(0000), decPar(1101)) + ordem(decPar(1101), decPar(0110))$$
$$+ ordem(decPar(0110), decPar(1001)) + ordem(decPar(1001), decPar(1111))$$
$$+ ordem(decPar(1111), decPar(0010)) + ordem(decPar(0010), decPar(0011))$$
$$+ ordem(decPar(0011), decPar(0101)) + ordem(decPar(0101), decPar(0111))$$
$$+ ordem(decPar(0111), decPar(1011)) + ordem(decPar(1011), decPar(1000))$$
$$+ ordem(decPar(1000), decPar(1010)) + ordem(decPar(1010), decPar(1100))$$
$$+ ordem(decPar(1100), decPar(0100))$$

$$adapt(C_2) = ordem((1,2),(2,3)) + ordem((3,4),(4,5)) + ordem((5,6),(6,7))$$
$$+ ordem((7,8),(8,9)) + ordem((9,10),(10,11)) + ordem((11,12),(12,13))$$
$$+ ordem((13,14),(14,15)) + ordem((15,16),(16,1))$$

$$adapt(C_2) = 0+1+0+0+1+0+1+0+1+0+1+1+0+1+1+0 = 8$$

$$\begin{aligned} adapt(C_3) = {} & ordem(decPar(1000), decPar(1010)) + ordem(decPar(1010), decPar(1100)) \\ & + ordem(decPar(1100), decPar(0100)) + ordem(decPar(0100), decPar(0001)) \\ & + ordem(decPar(0001), decPar(1110)) + ordem(decPar(1110), decPar(0000)) \\ & + ordem(decPar(0000), decPar(1101)) + ordem(decPar(1101), decPar(0011)) \\ & + ordem(decPar(0011), decPar(0101)) + ordem(decPar(0101), decPar(0111)) \\ & + ordem(decPar(0111), decPar(1011)) + ordem(decPar(1011), decPar(0110)) \\ & + ordem(decPar(0110), decPar(1001)) + ordem(decPar(1001), decPar(1111)) \\ & + ordem(decPar(1111), decPar(0010)) \end{aligned}$$

$$\begin{aligned} adapt(C_3) = {} & ordem((1,2),(2,3)) + ordem((3,4),(4,5)) + ordem((5,6),(6,7)) \\ & + ordem((7,8),(8,9)) + ordem((9,10),(10,11)) + ordem((11,12),(12,13)) \\ & + ordem((13,14),(14,15)) + ordem((15,16),(16,1)) \end{aligned}$$

$$adapt(C_3) = 0+1+1+0+0+1+0+0+1+0+1+1+1+0+1+1 = 9$$

$$\begin{aligned} adapt(C_4) = {} & ordem(decPar(0111), decPar(1100)) + ordem(decPar(1100), decPar(1001)) \\ & + ordem(decPar(1001), decPar(1110)) + ordem(decPar(1110), decPar(0000)) \\ & + ordem(decPar(0000), decPar(0110)) + ordem(decPar(0110), decPar(0001)) \\ & + ordem(decPar(0001), decPar(0100)) + ordem(decPar(0100), decPar(1101)) \\ & + ordem(decPar(1101), decPar(0101)) + ordem(decPar(0101), decPar(1010)) \\ & + ordem(decPar(1010), decPar(0011)) + ordem(decPar(0011), decPar(1000)) \end{aligned}$$

$$+ ordem(decPar(1000), decPar(1111)) + ordem(decPar(1111), decPar(0010))$$
$$+ ordem(decPar(0010), decPar(1011))$$

$$adapt(C_4) = ordem((1,2),(2,3)) + ordem((3,4),(4,5)) + ordem((5,6),(6,7))$$
$$+ ordem((7,8),(8,9)) + ordem((9,10),(10,11)) + ordem((11,12),(12,13))$$
$$+ ordem((13,14),(14,15)) + ordem((15,16),(16,1))$$

$$adapt(C_4) = 1+0+0+1+0+1+1+0+1+0+0+1+0+1+0+1 = 8$$

Em seguida o algoritmo genético de Holland seleciona os $h = 2$ cromossomos mais adaptados da população inicial P_0, que no caso são C_2 e C_3, os quais constituem a população dos cromossomos selecionados P_{sel}. Sobre estes cromossomos será realizada a operação de cruzamento e mutação, conforme segue:

Cromossomos mais adaptados da população inicial P_0 antes da operação de cruzamento

C_2 = **0001111000001101011010011111001000110101011110111000101011000100**

C_3 = 1000101011000100000111100000110100110101011110110110100111110010

Cromossomos gerados após operação de cruzamento em P_0 :

C_5 = **0001111000001101011010011111100**0100110101011110110110100111110010

C_6 = 1000101011000100000111100000111**0001101010111101110001010110001 00**

A população dos cromossomos descendente P_{des} da atual P_0 recebe os

cromossomos C_5 e C_6 com valor de adaptação respectivamente: $adapt(C_5)=5$ e $adapt(C_6)=7$. Logo após realizam-se operações de mutação nos cromossomos da população de selecionados P_{sel}, para gerar mais cromossomos para a população de descendentes P_{des}.

Cromossomos mais adaptados da população inicial P_0 antes da operação de mutação por complemento.

C_2 = 0001111000001101011010011111**1**0010001101010111101110001010110001 00

C_3 = 1000101011000100000**0**1111000001101001101010111101101101001111100 10

Cromossomos gerados após operação de mutação em P_0 :

C_7 = 0001111000001101011010011111**0**0010001101010111101110001010110001 00

C_8 = 1000101011000100001**1**1111000001101001101010111101101101001111100 10

Sendo o resultado dessa mutação colocado na população de cromossomos descendentes P_{des}. A nova população a ser trabalhada pelo algoritmo genético de Holland é formada pelos $k=4$ cromossomos mais adaptados da união da atual $adapt(C_7)=8$ com a população de descendentes gerados $adapt(P_{des})=\{C_5, C_6, C_7, C_8\}$, ou seja, a população $adapt(P_1)=\{C_2, C_3, C_4, C_8\}$. Dessa forma, o algoritmo genético executa sucessivas interações até encontrar a solução do problema dentro do espaço de busca que para o exemplo descrito anteriormente é o cromossomo com valor de adaptação igual a 15. Note que o valor de adaptação 15 é o maior valor que pode ser obtido em uma lista de 16 pares comparando dois a dois para verificar se os mesmos estão ordenados.3.5 - LIMITAÇÕES DO ALGORITMO GENÉTICO DE HOLLAND

Considere que o problema tratado na seção anterior necessitasse do acréscimo de mais uma escola: para atender a essa exigência, o número de bits para representar os pares $(escola_i, tamanho_j)$ fossem maior do que o número pares existentes na loja de fardamento. Nesta situação, existiriam casos em que uma sequência de bits não representaria nenhum par trabalhado pelo problema. O que poderia conduzir o algoritmo genético a encontrar um cromossomo que não é solução para o problema. Uma forma de manter o algoritmo de Holland, com a representação definida e ainda assim conseguir que os cromossomos encontrados representem uma solução para o problema, seria trabalhar a função de adaptação para que esta penalizasse os cromossomos indesejáveis, ou propor operadores genéticos que só gerem cromossomos válidos.

Outras limitações encontradas no algoritmo genético de Holland são:

- Sua forma de seleção e construção da nova população reduz a diversidade da população, o que limita a exploração do espaço de resultados do problema;
- Sua representação com tamanho fixo para o cromossomo e a população evita que sejam consideradas novas características ou cromossomos promissores;
- Seus operadores genéticos são subordinados a representação adotada para o cromossomo e a população.

Na tentativa de resolver alguns desses problemas vários trabalhos propondo alterações na representação, no comportamento dos operadores genéticos, na função de seleção e na função de substituição do algoritmo genético de Holland.

3.6 - CONSIDERAÇÕES FINAIS

O algoritmo genético de Holland e algumas de suas variações foram aplicados aos problemas de *job scheduling* com sucesso, motivo pelo qual escolheu-se este método para ser aplicado ao problema tema desta dissertação. No próximo capítulo será apresentado o algoritmo genético de Holland desenvolvido para problema de *job scheduling* deste trabalho.4 - O ALGORITMO GENÉTICO DE HOLLAND PARA O PROBLEMA DE *JOB SCHEDULING*

4.1 – ALGORITMO GENÉTICO APLICADO A PROBLEMAS DE *JOB SCHEDULING*

Algoritmos genéticos vem sendo usado com sucesso na resolução de alguns problemas, quando existem domínios onde o conhecimento específico não pode ser calculado computacionalmente em tempo aceitável, ora por permitir avaliar vários possíveis resultados simultaneamente, ora por serem abordagens computacionais globais, consequentemente não ficam presos a máximos locais.

A busca realizada por um algoritmo genético, não é totalmente aleatória, pois leva em conta a qualidade dos indivíduos da população corrente para determinar o próximo estado de busca. Também não é afetada por descontinuidade na função de avaliação dos indivíduos. Isto é muito útil para trabalhar problemas de *Job Scheduling*, motivo pelo qual existem vários trabalhos que propõem o uso de algoritmos genéticos, tais como:

- *A Two-Stage Genetic Algorithm for Large-Size Scheduling Problem:* propõe a resolução de problemas de programação em grandes escalas. Nele é proposto encontrar uma melhor solução para o problema de *job sheduling* através de uma abordagem computacional hierárquica de algoritmos genéticos, onde um algoritmo genético determina os parâmetros do segundo algoritmo genético, que é que encontra o resultado do problema (YIN,2007);
- *A New Approach for Planning and Scheduling Problems in Hybrid Distributed Manufacturing Execution System*: gera o planejamento e a programação industrial de

um produto. Como resultado os autores indicam que o planejamento obtido com um algoritmo genético reduziu em vinte por cento os prazos de vencimento, oito e meio por cento o armazenamento do produto de fabricação, e vinte e cinco por cento os problemas de qualidade (LIAU,2006);

- *An Improved Adaptive Genetic Algorithm for Job-Shop Scheduling Problem*: desenvolve uma variação do algoritmo genético de Holland para resolver o problema da programação de *job-shop*. Tal algoritmo trabalha a transmissão da hereditariedade das características excelentes de cromossomos pai e cria soluções melhores por operação periódica e não linear que ajustam o cruzamento e a probabilidade de mutação (XING,2007);
- *An accurate parallel genetic algorithm to schedule tasks on a cluster*: propõe uma abordagem computacional de algoritmo genético paralelo para o problema de programar tarefas múltiplas em um agrupamento de computadores. Tal algoritmo utiliza métodos de estimação matemática dos parâmetros de algoritmo genéticos paralelos (MOORE,2004);
- *An Indirect Genetic Algorithm for a Nurse Scheduling Problem*: faz uso de um algoritmo genético para o problema de programação dos serviços executados por enfermeiras em um hospital. Tal sistema trabalha um operador de cruzamento híbrido e uso de saltos simples para reduzir o tamanho do espaço de resultados a serem investigados na busca do resultado ótimo (AICKELIN,2004);
- *Building Better Nurse Scheduling Algorithms*: neste artigo é abordado o mesmo tema do artigo anterior, trata sobre programação dos horários das enfermeiras. No artigo, em resumo, o autor tenta desenvolver um método estatístico que compare os resultados obtidos e assim construir algoritmos mais eficientes (AICKELIN,2004).

Dada a natureza do problema desta pesquisa, optou-se pela implementação de uma variação do algoritmo genético de Holland, tendo em vista a complexidade encontrada em questões relacionadas ao problema de *Job Schedulin* de casos quando a cardinalidade do espaço de busca é grande.

4.2 - DESCRIÇÃO DO PROBLEMA DE *JOB SCHEDULING* IMPLEMENTADO

O setor de Gestão de Tecnologia da Informação do Instituto de Ciências Humanas Comunicação e Arte, pertencente a Universidade Federal de Alagoas, foi criado com o intuito de atender as solicitações de manutenção corretiva e preventiva do acervo tecnológico dessa unidade. Para tanto uma política de atendimento teve de ser definida, com as seguintes premissas: as solicitações de manutenção preventivas devem ser atendidas o mais rápido possível e as solicitações corretivas devem ser atendidas no prazo mínimo de vinte e quatro horas e máximo de quarenta e oito horas.

Todas as solicitações recebidas são divididas em duas filas uma para as solicitações preventivas e outra para as corretivas, a ordem em que as solicitações aparecem nessa lista deve respeitas a ordem em que as mesma foram realizadas. Para o presente trabalho foi considerado que o setor dispõe dos recursos materiais necessários para atender a todas as solicitações e que a prioridade das solicitações preventivas é maior do que a prioridade das solicitações corretivas.

