Universidade Católica Portuguesa

Faculdade de Engenharia

Planeamento de Estratégias de Salvaguarda e Reposição de Dados/Informação baseado em Algoritmo de Optimização de requisitos Multidimensionais

Luis Miguel Ferreira Fernandes

Dissertação para obtenção de Grau de Mestre em Segurança de Sistemas de Informação

Júri

Prof. Doutor Manuel José Martinho Barata Marques (Presidente)

Prof. Doutor Rui Jorge Correia Mendes Alves Pires

Prof. Doutor Tito Lívio dos Santos Silva (Orientador)

Fevereiro de 2014

Resumo

Estudos recentes publicados em revistas da especialidade demonstraram que 50% de todas as companhias que sofreram acidentes graves e que não recuperaram no espaço de 10 dias úteis, nunca recuperaram financeiramente, e 93% das empresas sem um plano de Continuidade de Negócio e Recuperação de Desastres (*Disaster Recovery*/*Business Continuity*), ficam fora do negócio no espaço de 5 anos após um desastre (perda relevante de dados) grave.

É, portanto, natural que a Continuidade de Negócio seja um tema cada vez mais presente na agenda da maioria dos decisores das organizações, independentemente da sua natureza (privadas vs. organismos públicos), dimensão e actividade.

O receio generalizado relativamente à ocorrência de situações imprevistas, como catástrofes naturais ou algum tipo de ataque informático, ou outro tipo de incidências mais banais mas com uma probabilidade de ocorrência relativamente mais elevada, como avarias de equipamentos ou incêndios, faz com que assegurar a continuidade dos processos de negócio, independentemente do tipo de incidente, seja uma preocupação fundamental.

Esta preocupação crescente surge como um objectivo não apenas tecnológico mas, desejável e principalmente, de negócio.

Em organizações (hoje praticamente todas) centradas na informação, a chave para a sobrevivência a uma situação classificada como "desastre" é simultaneamente preventiva (proteger a informação) e reactiva (conseguir recuperá-la em tempo útil). Como corolário, temos então uma dependência de uma convergência de acções, nomeadamente:

- Conhecer os principais negócios da organização e os principais factores de risco ao nível do negócio que afectam a organização, bem como o impacto ao nível das receitas
- Perceber que processos de negócio proteger e de que tipo de desastres/ameaças podem estar sujeitos
- Como proteger e respectivo grau de exposição ao desastre/ameaça
- Validar o estado de preparação da organização para responder a uma situação de perda de informação e assim manter a capacidade de continuidade de negócio

Pretende-se com este trabalho criar uma *Framework* que suporte superiormente as organizações com dependências desta natureza, proporcionando eficiências significativas nos esforços de salvaguarda e reposição de dados nomeadamente, na gestão das janelas temporais, na utilização de soluções tecnológicas (optimização da utilização de recursos, tarefas e operações de requisitos multidimensionais (servidores e robots, salvaguarda e reposição de informação, por exemplo) e, no limite, mitigando a necessidade destas estarem

dotadas de recursos humanos especializados no tema e toda a envolvente financeira necessário.

Neste trabalho obtiveram-se os seguintes resultados:

- Implementação de uma bancada de ensaios (ambiente de trabalho, gerador, simulador e visualizador de Gantt)
- O escalonamento automático
- A eficácia e eficiência de um conjunto limitado de abordagens algorítmicas para o cenário objectivo
- Elaboração de planos de salvaguarda e reposição devidamente optimizados com representação de Gantt

Descreve-se de forma sumária os resultados tangíveis:

- Redução de custos no recurso recorrente a especialistas de Continuidade de Negócio e Recuperação de Desastres (*CN/RD)*
- Melhor gestão da janela global de salvaguarda e reposição
- Permitir ao técnico (de salvaguarda e reposição) ter disponíveis os planos de salvaguarda e reposição mediante uma optimização de requisitos, recursos, operações e tarefas
- Possibilidade de executar simulações em sistemas de reais e medição de impacto de alterações recentes nos sistemas de informação ou de suporte à salvaguarda e reposição, assim como a identificação antecipada de constrangimentos futuros
- Ordenação dos melhores desempenhos com sucesso na procura de soluções através da seguinte demonstração de resultados:

Algoritmo	**T**arefas x **O**perações			
	20	**40**	**80**	**160**
Genético	2,84	4,65	11,86	30,53
Prog. Linear	3,56	9,56	38,96	65,79
Ganancioso	16,5	23,02	161,88	944.851,44
Regras	22,43	22,49	Inf	Inf
Aleatório	Inf	Inf	Inf	Inf

Palavras-chave: *Job-Shop*, Escalonamento, multidimensionais, salvaguarda, reposição, *framework*, continuidade, tempo crítico, tempo de execução, janela temporal, algoritmo, RTO, RPO.

Abstract

Recent studies published in peer-reviewed journals have shown that 50% of all companies that have suffered severe injuries and not recovered within 10 days, never recovered financially, and 93% of companies without a plan for Business Continuity and Disaster Recovery (Disaster Recovery / Business Continuity), are out of business within five years after a severe disaster (loss of relevant data).

It is therefore natural that Business Continuity is an increasingly issue on the agenda of most decision makers of organizations, regardless of their nature (vs. private. public), size and activity.

The widespread concern for the occurrence of unforeseen situations such as natural disasters or some kind of hacking, or other more mundane incidents but with a probability of occurrence relatively higher, such as equipment failure or fire, makes continuity business processes, regardless of the type of incident is a major concern. This comes as a growing concern objective not only technological but desirable and especially business.

Organizations (now virtually all) focused on information, the key to survival in a situation classified as "disaster" is both preventive (to protect the information) and reactive (able to retrieve it in time). As a corollary, we then have a dependency on a convergence of actions, including:

- Know the main business of the organization and the main risk factors at the level of business that affect the organization as well as the impact on revenue
- Understand business processes have to be protect and what kind of disasters / threats may be subject
- Protecting and their degree of exposure to disaster / threat
- Validation of the state of preparedness of the organization to respond to a situation of loss of information and thus maintain the business continuity

The better the quality level of these actions, the greater the probability of success in business continuity organizations.

The aim of this work is to create a framework that supports organizations with superior facilities of this nature, providing significant efficiencies in efforts to safeguard and replacement of data in particular, in the management of temporal windows, the use of technological solutions (optimal use of resources, tasks and operations requirements multidimensional (Backups robots, backup and restore information, for example) and, ultimately, alleviating the need for these are endowed with human resources specialized in the subject and all surrounding financial need.

The Framework will be supported by the best algorithm for each scenario, according to the Job-Shop model in which a set of scenarios is subjected to a staggering through a component (scheduler) set to test the algorithm with the aim to simplify and streamline procedures rehabilitation and safeguarding, through a management requirements, resources, tasks and operations in order to:

- Estimate time for the Protection and Recovery
- Manage plans for the Protection and Replacement
- Develop reporting procedures

Using this framework described and following the objectives listed is intended to demonstrate:

- The feasibility of the resource
- Automatic Scaling
- The effectiveness and efficiency of a limited set of algorithmic approaches to the objective scenario
- Plans for safeguarding and properly optimized replacement

It is described in summary form the tangible results:

- Cost reduction in recurrent appeal to specialists in Business Continuity and Disaster Recovery (BC / RD)
- Better management and safeguard global window replacement
- Allow the coach (safeguard and replacement) to have plans available to safeguard and replacement by an optimization requirements, resources, tasks and operations
- Ability to run simulations of real systems and measuring the impact of recent changes in information systems or support for the preservation and replacement, as well as early identification of future constraints
- Ordering of the best performances successfully in finding solutions through the following income statement:

Algorithm	**T**askd x **O**perations			
	20	**40**	**80**	**160**
Genetic	2,84	4,65	11,86	30,53
Prog. Linear	3,56	9,56	38,96	65,79
Greedy	16,5	23,02	161,88	944.851,44
Heuristics	22,43	22,49	Inf	Inf
Random	Inf	Inf	Inf	Inf

Keywords: Scaling, multidimensional safeguard replacement framework, continuity, critical time, run time, time window, Algorithm, RTO, RPO

Agradecimentos

Foram muitos os constrangimentos que nos últimos anos obrigaram a adiamentos sucessivos na entrega desta tese, desde incompatibilidades de disponibilidades, situações pessoais e profissionais, até às normais dificuldades de implementação, tudo contribuiu para a entrega prevista em Outubro de 2011 apenas acorra quase 2 anos depois.

No entanto e além do esforço pessoal, para reunir as condições de entrega, é merecida a referência e o agradecimento a pessoas, desde colegas de disciplinas deste mestrado, profissionais que acharam interesse no tema, particularmente na bancada de ensaios, e que de uma forma outra contribuíram para a implementação da bancada de testes, para todos fica o registo do agradecimento já transmitido pessoalmente.

Finalmente, o agradecimento muito especial pelo apoio e compreensão á minha esposa e ao meu filho, Cristina e Miguel.

Índice

Índice de figuras

Índice de Tabelas

Abreviaturas

Abreviaturas	Descrição
PCN	Plano de Continuidade de Negócio
POR	Perda admissível de Informação numa organização (*Recovery Point Objective*)
RTO	Tempo de indisponibilidade admissível ao nível de informação numa organização (*Recovery Time Objective*)
SI	Sistemas de Informação
SI/TI	Sistemas de Informação/Tecnologias de Informação
TI	Tecnologias de Informação
BIA	Análise de impacto de negócio (*Business Impact Analysis*)
PRD	Plano de Recuperação de Desastre
TICs	Tecnologias da informação e comunicação
AG	Algoritmo Genético
CPU	Central Processing Unit
GUI	Graphical User Interface
HTN	Hierarchical Task Network
BIA	Análise de Impacto de Negócio

Capitulo 1. Introdução

Os métodos de optimização através de algoritmos matemáticos foram muito apreciados na década de 60 por estes serem eficazes na resolução de problemas pequenos (dimensão), razão da necessidade de algo mais para problemas maiores e de grande importância nos diversos sectores do mundo real.

Começam então os métodos heurísticos a ganhar relevância e importância perante os bons resultados conseguidos [Jain, Meeran, 1999].

Num ambiente de monitorização e controlo de sistemas, o processo de gestão, planeamento e atribuição de operações e tarefas que asseguram o bom funcionamento de uma rede de máquinas, tem assim uma importância fulcral.

Surge assim a relevância do modelo de *Job-Shop scheduling* n x m (Wang e Zheng, 2002), que será utilizado neste trabalho como base para o objectivo proposto.

Este trabalho está estruturado em 7 capítulos para ilustrar da forma mais simples possível todo o trabalho, nomeadamente:

- Capitulo 1, uma antevisão do que é o objectivo deste trabalho, e os motivos na origem da sua escolha, a identificação do problema a resolver, e como resolver
- Capitulo 2, analisa-se o resultado de análise de mercado, e enumera-se conclusões
- Capitulo 3, aborda-se a metodologia seguida, descrição do problema *Job-Shop* em *Scheduling*, a preparação para a bancada de ensaios e respectiva estrutura, definição do simulador ao nível da estrutura e operacionalidade
- Capitulo 4, apresentação e análise de resultados obtidos, uma apresentação de resultados procedida de comentários e analise aos mesmos
- Capitulo 5, apresenta-se as conclusões gerais, contribuições do trabalho e perspectiva de trabalho futuro
- Capitulo 6, apresentação de informação importante para uma melhor compreensão do trabalho
- Capitulo 7, referencia a documentação utilizada e que contribuiu de forma significativa para a elaboração deste trabalho

1.1. Motivação

A escolha do tema deve-se às preocupações crescentes das organizações perante situações da actualidade que põem em causa a continuidade de negócio, e a eficiência dos planos de Salvaguarda e Recuperação, não só perante desastres naturais como também a eventuais falhas das tecnologias. Actualmente, embora existam soluções disponíveis para esta temática, estas encontram-se dispersas por soluções proprietárias e integradas, de elevado custo financeiro, que requerem a existência de um especialista da área nos quadros das organizações devido à complexidade de utilização, não sendo assim acessível a todas as organizações.

A mais-valia reside principalmente em preencher o vazio existente nas organizações nesta temática, através de soluções de código aberto de baixa complexidade, personalizáveis e consequentemente de reduzidos custos financeiros.

As matérias apresentadas nesta dissertação são de extrema relevância no actual contexto organizacional: as organizações têm sofrido um constante e significativo desgaste resultante de perda temporária ou permanente de informação, provocando prejuízos elevados desde as perdas financeiras até à desconfiança dos de clientes e fornecedores.

A capacidade para uma resposta eficaz e eficiente numa Gestão de Interrupções de serviço é cada vez mais um elevado factor decisivo na sobrevivência de uma organização (Sikich,2003).

1.2. Objectivos

O objectivo é resolver um problema actual e fulcral para as organizações através de uma solução simples recorrendo à utilização de um modelo de *Job-Shop scheduling* n x m (Wang e Zheng, 2002).

Procura-se nesta dissertação criar uma prova de conceito da exequibilidade de um sistema automático para escalonamento em tempo real (*online Scheduling*). A implementação será materializada através de um simulador criado para o efeito.

Com esse objectivo, utiliza-se um modelo de *Job-Shop* para testar a eficiência de diversos algoritmos. Pode conjecturar-se um cenário onde as tarefas de manutenção e funcionamento de um sistema complexo são escalonadas por uma aplicação de *software* – o escalonador, sendo encaminhada para as diversas máquinas que cumprem as tarefas (os recursos).

Usa-se uma abordagem dinâmica (*online Scheduling*), na qual as tarefas chegam ao escalonador ao longo de um período temporal.

Os objectivos desta tarefa de testes experimentais são os seguintes:

- Provar a exequibilidade do recurso a escalonamento automático no contexto de Recuperação e Continuidade
- Observar comportamentos e tendências no processo de escalonamento automático
- Determinar, de entre um conjunto limitado de abordagens, qual a mais eficaz e eficiente no contexto em causa.

Tem-se Tarefas heterogéneas que partilham uma *pool* de recursos, da qual retiram temporariamente (inteira ou parcialmente) um conjunto de recursos para realizar uma lista de Operações.

As tarefas, definidas pelos operadores, são organizadas por prioridades e/ou precedências, formando um grafo acíclico. Este grafo será submetido ao algoritmo para determinar qual a melhor sequenciação das tarefas através da utilização de algoritmos de utilização comum e de características multidimensionais de forma a minimizar o tempo de processamento total, designado por *makespan* (Gonçalves, 2005), o resultado será reflectido num gráfico de *Gantt*.

Até ao momento tem sido feito referência a um problema fulcral e actual das organizações, esse problema é o novo paradigma da Continuidade de Negócio e a Recuperação de Desastres ao nível dos seus sistemas de informação.

1.3. Continuidade de Negócio e Recuperação de Desastres (CN/RD)

Na nossa vida quotidiana protegemo-nos ao máximo, através da aquisição de seguros pessoais, seguros de imóveis, viaturas, saúde, e outras, com o objectivo de nos salvaguardarmos de situações imprevistas ou mesmo desastres. Similarmente é expectável que os responsáveis das organizações e em especial de TI tenham especial atenção às infraestruturas críticas que são a base de suporte da organização no modo como toleram e reagem a situações de desastre.

Os sistemas de Informação têm uma importância de tal forma relevante, que são decisivos na saúde das organizações, o que implica que a Continuidade de Negócio (*Business Continuity*) e a Recuperação de Dados (*Disaster* Recovery) seja uma preocupação fundamental.

A capacidade de responder de forma eficaz e eficiente é sem dúvida um pilar da sobrevivência de qualquer organização.

De acordo com um estudo da *Gartner* (*Computerworld* 2010), a maioria das pequenas e médias empresas (PME) não investem o suficiente (ou nada mesmo) no planeamento da Continuidade de Negócio (BC) e na recuperação em caso de desastres (RD). De acordo com este estudo, estima-se que apenas 35% das PME possuem Planos de Recuperação (PRD) em caso de desastre, e menos de 10% possuem planos de gestão de crise, contingência, recuperação e retorno do negócio

Segundo o mesmo estudo, duas em cada cinco empresas que sofrem um desastre, acabam por abandonar o negócio no espaço médio de cinco anos pelo que é crítico implementar um Plano de Recuperação em caso de desastre. Estando os desastres a acontecer com mais frequência do que esperado, 80 % da inactividade das aplicações está relacionada com falhas humanas e processuais e não em desastres naturais ou falhas tecnológicas.

O estudo em causa é referente a organizações em que os seus mercados possuem uma maior maturidade ao nível operacional e de gestão de risco em comparação com a realidade nacional, o que sugere que na realidade portuguesa o cenário descrito será ainda mais negativo.

Com planos de continuidade desactualizados, concebidas de forma incorrecta, ou mesmo inexistente, as empresas poderão perder negócios cruciais para a sua sobrevivência, ou até mesmo sair da área de mercado.

A humanidade tem vindo a prestar cada vez mais atenção a este tipo de preocupações, mas já os Faraós se preocupavam com o armazenamento de cereais para um determinado momento que achavam vir a precisar (Serrano&Jardim, 2007).

Existem várias fases que antecedem a implementação de uma solução de Continuidade de Negócio e Recuperação de Desastres e que são fundamentais para um correcto desempenho da solução, nomeadamente:

- Comprometimento dos responsáveis da Gestão de Topo, directores e administradores
- Sem um comprometimento e "sponsorização" da direcção/administração da organização na necessidade de continuidade de negócio e de recuperação de desastres não se conseguirá a implementação de uma solução adequada e que vá ao encontro das necessidades efectivas da organização.
- Elaboração de uma Análise de Impacto de Negócio (BIA)
  Esta análise, além de identificar o que deve ser recuperado e respectivas prioridades, permite ainda estabelecer a relação entre o impacto e o risco para o negócio e a infraestrutura tecnológica. Deve também produzir os tempos máximos admissíveis de paragem para as funções de negócio a partir do qual os prejuízos poderão ser muito significativos ou insustentáveis para a organização.
- Orçamentação
  Muito importante no processo de implementação, define as balizas orçamentais para a solução a implementar; todos os custos devem ser categorizados e claramente descritos.

- Definição de arquitecturas viáveis de DR, enquadradas com o BIA
  A definição das arquitecturas viáveis de Recuperação de Desastres ainda a alto nível, para fazer face aos requisitos de recuperação indicados pelo BIA. É este conjunto de cenários possíveis, e devidamente enquadrados nos objectivos de recuperação, que possibilita o início do processo da formulação do desenho da solução.

Por vezes, com o objectivo de reduzir custos, as organizações tendem a confundir os conceitos pelo facto das semelhanças dos efeitos de ambos (BC e RD).

A Recuperação após um desastre (*Disaster Recovery*) é um conceito associado à recuperação da Infraestrutura de TI, mas Continuidade de Negócio (*Business Continuity*) é referente a um nível dos processos de negócio da organização (Jackson,2007).

A Recuperação de dados (*Disaster Recovery*) e Continuidade de Negócio (*Business Continuity*) determinam a forma como a organização continuará com a sua disponibilidade e operacionalidade após ser afectada por um acontecimento de ruptura e até que as suas capacidades normais sejam recuperadas.

A junção de ambos possibilita que as organizações possam preparar-se para eventuais situações de ruptura, provocados por desastres naturais, tecnológicos ou originados por incorrecta ou intencional intervenção humana.

No contexto, Desastres não se resumem apenas a situações que ocorrem no dia-a-dia (inundações, fogo, sismos ….), mas também a situações de avarias ou deficiente operacionalidade e disponibilidade de hardware e software.

A recuperação de Desastres (*Disaster Recovery*) ocorre devido a uma disfunção anormal no funcionamento da empresa com efeitos de paralisação parcial ou total, provocando desde logo perdas directas ou indirectas à organização.

A recuperação de dados, de operações, de sistemas, de instalações, ou de recursos técnicos, resultam nos processos/procedimentos a executar e determinantes para a operacionalidade da organização. Quando existe esta necessidade, associada a uma análise de situação por grupos de trabalho específicos para o efeito, é anunciada a Declaração de Desastre, iniciando-se desde logo o plano de recuperação definido e testado previamente.

Qualquer organização está exposta a ameaças e a desastres que podem provocar a falha ou mesmo a interrupção das suas operações. A existência de um Plano de Recuperação (DRP) devidamente actualizado tem por objectivo mitigar estes efeitos negativos.

A Continuidade de Negócio (*Business Continuity*) consiste no conjunto de diferentes processos de recuperação e gestão dos ambientes de IT e de negócio, dentro da organização, independentemente do(s) factor(es) motivadores, em que a organização poderá continuar a operar mesmo em situações de emergência para garantir a relação existente com clientes e fornecedores e de continuar a facturar.

Para o sucesso de implementação de um ambiente de CN/RD, as organizações devem desde logo planear as acções a executar perante situações de desastre ou de continuidade do seu negócio, a acção inicial será uma Analise de Impacto de Negócio (BIA)

1.3.1. Análise de Impacto de Negócio (BIA)

A forma mais correcta de iniciar a Analise de Impacto de Negócio (BIA) é proceder à identificação de todos sistemas e processos cruciais e transversais ao negócio das organizações e saber através de uma BIA qual o efeito que a interrupção/ruptura terá na organização, através de uma Análise de Impacto de Negócio.

A BIA contribui assim para estabelecer os procedimentos e respectivas sequências de recuperação de forma a determinar as prioridades de reposição das áreas de negócio. A este primeiro passo associam-se outros factores e desafios determinantes para assegurar o sucesso da recuperação, dos quais destaca-se:

- Determinar a perda admissível de Informação numa organização (RPO)
- Determinar o tempo de indisponibilidade admissível ao nível de informação numa organização (RTO)
- Quem pode decidir em situações de desastres
- Classificar os Sistemas de Informação mediante a criticidade e relevância para o Negócio
- Escolher a melhor solução de Recuperação de Desastres/ Continuidade de Negócio (PRD/CN), e decidir sobre a necessidade de recorrer ou não a um Centro de Processamento de Dados secundário descentralizado

Surge então a necessidade de criar documentação que será a fonte de conhecimento para uma actuação de sucesso perante os cenários de interrupção de negócio, Recuperação de Desastres/ Continuidade de Negócio, em que a organizações pode:

- Implementar o Plano Recuperação de Desastres/ Continuidade de Negócio (PRD/CN)
- Validar e testar com curtos intervalos temporais (a determinar) o PRD/CN
- Dar continuidade ao Plano mediante actualizações e testes tantos quanto necessários

O PRD/CN implica a necessidade de gestão de recursos humanos e processos adequados para a recuperação do Hardware e Sistemas de Informação críticos quando ocorre um evento de ruptura/interrupção (desastre).

A saúde das organizações implica que a Recuperação de Desastres/Continuidade de Negócio sejam uma preocupação fundamental para conseguir uma capacidade suficiente para responder de forma eficaz e eficiente a eventuais falhas críticas provocadas por eventos de ruptura, dando continuidade à disponibilidade do seu negócio mesmo que não em condições

óptimas. Esta capacidade deve ser claramente vinculada na BIA através da sua aprovação formal pela Gestão de Topo da Organização.

O plano de CN/RD é sem dúvida um pilar da sobrevivência de qualquer organização no qual mantem uma relação estreita e importante.

1.3.2. Como se relacionam

Apesar de ambos serem estratégias e não tecnologias, a relação entre BC e DR é uma relação muito estreita, quase todos os processos de negócio das organizações modernas tem fortes dependências dos sistemas de informação e uma maior componente tecnológica.