A solicitação do serviço deverá ser feita mediante preenchimento do formulário próprio e encaminhamento do mesmo para o e-mail do responsável da Coordenação Gestão de Tecnologia da Informação. O coordenador deverá então no início de cada dia, cadastrar a lista de solicitações recebidas no dia anterior. Em seguida, o coordenador deverá informar as eventuais alterações de horário dos servidores dessa coordenação que já devem ter sido cadastrados anteriormente. Após estas tarefas, o coordenador deverá requerer que o sistema gere um plano de trabalho para o referido dia.

4.3 – O ALGORITMO GENÉTICO IMPLEMENTADO

Para o problema foco desta dissertação, baseado na associação entre as solicitações a serem atendidas, as habilidades funcionais dos servidores: as habilidades cadastradas para realizar os serviços de manutenção de computadores e o peso atribuído pelo administrador do sistema aos serviços fornecidos. Sendo de responsabilidade do coordenador do setor o

cadastramento dos serviços, funcionários e solicitações.

O peso dos serviços no momento atual é fixo em 3 para manutenção preventiva e 1 manutenção corretiva, sendo que estes valores podem ser alterados pelo administrador do sistema, podendo inclusive atribuir pesos diferentes aos serviços, em função de uma mudança da politica de atendimento do setor de Manutenção de Computadores.

No restante desta seção será especificado as modificações realizadas no modelo de Holland bem como a função de adaptação utilizada no algoritmo genético implementado para o presente trabalho.

4.3.1 - Cromossomo

Convencionou-se que o cromossomo é uma matriz com número de colunas igual ao número de solicitações cadastradas na base de dados, e com número de linhas igual ao período de funcionamento do Instituto, dividido em intervalos de trinta minutos, onde o conteúdo das suas células é um número natural correspondente ao número do servidor na base de dado que será alocado apara atender a solicitação. Devido a natureza deste problema ser de otimização combinatória optou-se por implementar uma variação do algoritmo genético de Holland primeiro na representação do cromossomo, como descrito acima, e em função desta representação foi necessário desenvolver novos operadores genéticos (cruzamento e mutação), ficando mantida a estratégia de seleção elitista e a nova população do algoritmo genético de Holland.

Por exemplo, considere a Tabela 4.1 que apresenta todas as solicitações que deverão ser atendidas pelos setor de Manutenção de Computadores com as respectivas habilidades técnicas necessárias para a sua realização (um total de quatro) e seu peso; a Tabela 2 que apresenta todos os servidores (um total de 5) com suas respectivas habilidades técnicas e seu horário.

Tabela 4.1 – Relação de solicitações de atendimento

$Habilidade_1$	$Habilidade_2$	$Habilidade_3$	$Habilidade_4$	$Habilidade_5$	$Peso$

$Solicitação_1$	1	1	1	1	1	1
$Solicitação_2$	1	0	0	0	1	1
$Solicitação_3$	0	0	0	1	0	3
$Solicitação_4$	1	1	0	0	1	1

Fonte: Autor em 2013

Segundo a Tabela 4.1 a *Solicitação₁* apresenta a *Habilidade₁*, *Habilidade₂*, *Habilidade₃*, *Habilidade₄* e *Habilidade₅* com peso 1; a *Solicitação₂* apresenta a *Habilidade₁* e *Habilidade₅* com peso 1; a *Solicitação₃* apresenta a *Habilidade₄* com peso 3; e a *Solicitação₄* apresenta as *Habilidade₁*, *Habilidade₂* e *Habilidade₅* com peso 1.

Tabela 4.2 – Relação de servidores

	$Habilidade_1$	$Habilidade_2$	$Habilidade_3$	$Habilidade_4$	$Habilidade_5$	*Início*	*Fim*
$Servidor_1$	1	1	1	1	1	08	121
$Servidor_2$	1	0	0	0	1	09	13
$Servidor_3$	0	0	0	1	0	08	12
$Servidor_4$	1	1	0	0	1	09	13
$Servidor_5$	0	1	1	1	0	08	12
$Servidor_1$	0	0	0	0	0	08	131

Fonte: Autor em 2013

Segundo a Tabela 4.2 acima o *Servidor₁* apresenta a *Habilidade₁*, *Habilidade₂*,

Habilidade$_3$, *Habilidade*$_4$ e *Habilidade*$_5$ com horário das 8 às 12 horas; o *Servidor*$_2$ apresenta a *Habilidade*$_1$ e *Habilidade*$_5$ com horário das 9 às 13 horas; o *Servidor*$_3$ apresenta a *Habilidade*$_4$ com horário das 8 às 12 horas; o *Servidor*$_4$ apresenta a *Habilidade*$_1$, *Habilidade*$_2$ e *Habilidade*$_5$ com horário das 9 às 13 horas; e o *Servidor*$_5$ apresenta as *Habilidade*$_2$, *Habilidade*$_3$ e *Habilidade*$_4$ com horário das 8 às 12 horas. O *Servidor*$_\lambda$ não tem habilidade; ele foi incluído nesta lista para ajudar a definir a função de adaptação do problema.

Um cromossomo é uma matriz de 10 linhas (porque o turno considerado no exemplo é de cinco horas) e 4 colunas cada uma representando uma solicitação da Tabela 4.3, onde cada célula $[i,j]$ contém o nome do servidor que irá atender a solicitação j no horário i, com $1 \leq i \leq 10$ e $1 \leq j \leq 4$, representa respectivamente o planejamento da execução das solicitações a serem atendidas.

Tabela 4.3 – Cromossomo para as solicitações da Tabela 4.1 com os servidores da Tabela 4.2

Horário	*Solicitação*$_1$	*Solicitação*$_2$	*Solicitação*$_3$	*Solicitação*$_4$
8:30-9:00	*Servidor*$_1$	*Servidor*$_\lambda$	*Servidor*$_\lambda$	*Servidor*$_\lambda$
9:00-9:30	*Servidor*$_1$	*Servidor*$_\lambda$	*Servidor*$_\lambda$	*Servidor*$_\lambda$
9:30-10:00	*Servidor*$_1$	*Servidor*$_5$	*Servidor*$_\lambda$	*Servidor*$_\lambda$
10:00-10:30	*Servidor*$_1$	*Servidor*$_5$	*Servidor*$_4$	*Servidor*$_\lambda$
10:30-11:00	*Servidor*$_\lambda$	*Servidor*$_5$	*Servidor*$_4$	*Servidor*$_\lambda$
11:00-11:30	*Servidor*$_\lambda$	*Servidor*$_\lambda$	*Servidor*$_4$	*Servidor*$_\lambda$
11:30-12:00	*Servidor*$_\lambda$	*Servidor*$_\lambda$	*Servidor*$_4$	*Servidor*$_\lambda$
12:00-12:30	*Servidor*$_\lambda$	*Servidor*$_1$	*Servidor*$_4$	*Servidor*$_2$
12:30-13:00	*Servidor*$_1$	*Servidor*$_1$	*Servidor*$_\lambda$	*Servidor*$_2$

Fonte: Autor em 2013

4.3.2 – Função de Adaptação

O valor da adaptação do cromossomo *Adaptação(c)* é dada pela seguinte

somatória das alocações em que o servidor é capaz de atender as habilidades exigidas pela solicitação dentro do seu horário de trabalho multiplicado pelo peso da solicitação, menos o valor do atendimento de uma solicitação por dois servidores ao mesmo tempo e menos o valor de alocar para o mesmo servidor duas solicitações no mesmo horário.

$$adaptação(c) = \sum_{i \in \{1,\ldots,N_{Período}\}} \sum_{j \in \{1,\ldots,N_{Solicitação}\}} peso(j) \times atende(c[i,j]) - \sum_{i \in \{1,\ldots,N_{Período}\}} \sum_{j \in \{1,\ldots,N_{Solicitação}\}} comparaLinha(i,j) \times 100 - \sum_{i \in \{1,\ldots,N_{Período}\}} \sum_{j \in \{1,\ldots,N_{Solicitação}\}} comparaColuna(i,j) \times 100$$

onde

- $N_{Período}$ é o número de períodos iguais em que foi dividido o tempo de funcionamento do setor de manutenção,
- $N_{Solicitação}$ é o número de solicitações a serem atendidas, e
- $peso(j)$ é o peso da j-ésima solicitação na Tabela 4.1.

A função atende $atende(i,j)$ informa se o funcionário alocado no planejamento está dentro do seu horário de trabalho e tem as habilidades necessárias para realizar a solicitação.

$$atende(j) = cargaHorária(c[i,j]) \times \#\{z \mid z \in (Habilidade(c[i,j]) \cup Habilidade(j))\}$$

onde

$Habilidade(s)$ é o conjunto de habilidades do servidor s,

$Habilidade(j)$ é o conjunto de habilidades da solicitação j, e

$\#\{z\}$ é a função cardinalidade, que recebe um conjunto e retorna o seu tamanho.

$$cargaHorária(i,s) = \begin{cases} 1 & \text{se } Início(s) \leq i \times 30 < Fim(s) \\ 0 & \text{caso contrário} \end{cases}$$

A função de $comparaLinha$ informa que se dois ou mais servidores são alocados para uma mesma solicitação eles devem ter alguma habilidade que os diferenciem dois a dois.

$$comparaLinha(i,j) = \begin{cases} 1 & \exists z \in \{1, \ldots, N_{Solicitação}\} \\ & (c[i,z] \neq Servidor_{\lambda}) \wedge (c[j,z] \neq Servidor_{\lambda}) \wedge (\#\, Habilidade(i,j,z) < 1) \\ 0 & \text{caso contrário} \end{cases}$$

onde

$$Habilidade(i,j,z) = (Habilidade(c[i,z]) - Habilidade(c[j,z])) \cup (Habilidade(c[j,z]) - Habilidade(c[i,z]))$$

A função de $comparaColuna$ informa se o mesmo servidor não pode ser alocado para duas ou mais solicitações no mesmo instantes de tempo.

$$comparaColuna(i,j) = \begin{cases} 1 & \#\{w \mid \forall z \in \{1, \ldots, N_{Período}\} (c[z,i] = c[z,j]) \Rightarrow (w = c[z,j])\} = 0 \\ 0 & \text{caso contrário} \end{cases}$$

Por exemplo, para o cromossomo c da Figura 4.3 considerando as Tabelas 4.1 e 4.2, o valor da adaptação do cromossomo é dada pela seguinte fórmula:

$$adaptação(c) = \sum_{i=1,\ldots,10} \sum_{j=1,\ldots,4} peso(j) \times atende(c[i,j]) - \sum_{i=1,\ldots,9} comparaLinha(i,i+1) \times 100 - \sum_{j=1,\ldots,3} comparaColuna(j,j+1) \times 100$$

$$
\begin{aligned}
adaptação(c) = {} & peso(1) \times atende(c[1,1]) + peso(1) \times atende(c[1,2]) + peso(1) \times atende(c[1,3]) \\
& + peso(1) \times atende(c[1,4]) + peso(2) \times atende(c[2,1]) + peso(2) \times atende(c[2,2]) \\
& + peso(2) \times atende(c[2,3]) + peso(2) \times atende(c[2,4]) + peso(3) \times atende(c[3,1]) \\
& + peso(3) \times atende(c[3,2]) + peso(3) \times atende(c[3,3]) + peso(3) \times atende(c[3,4]) \\
& + peso(4) \times atende(c[4,1]) + peso(4) \times atende(c[4,2]) + peso(4) \times atende(c[4,3]) \\
& + peso(4) \times atende(c[4,4]) + peso(5) \times atende(c[5,1]) + peso(5) \times atende(c[5,2]) \\
& + peso(5) \times atende(c[5,3]) + peso(5) \times atende(c[5,4]) + peso(6) \times atende(c[6,1]) \\
& + peso(6) \times atende(c[6,2]) + peso(6) \times atende(c[6,3]) + peso(6) \times atende(c[6,4]) \\
& + peso(7) \times atende(c[7,1]) + peso(7) \times atende(c[7,2]) + peso(7) \times atende(c[7,3]) \\
& + peso(7) \times atende(c[7,4]) + peso(8) \times atende(c[8,1]) + peso(8) \times atende(c[8,2]) \\
& + peso(8) \times atende(c[8,3]) + peso(8) \times atende(c[8,4]) + peso(9) \times atende(c[9,1]) \\
& + peso(9) \times atende(c[9,2]) + peso(9) \times atende(c[9,3]) + peso(9) \times atende(c[9,4]) \\
& + peso(10) \times atende(c[10,1]) + peso(10) \times atende(c[10,2]) + peso(10) \times atende(c[10,3]) \\
& + peso(10) \times atende(c[10,4]) - comparaLinha(1,2) \times 100 - comparaLinha(1,3) \times 100 \\
& - comparaLinha(1,7) \times 100 - comparaLinha(1,8) \times 100 - comparaLinha(1,9) \times 100 \\
& - comparaLinha(1,10) \times 100 - comparaLinha(2,1) \times 100 - comparaLinha(2,3) \times 100 \\
& - comparaLinha(2,4) \times 100 - comparaLinha(2,5) \times 100 - comparaLinha(2,6) \times 100 \\
& - comparaLinha(2,7) \times 100 - comparaLinha(2,8) \times 100 - comparaLinha(2,9) \times 100 \\
& - comparaLinha(3,1) \times 100 - comparaLinha(3,2) \times 100 - comparaLinha(3,4) \times 100 \\
& - comparaLinha(3,5) \times 100 - comparaLinha(3,6) \times 100 - comparaLinha(3,7) \times 100 \\
& - comparaLinha(3,8) \times 100 - comparaLinha(3,9) \times 100 - comparaLinha(3,10) \times 100 \\
& - comparaLinha(4,1) \times 100 - comparaLinha(4,2) \times 100 - comparaLinha(4,3) \times 100 \\
& - comparaLinha(4,5) \times 100 - comparaLinha(4,6) \times 100 - comparaLinha(4,7) \times 100 \\
& - comparaLinha(4,8) \times 100 - comparaLinha(4,9) \times 100 - comparaLinha(4,10) \times 100 \\
& - comparaLinha(5,1) \times 100 - comparaLinha(5,2) \times 100 - comparaLinha(5,3) \times 100 \\
& - comparaLinha(5,4) \times 100 - comparaLinha(5,6) \times 100 - comparaLinha(5,7) \times 100
\end{aligned}
$$

$$- comparaLinha(5,8)\times 100 - comparaLinha(5,9)\times 100 - comparaLinha(5,10)\times 100$$
$$- comparaLinha(6,1)\times 100 - comparaLinha(6,2)\times 100 - comparaLinha(6,3)\times 100$$
$$- comparaLinha(6,4)\times 100 - comparaLinha(6,5)\times 100 - comparaLinha(6,7)\times 100$$
$$- comparaLinha(6,8)\times 100 - comparaLinha(6,9)\times 100 - comparaLinha(6,10)\times 100$$
$$- comparaLinha(7,1)\times 100 - comparaLinha(7,2)\times 100 - comparaLinha(7,3)\times 100$$
$$- comparaLinha(7,4)\times 100 - comparaLinha(7,5)\times 100 - comparaLinha(7,6)\times 100$$
$$- comparaLinha(7,8)\times 100 - comparaLinha(7,9)\times 100 - comparaLinha(7,10)\times 100$$
$$- comparaLinha(8,1)\times 100 - comparaLinha(8,2)\times 100 - comparaLinha(8,3)\times 100$$
$$- comparaLinha(8,4)\times 100 - comparaLinha(8,5)\times 100 - comparaLinha(8,6)\times 100$$
$$- comparaLinha(8,7)\times 100 - comparaLinha(8,9)\times 100 - comparaLinha(8,10)\times 100$$
$$- comparaLinha(9,1)\times 100 - comparaLinha(9,2)\times 100 - comparaLinha(9,3)\times 100$$
$$- comparaLinha(9,4)\times 100 - comparaLinha(9,5)\times 100 - comparaLinha(9,6)\times 100$$
$$- comparaLinha(9,7)\times 100 - comparaLinha(9,8)\times 100 - comparaLinha(9,10)\times 100$$
$$- comparaLinha(10,1)\times 100 - comparaLinha(10,2)\times 100 - comparaLinha(10,3)\times 100$$
$$- comparaLinha(10,4)\times 100 - comparaLinha(10,5)\times 100 - comparaLinha(10,6)\times 100$$
$$- comparaLinha(10,7)\times 100 - comparaLinha(10,8)\times 100 - comparaLinha(10,9)\times 100$$
$$- comparaColuna(1,2)\times 100 - comparaColuna(1,3)\times 100 - comparaColuna(1,4)\times 100$$
$$- comparaColuna(2,1)\times 100 - comparaColuna(2,3)\times 100 - comparaColuna(2,4)\times 100$$
$$- comparaColuna(3,1)\times 100 - comparaColuna(3,2)\times 100 - comparaColuna(3,4)\times 100$$
$$- comparaColuna(4,1)\times 100 - comparaColuna(4,2)\times 100 - comparaColuna(4,3)\times 100$$

Para calcular o valor da adaptação tem-se antes que calcular o valor das funções $atende$, $comparaLinha$ e $comparaColuna$, já que a função $peso$ é simplesmente um acesso a um campo da Tabela 4.3, em seguida será mostrada na Tabela 4.4 com o valor da adaptação de cada célula do cromossomo, cuja adaptação é 20.

Tabela 4.4 – Peso atribuído as células do cromossomo da Tabela 4.3

Horário	$Solicitação_1$	$Solicitação_2$	$Solicitação_3$	$Solicitação_4$
8:30-9:00	3	0	0	0
9:00-9:30	3	0	0	0
9:30-10:00	3	0	0	0
10:00-10:30	3	1	0	0
10:30-11:00	3	1	0	0
11:00-11:30	0	1	0	0
11:30-12:00	0	0	0	0
12:00-12:30	0	0	0	2
12:30-13:00	0	0	0	2

Fonte: Autor em 2013

Note que a alocação do servidor $Servidor_4$ para a solicitação $Solicitação_3$ tem valor 0, pois o servidor não apresenta nenhuma das habilidades profissionais necessárias para atender a referida solicitação. A alocação do $Servidor_1$ para a solicitação $Solicitação_1$, faz com que a solicitação seja atendida parcialmente pois a solicitação requer todas as habilidades profissionais e o servidor que a está atendendo só possui o conjunto $\{Habilidade_1, Habilidade_4, Habilidade_5\}$. Ou seja, este cromossomo precisa da ação dos operadores genéticos para tentar melhorar sua adaptação. 4.3.2 – Operadores Genéticos de Cruzamento

Os operadores genéticos de cruzamento utilizados foram o de um ponto de corte na vertical e na horizontal, definidos da seguinte forma:

- o operador de cruzamento de um ponto de corte na vertical tem por objetivo combinar as colunas de dois cromossomos da seguinte maneira: sejam $A_{n\times m}$ e $B_{n\times m}$ dois

cromossomos pertencentes a população corrente do algoritmo genético; seleciona-se aleatoriamente um número x, pertencente ao conjunto $\{1,2,\ldots,m\}$ e então constrói-se dois $A'_{n\times m}$ novos cromossomos formado pela concatenação das colunas de 1 até x da matriz $A_{n\times m}$ com as colunas de $x+1$ até m da matriz $B_{n\times m}$ e $B'_{n\times m}$ formado pela concatenação das colunas de 1 até x da matriz $B_{n\times m}$ com as colunas de $x+1$ até m da matriz $A_{n\times m}$ (Figura 4.2).

Figura 4.2 - Cruzamento de um ponto de corte na vertical

Cromossomos originais

$$A=\begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \qquad B=\begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{bmatrix}$$

Cromossomos gerados

$$A'=\begin{bmatrix} a_{1,1} & \cdots & a_{1,x} & b_{1,x+1} & \cdots & b_{1,m} \\ \vdots & & & & & \vdots \\ a_{n,1} & \cdots & a_{n,x} & b_{n,x+1} & \cdots & b_{n,m} \end{bmatrix} \qquad B'=\begin{bmatrix} b_{1,1} & \cdots & b_{1,x} & a_{1,x+1} & \cdots & a_{1,m} \\ \vdots & & & & & \vdots \\ b_{n,1} & \cdots & b_{n,x} & a_{n,x+1} & \cdots & a_{n,m} \end{bmatrix}$$

Fonte: Autor 2013

A Figura 4.3 ilustra um exemplo deste cruzamento de um ponto de corte na vertical, considerando $x=3$, os cromossomos com células brancas são os originais e os com células cinzas são os construídos.

Figura 4.3 - Exemplo de cruzamento de um ponto de corte na vertical

		$\cdots x$		
3	0	0	0	3
3	0	0	0	3

		$\cdots x$		
3	0	0	**0**	**0**
3	0	0	**0**	**0**

3	0	0	0	3

3	0	0	**2**	**0**

0	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**2**	**0**

0	**0**	**0**	0	3
0	**0**	**0**	0	3
0	**0**	**0**	0	3

Fonte: Autor 2013

- operador de cruzamento de um ponto de corte na horizontal tem por objetivo combinar as colunas de dois cromossomos da seguinte maneira: seja $A_{n,m}$ e $B_{n,m}$ dois cromossomos pertencentes a população corrente do algoritmo genético; seleciona-se aleatoriamente um número y, pertencente ao conjunto $\{1,2,\ldots,n\}$ e então constrói-se dois novos cromossomos $A'_{n,m}$ formado pela concatenação das linhas de 1 até x da matriz $A_{n\times m}$ com as linhas de $x+1$ até m da matriz $B_{n\times m}$ e $B'_{n,m}$ formado pela concatenação das linhas de 1 até x da matriz $B_{n\times m}$ com as linhas de $x+1$ até m da matriz $A_{n\times m}$ (Figura 4.4).

Figura 4.4 - Cruzamento de um ponto de corte na horizontal

Cromossomos originais

$$\boldsymbol{A}=\begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \qquad \boldsymbol{B}=\begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{bmatrix}$$

Cromossomos gerados

$$A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \\ a_{x,1} & \cdots & a_{x,m} \\ b_{x+1,1} & \cdots & b_{x+1,m} \\ \vdots & & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{bmatrix} \qquad B' = \begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & & \\ b_{x,1} & \cdots & b_{x,m} \\ a_{x+1,1} & \cdots & a_{x+1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Fonte: Autor em 2013

A Figura 4.5 ilustra um cruzamento de um ponto de corte na horizontal, considerando $y = 2$, os cromossomos com células brancas são os originais e os com células cinzas são os construídos.

Figura 4.5 - Exemplo de cruzamento de um ponto de corte na horizontal

	3	0	0	0	3		3	0	0	0	3
$y \rightarrow$	3	0	0	0	3		3	0	0	0	3
	3	0	0	0	3		**0**	**0**	**0**	**2**	**0**

	0	**0**	**0**	**0**	**0**		**0**	**0**	**0**	**0**	**0**
$y \rightarrow$	**0**	**0**	**0**	**0**	**0**		**0**	**0**	**0**	**0**	**0**
	0	**0**	**0**	**2**	**0**		3	0	0	0	3

Fonte: Autor em 2013

4.3.3 – Operadores Genéticos de Mutação

Os operadores genéticos de mutação utilizados foram o de troca na vertical determinística, na horizontal determinística, na vertical aleatória e na horizontal aleatória. Definidos da seguinte forma:

- o operador de mutação de troca na vertical determinística tem por objetivo alterar a

cadeia de um cromossomo da seguinte maneira: seja $A_{n,m}$ um cromossomo da população corrente do algoritmo genético; seleciona-se aleatoriamente dois números x_1 e x_2, com $x_1 > x_2$ e ambos pertencentes ao conjunto $\{1,2,\ldots,m\}$, e então constrói-se um novo cromossomo $A'_{n,m}$ formado pela troca da coluna de x_1 com a x_2 da matriz $A_{n \times m}$ (Figura 4.6).

Figura 4.6 - Mutação de troca na vertical determinística

Cromossomos original

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Cromossomos gerado

$$A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,x_1-1} & a_{1,x_2} & a_{1,x_1+1} & \cdots & a_{1,x_2-1} & a_{1,x_1} & a_{1,x_2+1} & \cdots & a_{1,m} \\ \vdots & & & & & & & & & \cdots & \vdots \\ a_{n,1} & \cdots & a_{n,x_1-1} & a_{n,x_2} & a_{n,x_1+1} & \cdots & a_{n,x_2-1} & a_{n,x_1} & a_{n,x_2+1} & \cdots & a_{n,m} \end{bmatrix}$$

Fonte: Autor em 2013

A Figura 4.7 ilustra a mutação de troca na vertical determinística considerando $x_1 = 2$ e $x_2 = 5$, o cromossomo com células brancas é o original e o com células cinzas é o construído.

Figura 4.7 - Exemplo de mutação de troca na vertical determinística

		$\square x_1$		$\square x_2$
3	0	0	0	3
3	0	0	0	3
3	0	0	0	3

		$\square x_2$		$\square x_1$
3	0	3	0	0
3	0	3	0	0
3	0	3	0	0

Fonte: Autor em 2013

- o operador de mutação de troca na horizontal determinística tem por objetivo alterar a cadeia de um cromossomo da seguinte maneira: seja $A_{n,m}$ um cromossomo da população corrente do algoritmo genético; seleciona-se aleatoriamente dois números y_1 e y_2, com $y_1 > y_2$ e ambos pertencentes ao conjunto $\{1,2,\ldots,n\}$ e então constrói-se um novo cromossomo formado pela troca da linha de y_1 com a y_2 da matriz $A_{n\times m}$ (Figura 4.8).