Figura 1.1 - Relação entre RD e CN

É possível definir processos de negócio, que poderão estar assentes em processos manuais, o facto é que estes se tornam insustentáveis se prolongarem por tempo excessivo.

Assim, um processo de continuidade de negócio depende em larga escala do processo de recuperação de desastre para garantir a operacionalidade da empresa.

1.3.3. Plano de Recuperação de Desastre (PRD) versus Plano de Continuidade de Negócio (PCN)

Sendo o PRD um processo técnico, especifica o conjunto de equipamentos, localizações, procedimentos e recursos humanos envolvidos, enquanto o PCN é um processo de gestão que visa manter/repor os processos de negócio após eventual falha, também este inclui toda a gestão de recursos humanos e bens.

1.3.4. Fulcralidade da protecção dos dados

Apesar do paradigma da salvaguarda e segurança de dados, a protecção dos dados contínua essencial, uma vez que é o bem mais valioso para qualquer organização, independentemente da sua dimensão. A perda irremediável de dados acarreta normalmente custos muito significativos para as organizações ou mesmo a sua retirada do mercado de negócio.

De uma forma sumária, faz-se a divisão em duas áreas:

- Gestão de acessos
- Protecção de dados

A gestão de acesso, e a protecção de dados, englobam componentes tecnologia de *hardware* e *software*, no entanto em áreas diferentes.

A figura seguinte ilustra algumas das boas práticas no que concerne à protecção de dados:

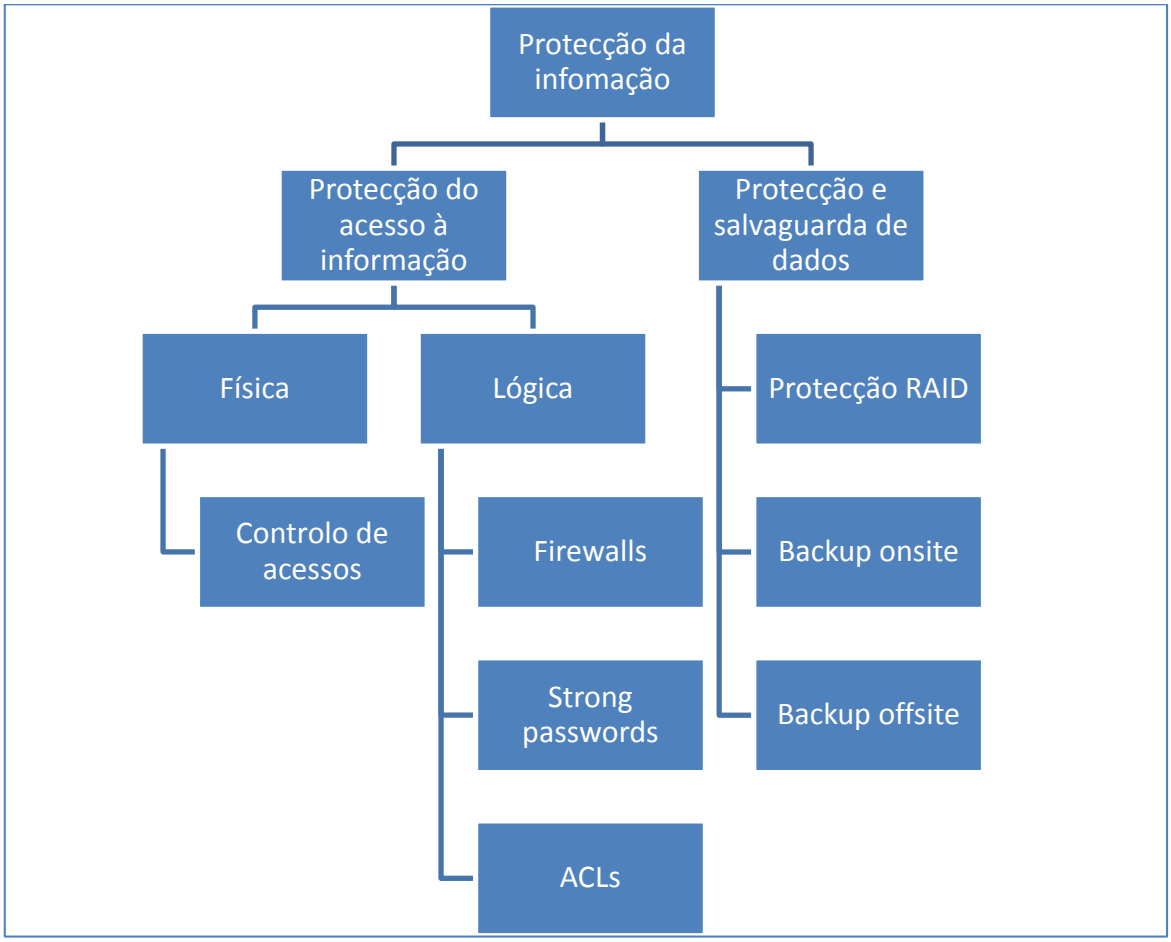

Figura 1.2 - Protecção da Informação

1.4. Objecto de estudo

Um sistema de informação é composto por vários e diferentes elementos, que da sua interacção produzem resultados, no entanto susceptíveis a falhas dos mais diversos tipos. Tendo em conta as suas características e relação com a Continuidade de Negócio da organização, pelo que em caso de necessidade de reposição destes sistemas o impacto negativo no negócio deverá ser o mínimo, desde a melhor sequencia até à maior rapidez da disponibilidade, operacionalidade e validação, pois a continuidade da facturação da organização está em causa, será um excelente contributo ter disponível forma de optimização de execução de PRD/CN.

Uma possível forma será a existência de um sistema que permita definir recursos e propriedades, operações, tarefas, bem como o tempo admissível de paragem (RTO) e perdas admissíveis (RPO), e como resultado, um plano para uma execução optimizada em janelas temporais disponíveis e respectivas sequências.

Para o este objecto de estudo define-se 2 tipos de objectivos, específicos de conteúdo ao nível da implementação e das mais-valias e os objectivos gerais com principal incidência para os resultados finais.

1.4.1. Objectivos específicos

Este documento contém uma proposta para implementação de uma *framework* que permita invocar um conjunto de algoritmos de optimização de requisitos Multidimensionais num modelo *Job-Shop*, registando as suas variáveis de entrada, tempo de execução e resultados. Com este registo de execução dos algoritmos, será possível por análise de dados, verificar qual o algoritmo mais vantajoso na obtenção da solução mais optimizada para um universo de cenários.

Verificar-se-á, também, que quanto maior for a base de conhecimento, mais assertivos serão os resultados e consequentemente, mais fácil será a escolha do algoritmo adequado, bem como a celeridade com que todo o processo será executado e apresentado.

Paralelamente existem outros tipos de objectivos derivados dos supra referidos, que acrescem um conjunto de mais-valias, das quais destacam-se na figura seguinte as mais determinantes:

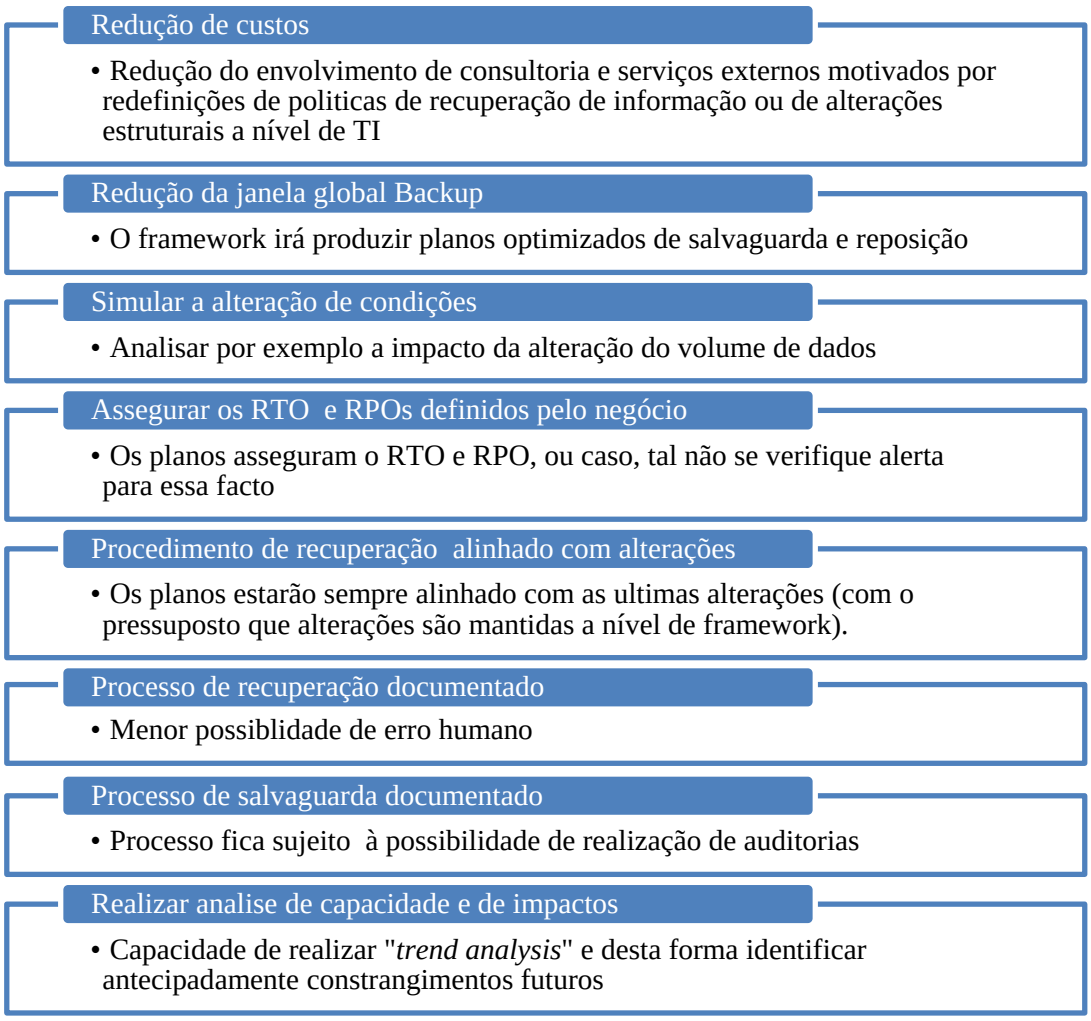

Figura 1.3 - Sumário dos objectivos específicos

1.4.2. Objectivos Gerais

Perante a insistência em muitas organizações de soluções que permitam a elaboração de Planos de Salvaguarda e Recuperação de simples utilização, o principal objectivo é a utilização de software de código aberto (OSS), mediante um modelo Job-Shop e através de uma *framework* baseada em algoritmos de optimização de requisitos Multidimensionais, ter como resultado a solução mais optimizada que servirá para a elaboração de um Plano de Salvaguarda e Reposição representado num diagrama de Gantt.

1.4.3. O conceito RTO, RPO e respectivas relevâncias

Num Plano de Recuperação, deverá ser decidir o ponto de recuperação – RPO - (Recovery Point Objective) e o tempo de recuperação – RTO - (Recovery Time Objective). O RPO dita a perda de dados admissível, enquanto o RTO o tempo em que os sistemas podem estar indisponíveis, ou seja a interrupção máxima tolerada, informação constante no BIA.

A hipótese básica por detrás de uma BIA é a de que cada elemento da organização está relacionado com a operação de outro elemento, mas alguns dos elementos são mais cruciais do que outros, considerando que para tal já é conhecido o BIA.

Se um desastre acontece, torna-se pertinente a resposta a algumas questões, nomeadamente, quanto tempo uma organização pode esperar, ou seja, quanto tempo poderá viver sem sistemas de informação (horas, dias, semanas). Uma organização que requeira recuperação imediata necessitará de maior investimento em soluções de recuperação do que uma organização que pode estar inactiva alguns dias. Do mesmo modo, quanto menor um RPO mais dispendioso será, pelo que as organizações devem comparar as despesas preventivas com o custo da perda de dados para que o POR a definir seja o mais optimizado mediante os custos possíveis de suportar na implementação da solução versus as perdas com a paragem.

Identificar o RPO e o RTO irá auxiliar a determinar os recursos apropriados e para formular o orçamento financeiro para a implantação da solução a seleccionar.
Estas variáveis são determinantes para a elaboração de estratégias de recuperação, e sendo que de uma forma mais detalhada:

- **RPO** - representa o ponto no tempo em que será usado para recuperação dos dados
  - Normalmente a altura em que foi realizado a última salvaguarda de dados
  - Quanto menor o RPO, menor será a perda de dados em caso de desastre.

- **RTO** - Ponto no tempo em que as operações são recuperadas, após a ocorrência de um desastre.
  - Quanto menor o RTO, menor será o tempo que os sistemas estão indisponíveis, logo menor a perda associada à indisponibilidade dos mesmos.

Numa estratégia de recuperação após um desastre, além de existirem todas as condições acima referidas, é determinante e essencial que exista uma salvaguarda de dados mediante as boas práticas para o efeito.

1.4.4. Salvaguarda de dados

A caracterização do tempo estimado de recuperação é de crucial importância para definir os procedimentos de salvaguarda e reposição e para determinar se estes vão de encontro aos requisitos definidos pelo negócio em termos de RTO e RPO.

O esquema que segue define de uma forma simplificada as variáveis, ao dados e interdependência que irão permitir definir o plano de salvaguarda.

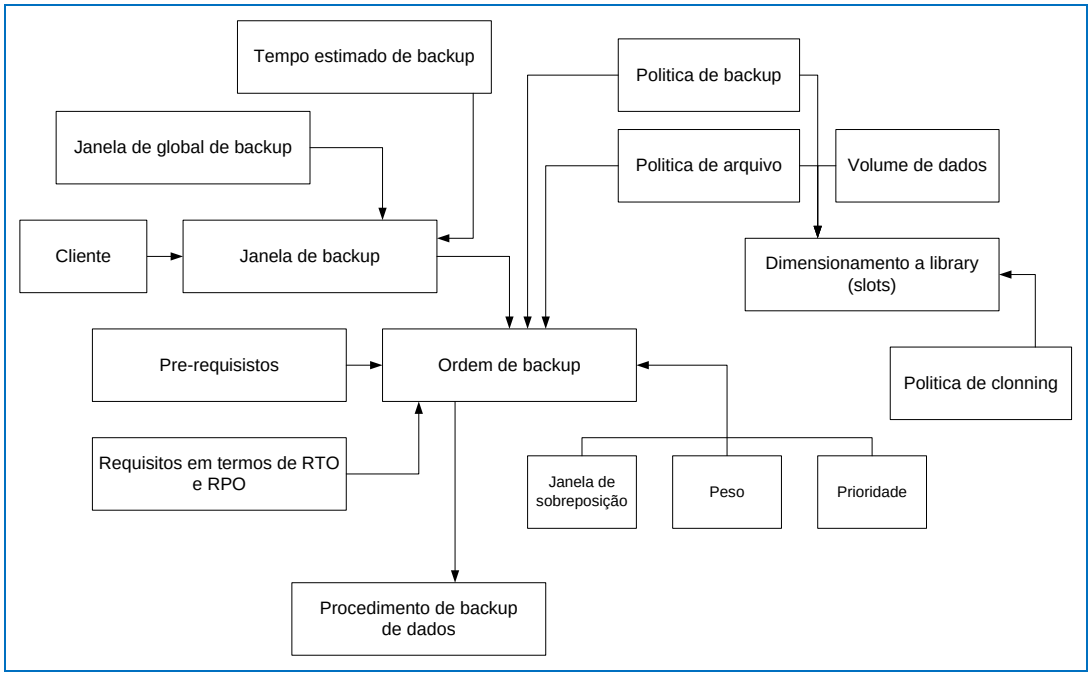

Figura 1.4 - Variáveis e interdependências que definem o Plano de Salvaguarda

1.5. O "problema"

A ainda ausência do âmbito deste trabalho nas organizações é o grande motivo da realização do mesmo, pois apesar da evolução tecnológica ser uma constante na inovação, e da presença cada vez mais forte da gestão de conhecimento organizacional, é notório que algumas organizações baixam a prioridade à execução de planos de recuperação de desastre ou testes dos mesmos perante eventuais estimativas de custos elevados. Ainda paira o mito que este tipo de precaução apenas pode ser feito mediante sistemas tecnológicos complexos e de elevados custos, e possuir nas suas equipas pessoal altamente especializado, e experiente para execução dos planos, acresce que os custos destes dois factores, só algumas organizações poderiam.

Para desmistificar este paradigma, surge a proposta deste trabalho no qual será demonstrado que sem ser necessário recorrer a elevados investimentos de origem financeira e humana, é

possível ter disponível uma ferramenta de software de trabalho simples, eficaz e eficiente, gerida pelos recursos humanos da organização, bastando dar-lhes alguma formação na nova ferramenta. A ferramenta contribua ainda fortemente para aferir os actuais procedimentos, aumentando seguramente a probabilidade das organizações sentirem a CN após recuperação.

1.6. Âmbito do trabalho

A disponibilização desta ferramenta, que privilegia o *Open Source*, e que utiliza algoritmos simples e clássicos, mas de possível evolução perante o ambiente e cenários onde irá servir.

Não é intenção de rever, analisar, ou classificar, detalhada de forma exaustiva os métodos de solução referidos pela metodologia clássica para cada tipo de problema de *Scheduling* na utilização dos vários tipos de algoritmos

Acresce o facto de não se pretender a utilização de algoritmos evoluídos e recentes, certamente com melhor *performance* mas também de mais difícil utilização por parte de programadores menos experientes, mas sim permitir que a possibilidade de evolução seja viável a partir de algoritmos clássicos, actualmente mais utilizados e documentados, sendo a evolução mais específica e complexa mediante os ambientes e cenários, e assim permitir que numa base simples a evolução e adaptação ás várias realidades seja simples. O actual conhecimento permite antever que quanto maior o universo/população mais evoluídos e complexos serão os algoritmos, o que constitui um motivo adicional de confirmar de que mediante o ambiente e cenário, o melhor resultado será fornecido pela utilização de diferentes algoritmos.

Pelo referido deverá ser sempre encarado num âmbito dinâmico e nunca como final, terminado e incapaz de evoluir.

Através de algoritmos de requisitos a combinações de Tempo e Recurso e de Tarefas e Operações, para a optimização de requisitos Multidimensional seleccionado de forma automática e entre os mais comuns, pretende-se obter uma *Framework* com o objectivo de simplificar a gestão dos processos de recuperação e de salvaguarda, garantindo o alinhamento com os requisitos impostos pelo negócio e pelas boas práticas, sem a necessidade de elevados custos financeiros, sem complexidade, personalizável aos requisitos de cada organização, com gestão e operacionalidade por parte das equipas locais, sem ter de recorrer a especialistas desta temática.

Na verdade, a existência do Plano de Salvaguarda e Plano de Recuperação, com a *framewok* poderá existir um único plano, dado que as tarefas de eventuais reposições serão escalonadas

em simultâneo com as tarefas de salvaguarda, sendo a sua execução dependente do resultado obtido pela *framework* e de acordo com as regras estabelecidas.

1.6.1. Plano de salvaguarda

A figura 1.4 ilustra uma possível abordagem para criar um Plano de Salvaguarda

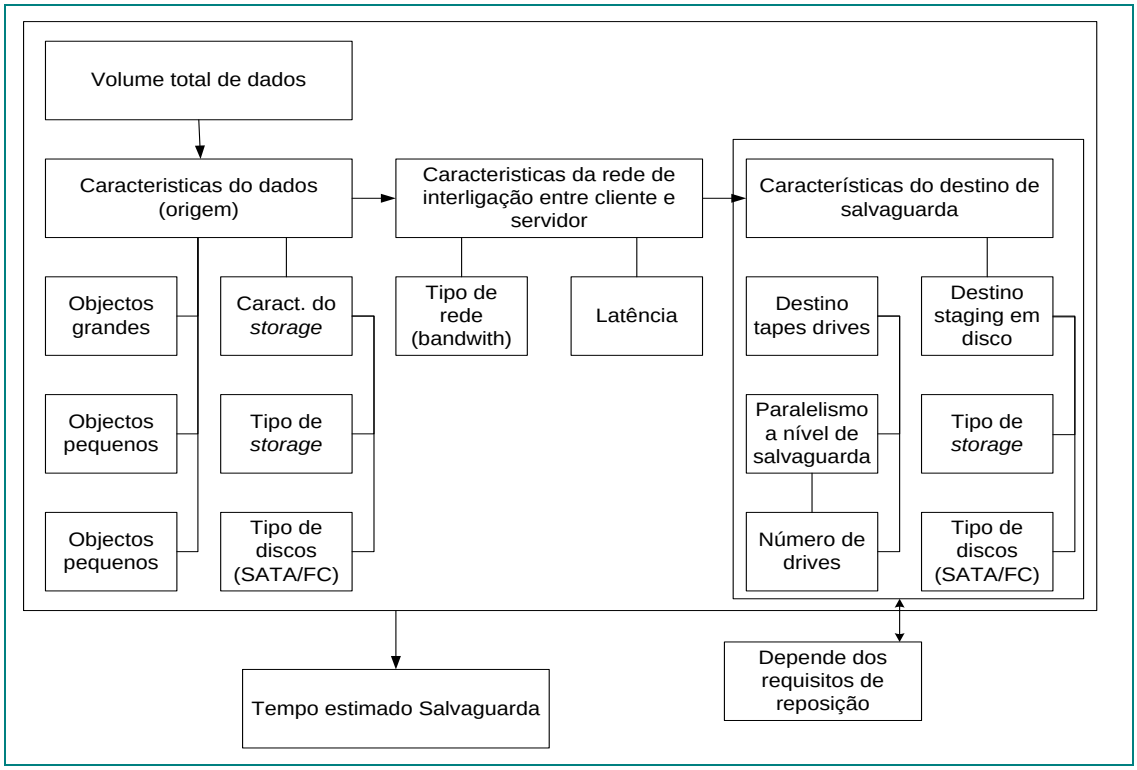

Figura 1.5 - Plano de Salvaguarda

Plano de Reposição

Existe um conjunto de características a ter em conta para que o levantamento de requisitos tenha a qualidade suficiente para o Plano de Reposição.

A figura seguinte mostra alguns dos requisitos a ter em conta para um plano de reposição.