Figura 4.8 - Mutação de troca na horizontal determinística

Cromossomos original

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Cromossomos gerado

$$A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{y_1-1} & \cdots & a_{y_1-1,m} \\ a_{y_2,1} & \cdots & a_{y_2,m} \\ a_{y_1+1} & \cdots & a_{y_1+1,m} \\ \vdots & & \vdots \\ a_{y_2-1} & \cdots & a_{y_2-1,m} \\ a_{y_1,1} & \cdots & a_{y_1,m} \\ a_{y_2+1} & \cdots & a_{y_2+1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Fonte: Autor em 2013

A Figura 4.9 ilustra a mutação de troca na horizontal determinística considerando $y_1 = 2$ e $y_2 = 4$, o cromossomo com células brancas é o original e o com células cinzas é o construído.

Figura 4.9 - Exemplo de mutação de troca na horizontal determinística

	3	0	0	0	3
$y_1 \rightarrow$	3	0	0	0	3
	3	0	0	0	3
$y_2 \rightarrow$	1	2	3	2	1

	3	0	0	0	3
$y_2 \rightarrow$	1	2	3	2	1
	3	0	0	0	3
$y_1 \rightarrow$	3	0	0	0	3

Fonte: Autor em 2013

- o operador de mutação de troca na vertical aleatória tem por objetivo alterar a cadeia de um cromossomo da seguinte maneira: seja $A_{n,m}$ um cromossomo da população corrente do algoritmo genético; seleciona-se aleatoriamente um números x, pertencentes ao conjunto $\{1,2,\ldots,m\}$ e então constrói-se um novo cromossomo $A'_{n,m}$ formado pela troca da coluna de x da matriz $A_{n \times m}$ por uma coluna gerada aleatoriamente (Figura 4.10).

Figura 4.10 - Mutação de troca na vertical aleatória

Cromossomos original

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Cromossomos gerado

$$A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,x-1} & ?_{1,x} & a_{1,x+1} & \cdots & a_{1,m} \\ \vdots & & & & & & \vdots \\ a_{n,1} & \cdots & a_{n,x-1} & ?_{n,x} & a_{n,x+1} & \cdots & a_{n,m} \end{bmatrix}$$

Fonte: Autor em 2013

A Figura 4.11 ilustra a mutação de troca na vertical aleatória considerando $x = 2$, o cromossomo com células brancas é o original e o com células cinzas é o construído.

Figura 4.11 - Exemplo de mutação de troca na vertical aleatória

	$\cdots x$			
3	0	0	0	3
3	0	0	0	3
3	0	0	0	3

	$\cdots x$			
3	0	1	0	3
3	0	2	0	3
3	0	1	0	3

Fonte: Autor em 2013

- o operador de mutação de troca na horizontal aleatória tem por objetivo alterar a cadeia de um cromossomo da seguinte maneira: seja $A_{n,m}$ um cromossomo da população corrente do algoritmo genético, seleciona-se aleatoriamente um números y, pertencentes ao conjunto $\{1,2,\cdots,n\}$ e então constrói-se um novo cromossomo $A'_{n,m}$ formado pela troca da linha de y da matriz $A_{n \times m}$ por uma linha gerada aleatoriamente (Figura 4.12).

Figura 4.12 - Mutação de troca na horizontal aleatória

Cromossomos original

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Cromossomos gerado

$$A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{y-1,1} & \cdots & a_{y-1,m} \\ ?_{y,1} & \cdots & ?_{y,m} \\ a_{y+1,1} & \cdots & a_{y+1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix}$$

Fonte: Autor em 2013

A Figura 4.13 ilustra a mutação de troca na horizontal aleatória considerando $y = 2$, o cromossomo com células brancas é o original e o com células cinzas é o construído.

Figura 4.13 - Exemplo de mutação de troca na horizontal aleatória

	3	0	0	0	3
$y \rightarrow$	3	0	0	0	3
	3	0	0	0	3
	1	2	3	2	1

	3	0	0	0	3
$y \rightarrow$	0	1	2	1	0
	3	0	0	0	3
	1	2	3	2	1

Fonte: Autor em 2013

4.3.4 – Garantia de Convergência

Deve-se reparar que dependendo do critério de parada adotado pelo algoritmo genético definido acima, uma solicitação pode não ser atendida completamente, devido a possibilidade de uma convergência prematura para uma solicitação ótima local. Por outro lado esta situação pode ocorrer quando o setor de Manutenção de Computadores não tiver condições de atender

totalmente a solicitação. O reflexo dessas duas situação é que o conjunto das habilidades dos funcionários alocados para uma solicitação não contém o conjunto das habilidades requeridas pelas operações a serem executadas para atendê-la. Tal situação possui vantagens e desvantagens. A vantagem é que o setor de Manutenção de Computadores sempre irá atender a solicitação no que for possível pelo seu conjunto de servidores. A desvantagem é que o trabalho do servidor pode ser inócuo, não resultando em alguma satisfação do solicitantes, e ainda ocasionando perda de tempo da atividade a ser realizada pelo funcionário.

Uma alternativa para impedir que uma solicitação seja atendida parcialmente por:

- convergência prematura é só parar a execução após 10 execuções em que a adaptação da população, que é a somatória da adaptação do cromossomos que a compõe, não seja alterada para mais;
- impossibilidade de atender e propor uma função de adaptação que faça a exigência de que uma solicitação só pode ter funcionários alocados a ela se ela for atendida completamente, como a função de adaptação abaixo:

$$
\begin{aligned}
adaptação(c) = & \sum_{i \in \{1, \ldots, N_{Período}\}} \sum_{j \in \{1, \ldots, N_{Solicitação}\}} peso(j) \times atende(c[i,j]) \\
& - \sum_{i \in \{1, \ldots, N_{Período}\}} \sum_{j \in \{1, \ldots, N_{Período}\}} comparaLinha(i,j) \times 100 \\
& - \sum_{i \in \{1, \ldots, N_{Solicitação}\}} \sum_{j \in \{1, \ldots, N_{Solicitação}\}} comparaColuna(i,j) \times 100 \\
& - \sum_{i \in \{1, \ldots, N_{Solicitação}\}} \#((\forall j \in \{1, \ldots, N_{Período}\}(\cup Habilidade(c[i,j]))) - Habilidade(i)) \\
& - \sum_{i \in \{1, \ldots, N_{Solicitação}\}} \#(Habilidade(i) - (\forall j \in \{1, \ldots, N_{Período}\}(\cup Habilidade(c[i,j]))))
\end{aligned}
$$

onde

- $N_{Periodo}$ é o número de períodos iguais em que foi dividido o tempo de funcionamento do setor de manutenção,

- $N_{Solicitação}$ é o número de solicitações a serem atendidas, e
- $peso(j)$ é o peso da j-ésima solicitação na tabela 1.

4.3 – CONSIDERAÇÕES FINAIS

O algoritmo genético desenvolvido para resolve o problema proposto não é uma aplicação direta do algoritmo genético de Holland, pois a representação adota para o cromossomo e a definição dos operadores de cruzamento e mutação não se adéquam ao problema de *Job Scheduling* tratado. Sendo necessário adotar uma outra representação para o cromossomo e novos operadores genéticos de cruzamento e mutação, os quais foram apresentados acima.

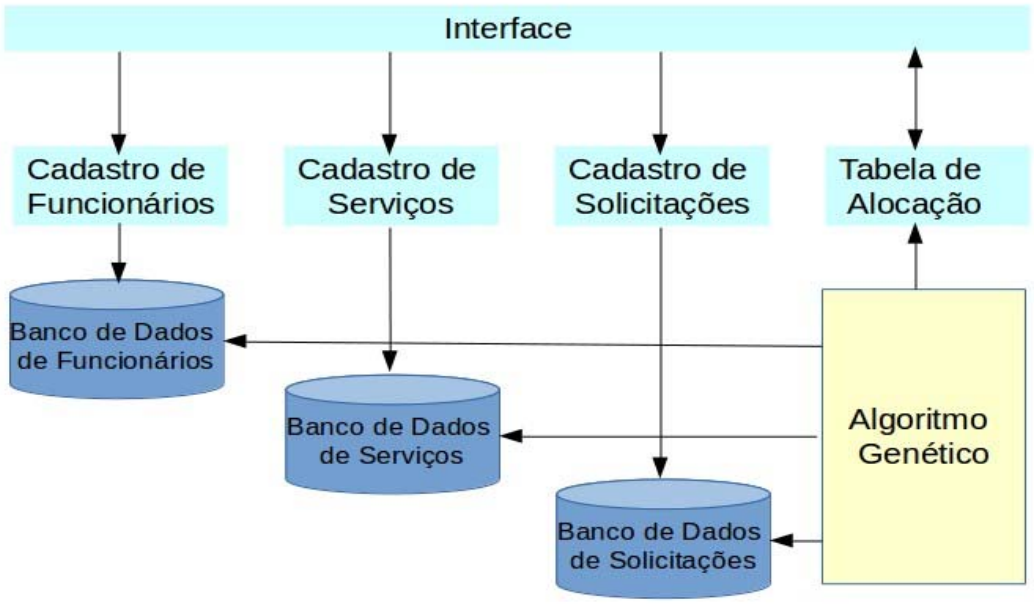
No próximo capitulo será apresentado o sistema desenvolvido cujo módulo principal é o algoritmo genético aqui especificado.

5 – O SISTEMAS DE GERENCIAMENTO DE *JOB SCHEDULING- SGJS*

5.1 - IMPLEMENTAÇÃO DA FERRAMENTA DE ALOCAÇÃO DE SERVIÇO PARA FUNCIONÁRIOS

A Figura 5.1 apresenta a arquitetura do sistema desenvolvido, a qual é composta de seis módulos denominados de *interface*, *cadastro de funcionários*, *cadastro de serviços*, *cadastro de solicitação*, *tabela de alocação* e *algoritmo genético*.; além de três bancos de dados denominados de *banco de dados de funcionários*, *banco de dados de serviços* e *banco de dados de solicitações*.

Figura 5.1 – Arquitetura do sistema



Fonte: Autor 2013

O sistem cadas trado s tenha m pelo meno s um servidor com as habilidades necessárias para realizá-lo.

O sistema foi desenvolvida utilizando o kit de desenvolvimento da linguagem Java versão 1.6.0 distribuído pela Sun Microsystens, em conjunto com a ferramenta de programação Eclipse versão 3.3.1.1. A escolha de uma implementação na linguagem Java

permitiu também que a ferramenta possa ser executada tanto no sistema operacional Windows como Linux.

No restante deste capítulo será apresentada uma descrição detalhada dos módulos e do funcionamento do sistema proposto.

5.2 - INTERFACE

No módulo *interface* foi adotado os elementos janelas, box de edição, box lista e botões das aplicações padrões do Windows, em função da popularidade deste sistema operacional, o que de certa forma torna a familiarização dos usuários ao sistema mais rápida, como mostra a Figura 5.2.

Ao acessar os ícones padrões da janela o usuário pode optar por:

- sair do sistema; ou
- minimizar a janela do sistema; ou
- redimensionar a janela do sistema, este ícone alterna entre a janela com dimensão igual a da tela e a janela com tamanho menor do que a tela, podendo este tamanho ser definido pelo usuário com o mouse.