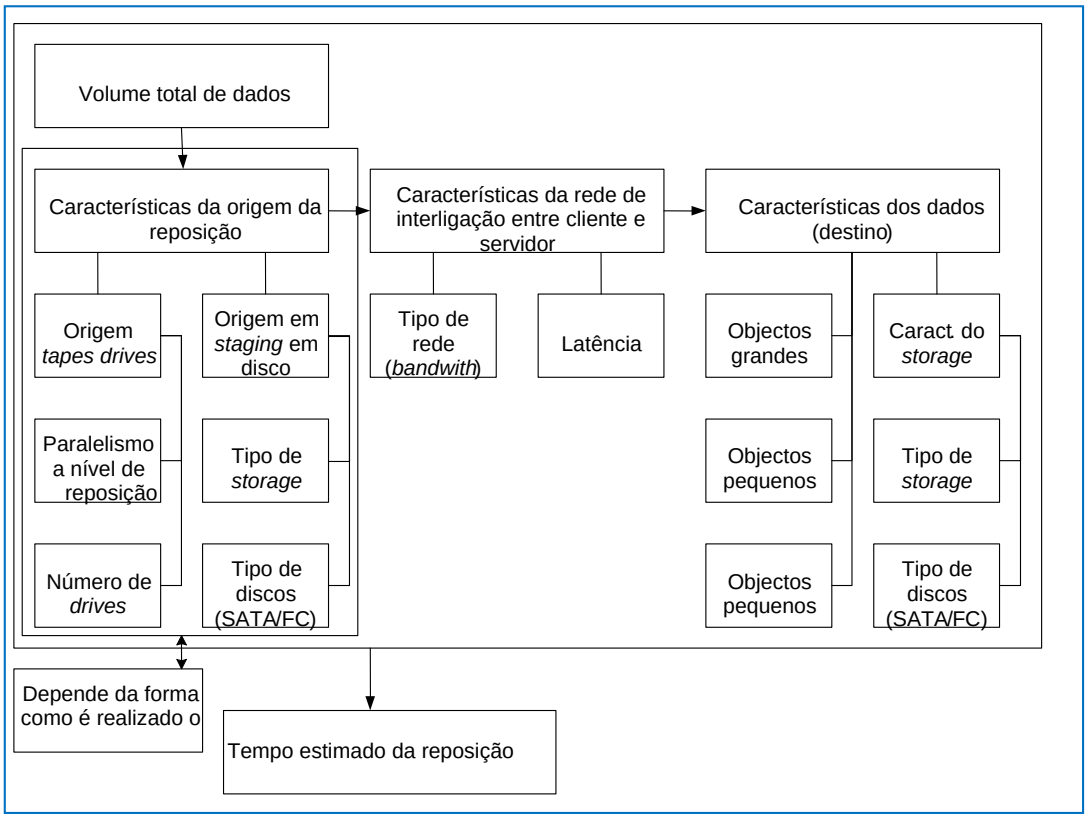

Figura 1.6 - Plano de Reposição

Tendo em conta o tipo de situação e os requisitos respectivos, é possível elaborar um plano de reposição de cada sistema, sendo este mais ou menos optimizado para a realidade da organização em causa, sendo assim necessário uma gestão eficiente e eficaz para aproveitar o máximo das janelas temporais na reposição através essencialmente da optimização dos tempos necessários e as sequencias mais correctas.

1.6.2. Gestão do plano de Reposição

O esquema seguinte define de uma forma simplificada de variáveis, dados e interdependências que irão permitir definir o procedimento de recuperação.

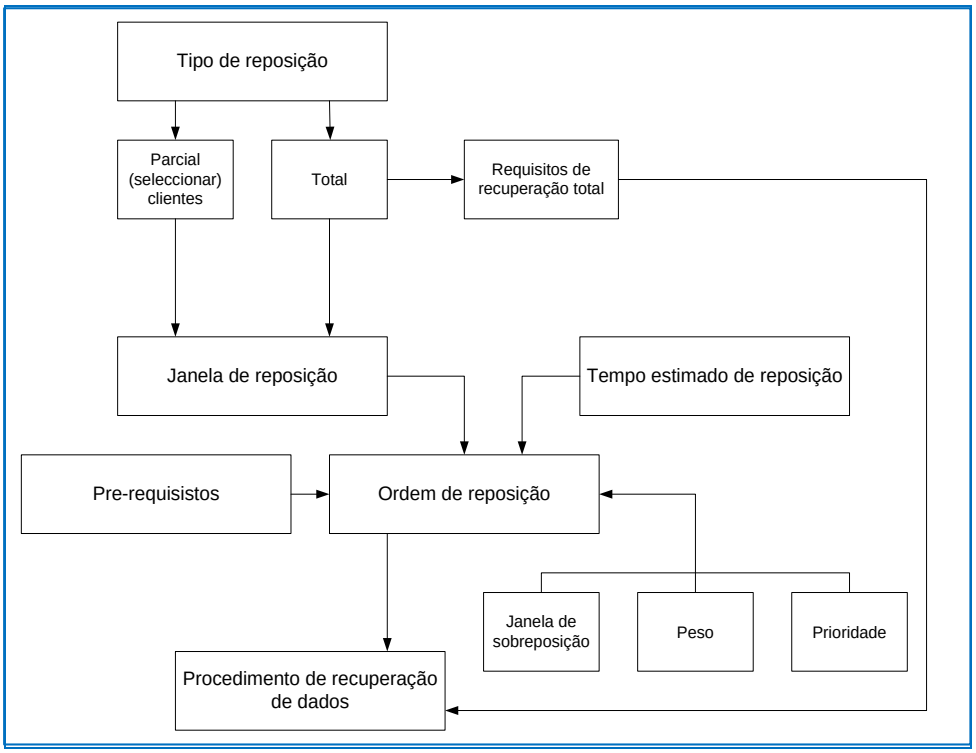

Figura 1.7 - Gestão do plano de Reposição

Capitulo 2. Estado da Arte

2.1. Análise de Mercado

Em Janeiro de 2010, a Symantec[1] publica o estudo "SYMANTEC STATE OF THE DATA CENTER REPORT 2010", com uma metodologia de pesquisa aplicada, que utilizou em Novembro de 2009 para realizar via telefone contacto com a sede de organizações de 26 países, tendo os entrevistados sido identificados através de 3 grupos pertencentes a empresas das mais variadas áreas de negócio, e de diversas responsabilidades profissionais (profissionais de IT, Administradores, colaborantes …):

- Empresas consideradas pequenas (1.000 – 1.999 funcionários)
- Empresas consideradas médias (2.000 – 9.000 funcionários)
- Empresas consideradas Grandes (>= 10.000 funcionários)

Na matéria de salvaguarda/reposição de dados, foram questionados os responsáveis destas áreas sobre vários temas, dos quais destacam-se:

a. Quantificação do número de funcionários e agregação por 3 grandes grupos
b. Distribuição dos entrevistados por 2 áreas chave (Dep. IT e Dep. colaborantes)
c. O Estado do Plano de Recuperação de Dados
d. As principais causas de *Downtime* na Organização
e. Taxa de iniciativa para 2010
f. Nível de Confiança no actual Plano de Recuperação de Dados
g. Validação / Testes do Plano de Recuperação de Dados nos últimos 12 meses
h. Motivos de Reavaliação do Plano de Recuperação de Dados
i. Inclusão de Locais Remotos no Plano de Recuperação de Dados Central
j. Tácticas e Métodos utilizados na Estratégia da Recuperação de Dados
k. A frequência de teste do Plano de Recuperação de Dados
l. O Resultado do último teste do Plano de Recuperação de Dados
m. Incidentes registados responsáveis por *downtime*
n. Na situação de Reposição, qual a classificação da mesma
o. Recursos de IT necessários e custo dos Incidentes (média)

a. Quantificação do número de funcionários e agregação por 3 grandes grupos

[1] Entidade que divulgou o estudo

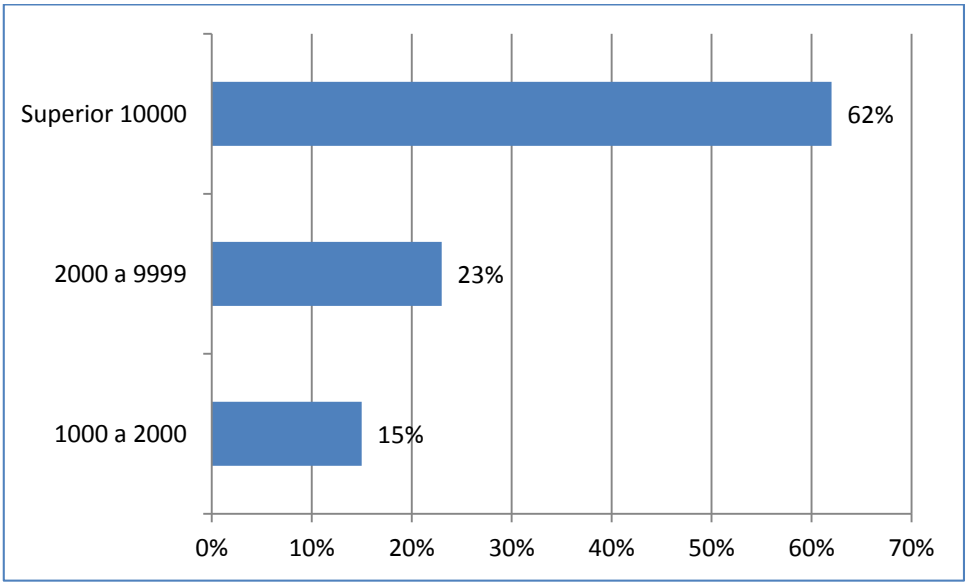

Figura 2.1 - Quantificação do número de funcionários e agregação por grupos

b. Distribuição dos entrevistados por 2 áreas chave (Dep. IT e Dep. colaborantes)