Figura 5.2 – Janela Principal do Sistema

Fonte: Autora em 2013

Ao clicar no botão:

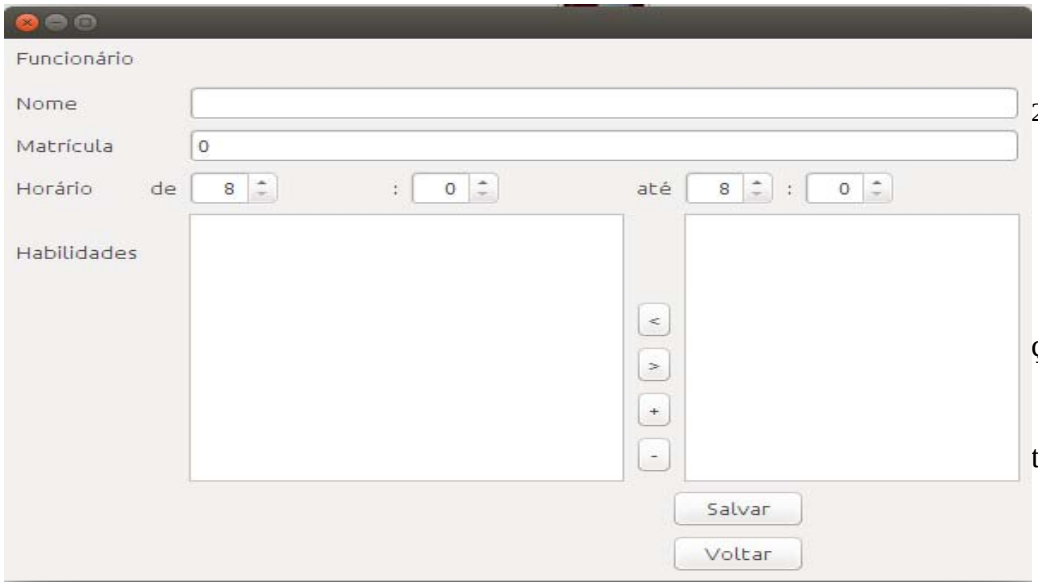
- **Funcionário** do sistema o usuário irá acionar o módulo *cadastro de funcionário*:
- **Serviços** do sistema o usuário irá acionar o módulo *cadastro de serviços*:
- **Solicitações** do sistema o usuário irá acionar o módulo *cadastro de solicitações*:
- **Tabela** do sistema o usuário irá acionar o módulo *tabela de alocação.*

5.2.1 - Modulo *cadastro de funcionários*

Ao ser acionado o módulo de *cadastro de funcionários* irá abrir a janela da Figura 5.3, composta por:

- um campo de edição para o nome do funcionário;
- um campo para edição da matrícula do funcionário;
- campos para registrar os horários de início e termino do turno do funcionário;
- uma lista com as habilidades já cadastradas;
- uma lista com as habilidades do funcionário;
- o botão

Figura 5.3 – Janela Cadastro de Funcionários



Fonte: Autora em 2013

A colocação de conceitos no mapa ocorre sempre que o professor clicar duas vezes o botão esquerdo do mouse em qualquer posição da área destinada a visualização do mapa conceitual, enquanto a colocação de uma relação ocorre quando o professor posicionar o mouse em cima de um conceito e pressionando o botão esquerdo do mouse levar o mouse até um outro conceito.

Ao ser inserido um novo conceito ou relação no mapa conceitual o Dedalus lhe atribui um nome padrão tipo: UndefinedX e EdgeY, com X e Y sendo um número natural correspondendo a ordem em que o conceito ou a relação foi inserida no mapa. Para alterar o nome o professor deve clicar uma vez o botão esquerdo do mouse sobre uma circunferência ou a seta de um arco para que apareça na área destinada a nomeação uma caixa exibindo o nome atual do elemento gráfico selecionado, ao professor escrever o novo nome neste local e pressionar a tecla <Enter> o novo nome sera exibido na mapa (Figura 4.7 e Figura 4.8).

Figura 4.7 – Tela do Dedalus com o Menu Módulos
Fonte: Autora em 2013

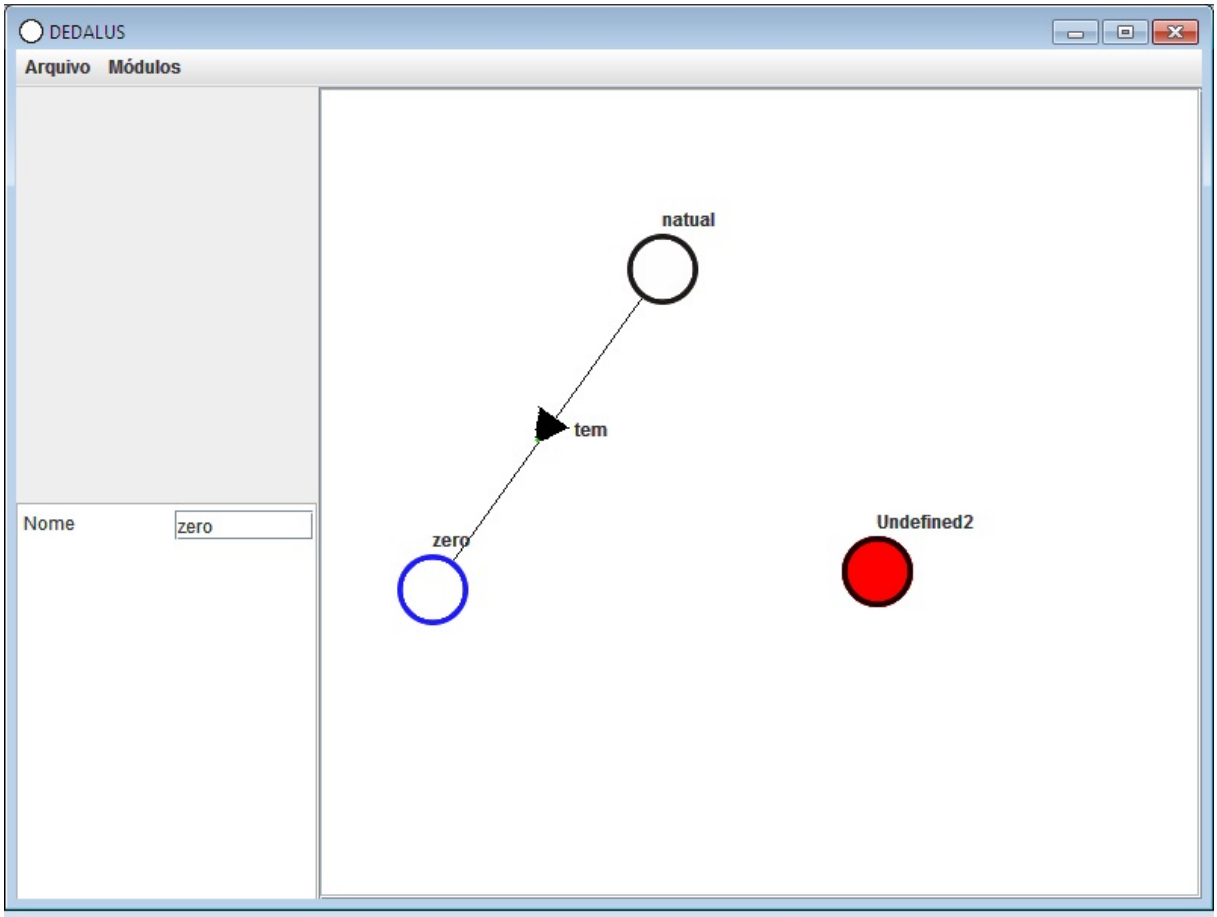
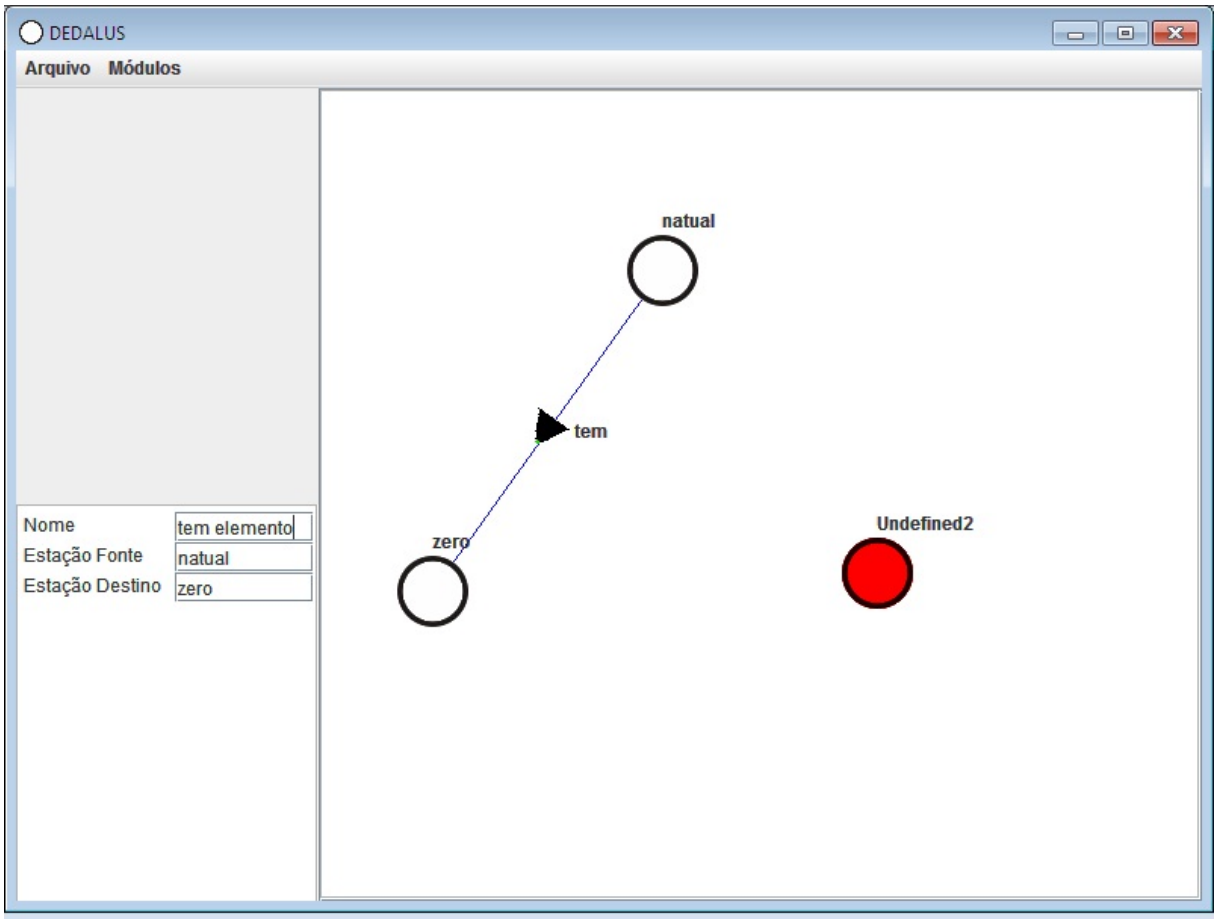


Figura 4.8 – Tela do Dedalus para nomear relações
Fonte: Autora em 2013



Caso o professor por algum engado forneça duas vezes um determinado nome para um conceito, o sistema ira colocar a área interna da circunferência que representa os dois conceitos de vermelho para mostrar que isso constituí um erro na construção do mapa por definição, ver seção 3.6. Mas se o professor não perceber este aviso e tentar salvar o mapa conceito como duplicidade de conceito, o Dedalus ira exibir uma caixa de dialogo com a seguinte mensagem "Não é permitido salvar enquanto houver dois ou mais nós com nomes repetidos".

Se o professor desejar apagar um conceito ou uma relação do mapa, ele deve clicar uma vez o botão esquerdo do mouse sobre a área interna da circunferência ou a seta do arco, e em seguida pressionar a tecla <Delete> para que o conceito ou a relação seja apagado do mapa conceitual. Ao apagar um conceito todos os arcos do qual estes conceito participar

também serão apagados do mapa.

O professor pode alterar a posição dos conceitos para evitar a sobreposição dos arcos que representam as relações do mapa, bastando para isso colocar o mouse sobre o conceito de partida ou chegada do arco desta relação, pressionar o botão esquerdo do mouse e movê-lo para a nova posição a ser ocupada pelo conceito, liberando então o botão esquerdo do mouse.

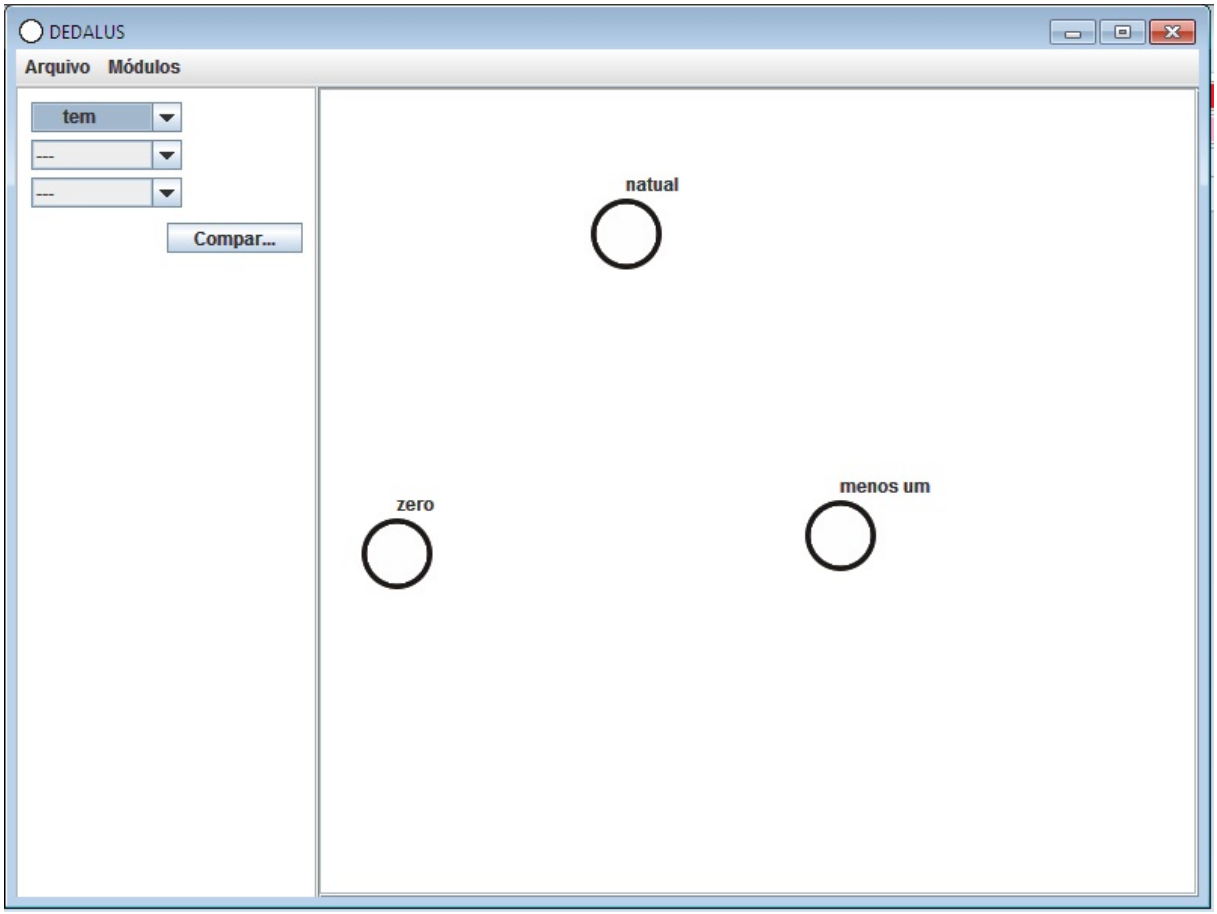
As circunferências ligadas por arcos tem sua área interna colorida com a cor branca e as que não encontram-se ligadas tem sua área interna colorida de vermelha (Figura 4.8). Dessa foma o usuário saberá quais conceitos ainda precisam ser relacionados aos conceitos já aprendidos.

4.3.2 - Modulo Organizador

O papel do aluno no Dedalus é organizar os conceitos e as relações presentes no mapa conceitual construído pelo professor. Sua interação com o Dedalus se dá por meio de uma interface gráfica dividida em duas partes a qual lhe permite visualizar os conceitos e as relações do mapa conceitual que esta sendo organizado. No lado direito da janela do Dedalus esta a área de visualização do mapa conceitual, enquanto no lado esquerdo superior da janela está a área destinada edição das relações do mapa conceitual.

O aluno deve acessar a opção **Arquivo** do menu do Dedalus e escolher a sub-opção de **Abrir**, para que uma janela com o nome dos diretórios e arquivos no computador apareça, em seguida o aluno devera escolher o nome do mapa conceitual que deseja organizar. Então o Dedalus ira mostrar na área destinada a visualização de sua janela os conceitos presentes no mapa escolhido e na área de organização das relações uma lista com o nome das relações e os possíveis conceitos a serem associados como conceito de partida e chegada de uma relação (Figura 4.9).

Figura 4.9 – Tela do Dedalus para nomear relações: exemplo números naturais
Fonte: Autora em 2013



Para organizar o mapa o aluno deve escolher o nome de uma relação, para depois colocar o nome do primeiro conceito da relação na lista de conceitos de partida, e em seguida colocar o nome do segundo conceito da relação na lista de conceitos de chegada, neste instante surgira na área da janela do Dedalus destinada a visualização a relação que acaba de ser construída.

A qualquer momento o aluno pode solicitar que o seu mapa seja comparado com o mapa do professor, com o intuito de saber se seu mapa esta correto, para isso basta pressionar o botão **Compara** na área da janela do Dedalus. O Dedalus então irá apresentar uma janela com um valor informando o número de incompatibilidades entre o mapa do aluno e o do professor.

Caso o aluno deseje remover do mapa que esta sendo organizado uma relação ele pode escolher a relação e fornecer como conceitos de partida ou chegada o nome "----". Mas se ele desejar somente alterar o nome do conceito de partida ou de chegada da relação, então o aluno deve escolher o nome da relação e na lista do conceito que deseja alterar ele deve fornecer o nome de outro conceito. Como resultado desta ação o arco da relação a ser alterada ira ser removido e um novo arco surgira com o nome de relação ligando dois outros conceitos.

4.4 - MÓDULO DE ARMAZENAMENTO

Os mapas conceituais são grafos que precisam ser convertidos em estruturas de dados apropriadas para a sua escrita e leitura em arquivos do tipo texto. No Dedalus a tarefa de codificar e decodificar um mapa conceitual do formato texto em sua estruturas de dados e vice-versa é de responsabilidade do modulo de armazenamento.

O entendimento de como o processo de codificação e decodificação de mapas conceituais em texto ocorre reque que o leitor compreenda tanto as estruturas de dados interna usada pelo Dedalus para representar o mapa conceitual, como o comportamento do seu modulo de escrita e leitura de arquivo no formato texto.

4.4.1 - Estrutura de Dados do Dedalus

As estruturas de dados manipuladas pelo Dedalus são: lista simplesmente encadeada, n-uplas e matriz de adjacência. Uma lista é uma estrutura de dados que contém uma coleção de objetos da mesma natureza definidos por um conjunto de parâmetros, tendo como propriedade a possibilidade de ter um tamanho não limitado e a repetição do mesmo elemento na coleção, desde que a operação de inserção de novos elementos na lista não faça nenhuma

exigência com relação ao número de ocorrências de um determinado objeto na lista.

Na lista os elementos são disposto em uma sequência, no caso da lista simplesmente encadeada cada elemento da lista sabe qual é a lista de elementos que o sucede, sendo que o ultimo elemento da lista aponta para a lista vazia representada por “[ ]”. Por convenção neste trabalho a lista será representada por uma sequencia de elementos separados por virgula e delimitados pelo caracteres “[“ e “]”. Por exemplo, a lista dos cinco primeiros números impares é [**1**,**3**,**5**,**7**,**9**].

O primeiro elemento de uma lista é denominado de cabeça da lista e a lista resultante da remoção da cabeça da lista é denominada de corpo da lista. Para a lista [**1**,**3**,**5**,**7**,**9**] a cabeça é o elemento **1** e o corpo a lista [**3**,**5**,**7**,**9**].

A busca de um elemento na lista se dá pela acesso do elemento da cabeça da lista, se este for o elemento que se procura basta retornar o seu conteúdo, caso contrário o elemento da cabeça do corpo da lista deverá ser acessado. Este processo deve ser repetido até que o corpo da lista seja igual a lista vazia, informando que a lista foi toda percorrida e que o elemento buscado não foi encontrado. Por exemplo, a busca pelo terceiro número impar na lista [**1**,**3**,**5**,**7**,**9**] se processa da seguinte forma:

- passo 1, de posse da lista [**1**,**3**,**5**,**7**,**9**] acessa-se a cabeça dela que é o primeiro número impar, como este não é o número buscado então a busca continua
- passo 2, de posse do corpo da lista [**1**,**3**,**5**,**7**,**9**] que é a lista [**3**,**5**,**7**,**9**], acessa-se a cabeça dele que é o segundo número impar, como este não é o número buscado então a busca continua
- passo 3, de posse do corpo da lista [**3**,**5**,**7**,**9**] que é a lista [**5**,**7**,**9**], acessa-se a cabeça dele que é o terceiro número impar, retornando o elemento **5** como resposta da busca.

Uma relação é uma n-úpla, onde n é a aridade dos termos relacionados, no caso especifico dos mapas conceituais as relações são binárias. A estrutura de dados usadas para armazenar as relações binárias dos mapas conceitual manipulados pelo Dedalus é uma matriz R_{NOME} de dimensão $m \times w$, onde n é a cardinalidade do conjunto de conceitos de partida da relação (denominado de domínio da relação), w é a cardinalidade do conjunto de conceitos de

chegadas da relação (denominado de imagem da relação) e *NOME* é o nome da relação. A célula $R_{NOME}[i,j]$ da matriz R_{NOME} armazena o valor 1 se o i-ésimo conceito do domínio da relação relaciona-se através da relação *NOME* com o j-ésimo conceito da imagem da relação.

A matriz de adjacência para mapas conceituais A é uma variação da estrutura de dados matriz de adjacência usada para representar grafos em procedimentos computacionais. Basicamente uma matriz de adjacência para mapas conceituais é uma matriz quadrada $n \times n$, onde n é igual ao número de nós de um mapa. O valor armazenado no elemento A[i,j], com $1 \leq i \leq \mathbf{n}$, é uma lista vazia (representado aqui por "[ ]") se não existir nenhum arco que comece no nó i e chegue ao nó j no mapa, e uma lista das relações que ligam o nó i ao nó j no mapa caso contrário.

A matriz abaixo ilustra a matriz de adjacência do mapa conceitual da Figura 4.8.

$$\bar{A} = \begin{bmatrix} [\,] & [r_1(natural,zero), r_2(natural,zero)] & [r_3(natural,um), r_4(natural,um)] \\ [\,] & [\,] & [\,] \\ [\,] & [\,] & [\,] \end{bmatrix}$$

As relações contidas na matriz de adjacência acima são representadas nas relações abaixo:

- Relação elemento neutro da operação de adição (r_1)

natural	zero

- Relação elemento neutro da operação de subtração (r_2)

natural	zero

- Relação elemento neutro da operação de multiplicação (r_3)

natural	um

- Relação elemento neutro da operação de divisão (r_4)

natural	um

- Relação têm (r_5)

natural	zero
natural	um

A posição ocupada por um conceito é armazenada em uma tabela de posições que armazena o nome do conceito, sua coordenada X e sua Coordenada Y.

4.4.2 - Codificação e Decodificação de Mapas Conceituais em Texto

Para que os usuários da ferramenta Dedalus tenham capacidade de salvar, abrir e copiar seu mapa conceitual, é necessário que o mesmo encontre-se armazenado em um dispositivo de memória de forma apropriadas. No Dedalus os conceitos e relações dos mapas conceituais são armazenados em um arquivo texto de acordo com o seguinte fluxograma:

Figura 4.10 – Fluxograma do Módulo de Codificação do Dedalus
Fonte: Autora em 2013

Segundo esta figura, o processo de codificação das estruturas do Dedalus que representam um mapa conceitual em um arquivo texto começa no módulo de escrever a posição dos conceitos do mapa. Este módulo escreve em cada linha do arquivo texto o seguinte termo [*posicao*(<*coordenada*$_X$>, <*coordenada*$_Y$>, <*nome-conceito*>)] onde :

- <*coordenada*$_X$> informa a posição da coordenada do eixo cartesiano *X* do centro da circunferência que representa o conceito <*nome-conceito*>,
- <*coordenada*$_Y$> informa a posição da coordenada do eixo cartesiano *Y* do centro da circunferência que representa o conceito <*nome-conceito*>, e
- <*nome-conceito*> informa o nome do conceito cuja posição do centro da circunferência é o *ponto*(<*coordenada*$_X$>, <*coordenada*$_Y$>).

No próximo módulo do processo de codificação será escrito o nome das relações presentes no mapa. Este módulo escreve em cada linha do arquivo texto o seguinte termo [*relacao*(<*conceito*$_1$>, <*conceito*$_2$>, <*nome-relacao*>)] onde :

- <*conceito*$_1$> informa o nome do conceito cujo arco que representa a relação irá partir,
- <*conceito*$_2$> informa o nome do conceito cujo arco que representa a relação irá chegar,
- <*nome-relacao*> informa o nome da relação que parte do conceito <*conceito*$_1$> e chega no conceito <*conceito*$_2$>.