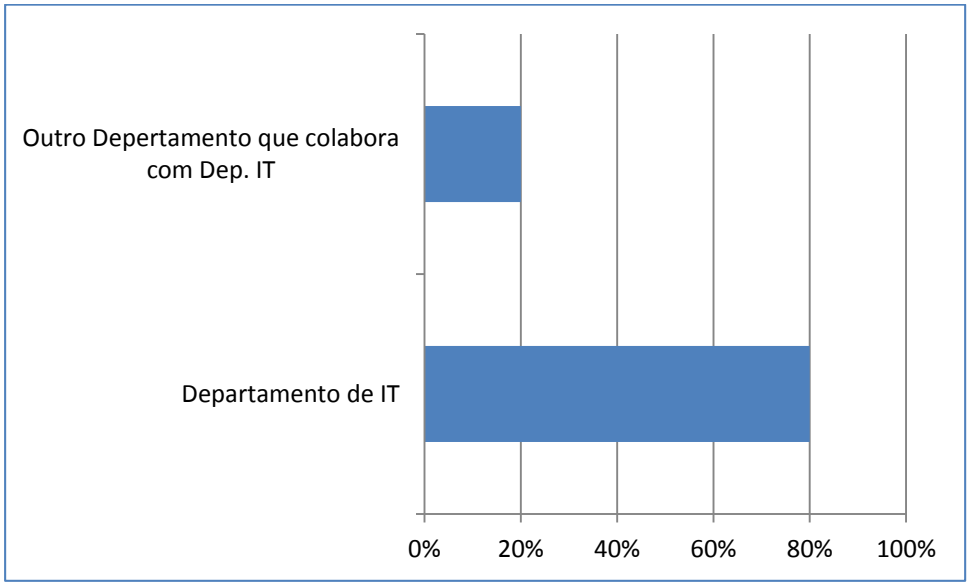

Figura 2.2 - Distribuição dos entrevistados

c. O Estado do Plano de Recuperação de Dados

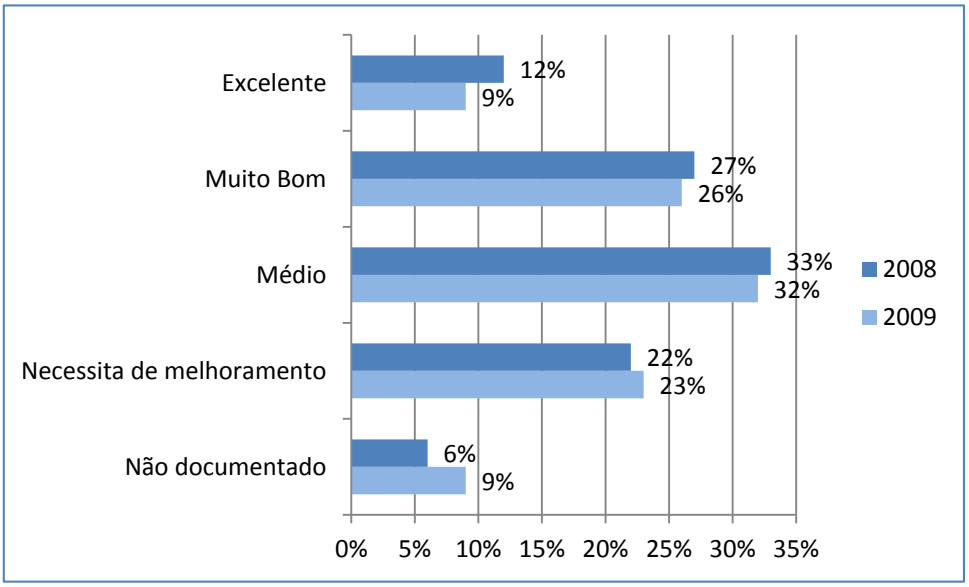

Figura 2.3 - O Plano de Recuperação

d. As principais causas de *Downtime* na Organização

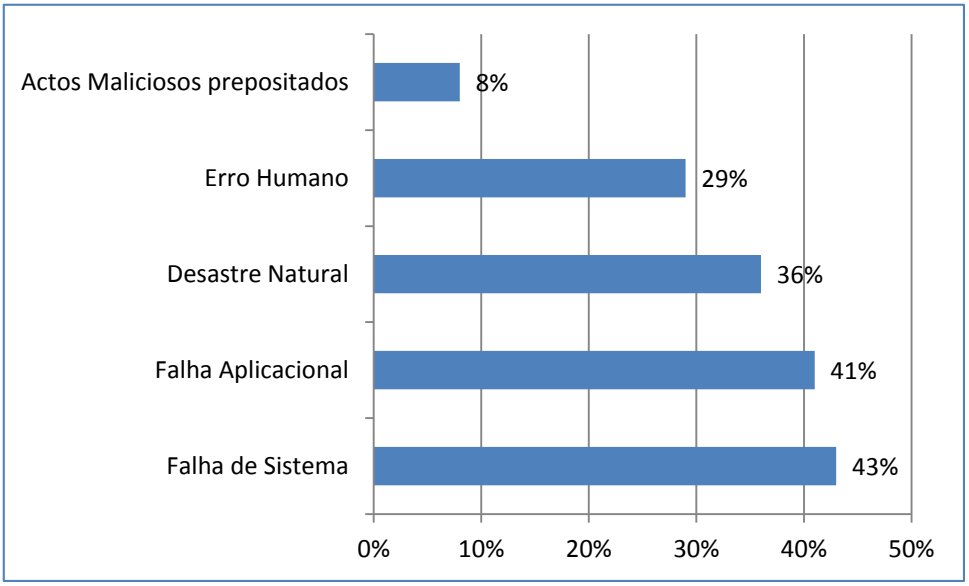

Figura 2.4 - Causas de *Downtime*

e. Taxa de iniciativa para 2010

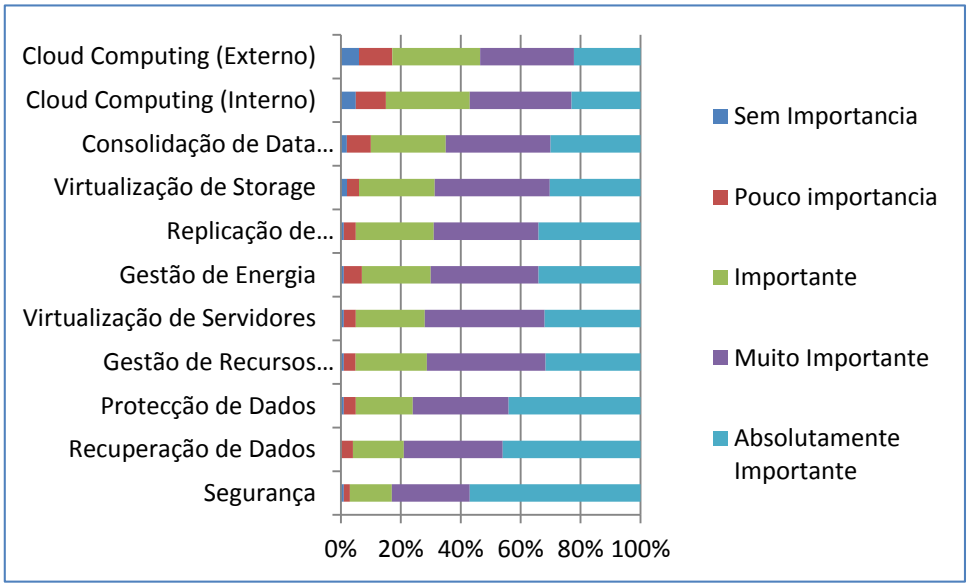

Figura 2.5 - Taxa de iniciativa

f. O Nível de Confiança no actual Plano de Recuperação de Dados

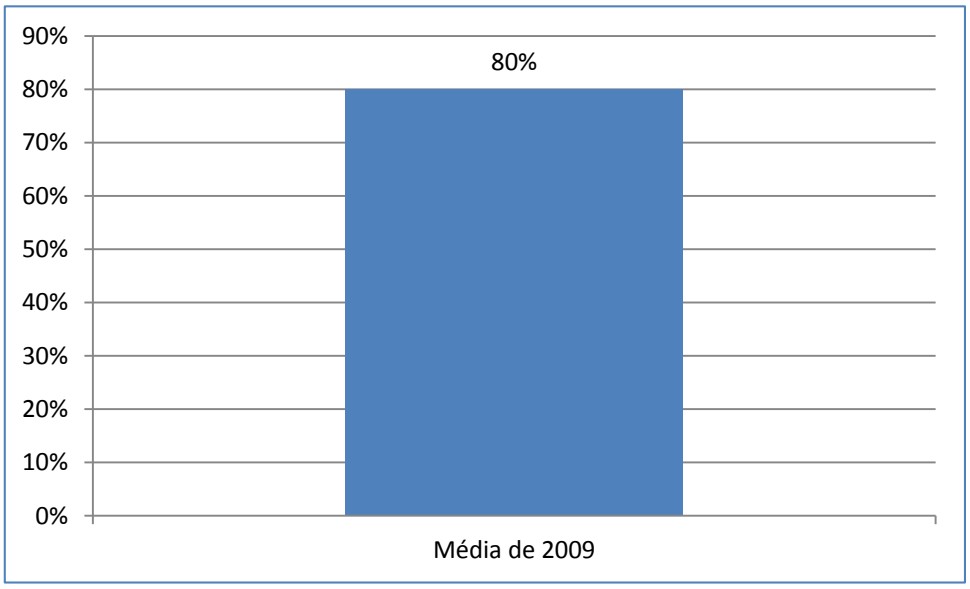

Figura 2.6 - Nível de Confiança

g. Validação / Testes do Plano de Recuperação de Dados nos últimos 12 meses

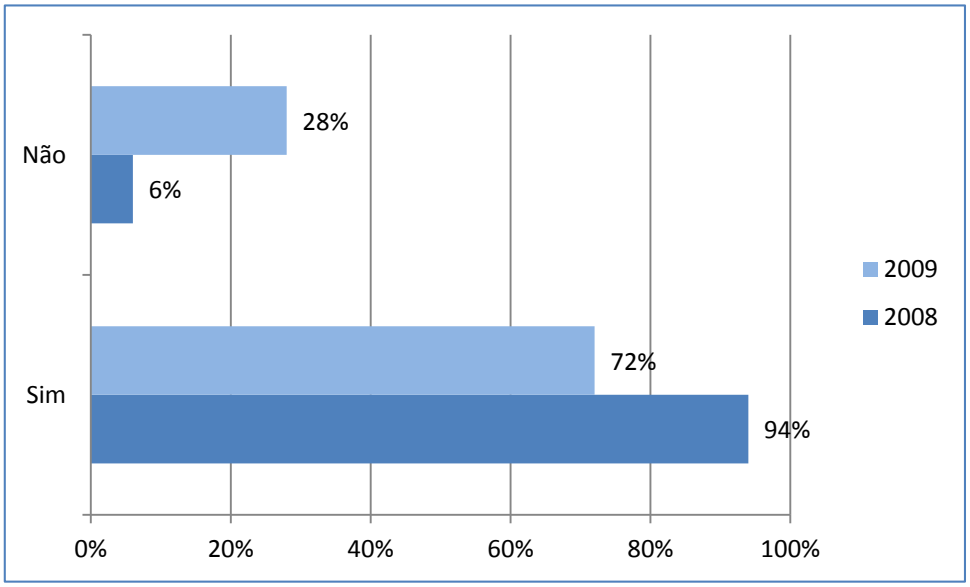

Figura 2.7 - Validação / Testes do Plano de Recuperação

h. Motivos de Reavaliação do Plano de Recuperação de Dados

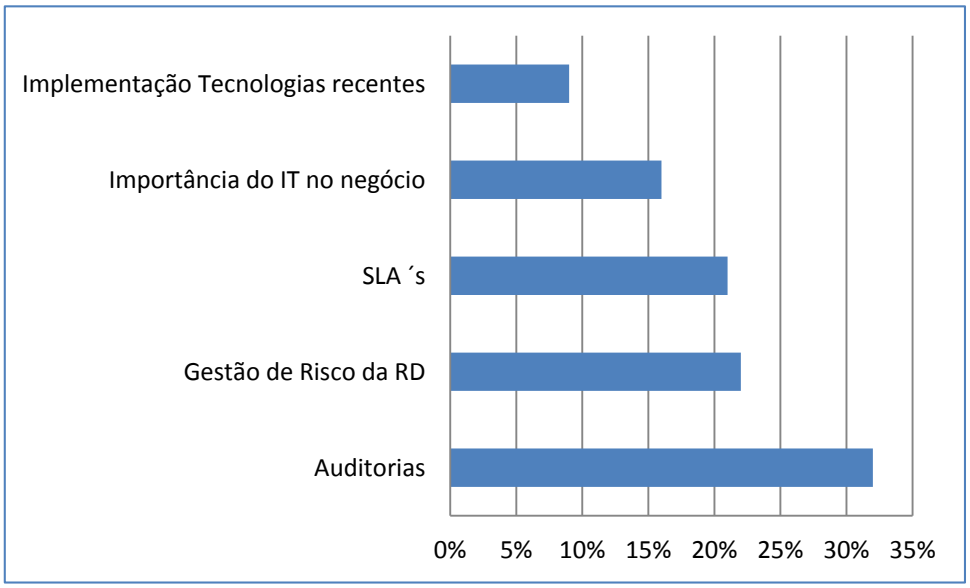

Figura 2.8 - Motivos de Reavaliação do Plano de Recuperação de Dados

i. Inclusão de Locais Remotos no Plano de Recuperação de Dados Central

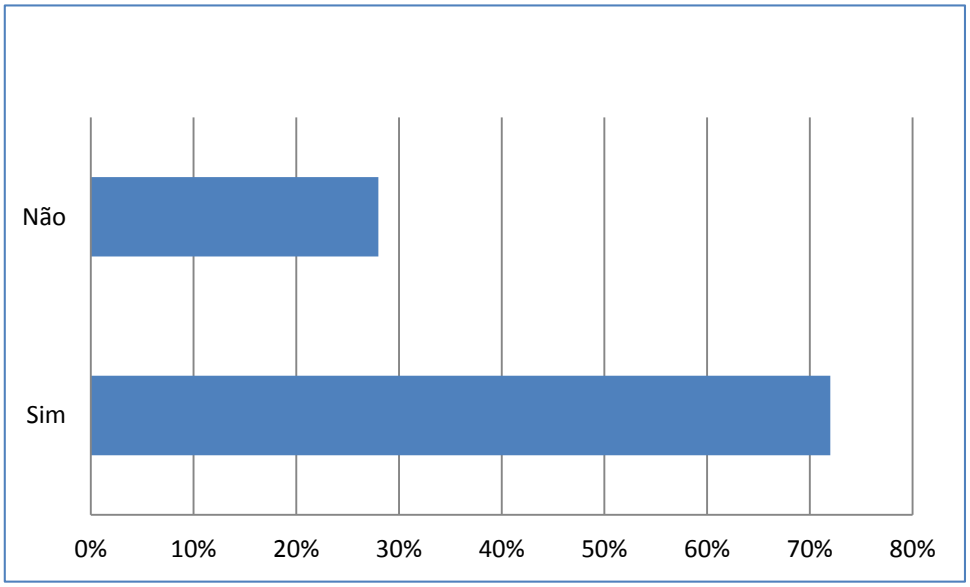

Figura 2.9 - Locais Remotos no Plano de Recuperação

j. Tácticas e Métodos utilizados na Estratégia da Recuperação de Dados

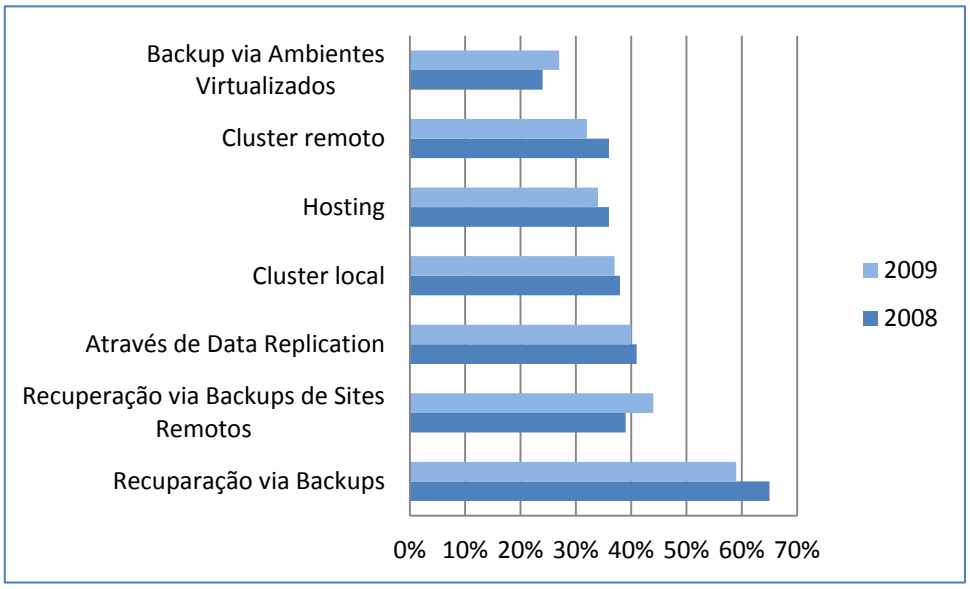

Figura 2.10 - Tácticas e Métodos na Estratégia da Recuperação de Dados

k. A frequência de teste do Plano de Recuperação de Dados

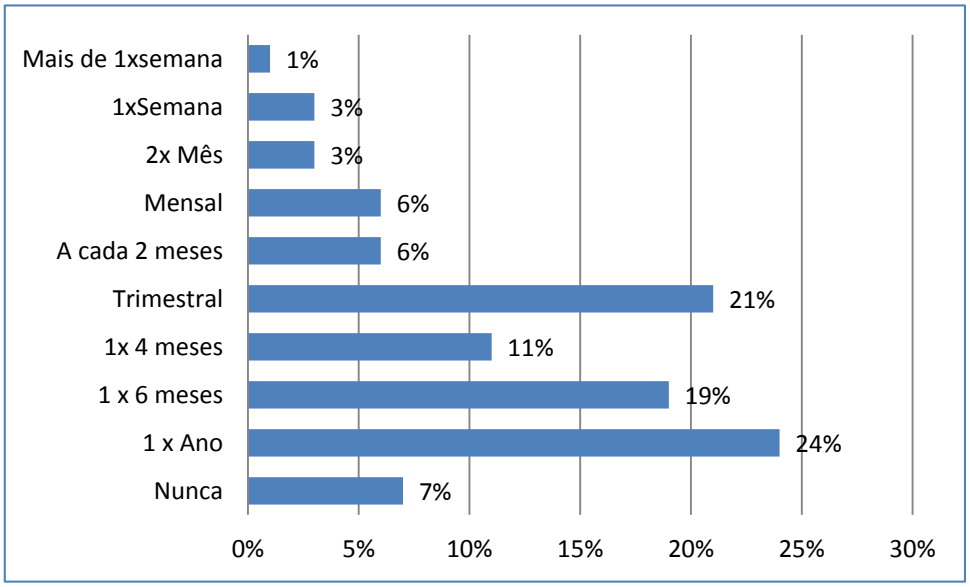

Figura 2.11 - A frequência de testes do Plano de Recuperação de Dados

l. O Resultado do último teste do Plano de Recuperação de Dados

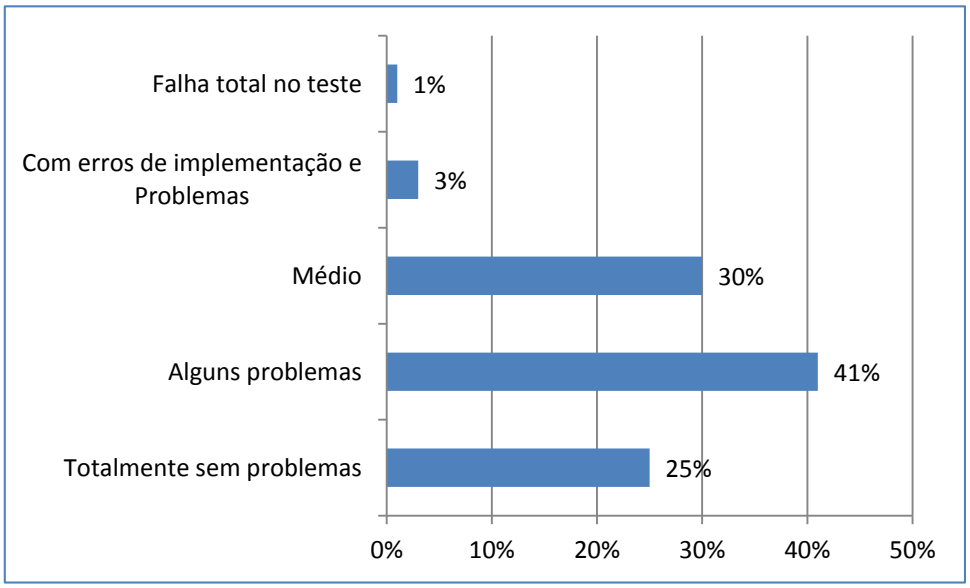

Figura 2.12 - O Resultado do Plano de Recuperação de Dados

m. Incidentes registados responsáveis por *downtime*

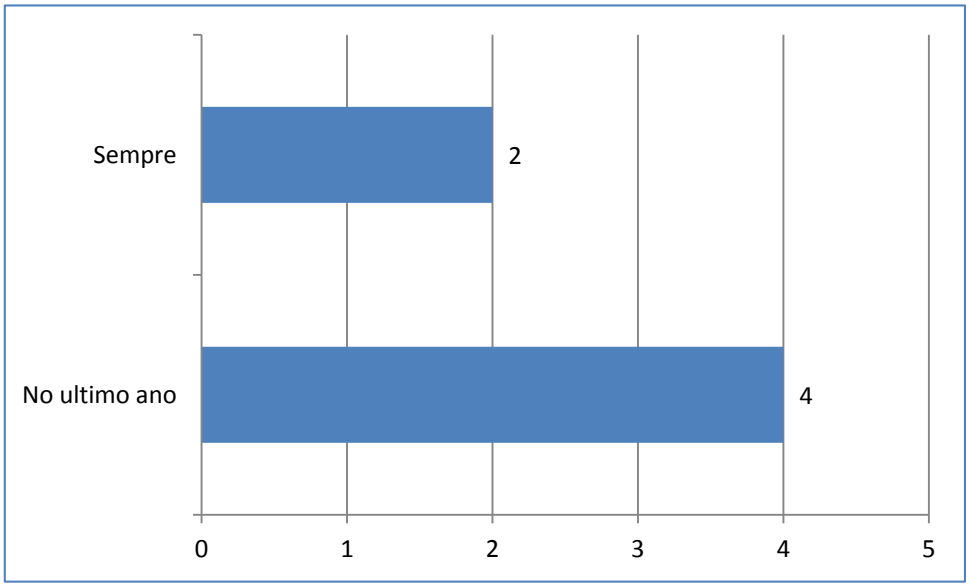

Figura 2.13 - Incidentes por *downtime*

n. Na situação de Reposição, a classificação da mesma

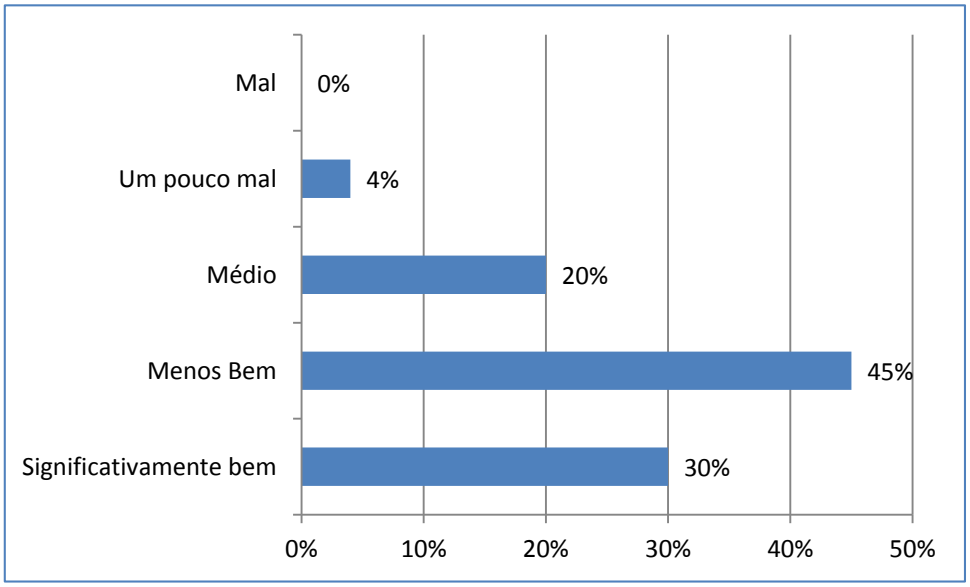

Figura 2.14 – Classificação da Reposição

o. Recursos de IT necessários e custo dos Incidentes (média)

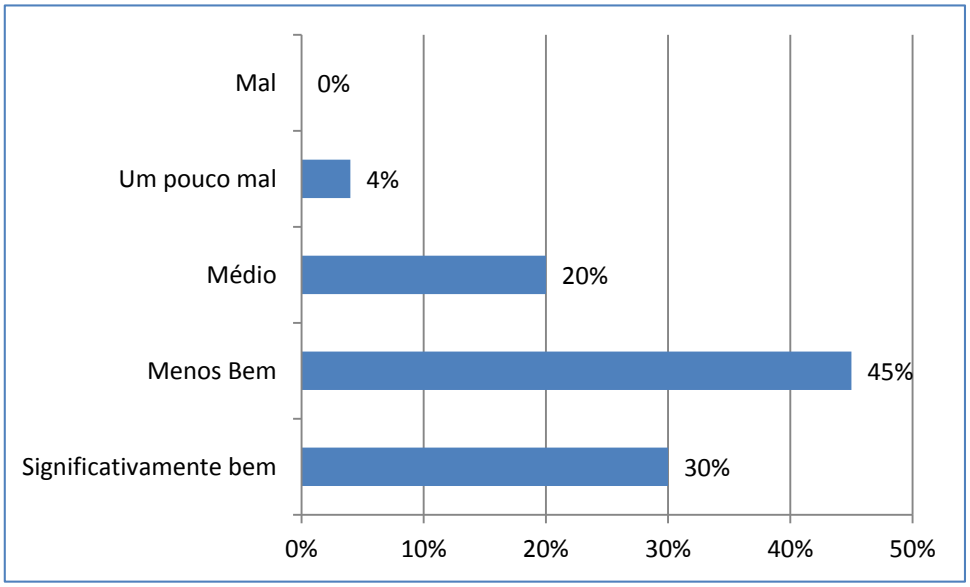

Figura 2.15 - Recursos de IT necessários e custo dos Incidentes

Em conclusão:

- O nível de confiança nos seus planos de Salvaguarda/Reposição de Dados, é a média das respostas foi de 80%
- Ao nível de testes/validação apenas ocorrem 2 nos últimos 12 meses, em que as principais causas de *downtime* foram de falha de sistemas, aplicacionais e de desastres naturais.
- Teste de recuperação de desastres pode ter um impacto significativo negócio. As empresas devem procurar melhorar o sucesso dos testes através da avaliação e implementação de métodos de ensaio.
- Nas causas do *downtime*, apesar de os ataques exteriores serem algo que deve ser tido em conta pelas organizações, apenas 8% são actos maliciosos propositados, enquanto que as falhas em sistemas e aplicações são o foco dos problemas de *downtime*.
- Combinando a questão das causas do *downtime* com a taxa da iniciativa para 2010 vemos que a segurança é aquela que é considerada "Absolutamente Importante". Concluindo-se assim que os responsáveis andam preocupados com os temas "errados".

- No actual mundo em que a recuperação de dados é fundamental, apenas 25% dos testes de recuperação de dados sejam "Totalmente sem problemas", quando mais de 75% destes testes apenas são feitos com mais 3 meses de distância, nomeadamente:

trimestralmente (21%), quadrimestralmente (11%), semestralmente (19%) ou anualmente (24%).

- As PME´s assistem a todos estes cenários com um olhar diferente, pois vivem outra realidade ao nível de dimensão e financeiro, no entanto e apesar da diferença de dimensão estão também sujeitas aos mesmos fenómenos e evidentemente aos mesmos efeitos e com capacidades financeiras insuficientes para uma resposta no mínimo suficiente.

A mesma fonte (Symantec) auscultou o mercado ao nível das Pequenas e Médias Empresas (PME), e conclui que estas organizações entendem que a prevenção de desastres não é uma prioridade

Segundo o estudo realizado entre Outubro e Novembro de 2010, junto de profissionais de IT responsáveis por Redes Informáticas e Recursos Tecnológicos de Pequenas e Médias Empresas, num universo de 1840 inquiridos de 23 países (América do Norte, América Latina, região EMEA (Europa, Médio Oriente e África) e região Ásia-Pacifico) é elevado o número de organizações que ainda não atribuem a importância devida a preparações contra desastres, pois metade das organizações que tem plano de preparação contra desastres, apenas o fez após sofrerem as consequências de perda de dados ou interrupções de disponibilidade de serviço. (*SMB Disaster Preparedness Survey*, 2011)

De acordo com o mesmo estudo, e mediante o universo inquirido, apresenta-se de seguida alguns dos resultados pertinentes:

a. Existência de Plano de Recuperação implementado
b. Existência de Plano de Recuperação implementado nos últimos 6 meses
c. Testes efectuados aos Planos de Recuperação implementados
d. Actividade em áreas consideradas critica e com elevadas hipóteses de desastres naturais
e. Frequência de Cópias de segurança semanais
f. Frequência de Cópias de segurança diários
g. Preparação de desastre como prioridade
h. Desastre poderia corresponder a pelo menos 40% de perda de dados

O gráfico seguinte ilustra parte dos resultados mais significativos do estudo:

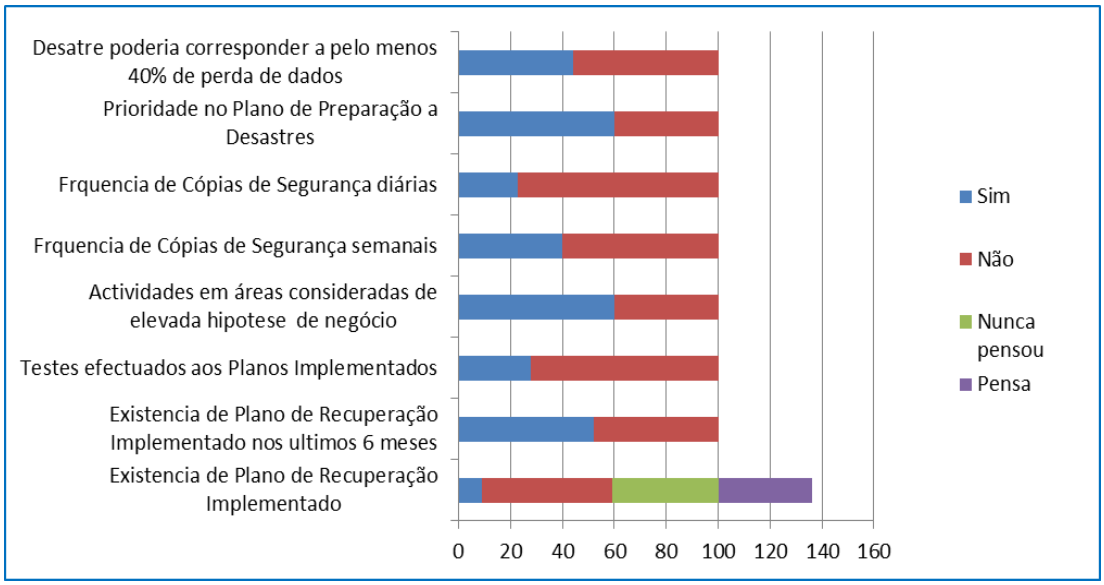

Figura 2.16 - Resultados do Estudo

.

2.2. Conclusão

Em conclusão, e de acordo com o estudo realizado, o risco é elevado, e a acontecer o que indica o estudo, algumas organizações serão forçadas a encerrar a sua actividade, mas aparentemente é uma realidade que parecem não acreditar. Um simples planeamento bem estruturado permitirá a sua protecção, da sua imagem e dos seus clientes.

Apesar do resultado apresentado no estudo, perante acontecimentos recentes de desastres naturais ou terroristas, perante a sua exposição a este tipo de risco iminente, é verificado um aumento do número de organizações que começam a incluir nas suas estruturas, metodologias, estratégias e ferramentas de Continuidade de Negócio e Recuperação de Desastres, pois o impacto seja este operacional ou estratégico ao nível do negócio traduz-se numa possível perda financeira e de clientes bem como quota de mercado e eventuais obrigações legais.

Capitulo 3. Metodologia

3.1. Modelo Metodológico

De acordo com o âmbito e objectivos propostos, apresenta-se a metodologia adoptada, a qual possibilitou a obtenção da informação e dos resultados apresentados neste trabalho através do Modelo Metodológico em Cascata (Silva E Videira, 2001).

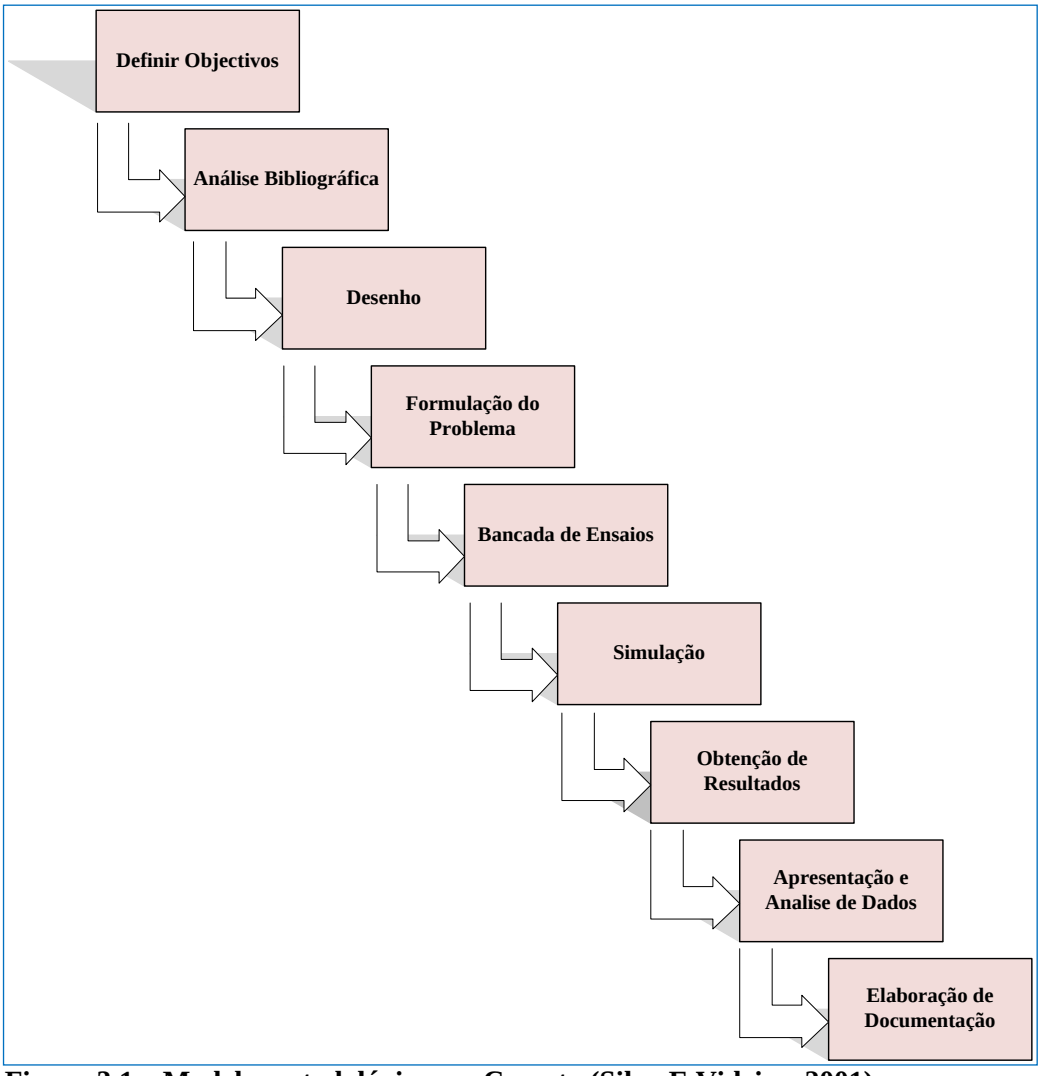

Figura 3.1 – Modelo metodológico em Cascata (Silva E Videira, 2001)

Neste capítulo são apresentadas as abordagens seguidas, representações de Input e Outputs, apresentação do Algoritmo de requisitos Multidimensionais, o modelo, conceito e descrição de *Job-Shop* utilizado.

É ainda feita uma descrição da bancada de ensaios a implementar em laboratório para a obtenção dos resultados e respectivas conclusões.

3.2. Representação do Input

De acordo com o modelo metodológico utilizado, e pesquisas várias sobre problemas de escalonamento e planeamento de tarefas e operações, considerou-se uma abordagem recorrendo a uma *Hierarchical Task Network* (HTN).

Verificou-se que o problema da optimização dos Planos de Salvaguarda e Reposição, combina com esta abordagem sugerida, ou seja, tarefas heterogéneas que partilham uma *pool* de recursos, da qual retiram temporariamente (inteira ou parcialmente) um conjunto de recursos para realizar uma lista de Operações.

As tarefas, definidas e referenciadas pelos responsáveis ou utilizadores de um determinado sistema ou ambiente, são organizadas por prioridades e/ou precedências, formando um grafo acíclico. Este grafo será submetido ao algoritmo para determinar qual a melhor sequenciação das tarefas numa janela temporal.

É importante referir que o algoritmo não define as Tarefas e as suas precedências, mas sim implementa e respeita a hierarquia definida pelo utilizador, optimizando e paralelizando operações segundo a sua utilização de recursos disponíveis.