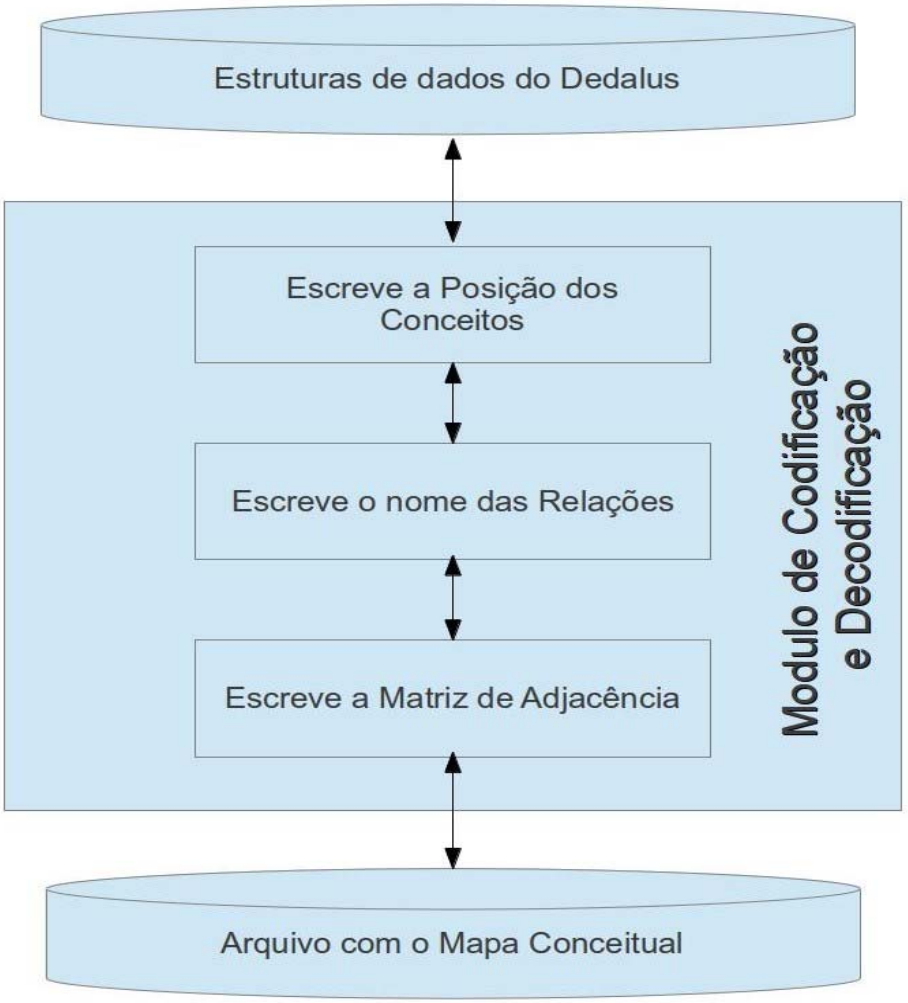
O último módulo do processo de codificação será responsável por registrar a matriz de adjacência de mapas conceituais conforme descrito na Seção 4.4.1. Este módulo escreve em cada linha do arquivo texto o seguinte termo [*celula*(<*conceito*$_1$>, <*conceito*$_1$>, [<*lista_relacao*>])] onde :

- <*conceito*$_1$> informa o nome do conceito da linha da matriz,
- <*conceito*$_2$> informa o nome do conceito da coluna da matriz,
- <*lista-relacao*> informa o nome da relação separadas por virgula que parte do conceito

<*conceito*$_1$> e chega no conceito <*conceito*$_2$>.

Na Figura 4.11, o processo de decodificação do arquivo texto que representam um mapa conceitual nas estruturas do Dedalus começa no módulo Ler a Posição dos Conceitos, cujo comportamento é de ler em cada linha do arquivo texto o termo [*posicao*(<*coordenada*$_X$>, <*coordenada*$_Y$>, <*nome_conceito*>)] até não existir mais este termo. Em seguida, o módulo Ler o Nome das Relações presentes no mapa, cujo comportamento é ler em cada linha do arquivo texto o termo [*relacao*(<*conceito*$_1$>, <*conceito*$_2$>, <*nome-relacao*>)] até que não exista mais registro do referido termo. Por fim, o módulo Ler Matriz de Adjacência de mapas conceituais, cujo comportamento é ler em cada linha do arquivo texto o termo [*celula*(<*conceito*$_1$>, <*conceito*$_2$>, [<*lista_relacao*>])] até que o fim do arquivo seja encontrado.

Figura 4.11 – Fluxograma do Módulo de Decodificação do Dedalus
Fonte: Autora em 2013



4.5 - MÓDULO DE COMPARAÇÃO

Dados duas matrizes de adjacência A_P e A_A para uma mesma planta baixa, onde A_P é a matriz de adjacência de professor e A_A é a matriz de adjacência do aluno. Considerando que os conjuntos de conceitos e relações das A_P e A_A são os mesmos. Logo deve-se esperar que elas correspondam a o mesmo mapa conceitual. Mas durante o processo de ensino- aprendizagem podem ocorrer entendimentos equivocados ou mesmo a ausência do registro de uma determinada propriedade do conceito que esta sendo apreendido, sendo necessário então que o aluno possa ter uma forma de avaliar como esta seu aprendizado de desenho ao concluir uma tarefa.

No Dedalus, o módulo responsável por ajudar o aluno a avaliar seu aprendizado é o módulo de comparação. A finalidade deste módulo é verificar o número de divergências entre

o mapa conceitual do aluno e o do professor. Para tanto o Dedalus faz o seguinte calculo:

$\exists\ comparacao \in \square\ (\ (zera(comparacao)) \wedge$

$(\forall\ i,j \in \{1..,n\}\ ((A_P[i,j] \neq_{lista} A_A[i,j]) \rightarrow incrementa(comparacao))))$

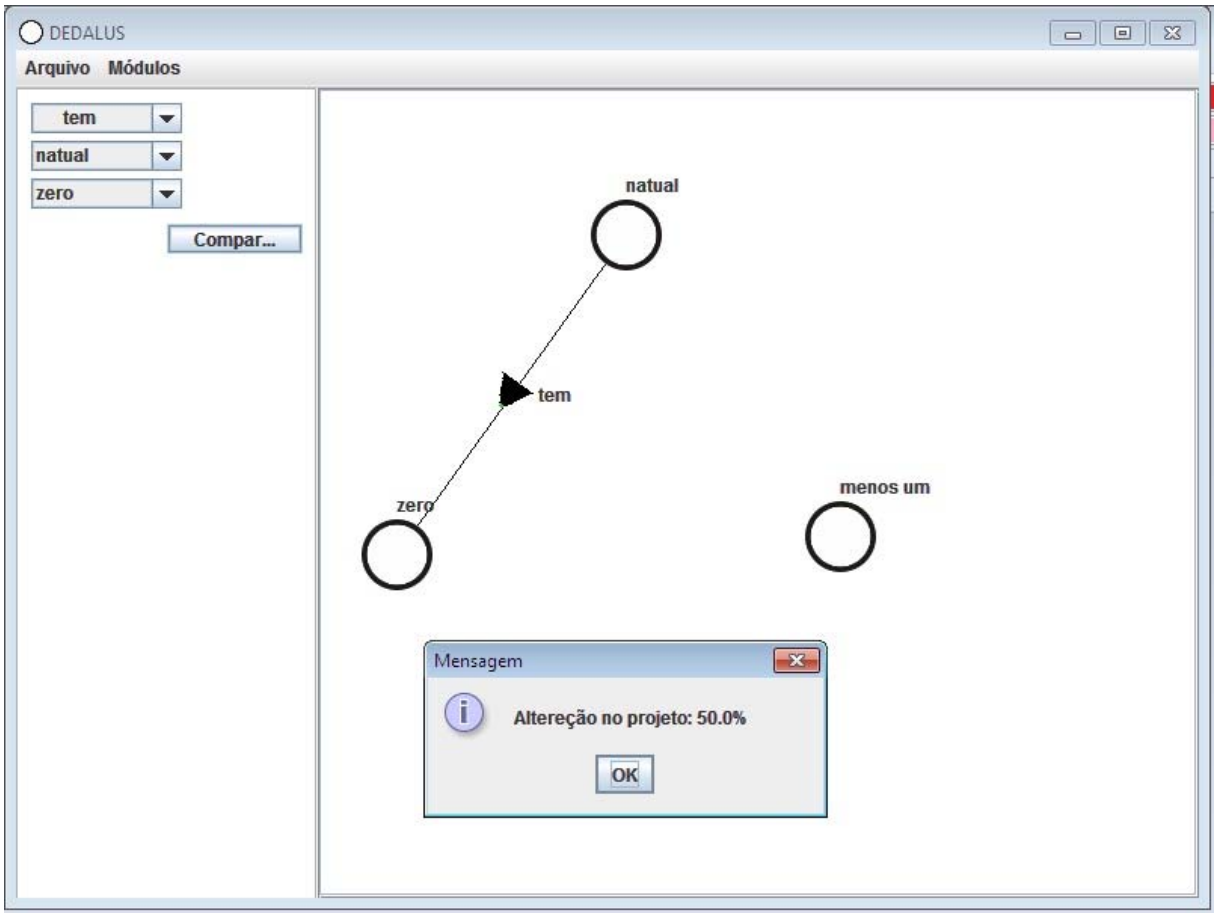
onde *n* é a cardinalidade do conjunto de conceitos presentes no mapa conceitual do professor, *zera*(*comparacao*) é um predicado que atribui o valor zero a variável *comparacao*, e *incrementa*(*comparacao*) é um predicado que incrementa de uma unidade a variável *comparacao*.

Em seguida é realizado o seguinte calculo

$$avaliacao = \frac{comparacao \times 100}{n_{arcos}^{professor}}$$

onde $n_{arcos}^{professor}$ é o número de arcos no mapa conceitual do professor. O Dedalus então exibe uma mensagem como a da Figura 4.12.Figura 4.12 – Resposta do módulo de comparação do Dedalus

Fonte: Autora em 2013



4.6 - CONSIDERAÇÕES FINAIS

O professor deverá acessar a ferramenta Dedalus para editar o mapa conceitual correspondente ao assunto visto na sala de aula, ou a atividade proposta como exercício para os alunos, ou ao texto lido, etc. O mapa construído pelo professor deverá ser convertido em um arquivo texto para ser armazenado no repositório de mapas conceituais (memória do computador ou unidade de armazenamento secundária).

O aluno devera escolher um mapa já construído pelo professor para organizar, de modo a ratificar seu processo de aprendizado correto dos novos conhecimentos apresentados em sala de aula. A ferramenta desenvolvida fornecera ao aluno os conceitos e as relações existentes no mapa conceitual do professor referentes ao mapa escolhido. Nesse momento o aluno deverá iniciar o processo de organização dos conceitos e relações apresentados na tela

do sistema.

O Dedalus fornecerá ao aluno uma operação denominada de comparação, que pode ser usada a qualquer momento durante o processo de organização de um mapa conceitual, com o único objetivo de quantificar para o aluno as relações presentes no seu mapa que encontram-se associada a conceitos diferentes no mapa do professor.

No próximo capítulo sera descrito a metodologia de uso do Dedalus aplicada na presente pesquisa bem como uma análise quantitativa dos resultados obtidos com esta metodologia aplicada a uma turma do curso de Arquitetura e Urbanismo do Centro Universitário CESMAC na disciplina Desenho Técnico I.

A abordagem computacional desenvolvida apresenta algumas limitações, tais como: os serviços oferecidos pelo setor de Manutenção de Computadores devem ser decompostos em tarefas que possam ser realizadas em um intervalo de tempo múltiplo de trinta minutos.

5.7 - CONSIDERAÇÕES FINAIS

Durante a pesquisa foi possível observar que o uso da ferramenta Dedalus libertou o aluno da percepção sistematizada de uma planta baixa, os alunos percebem as características da edificação representada pelo desenho de forma livre, neste momento foi observado ainda que cada aluno iniciou sua leitura da planta baixa de uma forma diferente, gerando no final o mesmo mapa conceitual.

A melhora nas notas do grupo significativo deve-se ao fato de que o aluno pode "checar" os conhecimentos obtidos durante o processo de leitura do desenho com os conhecimentos do professor através da ferramenta Dedalus. Mesmo assim, pode-se considerar que as habilidades manuais dos alunos precisam ser melhor trabalhadas, ou os mapas conceituais do professor devem ser mais elaborados, isso porque as notas das provas dos alunos que utilizaram o Dedalus não foi dez.

6 - CONCLUSÃO

6.1 - CONSIDERAÇÕES SOBRE O TRABALHO REALIZADO

Após propor a adoção de mapas conceituais como ferramenta computacional para o processo de aprendizagem significativa da disciplina Desenho Técnico I para os alunos dos curso de Arquitetura e Urbanismo do CESMAC, pode-se destacar 2 (dois) aspectos: o primeiro é do ponto de vista metodológico, quando é utilizada uma teoria de aprendizagem bem consolidada no ensino de disciplinas bases para o desenho técnico como geometria (Soares,2009) (Gerson,1995). Um segundo aspecto a ser considerado é que, o uso do módulo aluno não permitir que o usuário tivesse liberdade para edição de seu mapa conceitual na ferramenta Dedalus, limitando sua experiência e conhecimentos aos conceitos e relações presentes no mapa conceitual do professor.

Quanto a proposta de ensino baseada na teoria de Ausubel para a disciplina Desenho Técnico I assistida pela ferramenta Dedalus, percebe-se que este se distingue da maior parte das propostas de ensino para a referida disciplina assistida por computado, uma vez que a ferramenta Dedalus não se propõe a execução do desenho mais a sua leitura, pois se o aluno não conseguem ler uma planta baixa não adianta o professor solicitar que o mesmo vá a prancheta e a execute.

Apesar dos resultados satisfatórios para esta pesquisa, entende-se que o uso da ferramenta Dedalus no ensino da disciplina Desenho Técnico I, poderiam gerar informações ainda mais consistentes, principalmente se forem levados em consideração as exigências da ABNT associadas aos conceitos presentes no mapa conceitual.

Através da proposta do modelo desenvolvido espera-se que, a partir de agora, que os professores da disciplina Desenho Técnico I utilizam a ferramenta Dedalus como uma alternativa computacional que os auxilie na minimização de problemas presentes no processo de ensino e aprendizado como a compreensão equivocada ou incompleta de um conceito,

buscando uniformizar o conhecimento mínimo necessário aos alunos dessa disciplina para que os mesmos possam continuar sue processo de aprendizado nas próximas disciplinas que tenham a disciplina desenho Técnico I como pré-requisito.

6.2 - TRABALHOS FUTUROS

Para trabalhos futuros, pretende-se aprofundar os estudos relacionados aprendizagem significativa e ao ensino assistido por computador para a disciplina Desenho Técnico I, como também buscar-se-á a minimização de problemas relacionados às interações entre professores-alunos e alunos-alunos, a medida que a linguagem do desenho passa a fazer parde do conhecimento de todos.

No que concerne às possibilidades de trabalhos voltados para a melhoria da ferramenta Dedalus, há o interesse de que seja trabalhada uma circunferência com um ícone gráfico para os conceitos do mapa conceitual. Existe ainda a possibilidade de que sejam desenvolvidas uma visão estratificada do mapa conceitual, de modo que em uma visão macro os conceitos sejam correspondentes as partes que compões a edificação, e ao expandir o nó de uma determinado conceito se tenha acesso ao mapa conceitual dos detalhes da edificação do conceito selecionado.

7 - REFERÊNCIA BIBLIOGRAFICA

ANDERSSON, B., The experimental gestalt of causation: *a common core to pupils preconceptions in science*, **European Journal of Science Education**, volume 8, páginas 155-171, 1986.

ANOHINA, A. E GRUDSPENKIS, J., A Concept Map Based Intelligent System for Adaptive Knowledge Assessment, **Proceeding of the 2007 conference on Databases and Information Systems**, IV: Selected Papers from the Seventh International Baltic Conference, 2007.

AUSUBEL, D. P., NOVAK, J. D. E HANESIAN, H., **Psicologia educacional**, Rio de Janeiro, Interamericana Ltda, 623p, 1980

AUSUBEL, D. P., NOVAK, J. D. E HANESIAN, H., **Educational psychology: *A cognitive view***, 2ª edição, New York: Holt, Rinehart and Winston, 733, 1978.

AUSUBEL, D. P., In Defense of Advance Organizers: *A Reply to the Critics*, **Review of Educational Research**, número 2, Spring, páginas 251-257, volume 48, 1978.

AUSUBEL, D. P., NOVAK, J. D. E HANESIA, H., **Psicologia educacional.** Tradução de Eva Nick, Rio de Janeiro, Interamericana Ltda, 1980.

BACKES, R. J., O Ato de desenhar: *do desenvolver da percepção à construção da representação*, **XIV- Congresso Internacional de Ingenieria Gráfica**. Universidade de Santa Cruz, RS, Brasil. Departamento de História e Geografia. Santander, Espanha, 2002.

BARROS, T. F. G. E CORREIA, A. M. A., Quebrando Tabus: o ensino do Desenho Arquitetônico no curso de Engenharia Civil, **Graphica**, 2007.

SAVIANI, D., **Política e educação no Brasil**, 3ª edição, Campinas, Autores associados, 1996.

BRAULT, F., **Vectorworks Architect Getting Started Guide**, Disponível em: <http://www.vectorworks.net/training/free_resource.php>. Acesso em: 12 novembro 2012.

BÜRDEK, B. E., **Diseño. Historia, teoría y práctica del diseño industrial.**, Gustavo Gili,

S.A., 1994.

CHING, F. D. K., **Representação gráfica em arquitetura**, 5ª edição, Bookman, 2011.

CUNHA, L. V., **Desenho Técnico**, 14ª edição, Fundação Calouste Gulbenkian. ISBN: 9789723110661, 2008.

PAVEL SHVAIKO, JÉRÔME EUZENAT, TOM HEATH, CHRISTOPH QUIX, MING MAO ANDISABEL F. CRUZ, **Proceedings of the 6th International Workshop on Ontology Matching**, Bonn, Germany, October 24, 2011.

QUENTIN REUL, JEFF Z. PAN AND DEREK H. SLEEMAN, **Towards a framework for ontology mapping based on description logic reasoning**, OM, CEUR-WS.org, CEUR Workshop Proceedings, volume 814, 2011.

LEONARDO CARBONARA E DEREK H. SLEEMAN, Effective and Efficient Knowledge Base Refinement, **Machine Learning**, volume 37, número 2, páginas 143-181, 1999.

DRIVER, R., Psicología cognoscitiva y esquemas conceptuales de los alunnos, **Enseñanza de las Ciencia**, páginas 3-15, volume 4, 1986.

DRIVER, R., Un enfoque constructivista para el desarrollo del currículo de ciencias, **ENSEÑANZA DE LAS CIENCIAS**, páginas 109-120, volume 6, 1988.

EKLUND, J., SAWERS, J. E ZEILIGER, R., A tool for the collaborative construction of knowledge through constructive navigation, **The Fifth Australian World Wide Web Conference, Southern Cross University Press**, páginas 396-408, R. Debreceny & A. Ellis, 1999..

ESTEPHANIO, C., **Desenho Técnico: *uma linguagem básica***, 2ª edição, C. Estephanio, 1994.

FRENCH, T. E. E VIERCK, C.. **Desenho técnico e tecnologia gráfica**, Tradutor: Eny Ribeiro Esteves, 8ª edição, Globo, ISBN:8525007331, 1989.

GERSON, H. B. P., **Aplicação de novas tecnologias no ensino e aplicação do desenho**,

Dissertação no Departamento de Construção Civil e Urbana, Universidade de São Paulo, 1995.

GERSON, H. B. P., Uma comparação crítica entre os cursos de desenho que utilizam o computador, **XXVI CONGRESSO BRASILEIRO DE ENGENHARIA**, 1998.

GILBERT, J. K., Children's science and its consequences for teaching, **Research in Science Education**, Spring, páginas 623-633, volume 66, 1982.

GRIZ, C.; CARVALHO, G. L. E PEIXOTO, A., Cognitive Ergonomy: *A combination of many didactics resources for the Architectural Drawing teaching*, **Proceedings of the 11th Iberoamerican Congress of Digital Graphics**, México, 2007.

HEMERO, SOFTWARE LTDA, **ArchiStation 2012 ® Tutorial Básico**, Disponível em: <http://www.archistation.com/aprendizagem.asp>. Acesso em: 7 janeiro 2012.

LINKE, R. D. E VENZ, M. I., Misconceptions in physical science among non-science background students, **Research in Science Education**, Spring, páginas 103-109, volume 9, 1979.

LINSINGEN, I. E ET. ALL, **Formação do engenheiro: *desafios da atuação docente, tendências curriculares e questões da educação tecnológica***, Universidade Federal de Santa Catarina, 1999.

MACKENZIE, S. E GILBERT, S., **ArchiCAD for AutoCAD Users**, Graphisoft SE, ISBN:9789638875051, 2011.

MICELI, M. T. E FERREIRA, P., **Desenho Técnico Básico**, 3ª edição, Imperial Novo Milenio, ISBN: 9788599868393, 2008.

MOREIRA, M. A, MASINI, E. F. S., **Aprendizagem Significativa: A teoria de David Ausubel**, Moraes, 1982.

MOREIRA, M. A. E MASINI, E. F. S., **A teoria da Aprendizagem Significativa e sua implementação em sala de aula**, Universidade de Brasília, 2006.

MOREIRA, M. A., **Ensino e Aprendizagem - Enfoques Teóricos**, 3ª edição, Moraes, 1983.

MOREIRA, M. A., **Teorias de aprendizagem**, EPU, 1999.

NOVAK, J. E GOWIN, D., **Learning How to Learn**, Cornell Univ. Press, 1984.

NOVAK, J. D. E CANAS, A. J., The Theory Underlying Concept Maps and How to Construct and Use Them, **Technical Report IHMC CmapTools 2006-01 Rev 2008-01**. Institute for Human and Machine Cognition, 2006.

NOVAK, J. D. E GOWIN, D.B., **Aprender a aprender**, Plátano Edições Técnicas, 1996. Tradução ao português, de Carla Valadares, do original **Learning how to learn**. 212p.

NOVAK, J. D., GOWIN, D. B., E JOHANSEN, G. T., The use of concept mapping and knowledge vee mapping with junior high school science students, **Review of Science Education**, Spring, páginas 625-645, volume 67, 1983.

NOVAK, J. D., **Learning, Creating, and Using Knowledge: *Conceptual Maps as Facilitative Tools in Schools and Corporations***, Lawrence Erlbaum Associates, 1998.

NOVAK, J. D., **Theory of education**, Cornell University Press, 1977.

NOVAK, J. D., How do we learn our lesson? *Taking students through the process*, **Science Teacher**, número 3, Spring, páginas 50-55, volume 60, 1993.

OLIVEIRA, B. T. E AITA, T., O ensino da matéria de desenho no 1º, 2º e 3º graus, **Revista do Centro de Tecnologia**, 1985.

PEREIRA, F., **Tutorial rápido -- CMAP TOOLS**, Universidade Federal de Alagoas, Apostila da Faculdade de Economia, Administração e Contabilidade, 2007.

PÉREZ, C. C. C. E VIEIRA, R., Mapas Conceituais: *geração e avaliação*, **XXV Congresso da Sociedade Brasileira de Computação**, Sociedade Brasileira de Computação, Universidade do Vale do Rio dos Sinos, 2005.

PÉREZ, D. G., La metodología científica y la enseñanza de las ciencias. Unas relaciones controvertidas, **Enseãanza de las Ciencias**, páginas 111-121, volume 4, 1986.

POZO, J. I., A aprendizagem e o ensino de fatos e conceitos, Os conteúdos na reforma, páginas 17-71, **Artes médicas**, 1998.

PREECE, P. F. W., Mapping Cognitive Structure: *A Comparison of Methods*, **Jornal of Education Psychology**, número 1, volume 86, 1976.

ROCHA, F. E. L., LOPES, R. V. V., COSTA JUNIOR, J. V. E FAVERO, E. L., Especificação de um Algoritmo Genético para auxiliar na Avaliação da Aprendizagem Significativa com Mapas Conceituais, **Simpósio Brasileiro de Informática na Educação**, páginas 139-148, Editora da Universidade Federal do Amazonas, volume 1, 2004.

ROSA, M. A. AND BUCHWEITZ, B., Alunos bons solucionadores de problemas de física: *Caracterização a partir da análise de testes de associação de conceitos*, **Revista Brasileira de Ensino de Ensino de Física**, número 1-4, páginas 52-60, volume 15, 1993.

RUSSEL, S. E NORVIG, P., **Inteligência Artificial**, 2ª edição, Campus, 2004.

SANTOS, E. O., Ambientes virtuais de aprendizagem, **Revista FAEBA**, número 18, Autorias livre, plurais e gratuitas, volume 29, 2003.

SANTOS, M. E. V. M., **Mudança conceitual na sala de aula: um desafio epistemologicamente fundamentado**, Livros Horizonte, 1998.

SILVA, A. E TAVARES, C., **Desenho Técnico Moderno,** 4ª edição, LTC, 2006.

SOARES, L. H., **Aprendizagem Significativa na Educação Matemática: *uma proposta para a aprendizagem de Geometria Básica***, Dissertação em Educação da Universidade Federal da Paraíba, 2009.

TABER, K. S., Student reaction on begin introduced to concept mapping, **Physics Educations**, volume 29, 1994.

ULBRICHT, S. M., **Geometria e desenho: *história, pesquisa e evolução***, Florianópolis, 1998,

VARGAS, M., **Tecnologia, técnica e ciência, Metodologia da Pesquisa Tecnológica**, Globo,

1985.

VIEIRA, R. V., **Um Algoritmo Genético baseado em Tipos Abstratos de Dados e sua especificação em Z**, Centro de Informática, Universidade Federal de Pernambuco, 2003.

Zahar, J., **História Ilustrada da Ciência. Colin A. Ronan. Da Renascença a Revolução Científica**, edição 1, 2001.