3.3. Representação do Output

Para um melhor entendimento da representação de saída, apresenta-se de forma esquemática uma estrutura simples de salvaguarda e de reposição.

Essencialmente é necessário proceder à recolha de vários valores de variáveis previamente identificadas, seleccionar e execução do método de geração do Plano de Salvaguarda a utilizar perante um determinado ambiente.

O objectivo é proceder à salvaguarda completa ou parcial e segundo as melhores práticas recomendadas para o efeito, para que em caso de falha, o ambiente/sistema possa ser reposto com a maior qualidade (menor perda).

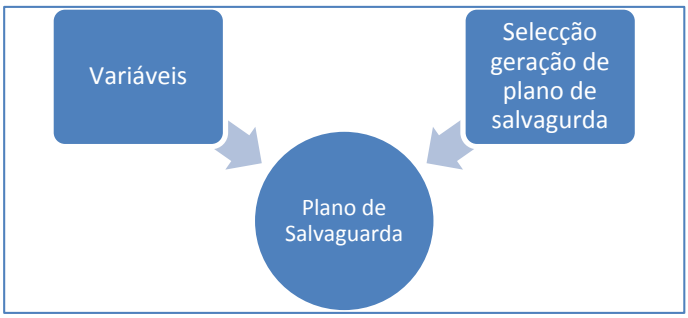

Figura 3.2 - Plano de Salvaguarda

De uma forma muito semelhante ao Plano de salvaguarda, o plano de reposição igualmente recolhe um conjunto de valores fornecidos por variáveis, aos quais são adicionados requisitos específicos dos ambientes em causa e executado o plano**.**

A reposição poderá ser total ao parcial e deverá ocorrer numa janela temporal aceitável perante as regras definidas pela organização, bem como a taxa de sucesso dessa reposição ser o mais próximo de 100%.

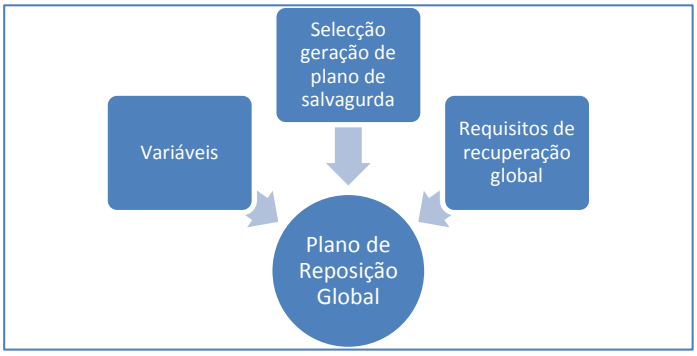

Figura 3.3 - Plano de reposição global

3.4. Algoritmo de requisitos multidimensionais

Para definir, com sucesso um Plano, seja de Salvaguarda ou Reposição, é necessário definir as Propriedades que serão utilizadas para caracterizar os Recursos, salientando-se que as Propriedades são genéricas e podem ser aplicadas a quaisquer Recursos que venham a ser definidos, e que são caracterizadas por uma unidade de medida, que pode ser de um de 3 tipos pré-estabelecidos: Volume, *Throughput* e Tempo.

Quando as propriedades forem associadas a Recursos, é especificado o valor dessa Propriedade para o Recurso em causa.

Os Recursos são posteriormente associados a Operações, não sendo estabelecida uma ordem pela qual os Recursos são utilizados numa Operação.

Se a uma Operação não for atribuída a nenhum Recurso, estamos perante uma validação, neste caso, é necessário atribuir a duração dessa validação, caso contrário, neste passo, é calculada a duração da Operação, consoante as Propriedades dos Recursos associados. Não será, nesta fase, implementada qualquer restrição quanto aos Recursos, depende apenas do utilizador a correcta definição das Propriedades de cada recurso e dos Recursos associados a cada Operação.

A duração de uma Operação é dada pela seguinte expressão:

$$T_{Operação} = \frac{\sum Propriedades_{Volume}}{min(Propriedades_{Throughput})}$$

Esta expressão traduz a noção de que a duração de uma Operação é dada pelo quociente entre o somatório de todas as Propriedades do tipo Volume, dos Recursos associados à Operação, e o valor mais baixo para as Propriedades do tipo *Throughput*, para esses mesmos Recursos associados.

O Plano é realizado por passos, após a optimização, que encadeia as Operações de cada Tarefa segue-se uma fase de análise desse Plano. O resultado dessa análise é apresentado num diagrama de *Gantt*. A fase de análise contempla as restrições identificadas no parágrafo anterior, montando o Plano por estágios. Estes estágios definem um intervalo de tempo, durante o qual são executadas Operações paralelizáveis.

A validação do tempo total de execução ser inferior ao especificado no RTO/RPO, é realizada após o cálculo da duração total do Plano optimizado. Como este é o Plano óptimo, apesar que não pode haver nenhum outro Plano que permita paralelizar as Operações de tal forma a que a duração total seja inferior à deste.

Enuncia-se seguidamente, a metodologia seguida para consolidação do Plano de salvaguarda ou reposição. Salienta-se que o plano produzido é sempre exequível, em termos de dependências.

Dada uma lista de Operações de salvaguarda ou reposição por executar deve-se:

- Criar um novo estágio de execução, vazio;
- Criar uma lista de Operações executadas;
- Criar uma lista de Operações não executáveis no estágio actual;
- Se a lista definida em 3. for vazia, apresentar esta lista de Operações resultante da optimização do Problema;
- Escolher uma Operação da lista de não executadas;
- Se o estágio actual contém a mesma Operação associada a outra Tarefa, ou uma Operação diferente da mesma Tarefa, adicionar a Operação à lista de Operações não executáveis. Voltar ao Passo 5;
- Adicionar a Operação ao estágio actual;
- Se existem mais Operações na lista de por executar, voltar ao Passo 5.;
- Se a lista de Operações não executáveis no estágio actual não for vazia, tomar esta como a lista de Operações por executar. Voltar ao Passo 1.;
- Para cada estágio, aplicar a fórmula $T_{Estágio_n} = \max(T_{Operação_A}) : Operação_A \in Estágio_n$ para obter a duração do estágio (duração do estágio é igual à duração da mais longa Operação contida no estágio);
- Obter a duração total do Plano através da fórmula $T_{Plano} = \sum_{i=0}^{n_{Estágios}} T_{Estágio(i)}$ (somatório das durações dos estágios).
- Do resultado visível deste algoritmo de análise aplicado ao resultado da optimização, será um diagrama de *Gantt* com a representação da sequência das tarefas.

Apresenta-se de seguida um diagrama que ilustra o fluxo de informação e de acção na criação de um plano.

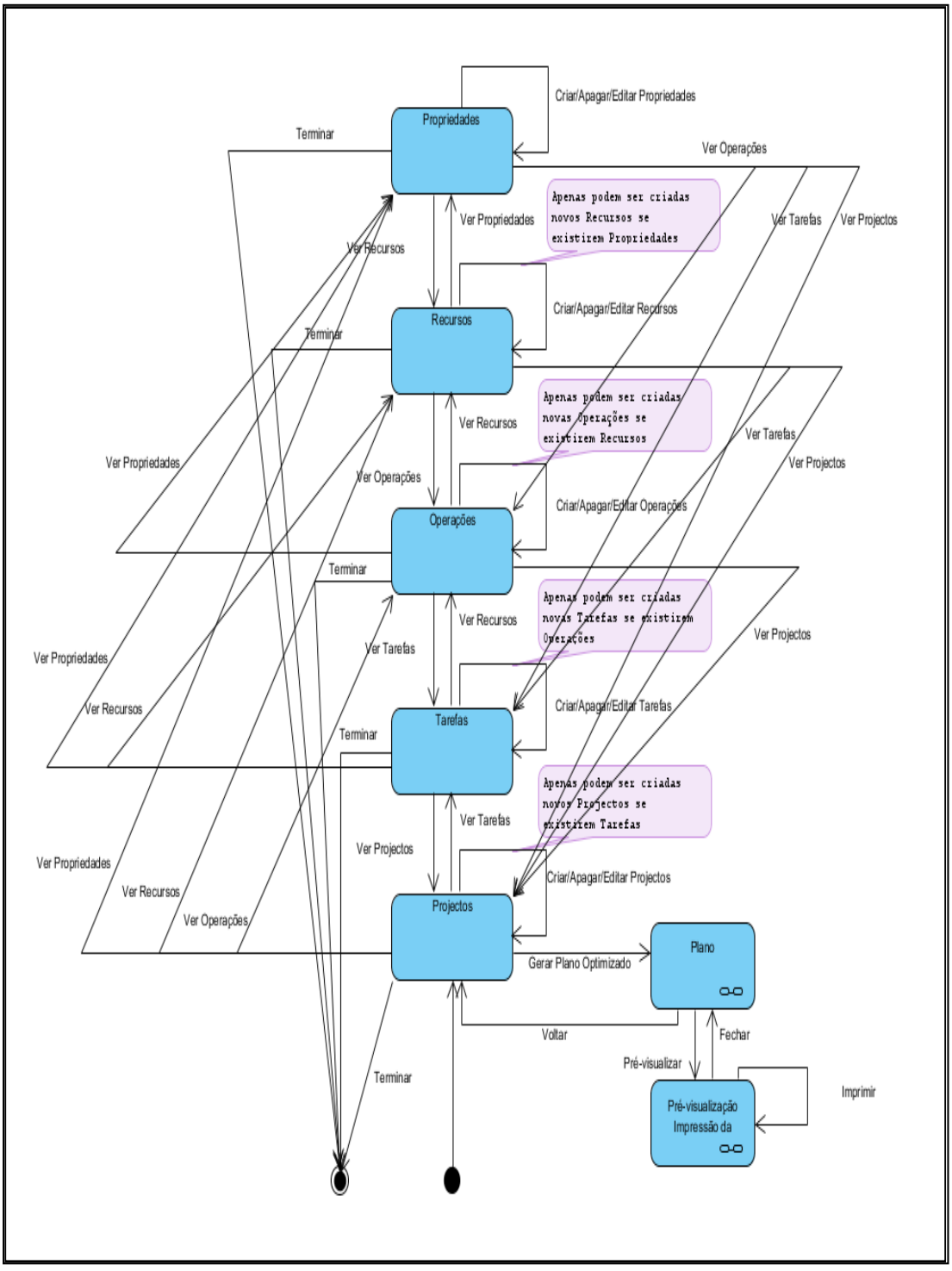

Figura 3.4 - Estado do Fluxo de informação e acções no Diagrama de Estados

Perante a breve descrição anterior, estão assim caracterizados de forma sumária os requisitos multidimensionais a utilizar pelos algoritmos em âmbito da Bancada de Ensaios a detalhar posteriormente, falta ainda definir como será feito o escalonamento e planeamento.

3.4.1. Modelo *Job-shop*

No momento para se iniciar a implementação de laboratório, é necessário apresentar como será feito o escalonamento e planeamento das tarefas e a respectiva apresentação do plano.

De acordo com os resultados obtidos nas pesquisas prévias referenciadas no capítulo respectivo deste trabalho, houve uma exaustiva procura de solidificar conhecimento para o âmbito e objectivos em causa, nomeadamente nas questões de alocação temporal de tarefas e operações a recursos e optimização e representação de resultados.

Um problema de *Job-Shop* clássico, consiste num conjunto de recursos (servidores para o caso), com a capacidade de processar as várias operações das tarefas previstas (*Jobs*) mediante um determinado tempo.

É assim identificado um possível modelo que revela características para que a sua utilização se enquadre no âmbito pretendido, o modelo *Job-Shop* numa vertente clássica.

Neste capítulo, procurar-se-á criar uma prova de conceito da exequibilidade de um sistema automático para escalonamento em tempo real (*online Scheduling*). A implementação será materializada através de um simulador criado para o efeito.

Na literatura existem alguns exemplos de experiências idênticas deste processo de naturezas diversas, mas quase sempre modelados na área da investigação operacional.

Uma experiência semelhante foi feita com encaminhamento de tráfego e envio de encomendas (Bent, 2004). Nesta experiência, um modelo de *Job-Shop* foi usado para testar a eficiência de diversos algoritmos. Pode-se conjecturar um cenário onde as tarefas de manutenção e funcionamento de um sistema complexo são escalonadas por uma aplicação de *software* – o escalonador, sendo encaminhada para as diversas máquinas (os recursos) que cumprem as tarefas.

Usa-se uma abordagem dinâmica (*online Scheduling*), na qual as tarefas chegam ao escalonador ao longo de um período temporal para a respectiva execução.

O modelo *Job-Shop* desempenha um papel importante no interesse da comunidade científica, comprovando-se tal facto pela diversidade de casos de estudo existentes. Existe ainda a tentativa da utilização do modelo *Job-Shop* em ambientes considerados de difíceis para este modelo. O problema consiste na atribuição de operações/tarefas ao escalonador com o objectivo de minimizar o tempo total de execução, podendo ser utilizadas várias funções de avaliação, a titulo de exemplo o *makespan*, (Gonçalves e tal., 2004). Cientificamente não está provado qual a melhor medida de avaliação, no entanto o *makespan*

é a mais utilizada (Park et al., 2003), pelo que esta será a avaliação a utilizar na Bancada de Ensaios.

3.5. Bancada de ensaios

A bancada de ensaios descrita no presente documento baseia-se num ambiente real em Open Source Software (OSS) criado para o efeito, servindo de laboratório a um ambiente de Planeamento de Estratégias de Salvaguarda e Reposição de Dados/Informação.

Com a bancada de ensaios pretende-se inovar a partir de resultados (algoritmos) devidamente comprovados de outros investigadores, e não desenvolver métodos específicos e trabalhos de forma independente, para que seja acessível a continuidade do desenvolvimento deste âmbito ou mesmo a sua adequação.

Para medição de eficácia dos processos a testar, usou-se um modelo idêntico ao mostrado da figura 3.4. Este tipo de modelo é idêntico ao utilizado por Nazarathy (2001), onde são testados vários cenários.

Devem ser considerados quatro tipos de objectos

- Recurso, representa uma máquina (computador) no mundo que se pretende modelar;
- Tarefa;
- Escalonador, responsável por associar recursos a tarefas, seguindo um algoritmo;
- Cenário, sequência de tarefas

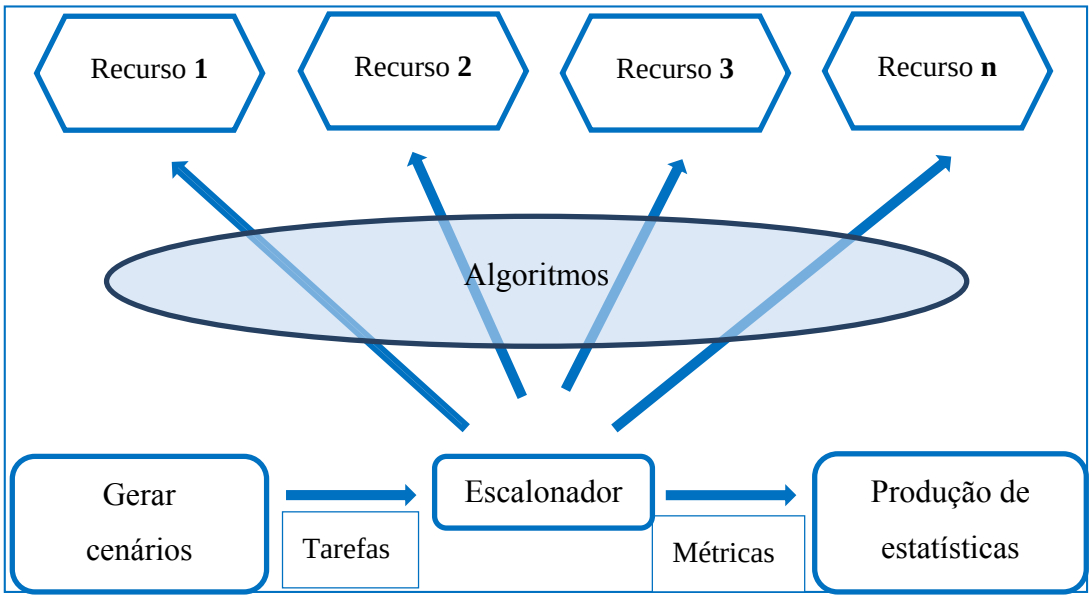

Figura 3.5 - Escalonador

Neste modelo, um conjunto de cenários é passado a um escalonador. Este componente está configurado para testar os algoritmos em cada medição, produzindo estatísticas para cada cenário.

O escalonador funciona segundo o seguinte fluxo de trabalho:

1. O gerador de cenários gera um conjunto aleatório de tarefas a ser executadas. Contém tanto a tarefa propriamente dita (com a sua duração, *deadline* e recursos que consome) como o instante ao qual é passada ao escalonador;
2. O escalonador, a cada instante, pode receber uma ou mais tarefas, que deverá despachar para um recurso. O despacho da tarefa para cada recurso é efectuado segundo um método, materializado num algoritmo;
3. Após o instante final, o plano (*schedule*) gerado com as tarefas efectuadas pelos recursos, é analisado e são geradas estatísticas sob a forma de ficheiros do tipo csv e xml. As tarefas não efectuadas são tidas em consideração para esta estatística

Um cenário é uma sequência de tarefas (*jobs*), em que cada tarefa pode ser de um dos seguintes tipos:

- Salvaguarda de dados
- Reposição de dados
- Validações / Verificações

Seguidamente nos pontos 3.5.1 a 3.5.3 detalha-se as componentes principais do escalonador, nomeadamente:

- Requisitos
- Gerador de cenários
- Simulador
- Produção de resultados

3.5.1. Requisitos

Também os requisitos fazem parte dos objetivos da simplicidade e facilidade, embora estes enquadrados num âmbito de utilização de recursos de requisitos normais disponíveis. Descreve-se os recursos utilizados na bancada de ensaios ao nível do:

- Equipamento
- Software

Na bancada de ensaios, foi utilizado um equipamento de características mínimas, nomeadamente um PC portátil com as seguintes características principais e mais relevantes:

- CPU2,53 Ghz 1 Core
- 2 GB de Ram
- Software (OSS - Open Source Software)
- Java 1.6.0.26
- Base de Dados Mysql
- MySql WorkBench
- Apache Tomcat
- Sistema Operativo Linux Ubuntu
- Oracle VM VirtualBox
- Mobile Android
- Browser FireFox

3.5.2. Cenários a utilizar

Na bancada de ensaios foi igualmente utilizado um gerador de cenários, este gerador possibilitou o carregamento automático das tarefas e operações e respectivos recursos na BD especifica para o efeito (BD – MysSQL como modelo de dados no capitulo de anexos).

Como valores de entrada é solicitado pelo gerador o número de tarefas e operações a utilizar no momento, sendo os recursos utilizados de forma fixa e estática, ou seja, para a mesma tarefa e operação os recursos atribuídos não alteram.

Na utilização do gerador não é tido em conta que tipo de tarefas ou de operação, no entanto as relações de ambas são consideradas (procedências e prioridades).

Revendo informação anterior e sumarizando para um melhor entendimento, recorda-se que, cada cenário tem um conjunto de tarefas, tarefas estas têm associadas operações, por sua vez, estas operações tem associado um lista de recursos, onde é afecto a percentagem de ocupação deste à tarefa em causa.

Como exemplo um cenário simples, se considerarmos um universo de 10 tarefas em que cada tarefa tem 4 operações, será obtido um total de 40 combinações, ou seja, o *scheduling* terá uma representação de 40 combinações, para o efeito, na bancada de ensaios será utilizado apenas 1 recurso.

Como exemplo, o esquema da figura 3.6 ilustra o funcionamento do gerador anteriormente descrito, e tendo como parâmetros de entrada o nome do projecto (ex: mssi_10x2), número de tarefas (10) e número de operações (2).

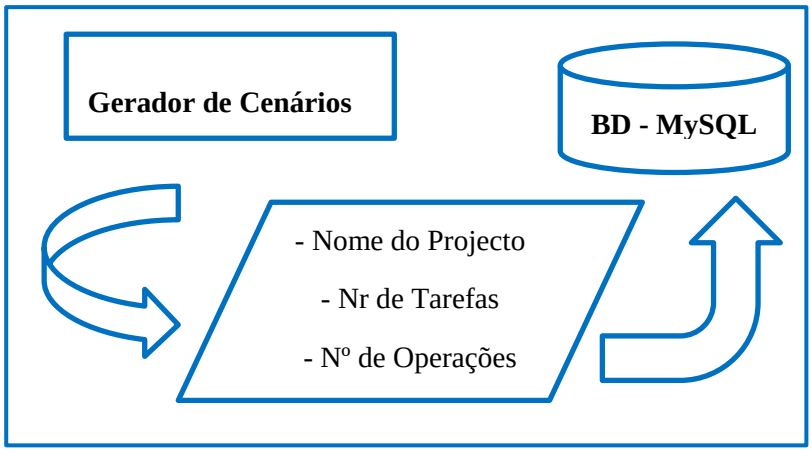

Figura 3.6 – Diagrama Gerador de cenários

3.5.3. Simulador

Para criar planos de escalonamento de tarefas e operações em tempo real e de uma forma automática foi construído para o efeito um sistema automático, o simulador.

O simulador usa um modelo Job-Shop para testar a eficiência de diversos algoritmos. O resultado do simulador consiste num cenário de execução de tarefas de manutenção e funcionamento de um sistema complexo, as quais poderão ser escalonadas para serem executadas pelas diversas máquinas envolvidas. É utilizada uma abordagem dinâmica (online scheduling), na qual as tarefas chegam ao escalonador ao longo de um período temporal.

Para mostrar os resultados da *framework*, e comprovar a escolha do algoritmo, procedeu-se à divisão da *framework* em duas componentes, o simulador e os utilitários que permite a visualização de resultados através de diagramas de *Gantt*.

Importante sublinhar que esta divisão permite facilitar a produção de dados de bancada para posteriormente serem interpretados e originar as conclusões possíveis.

É necessário indicar ao simulador qual o Projecto a executar (existente na BD), e o número de iterações pretendido como repetição. Após a execução do número de iterações para um

determinado projecto, o resultado da execução para o universo de algoritmos previsto, será guardado em dois ficheiros distintos, XML para posteriormente serem interpretados pelo utilitário *Gantt* e ficheiro de texto de formato csv para trabalho estatístico que dará origem a valores em tabela e interpretação gráfica.

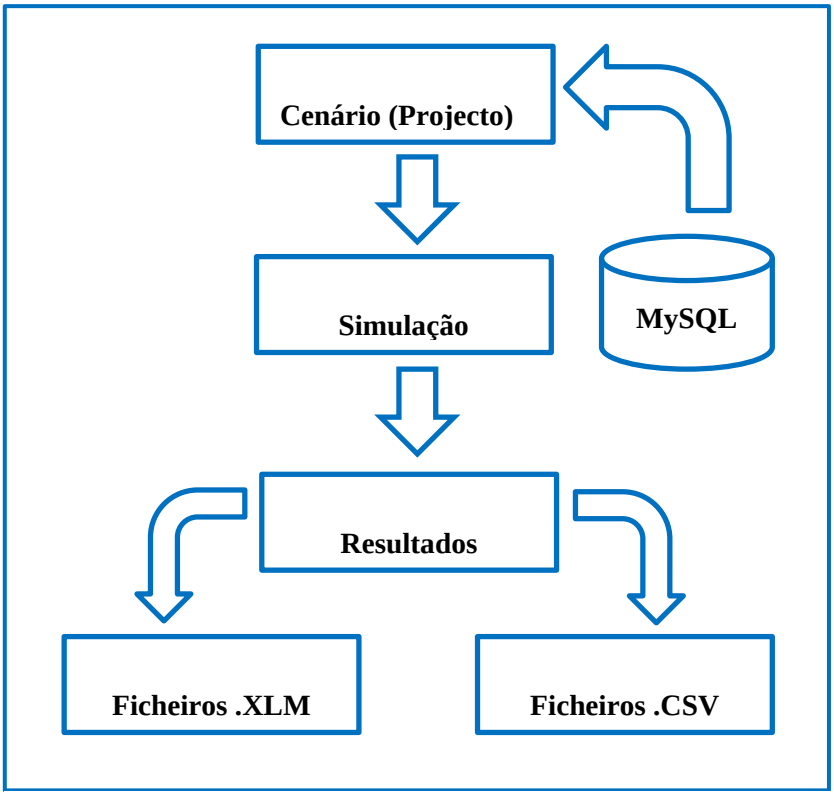

Figura 3.7 - Diagrama de simulação

Em resumo, o simulador tem como objectivo gerar um plano optimizado de execução das tarefas de um determinado problema, o plano gerado será o plano mais otimizado, ou seja, o plano que respeita todas as restrições especificadas e executa no menor tempo possível.

O processo de obtenção do plano mais optimizado recorre à execução de múltiplos algoritmos num modelo *Job-Shop*, aos quais foram passadas diversos parâmetros.

O objectivo do simulador é encontrar, em tempo útil, o máximo (ou mínimo) global. Um potencial algoritmo consistiria em testar todos os caminhos, o que o torna pouco eficiente já que o número de combinações é muito grande. Para resolver problemas deste tipo, têm vindo a ser pensados e desenhados algoritmos específicos, a maior parte baseado em heurísticas, os quais permitem encontrar combinações que respeitam todas as restrições de forma mais rápida do que testar todos os caminhos.

Descreve-se as principais componentes do resultado produzido pelo simulador:

- Data de início
- Algoritmo seleccionado
- ID de Projecto
- Total de Recursos
- Total de Operações
- Total de tarefas
- Parâmetros do algoritmo
- Detalhe do Projecto (XML com a definição total do projecto)
- Detalhe do *schedule* resultado
- Valor da função utilidade
- Data de fim
- Tempo total de execução
- Consumo médio de CPU
- Consumo médio de memória

Sendo um dos objectivos a utilização de OSS (Open Source Software), a geração de ficheiros do tipo xml e csv permitem com facilidade que outras ferramentas existentes e fora do âmbito deste trabalho, consigam carregar a informação existente nos ficheiros e assim possibilitar a sua análise em ferramentas já existentes, facilitando a interoperabilidade da *framework* com outras ferramentas existentes no mercado ou especificas a cada ambiente.

No ponto 3.5.4 foi descrito o modelo utilizado pelo simulador para testar a eficiência de diversos algoritmos, bem conhecidos das comunidades de programadores possuidores de características, adequação e complexidades diferentes tendo em conta as mais variadas situações de utilização.

Para uma demostração dos objetivos deste trabalho utilizou-se vários tipos de algoritmos mediante o modelo Job-Shop sem que fosse incluído qualquer tipo de especificidade que provocasse complexidade ou dificuldade de interpretação de programadores menos experientes.

Seguidamente pretende-se descrever os algoritmos utilizados no simulador, bem como a simplicidade da sua adequação de situações comuns para uma situação de escalonamento no modelo de *Job-Shop*.

3.5.4. Algoritmos

Nesta secção vamos mostrar a utilização de 5 diferentes algoritmos e posteriormente verificar a sua eficácia perante a respectiva actuação no simulador na bancada de ensaios, e proceder à análise de resultados.

Sendo o âmbito deste trabalho evitar matérias específicas e de alguma complexidade possibilitando eventuais ajustes a cenários individualizados de diferentes necessidades por parte das equipas de desenvolvimento aplicacional, utilizou-se algoritmos bem conhecidos do mundo académico para servirem de ponto de partida, não sendo do âmbito procurar/implementar um algoritmo específico para fornecer os melhores resultados mediante os cenários de utilização.

Assim foram testados na bancada de ensaios os seguintes algoritmos:

- Aleatório;
- Ganancioso;
- Regras / Heurísticas
- Genético;
- Programação linear

Para uma melhor compreensão vamos ainda incluir no capítulo 6 (anexos) uma representação de implementação em pseudo-código devidamente comentada em linguagem natural para cada um dos algoritmos utilizados na bancada de ensaios.

Seguidamente e para cada um dos algoritmos, haverá uma breve descrição textual dos algoritmos ao nível de conceito.

- Aleatório

Neste algoritmo as soluções são geradas alectoriamente, para cada iteração uma solução é gerada aleatoriamente e comparada com uma solução da iteração anterior. A geração de uma boa solução pode ser morosa devida à diversidade de soluções geradas, incluindo as soluções menos boas.

Na bancada de ensaios o escalonador gera cenários válidos, escolhendo aquele com que minimiza os atrasos, de um conjunto de simulações geradas, sendo as tarefas associadas aos recursos necessários para a sua execução.

O mecanismo de procura pode ser resumido em:

1. Quando uma tarefa T chega, é procurado um instante aleatório, entre o instante actual e o instante Tdeadline – Tduration
2. Se não é possível, é procurado qualquer instante a partir do momento actual
3. Se tal não for possível, a tarefa é colocada na lista das tarefas não efectuadas

Os passos anteriores são efectuados num número finito de tentativas aleatórias. Este algoritmo tem pouca racionalidade na atribuição, é usado como base de comparação em relação aos restantes algoritmos testados neste capítulo, prevendo-se desde já que, tendo em conta a existência do factor tempo de execução, a sua eficiência tende a baixar com cenários mais exigentes, mesmo nos de média exigência.

Assume-se que o escalonador tem um array de tarefas não associadas a recursos, o *unalocated_tasks*, onde as ultimas são adicionadas pelo escalonador através do método *Scheduler.addTask(task).* Outro vector global ao objecto, o *dropped_tasks*, guarda as tarefas que foram ignoradas, por não se ter conseguido encontrar escalonamento possível, os recursos de uma tarefa (task) são guardados no vector *task.resources*.

Perante as características deste algoritmo, antes da sua implementação é desde já esperado que não será certamente uma hipótese de solução, no entanto será considerado.

- Ganancioso

Para este tipo de algoritmo, o escalonador escolhe sempre o "óptimo local", para cada nova tarefa que chegue num dado instante.

No contexto do escalonamento dinâmico, o escalonador escolhe sempre o instante mais próximo em que os recursos necessários estão disponíveis.

A racionalidade deste algoritmo relaciona-se com a optimização local ser uma optimização simples, com optimização algo óbvia.

A computação deste algoritmo é a mais rápida de entre os testes, a utilização de abordagens do tipo ganancioso é descrita por Dantzig (1957), como solução para problemas do tipo *knapsack* (alocação de recursos com restrições perante um limite)

Este algoritmo tenta encontrar o instante mais cedo onde é possível escalonar a tarefa em todos os recursos. Para isso usa uma lista com todas as tarefas não alocadas e por cada tarefa, tenta encontrar o instante mais cedo onde seja possível alocar a tarefa em todos os recursos. Caso consiga uma alocação correta, a tarefa é escalonada, caso contrário é descartada.

Espera-se que este algoritmo produza uma solução para o problema mas que essa solução não seja a solução ótima do problema. Isto deve-se ao facto de pegar nas tarefas e aloca-las independentemente das consequências que essa alocação tenha no futuro.

- Regras / Heurísticas

Em Montana (2007), foi elaborado estudo sobre a eficácia a adopção de regras em cenários do tipo *job-Shop*. Para este algoritmo são determinadas regras de alocação de tarefas a recursos, sendo os recursos atribuídos segundo regras.

Neste algoritmo, cada vez que uma tarefa é enviada ao escalonador, o melhor instante disponível é determinado seguindo duas heurísticas:

1. Instante possível mais próximo do instante actual, para todos os recursos necessários para levar a tarefa a cabo (idêntico ao algoritmo ganancioso);
2. Instante de atribuição que resulte no menor número de recursos disponíveis.

A racionalidade por detrás deste algoritmo não só é a mesma do algoritmo ganancioso como também procura optimizar mais o processo, além do óptimo local.

É introduzido o método de recombinação das tarefas ainda não efectuadas da seguinte forma:

- Todas as tarefas já associadas a recursos são "retiradas" do escalonamento e colocadas numa determinada lista
- A lista é reordenada, onde as tarefas com maior prioridade e duração mais curta são colocadas em primeiro na lista

O algoritmo é corrido sobre a lista de tarefas do problema previamente ordenada, ordenação essa é efectuada mediante um peso relativo de vários factores como por exemplo, a duração da tarefa, a dependência entre tarefas, etc.. As tarefas são então escalonadas de acordo com o seu peso.

O primeiro factor que se tem em conta é a dependência entre tarefas. Caso existam dependências entre tarefas, é atribuído um factor máximo de importância na comparação. Caso não existam dependências entre tarefas, é dada uma importância a cada tarefa baseada na seguinte fórmula para ordenar a lista:

$$\text{task importance} = \frac{\text{task duration}}{10} \times \left(10 - \frac{\text{number of tasks}}{\text{task index}}\right)$$

- Genético

O escalonador considera a população inicial e um qualquer universo de vários *Schedules* aleatórios, dos quais escolhe 2 de forma aleatória como a selecção de pais que permitirão a reprodução inicial. Cada tem um valor de *Fitness* [2] calculado mediante a métrica do *makespan*[3]. Para cada iteração existe o pressuposto da existência de uma geração, que consiste no conjunto de sobreviventes das gerações anteriores. Em cada iteração será reproduzido um novo elemento que resulta da mutação[4] ou cruzamento (crossover)[5] dos pais, e assume-se que este terá um valor de *fitness* optimizado relativamente aos seus pais. O algoritmo terminará quando nenhuma iteração consegue optimizar o *fitness* da geração anterior.

Este algoritmo tem uma sequência de execução que pode ser representada de forma faseada em 8 fases.

Descrição de uma iteração simples do AG mediante as fases anteriormente anunciadas:

Passo 1: **Inicialização**

Ler o tamanho da população, K, e taxa de mutação, *pm*.

Inicializar cromossomas gerando soluções factíveis no tamanho da população.

Passo 2: **Cálculo (utilização da função fitness)**

Calcular os valores de *fitness* de cada indivíduo da população inicial e validar se a condição de paragem foi conseguida, se sim dirige-se para 8.

Passo 3: **Selecção dos pais**

Seleccionar aleatoriamente dois cromossomas da população, considerando a

---

[2] Total obtido através da função *Fitness* sobre um determinado *schedule* resultado, com base no tempo total do *schedule* minorado pelo total da ocupação dos recursos envolvidos

[3] Diferença entre o início e o fim de sequências de tarefas

[4] Considera um *schedule* como potencial solução, a operação de mutação incide no encandeamento de operações, se possível alterando a sequência destas de forma a minimizar o tempo total de *schedule*

[5] Considera 2 *schedules* candidatos a solução, a operação de cruzamento tenta que estes candidatos a combinação para encontrar um novo *schedule* que minimize o tempo total das várias operações

probabilidade de escolha associada ao *fitness* de cada um.

Passo 4: **Geração de descendência**

Aplicando o operador *crossover*, gerar dois cromossomas a partir dos pais seleccionados no passo 3.

Passo 5: **Fim da geração de descendência**

Repetir os passos 3 e 4 se o tamanho da geração de descendentes for < K; caso contrário, ir para o passo 6.

Passo 6: **Mutação**

Para cada indivíduo da população, varrer os elementos de cada cromossoma, modificando-os aleatoriamente, com determinada probabilidade, onde é possível introduzir nova informação genética na população

Passo 7: **Cálculo do fitness**

Calcular o *fitness* para os cromossomas descendentes.

Passo 8: **Finalização**

Caso o critério de finalização seja alcançado, parar; caso contrário, dirigir-se ao a 3.

- Programação Linear

É o algoritmo que à partida se posicionará como o menos *evoluído* em termos computacionais. O seu funcionamento é básico: calcula todas as combinações possíveis de instantes x operações, e iterativamente percorre todas as soluções analisando qual a que tem melhor tempo.

Num problema de x máquinas, e que existem y operações a serem executadas em cada máquina, o total de possibilidades por cada iteração será dado pela fórmula de Alan S. Manne, na qual teremos n tasks + m possíveis conflitos

N=y*x

M= ½(x)(y)(y-1)

Comportamentalmente, gera todas as combinações possíveis para a resolução de um problema. Este algoritmo termina mediante um parâmetro definido que representa o número máximo de combinações calculado (caso nesse número máximo de combinações não seja encontrado um resultado válido, o algoritmo continua a correr até encontrar pelo menos um válido).

Este algoritmo se não estiver restringido por um número máximo de combinações irá encontrar sempre a melhor solução possível para o problema. Devido a este facto encontrar a melhor solução poderá necessitar de um tempo considerável até obter a solução ideal.

Ao nível do cálculo e de condição de paragem, o valor pré-definido atribuído foi de 50000. Este valor pode ser ajustado no ficheiro externo de configuração. Aumentando o número máximo de combinações aumenta-se a probabilidade da solução óptima estar entre uma das combinações visitadas.

3.5.5. Função Utilidade

Uma primeira avaliação das soluções encontradas é feita através do recurso a uma função de utilidade, função essa responsável pela avaliação final da utilidade de determinada solução para todos os algoritmos da bancada de ensaio, levando em linha de conta os seguintes factores:

- Ocupação média ponderada com ocupação de CPU e gasto de memória (+)
- Instante final i.e., tempo total do *schedule* seleccionado como solução final (-)
- Tarefas não alocadas, i.e., tarefas que deveriam fazer parte do s*chedule* mas que tiveram que ser descartadas para gerar um *schedule* válido e optimizado (-)

Para cada um destes factores existe um coeficiente que indica o peso na avaliação final da utilidade de uma solução. Todos estes coeficientes estão disponíveis como parâmetros do sistema:

- Ocupação média ponderada - utility.allocation_weight = 0,55 (no intervalo [0, 1]);
- Instante final - utility.end_time_weight = 0,30 (no intervalo [0, 1]);
- Tarefas não alocadas - utility.dropped_tasks_weight 0,15 (no intervalo [0, 1]).

O valor da função utilidade deverá estar compreendido num intervalo configurável, por omissão entre 0 e 100.

A fórmula de utilidade usada, conta com 55% para a solução apresentada (plano de ressalva que execute em menos tempo), 30% para o tempo total de cálculo e 15% para os recursos ocupados (Memória + tempo de CPU)

Após vários ensaios conclui-se que tendo em conta as atuais características dos recursos (equipamentos para execução) e a respectiva ocupação/utilização dos factores (Ram e CPU), a ocupação nunca seria um problema, dado que com facilidade é possível aumentar as características dos recursos, o mesmo não acontecendo quando se atinge o tempo máximo disponível.

Pelo exposto formulou-se uma outra função utilidade mais adequada ao problema e aos objectivos da bancada de ensaios. Tal pretende "premiar" o algoritmo que gastou menos tempo na execução e formulação do plano, i.e., promovendo o que tem melhores desempenhos.

A função utilidade só calcula de forma isolada cada algoritmo, permitindo esta correção atingirem-se valores mais justos e penalizando de forma exponencial as diferenças de tempo adicionais.

Considerou-se que:

- TE – tempo de execução do algoritmo na procura da solução a utilizar (algoritmo mais adequado)
- TJ – Tempo do *job-Shop/Schedule*, também possível de designar por *Timespan* (elaboração do plano e/ou execução do plano)
- TC – Tempo limite (Janela temporal para a elaboração do plano ou para a execução)
    - TC de 5 minutos que corresponde a 300 segundos para a elaboração do plano
    - TC de 8 horas ou 28.800 segundos para a elaboração e execução do plano

Se a solução encontrada adicionada ao tempo necessário para calcular (algoritmo de otimização) apresenta um tempo menor que TC, nesta situação, o "o resultado da função utilidade corrigida é dado por TJ+TE.

Se TJ+TE<TC, então a solução encontrada mais o tempo necessário para calcular (algoritmo de otimização) são menores que TC, nesta situação, o "resultado" (o mais eficiente) corrigido é dado por TJ+TE.

No entanto se TJ+TE > TC, a duração da solução encontrada mais o tempo do processamento é maior que o tempo máximo de duração do problema, haverá lugar a uma penalização de forma exponencial na diferença de tempo adicional.

O tempo adicional é dado por (TJ+TE-TC) e o valor de correção por TC + (TJ+TE-TC)^2

Tal resulta que, a melhor solução corresponderá ao melhor valor da função utilidade após a aplicação da correção.

Cada um dos resultados foi obtido através da média ponderada resultante da execução por 10 vezes de cada algoritmo para cada população.

A população é caracterizada por múltiplos de 10.

Tarefas	**Operações**	**População**
10	2	20
10	4	40
10	8	80
10	16	160

Tabela 3.1 Caracterização da "população"

3.5.6. Geração de Cenários

De acordo com as entradas fornecidas ao simulador, é esperada a produção de resultados que indiquem o plano de execução mais eficaz (com tempo total mais reduzido). Este plano descreve a alocação de tarefas e operações associadas a propriedades e recursos ao longo de um determinado período de tempo (*timespan*).

Detalha-se as componentes principais do plano, mais em concreto:

- Propriedades
- Recursos
- Operações
- Tarefas

3.5.6.1. Propriedades

As características mais determinantes das operações foram agrupadas em Propriedades, características essas definidas tal como na tabela seguinte:

Nome	**Descrição**	**Valor de utilização (%)**
Ocupação de CPU	% de ocupação máxima de 100%	100%
Ocupação de Memória (RAM)	% de ocupação máxima de 100%	100%
Largura de Banda	% de ocupação máxima de 100%	100%
Unidade de Leitura/Escrita	1 utilização = 100%	100%

Tabela 3.2 Caracterização das propriedades

3.5.6.2. Recursos

Para a execução das tarefas serão necessárias máquinas, as quais são caracterizadas também por um conjunto de características (propriedades), os recursos.

Nome	Características	Valor de utilização (%)
Servidor de Salvagurada/Reposição	CPU	100%
	RAM	100%
	Rede (largura de Banda)	100%
Library	Unidade de Leitura/Escrita	100%
Armazenamento (*Storage*)	Disponibilidade de Espaço Livre	100%
	Rede (largura de banda)	100%
Bases de Dados	Volume de dados	100%

Tabela 3.3 Caracterização dos recursos

3.5.6.3. Operações

Do resultado da combinação de propriedades e recursos surgem as operações.

Nome	Recurso	Pré-requisitos	Efeito/consequência	Duração
- Salvaguarda 1 (armazenamento) - Reposição 1	Servidor de Salvagurada /Reposição *Tape* Disco Base Dados	- Procedência - Dependências	- Sucesso (avança) - Sem Sucesso (aguarda)	Estimada pelo plano e ajustado após operações seguintes
- Salvaguarda n - Reposição n	Servidor de Salvaguarda /Reposição	- Procedência - Dependências	- Sucesso (avança) - Sem Sucesso (aguarda)	Estimada pelo plano e ajustado após operações seguintes

Tabela 3.4 Caracterização das Operações

3.5.6.4. Tarefas

O simulador irá utilizar um conjunto de operações (trabalho) com uma determinada duração, tarefa.

Nome	Operações	Duração da tarefa	Nr de Operações
Salvaguarda / Reposição	Salvaguarda 1	x horas	y operações
	Reposição 1		
	Salvaguarda 1 / reposição 1		
	Salvaguarda n / reposição n		
Validações	Validação 1	x horas	y operações
	Validação n		

Tabela 3.5 Caracterização das Tarefas

3.6. Implementação

Para um determinado "problema de salvaguarda ou reposição" mantêm-se os mesmos recursos, incrementado o número de tarefas a executar num determinado momento.

Cada projecto possui um conjunto de tarefas, e por sua vez, cada uma destas tem operações, as quais tem associados recursos caracterizados por propriedades. Se a *x* tarefa estão afectas *y* operações, então estamos perante uma combinação de *x* tarefas por *y* operações utilizando os recursos afectos às mesmas, ou seja, o *Scheduling* terá uma representação de *x*y*.

A tabela seguinte apresenta uma matriz de relação entre Tarefas, operações, recursos e propriedades, bem como os eventuais efeitos e necessidades de pré-requisitos e repectivas durações de execução e elaboração do plano.

	Tarefas	**Operações**	**Recursos**	**Propriedades**	**Pré-requisitos**	**Efeito**	**Duração**
Tarefas		X					X
Operações			X		X	X	X
Recursos				X			

Tabela 3.6 Matriz de relação (Tarefa, Operação e Recurso)

3.6.1. Métricas

Como alvo de simulação propõe-se as seguintes execuções sendo o objectivo gerar um plano optimizado de um determinado problema. Na bancada de ensaios foram realizados:

Tarefas	**Operações**	**População**
10	2	20
10	4	40
10	8	80
10	16	160

Tabela 3.7 Métricas

Capitulo 4. Resultado

4.1. Análise e demonstração de resultados

4.1.1. Cenário de Salvaguarda e Reposição

O simulador tem como objectivo gerar um plano optimizado de execução das tarefas de um determinado problema; o plano gerado será o plano mais otimizado, ou seja, o plano que respeita todas as restrições especificadas e executa no menor tempo possível. O processo de obtenção do plano mais optimizado recorre à execução de múltiplos algoritmos num modelo *Job-Shop*, aos quais foram passadas os diversos parâmetros.

É descrito um problema de decisão de ordem NP (*Non-Deterministic Polynomial time*), podendo também ser visto como uma função não linear com vários máximos (ou mínimos). O objectivo do simulador é encontrar, em tempo útil, o máximo (ou mínimo) global.

Um potencial algoritmo consistiria em testar todos os caminhos, o que o torna pouco eficiente para o caso de o número de combinações ser muito grande.

4.1.2. Condições e parâmetros para execução

Na bancada de ensaios da execução do simulador, este recebe três parâmetros de entrada, descritos na tabela seguinte:

Parâmetros de entrada	
Tipo	**Descrição**
1. Algoritmos a utilizar	Identifica a lista de algoritmo a serem usados na execução
2. Parâmetros para os diferentes algoritmos	Ficheiro que contém as configurações necessárias para a execução do simulador
3. Tipo de Output a utilizar para fornecer o resultado	Especifica qual o ficheiro de output onde serão gravados os resultados

Tabela 4.1 Execução do Simulador

4.1.3. Execução

O "problema" descreve as tarefas, recursos e relacionamentos; baseando-se o simulador nessa descrição para recorrer a algoritmos específicos de modo a calcular um plano de execução optimizado.

Com recurso a uma Base de Dados Mysql, foi carregado toda a informação necessária, da sua estrutura, destacando-se os seguintes elementos:

- Nome;
- Duração máxima;
- Lista de recursos
- Listas de tarefas, estas dependem da disponibilidade dos recursos para terem condições de serem iniciadas. Tanto as tarefas como os recursos

4.1.4. Resultados

De acordo com as entradas fornecidas ao simulador, é esperada a produção de um ficheiro (ou mais) que indique o plano de execução mais eficaz (com tempo total mais reduzido). Este ficheiro descreve a alocação de tarefas associadas a recursos ao longo de um determinado período de tempo (timespan).

Imaginando um cenário hipotético representa-se de seguida uma interpretação visual de um plano de execução que permitiria uma recuperação de um desastre no mínimo tempo possível.

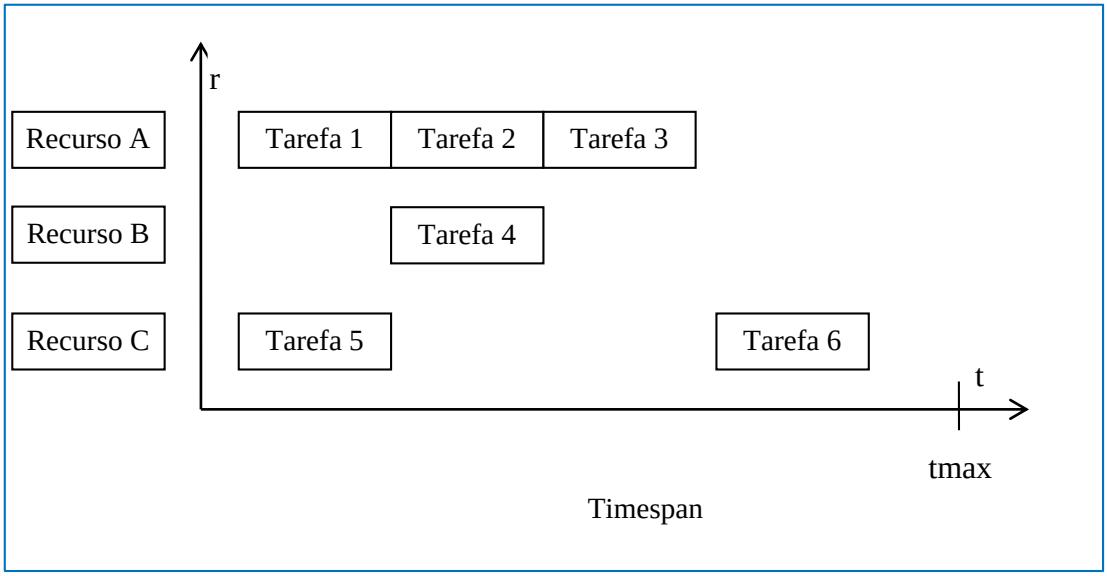

Figura 4.1 - Resultados para a comparação dos Algoritmos em análise

O problema em causa seria composto por 6 tarefas e 3 recursos. Neste cenário a tarefa 4 está dependente da tarefa 1 e a tarefa 6 está dependente da tarefa 3. O simulador, com base nessas restrições, chegou à sequência de tarefas acima

Com base nos resultados do simulador, é possível visualizar graficamente a ordem sugerida, a qual representa um plano que respeita as várias restrições (paralelismo dos recursos, dependências de tarefas, tempo máximo de conclusão do problema) pelo algoritmo utilizado.

4.2. Demonstração de resultados

A demonstração de resultados de desempenho de todos os algoritmos utilizados no simulador na bancada de ensaios será apresentada de duas formas distintas:

- Tabular;
- Gráfica

A apresentação mediante tabela permite visualizar os resultados na sua totalidade, estes resultados são posteriormente representados através de gráficos, permitindo uma melhor percepção através da visualização numa perspectiva evolutiva e conclusiva.

Fazendo uma aproximação á realidade, foram simulados três cenários diferentes e de elevada probabilidade de ocorrência no mundo real, para uma melhor compreensão foram ainda adicionados variáveis (x, y e z) nas fórmulas de cálculo da função utilidade.

Seguidamente descreve-se os cenários três cenários, nomeadamente:

1. Encontrar uma solução com o algoritmo mais apropriado (TE(x)) e elaborar um plano (TJ(x)), em que o tempo crítico (TC(x)) controla o tempo máximo na procura da solução e disponibilização do plano mediante gráfico de Gantt, ou seja, utilizar o simulador para verificar qual o algoritmo mais adequado ao cenário introduzindo um requisito limitador na janela temporal permitida para a execução do simulador (TC(x)), seguidamente formular o plano para o formato XML para que este possa ser visualizado pelo visualizador de Gantt;
2. Encontrar uma solução (TE(y)), elaborar e executar o plano (TJ(y)), sendo TC(y) a janela temporal disponível para executar o plano do cenário com sucesso, ou seja, idêntico ao ponto anterior adicionando ao TJ(y) o tempo de execução do plano. Desta forma o TJ(y) é o tempo de elaboração do plano adicionado ao tempo final de execução do plano (tempo final da ultima tarefa).
3. Reutilização de um plano anterior numa determinada janela temporal (TC(z)). Neste cenário não é feita a procura da melhor solução, mas sim a reutilização de um plano através da adaptação de um plano existente, desde que o TC(z) a utilizar seja =< ao TC(z), ou seja, o novo plano (ex: adicionar ou substituir tarefas) não poderá ter uma janela temporal de execução superior ao plano que a reutilizar.

4.2.1. Resultados obtidos para os vários algoritmos

Após o descritivo dos três cenários (ponto 4.2), seguidamente serão apresentados os resultados tendo em conta as características de cada um deles.

Na tabela 4.2 é sumarizada a produção de resultados pelo simulador mediante as condições já referidas, função utilidade e cenários para cada algoritmo.

1. Cenário (x) - Encontra solução TE(x), elabora plano TJ(x), e terá como resultado fu(x)= TE(x)+TJ(y) em unidades de segundos.

Cálculo: Fu(x)= SE((TJx+TEx)<TCx;(TJx+TEx);TCx+((TJx+TEx)-TCx)^2)

Pseudo-código: Fu(x)=TE(x)+TJ(x) SE TJ(x)<TC(x)

Fu(x)= TC(x)+(TE(x)+TJ(x)-TC(x))^2 **se** TJ(x)>TC(x)

Em que **TC(x)=** Tempo máximo admissível para procura de solução e elaboração do plano, e **TJ(x)**= Tempo de elaboração do Plano. A tabela 4.2 ilustra os resultados obtidos:

Encontra solução TE(x) e elabora plano TJ(x) (Fu(x)= TE(x)+TJ(x)	**Algoritmos**				
Cenários	**Genético**	**Ganancioso**	**Linear**	**Aleatório**	**Regras (Heurísticas)**
Nº combinações executadas em simultâneo (Tarefa x Operações)	**Valor da função utilidade – Fu(x)** **Fu(x)=**TE(x)+TJ(x) **se** TJ(x) < TC(x) **e Fu(x)=**TC>(x)+(TE(x)+TJ(x))-TC(x))^2 **se** TJ(x)>TC(x)				
20	2,84	16,90	3,56	Inf	22,43
40	4,65	23,02	9,56	Inf	22,49
80	11,86	161,88	38,96	Inf	Inf
160	30,53	944.851,71	65,79	Inf	Inf
Média	12,47	236.263,38	29,47	Inf	Inf
Desvio Padrão	12,66	472.392,22	28,73	Inf	Inf

Tabela 4.2 - Demostração de Resultados para cada um dos algoritmos (cenário 1)

2. Encontra solução TE(y), elabora plano e executa plano TJ(y), e terá como resultado fu(y)= TE(y)+TJ(y) em unidades de segundos.

Cálculo: Fu(y)= SE((TJy+TEy)<TCy;(TJy+TEy);TCy+((TJy+TEy)-TCy)^2)

Pseudo-código: Fu(y)=TE(y)+TJ(y) **se** TJ(y)<TC(y)

Fu(y)= TC(y)+(TE(y)+TJ(y)-TC(y))^2 **se** TJ(y)>TC(y)

Em que:

TC(y)= Janela temporal disponível para a execução do plano

TJ(y)= Tempo de execução do plano gerado

Convém desde já clarificar que este cenário diverge do anterior devido ao universo de TJ, no ponto anterior é considerado apenas o tempo de elaboração do plano, enquanto neste cenário é considerado o tempo de elaboração do plano e o tempo de execução do mesmo, tal como apresentado na tabela 4.3.

Encontra solução TE(y), elabora plano e executa plano TJ(y) Fu(x)= TEy+TJy)	**Algoritmos**				
Cenários	**Genético**	**Ganancioso**	**Linear**	**Aleatório**	**Regras (Heurísticas)**
Nº combinações executadas em simultâneo (Tarefa x Operações)	**Valor da função utilidade Fu(y)** **Fu(y)**=TE(y)+TJ(y) **se** TJ(y) < TC(y) **e** **Fu(y)**=TC(y)+(TE(y)+TJ(y))-TC(y))^2 **se** TJ(y)>TC(y)				
20	11.919,82	26.016,17	24.953,68	Inf	27.798,67
40	21.198,90	26.541,19	24.897,75	Inf	26.540,67
80	26.606,03	28.466,07	27.322,88	Inf	Inf
160	27.131,71	1.123.372,31	27.618,94	Inf	Inf
Média	21.714,12	301.098,94	26.198,31	Inf	Inf
Desvio Padrão	7.058,68	548.183,26	1.474,61	Inf	Inf

Tabela 4.3 - Demostração de Resultados para cada um dos algoritmos (cenário 2)

3. Plano Repetitivo, executa plano, terá como resultado fu(z)= TJ(z) em unidades de segundos.

   C**álculo:**
   Fu(z)= SE((TJz)<TCz;TJz;TCz+(TJz-TCz)^2)

   Pseudo-código:

   Fu(z)=TJ(z) se TJ(z)<TC(z)

   Fu(z)= TC(z)+(TJ(z)-TC(z))^2 SE TJ(z) >TC(z)

Em que:

TC(z)= Janela temporal disponível

TJ(z)= Tempo de execução do plano gerado

Neste cenário não é feita a procura da melhor solução, mas sim a reutilização de um plano já disponível através da adaptação de um plano existente, desde que o TC(z) a utilizar seja =< ao TC(z). A tabela 4.4 apresenta os resultados de Fu(z).

Plano Repetitivo, executa plano Fu(z)= TJ(z)	**Algoritmos**				
Cenários	**Genético**	**Ganancioso**	**Linear**	**Aleatório**	**Regras (Heurísticas)**
Nº combinações executadas em simultâneo (Tarefa x Operações)	**Valor da função utilidade Fu(y)** **Fu(y)**=TJ(z) **se** TJ(z) < TC(z) **e** **Fu(z)**=TCz+(TJz - TCz)^2 **se** TJz>TCz				
20	11.918,00	26.000,00	24.951,00	Inf	27.777,00
40	21.195,00	26.519,00	24.889,00	Inf	26.519,00
80	26.595,00	28.305,00	27.285,00	Inf	Inf
160	27.102,00	28.575,00	27.554,00	Inf	Inf
Média	21.702,50	27.349,75	26.169,75	Inf	Inf
Desvio Padrão	7.049,47	1.281,37	1.447,48	Inf	Inf

Tabela 4.4 - Demostração de Resultados para cada um dos algoritmos (cenário 3)

Da analise da tabela 4.2, 4.3 e 4.4, concretamente ao nível de:

- O desvio padrão – O desvio padrão, é uma medida de dispersão que avalia na amostra a dispersão dos valores em relação à média. Os resultados obtidos indicam desvios padrão normalmente superiores à média, significando que o algoritmo não é consistente, ou seja, o resultado depende muito do número de tarefas e operações, ou seja, do número de combinações destas.

- Função utilidade - Os resultados da função utilidade indicam qual a melhor solução, sendo esta a que apresentar valores mais baixos, significando que o tempo de utilização foi o mais optimizado.
  Convém relembrar de que o objectivo é premiar os algoritmos que gastaram menos tempo, logo, penalizando os que gastaram mais tempo.
  Considerando tal, foram obtidos 3 tipos de resultados na função utilidade:
  - Resultados que não sofreram influência da penalização nos desempenhos (valores mais baixos), compreendidos entre 2,84 e 65,75
  - Resultados onde foi aplicada a penalização nos desempenhos (valores superiores), acima de 161,88

- Inf – Alguns resultados foram etiquetados com a sigla Inf, abreviatura de infinito. Neste universo foram incluídos os resultados do simulador utilizando algoritmos em que não encontraram soluções no decorrer das 10 execuções sequenciais.

Em conclusão e ainda mediante dos resultados da tabela 4.5, é possível ordenar o desempenho dos algoritmos da seguinte forma:

Algoritmo	**T**arefas x **O**perações			
	20	**40**	**80**	**160**
Genético	2,84	4,65	11,86	30,53
Prog. Linear	3,56	9,56	38,96	65,79
Ganancioso	16,5	23,02	161,88	944.851,44
Regras	22,43	22,49	Inf	Inf
Aleatório	Inf	Inf	Inf	Inf

Tabela 4.5 - Demostração de Resultados ordenados, sendo os resultados em segundos

4.2.2. Representação gráfica dos resultados

A representação gráfica é apresentada através de gráficos de linhas, estando representados nos seus eixos o valor da função utilidade e o universo de Tarefas x Operações (20 a 160).

Para um melhor visibilidade dos valores na representação gráfica, apresenta-se a tabela 4.6, sendo esta o detalhe da tabela 4.2 e 4.5, referente aos valores obtidos através da função utilidade Fu(x). Sendo a Fu(x) pertencente ao cenário que mais se enquadra no âmbito da bancada de ensaios, e que influencia diretamente os restantes cenários (2 e 3), será o cenário 1 o que será analisado, no entanto as tabelas de detalhes das tabelas 4.3 e 4.4 serão apresentadas no capitulo de anexos. A informação seguinte na tabela 4.6, é como já referido o detalhe dos resultados obtidos para o cenário de Fu(x).

			Tempo (segundos)			
Algoritmos	Tarefas (TxOp)	Valor Médio de Execução (TE)	Critico (TC)	Execução TJ	TJ+TE	Resultado TR (TJ+TE)
Ganancioso	20	16,17	300	16,90	16,90	16,90
Prog. Linear	20	2,68	300	3,56	3,56	3,56
Aleatório	20	Inf	300	Inf	Inf	Inf
Regras (heurísticas)	20	21,67	300	22,43	22,43	22,43
Algoritmo Genético	20	1,82	300	2,84	2,84	**2,84**
Ganancioso	40	22,19	300	23,02	23,02	23,02
Prog. Linear	40	8,75	300	9,56	9,56	9,56
Aleatório	40	Inf	300	Inf	Inf	Inf
Regras (heurísticas)	40	21,67	300	22,49	22,49	22,49
Algoritmo Genético	40	3,90	300	4,65	4,65	**4,65**
Ganancioso	80	161,07	300	161,878	161,878	161,88
Prog. Linear	80	37,88	300	38,96	38,96	38,96
Aleatório	80	Inf	Inf	Inf	Inf	Inf
Regras (heurísticas)	80	Inf	300	Inf	Inf	Inf
Algoritmo Genético	80	11,03	300	11,857	11,857	**11,86**
Ganancioso	160	1.271,22	300	1.271,88	1.271,88	944.851,71
Prog. Linear	160	64,94	300	65,79	65,79	65,79
Aleatório	160	Inf	300	Inf	Inf	Inf
Regras (heurísticas)	160	Inf	300	Inf	Inf	Inf
Algoritmo Genético	160	29,71	300	30,53	30,53	**30,53**

Tabela 4.6 - Demostração de Resultados para cada um dos algoritmos de Fu(x)

Para o universo de algoritmos em análise, apresenta-se os respetivos gráficos de linhas, nos quais foi adicionado a linha de tendência para verificar a precisão, onde o seu valor (R2) varia entre 0 e 1, o que revela a proximidade da correspondência dos valores previstos para a linha de tendência relativamente aos dados utilizados, quanto mais próximo de 1 mais precisa e fidedigna (R^2 tem o valor 1 ou o mais próximo deste).

4.2.2.1. Algoritmo Genético

O comportamento do Algoritmo Genético (AG) foi praticamente constante e linear nos melhores desempenhos nos vários cenários. De salientar a precisão através do valor de R^2 que tende para 1.

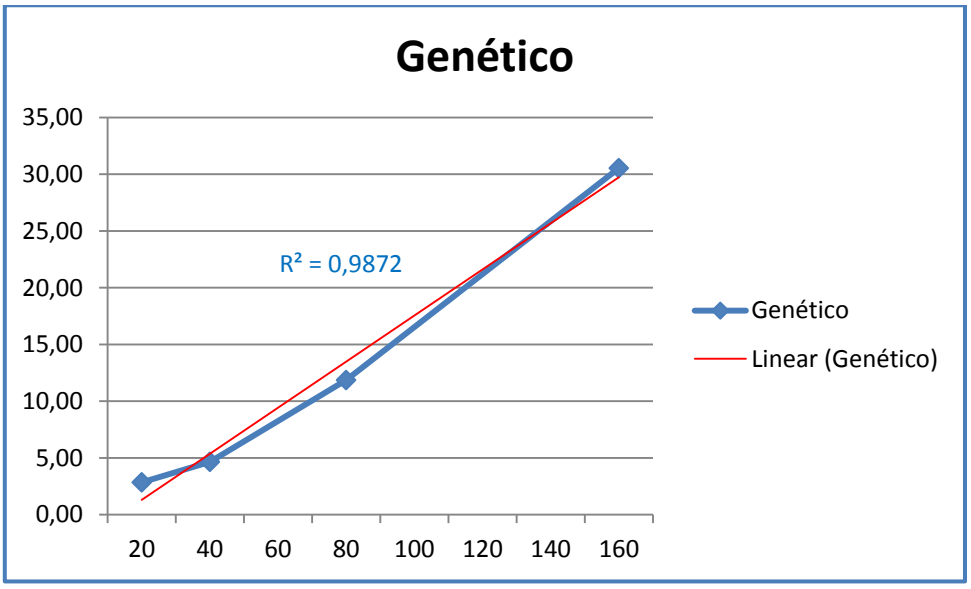

Figura 4.2 - Resultados referentes ao Algoritmo Genético

4.2.2.2. Algoritmo Ganancioso (Ganancioso)

No algoritmo ganancioso denota-se que o seu comportamento para problemas mais complexos apresenta um desempenho não aconselhável para os objetivos pretendidos, apesar de conseguir soluções para todas as simulações executadas na bancada de ensaios.

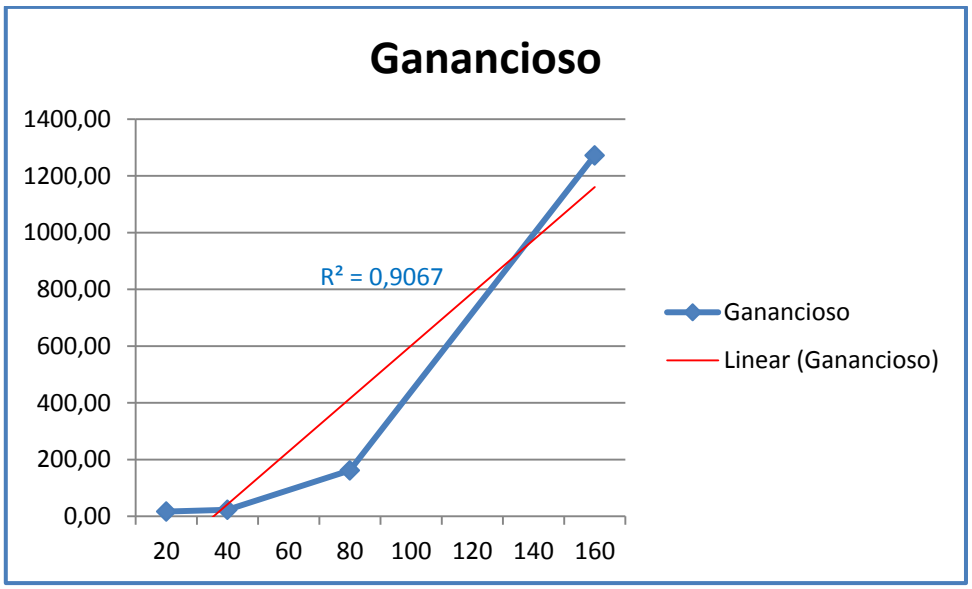

Figura 4.3 - Resultados para o Algoritmo Ganancioso (Ganancioso)

4.2.2.3. Algoritmo Linear

Apesar de uma precisão (R^2) de 0,97, o algoritmo linear apresenta-se como desajustado para a resolução deste problema em tempo útil. Quanto maior for a amostra da população, maior terá que ser o nº de iterações do algoritmo. Como o algoritmo está limitado no nº de iterações (para resolução em tempo útil do problema), estas iterações revelam-se insuficientes para encontrar a solução óptima. Aumentando o número de iterações disponíveis para a resolução, este algoritmo revela-se incapaz de produzir soluções óptimas em tempo útil.

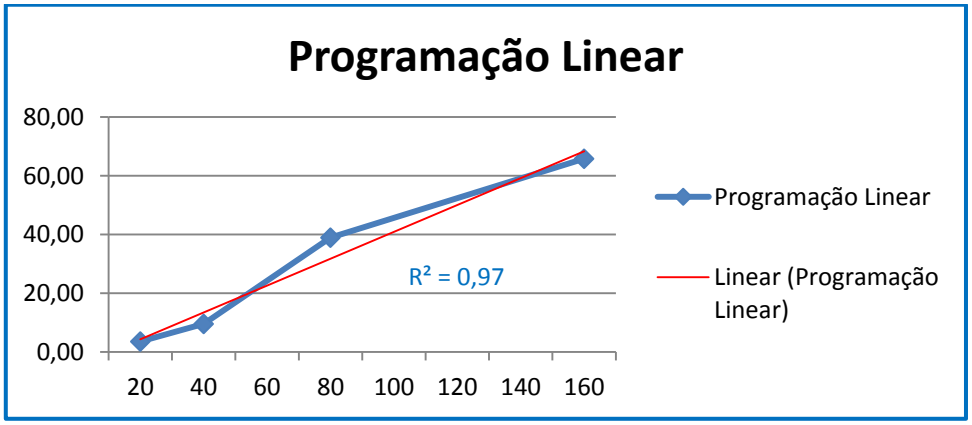

Figura 4.4 - Resultados para o Algoritmo Linear

4.2.2.4. Algoritmo Aleatório

O algoritmo aleatório pela sua génese é inconstante e é ainda difícil de caracterizar a sua aplicabilidade neste género de problemas.

Tal como se previa nas várias simulações os resultados obtidos não foram conclusivos quanto a possíveis soluções.

4.2.2.5. Algoritmo de Regras (Heurísticas)

O algoritmo baseado em heurísticas apresenta ao longo das diferentes populações, resultados inconstantes. Pelo facto de não terem sido apurados resultados para todos os cenários não é apresentada a representação gráfica, (apenas foram conseguidos resultados para os cenários de 20 e 40, não tendo havido soluções para o cenário de 80e 160).

Conclusão

É complexo obter uma função de avaliação (função utilidade) que possa ser aplicada de forma repetitiva. Encontrar a melhor solução para problemas complexos em situações reais requer funções de avaliação extremamente pesadas que por vezes necessitam de diversas horas para serem computadas e por vezes nem conseguem ser reproduzidas.

A figura 4.6 através de um gráfico de linhas único mostra a representação para todos os algoritmos em análise.

4.2.3. Comparação de resultados

Numa primeira consulta ao gráfico de linhas da figura 4.6 não faria sentido que o desempenho de algoritmos como GA fosse mais pobre que, por exemplo, do *de Linear Programming*.

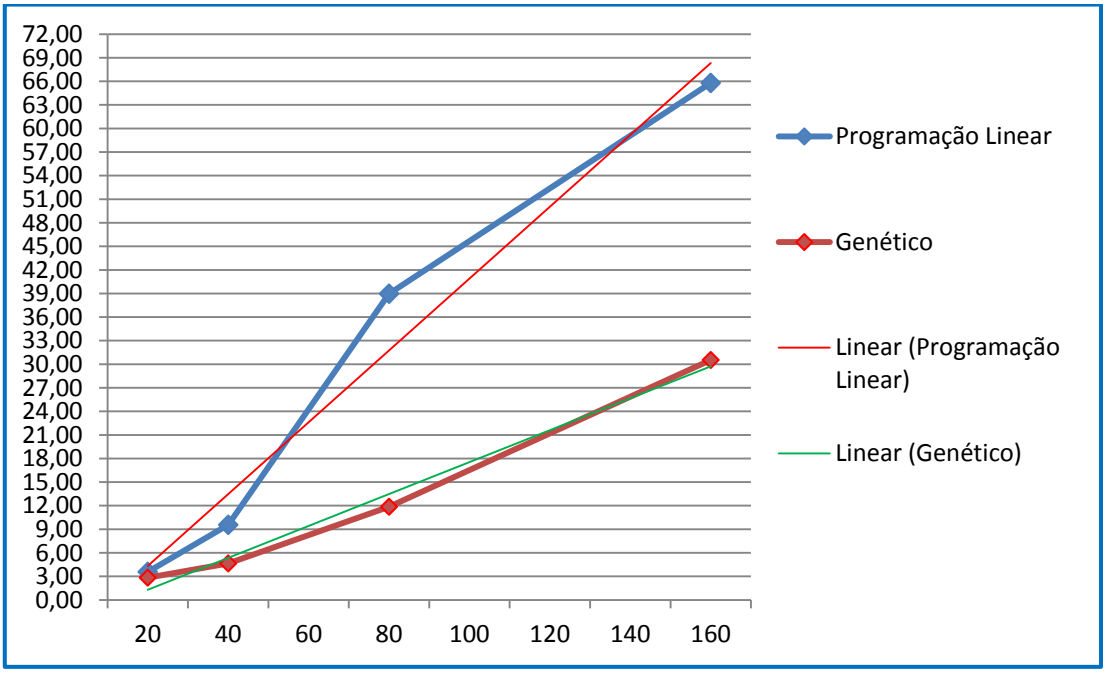

Figura 4.5 - Resultados para a comparação dos Algoritmos em análise

No entanto, poderíamos estar na presença de um tópico que tem alimentado várias controvérsias e discussões no seio do mundo da informática. Trata-se precisamente da aplicabilidade e performance deste mesmo algoritmo. Estudos e opiniões têm surgido e destacamos algumas críticas / problemas que têm vindo a ser apontadas a este algoritmo, como o estudo do Massachusetts Institute of Technology "A Comparison of Article Swarm Optimization and the Genetic Algorithm" (http://www.mit.edu/~deweck/PDF_archive/3%20Refereed%20Conference/3_50_AIAA-2005-1897.pdf.). No entanto tal conclusão não é aplicável aos resultados de desempenho do algoritmo Genético na obtenção de soluções.

Tendo como base as experiências feitas neste capítulo, pode-se inferir as seguintes conclusões:

- Algoritmos aleatórios não são adequados para este tipo de problema
- Um algoritmo ganancioso tem uma eficácia um pouco melhor à dos algoritmos baseados em regras
- Algoritmo Genético potencia-se como o melhor algoritmo para os cenários utilizados.

O desvio padrão também permite perceber se o algoritmo em termos de função utilidade tem muita ou pouca variação. Percebe-se que o algoritmo Genético é o que varia menos.

As expectativas no algoritmo baseado em regras, heurísticas, eram altas, e esperavam-se resultados superiores para populações maiores.

4.3. Representação do Plano de Salvaguarda e Reposição

Para a representação do problema de Salvaguarda e Reposição foi escolhido o diagrama de Gannt desenvolvido em 1917 por Henry Gannt.

O objetivo é a representação dos intervalos de tempo de cada combinação de tarefa e operação num determinado momento. Os diagramas de Gannt serão municiados por ficheiros do tipo XML gerados aquando da execução das simulações na bancada de ensaios.

A tabela 4.7 apresenta um exemplo da correspondência entre os ficheiros do tipo XML e a respetiva situação na execução (*run*) para o algoritmo genético.

Execução 5 (run)	
TxO	**Ficheiro**
10x2	2013_07_21_03_30_54_190-ga.xml
10x4	2013_07_21_03_27_47_336-ga.xml
10x8	2013_07_21_03_23_40_35-ga.xml
10x16	2013_07_21_03_09_59_193-ga.xml

Tabela 4.7 – Correspondência dos ensaios na Bancada de Ensaios

Apesar de muito interessantes os resultados obtidos em T10x16, com E=160, não é legível a sua representação para o formato deste documento, assim e para facilitar a boa visualização do resultado apresentado na figura 4.9 na forma de Gannt foi selecionado um exemplo do tipo T10xO2, com E=20, em que T representa as tarefas, O, representa as operações e finalmente E o resultado da execução.

Através do ficheiro XML, o visualizador procede ao carregamento dos dados tal como apresentado na tabela 4.8 seguinte referente à combinação Tarefa x Operação, com o tempo de início e fim gerando a duração e respetivo escalonamento.

Informação				
Resultado	**Fonte de Dados**	**Tarefas**	**Operações**	**Total**
R1	2013_07_21_03_30_41_196-ga.xml	**10**	**2**	**20**
R2	2013_07_21_03_09_59_193-ga.xml	**10**	**16**	**160**

Tabela 4.8 – Tabela de referência à fonte de dados utilizada

Para uma maior facilidade de visualização utiliza-se um exemplo do tipo 10x2, por cada tarefa existirá 2 operações, uma das operações de verificação e a outra de escrita (validação de espaço de armazenamento ou disponibilidade e salvaguarda ou reposição).

A tabela 4.9 apresenta os valores para o plano através do Gantt dos 2 algoritmos de melhor desempenho.

Genético			**Taredas x Operações**	**Programação Linear**		
Tinicial	**Tfinal**	**Duração**		**Duração**	**Tinicial**	**Tfinal**
7292	9114	1822	mssi_10x2_t0-o0	1822	21306	23128
1823	3645	1822	mssi_10x2_t0-o1	1822	23129	24951
1823	3188	1365	mssi_10x2_t1-o0	1365	19078	20443
10938	12303	1365	mssi_10x2_t1-o1	1365	20444	21809
0	3227	3227	mssi_10x2_t2-o0	3227	17274	20501
5469	8696	3227	mssi_10x2_t2-o1	3227	21810	25037
0	4115	4115	mssi_10x2_t3-o0	4115	0	4115
7292	11407	4115	mssi_10x2_t3-o1	4115	14962	19077
5469	8324	2855	mssi_10x2_t4-o0	2855	11302	14157
9115	11970	2855	mssi_10x2_t4.o1	2855	14418	17273
3646	7262	3616	mssi_10x2_t5.o0	3616	0	3616
5469	9085	3616	mssi_10x2_t5-o1	3616	4116	7732
0	803	803	mssi_10x2_t6-o0	803	14158	14961
0	803	803	mssi_10x2_t6-o2	803	20502	21305
3646	6497	2851	mssi_10x2_t7-o0	2851	7733	10584
0	2851	2851	mssi_10x2_t7-o1	2851	10585	13436
10938	11918	980	mssi_10x2_t8-o0	980	10321	11301
10938	11918	980	mssi_10x2_t8-o1	980	13437	14417
1823	5174	3351	mssi_10x2_t9-o0	3351	3617	6968
1823	5174	3351	mssi_10x2_t9-o1	3351	6969	10320

Tabela 4.9 – Tabela de referência à fonte de dados utilizada

4.3.1. Visualizador de resultados

Os dados recolhidos para os ficheiros do tipo XML gerados alimentaram um visualizador adaptado para o efeito e através de tecnologia de SSO (software Open Source) e em código Java conforme visualizado na figura 4.10 e tem como finalidade apresentar os resultados, tal como ilustrado nas figuras 4.9 e 4.10.

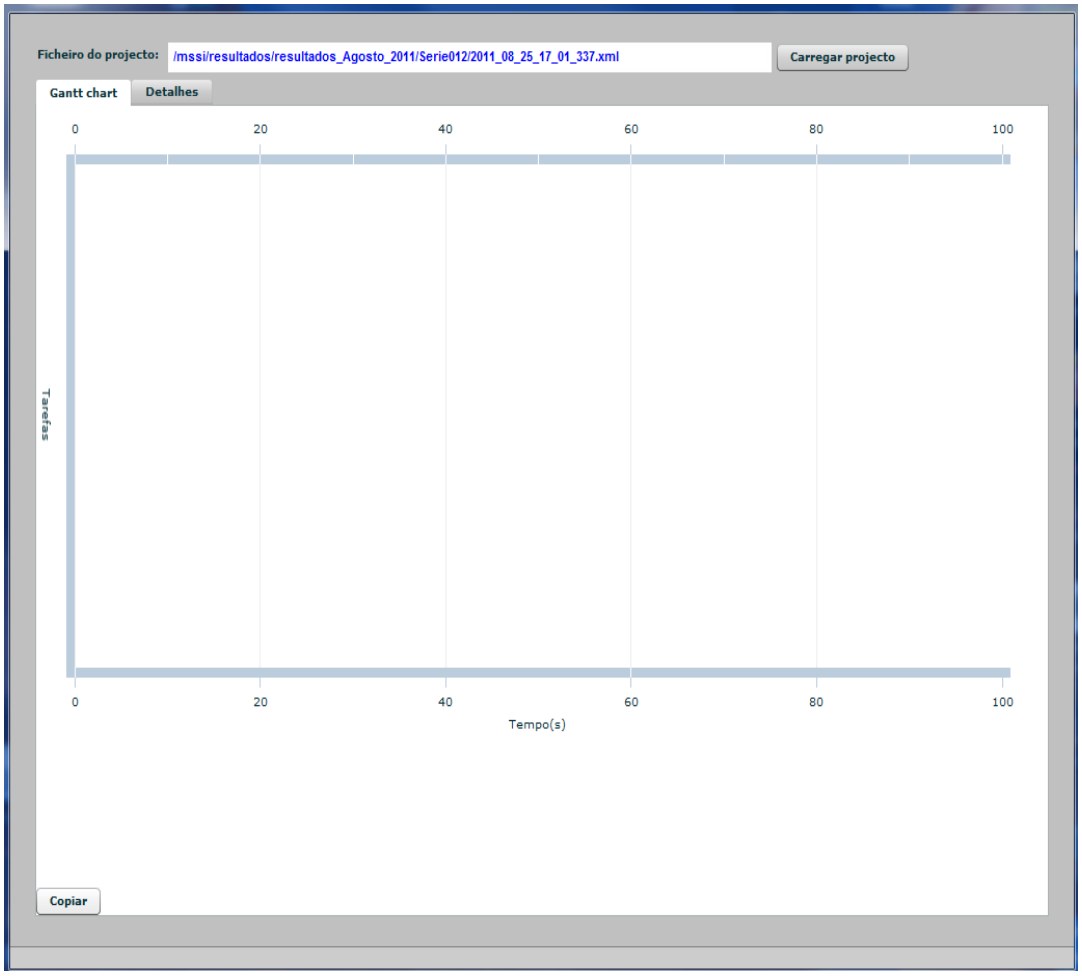

Figura 4.6 - Resultados para a comparação dos Algoritmos em análise

Após o carregamento do ficheiro xml, devolve o resultado apresentado na figura 4.8 e 4.9, mais concretamente um escalonamento de tarefas de salvaguarda ou reposição, sendo a primeira de menor complexidade de forma a permitir uma melhor visualização.

4.3.1.1. Representação de Gannt para o algoritmo Genético do resultado do tipo E=20

Figura 4.7 – Resultado para E=20 (TxO) com fonte de dados em 2013_07_21_03_30_41_196-ga.xml com TJ=12.303 (s)

4.3.1.2. Representação de Gannt para o algoritmo Programação Linear do resultado do tipo E=20

Figura 4.8 – Resultado para E=20 (TxO) com fonte de dados em 2013_07_24_09_39_49_369-linear_programming.xml, com TJ=25037 (s).

Capitulo 5. Conclusões

5.1. Contribuições

Com os resultados apresentados ficam demonstradas as eficiências significativas nos esforços de salvaguarda e reposição de dados nomeadamente, na gestão das janelas temporais, na utilização de soluções tecnológicas (optimização da utilização de recursos, tarefas e operações (servidores e robots, salvaguarda e reposição de informação, por exemplo), a própria validação dos actuais planos e, no limite, demonstrando ainda a necessidade destas estarem dotadas de recursos humanos especializados apenas no inicio para que a recolha de informação seja o mais precisa possível. Efectivamente, comprova-se que a qualidade de um bom plano depende da qualidade da informação recolhida e produzida para o efeito.

5.2. Síntese conclusiva

Considera-se que o estabelecimento de planos optimizados, com paralelização e ponderação das acções e recursos, podem contribuir activamente para o aumento da segurança da informação e rentabilidade dos equipamentos.

Com alguma facilidade foi verificado que para se obter um bom plano de salvaguarda ou reposição não foi necessário recorrer à utilização de algoritmos de elevado nível de sofisticação ou especificidade, mas sim apenas recorrer ao que já existe, sendo necessário alguma capacidade de adaptação.

Conclui-se ainda que ser possível uma evolução mais exigente e por isso deve ser investido tempo na investigação de melhores fórmulas de relacionar as diferentes métricas, a atribuição de ponderações diferentes aos recursos e operações e alguma automatização com o objetivo de reduzir potenciais erros humanos.

Perspectiva de Trabalho futuro

As experiências descritas neste capítulo procuraram modelar um contexto de gestão automatizada mediante um âmbito perfeitamente identificado. O avançar da bancada de ensaio e os resultados obtidos permitiram ter uma visão mais especifica do âmbito e assim desde logo ter noção mais exacta de possíveis evoluções e inovações. Uma evolução ao actual âmbito será a utilização de Redes Neuronais[6], permitindo assim adquirir um conhecimento sem a intervenção directa humana, permitindo:

- Recolha de informação comportamental através dos agentes
- Criação de padrões de comportamento mediante as situações conhecidas

Desta forma os valores das variáveis de entrada baseiam-se em dados reais recolhidos em observações. Desta forma a *framework* faz uma auto-aprendizagem (aprende com ela própria) e com os dados fornecidos pelo utilizador.

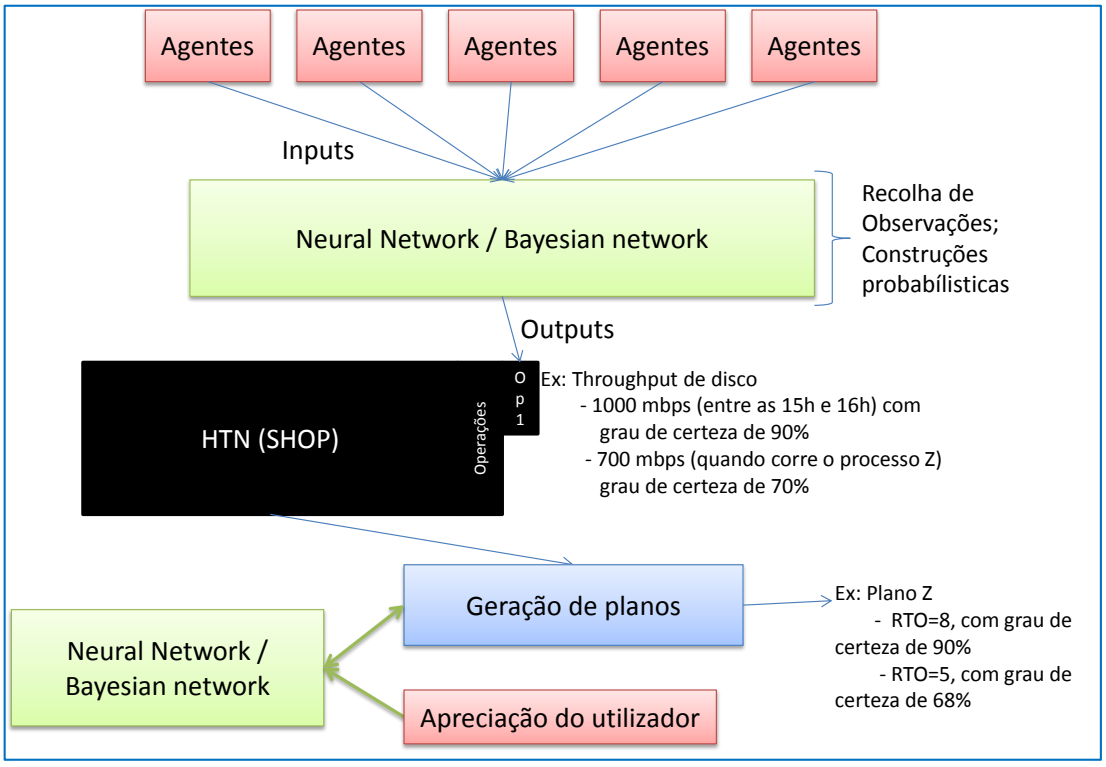

Figura 5.1 – Diagrama de uma possível solução usado Redes Neuronais

[6] Modelo inspirado na estrutura neuronal de organismos inteligentes e que adquirem conhecimento através da experiência

Como o decorrer do trabalho, e confirmado com os resultados obtidos, sugere-se que numa perspectiva de trabalhos futuros, deverá ser estudada uma representação mais complexa preparada para dar resposta a realidades de uma dimensão superior à situação que serviu para a realização deste trabalho de estudo.

Ainda como próximos passos, será importante fazer saltos exponenciais no tamanho da população /universo de experimentação de forma a termos mais resultados concretos que permitam aferir o verdadeiro comportamento dos principais algoritmos em populações grandes, populações estas que podem acontecer no mundo do IT em grandes instituições e empresas – e.g., um plano de salvaguarda e reposição de uma infraestrutura na Cloud atingirá facilmente mais de 1000 tarefas com mais de 100 operações que competem intensivamente por recursos de rede, CPU, memória e disco, razão pela qual não se abandonou a primeira hipótese de função utilidade, pois para estes cenários mais exigentes pode ser uma boa solução a conjugação dessa função utilidade com a função utilidade utilizada no simulador desta bancada de ensaios.

Anexos

Neste capítulo pode ser consultada informação deste trabalho referente à implementação na bancada de ensaios bem como os restantes resultados obtidos e que não foram referenciados no capítulo de resultados.

6.1. Código Utilizado (pseudo-código)

Sendo uma linguagem de fácil leitura, foi escolhido o pseudo-código para documentar a execução dos algoritmos utilizados na bancada de ensaios.

Será apresentado o código utilizado para cada uma dos algoritmos, adicionalmente foram incluídos comentários.

6.1.1. Algoritmo Aleatório

```
FUNCTION RESCHEDULE_RANDOM ( )
# Percorrer o vector de tarefas não-escalonadas
FOR EACH task IN unallocated_tasks:
tries = 0
DO
# nas primeiras 20 tentativas, tenta-se encontrar um escalonamento
# entre o instante corrente e o final da tarefa, multiplicado por 2
IF tries < TRIES_INTERVALS[0]:
random_instant = random(current_instant, current_instant + task.duration * 2)
# entre a tentativa 20 e a tentativa 40,
# tenta-se encontrar um escalonamento
# entre o instante corrente e o prazo da tarefa
ELSEIF tries < TRIES_INTERVALS[1]:
random_instant=random(current_instant, task.deadline)
```

(Continua)

```
# nas tentativas acima de 40, tenta-se encontrar um escalonamento
# em qualquer instante válido
ELSE:
random_instant = random(current_instant, max_instant -
task.duration - 1)
END IF
# se a adição da tarefa é válida para o instante, em todos os recursos
# a tarefa é associada, e passa à próxima tarefa (continue)
IF isTaskAdditionValid(task, task.resources, random_instant):
addTask(task, task.resources, random_instant)
CONTINUE
END IF
# número de tentativas é incrementado
tries = tries + 1
# se não se consegue encontrar um escalonamento válido,
# após 50 tentativas
# a tarefa é ignorada
IF tries > MAXIMUM_TRIES:
dropped_tasks = dropped_tasks + task
END IF
WHILE TRUE
END
END FOR
END FUNCTION
```

(Fim)

Figura 0.1 – Pseudo-código do Algoritmo Aleatório

6.1.2. Algoritmo Ganancioso

```
# Para cada tarefa tenta encontrar o instante mais cedo
# onde é possível escalonar a tarefa em todos os recursos
FUNCTION SCHEDULE_GREEDY ( )
# percorrer o vector de tarefas não-escalonadas
FOR EACH task IN unalocated_tasks:
# procurar o instante mais cedo onde é possivel escalonar a tarefa,
# para os recursos testados
best_match = findEarliestInstantForTask(task, task.resources)
# se tiver sido encontrado um instante possível,
# é escalonado com o melhor tempo
IF best_match IS NOT NULL:
addTask(task, task.resources, best_match)
# caso contrário, a tarefa é ignorada
ELSE:
dropped_tasks = dropped_tasks + task
END IF
END FOR
END FUNCTION
```

(Fim)

Figura 0.2 – Pseudo-código Ganancioso

6.1.3. Algoritmo de Regras

```
FUNCTION SCHEDULE_RULES1 ( )
# percorrer o vector de tarefas não-escalonadas
FOR EACH task IN unalocated_tasks :
# a melhor opção é inicializada a zero.
# funciona num referencial invertido em relação ao algoritmo GREEDY
best_match = 0
# procura os instantes válidos para alocação da tarefa aos seus recursos
# por ordem cronológica
instants = findInstantsForTask(task, task.resources)
FOR EACH tmp_instant IN instants:
# encontra o "fitness" da tarefa no instante candidato, nos respectivos recursos
fitness_for_task = getFitnessForTaskAt(task, task.resources,
tmp_instant)
# neste algoritmo, o best_match é uma combinação de heurísticas
# os coeficientes de equilibro foram determinados a partir de testes manuais
tmp_match = ((640800 – tmp_instant) * 1000) + ((1000 –
resource.getTaskCount(tmp_instant)) * 10) + (1000 – fitness_for_task)
# se o match encontrado é o melhor, passa a ser o escolhido
IF tmp_match > best_match :
best_match=tmp_match
END IF
END FOR
```

(Continua)

```
# actualiza a disponibilidade média
average_resource_availability = (resource_availability + (average_
resource_availability * number_resources)) / (number_resources + 1)
END IF
# actualiza o número de recursos
number_resources = number_resources + 1
END FOR
# normaliza a disponibilidade para obter o fitness
fitness = average_resource_availability * 1000
RETURN fitness
END FUNCTION
```

(Fim)

Figura 0.3 – Pseudo-código Regras

6.1.4. Algoritmo Genético

```
FUNCTION SCHEDULE_AG( )
no_preselect_schedules = 10
no_iterations = 100
current_iter = 0
dg_arg = 34 # valor de referência para decisão de algoritmo
list_os_scheduller=scheduller.getAleatorySchedulles(no_preselect_scheles)
#  obter 10
Schedulles aleatórios
p1 = list_os_scheduller[random(0,no_preselect_schedules)] # obter um primeiro
schedule aleatório
list_os_scheduller.remove(p1) # remover p1 da lista de schedules
p2 = list_os_scheduller[random(0,9)] # obter um segundo schedulle aleatório
list_os_scheduller.remove(p2) # remover p2 da lista de schedulles
WHILE (current_iter < no_iterations) # enquanto o nº de iterações for permitido
DG = CalculateDG(p1,p2) # calcular a distância entre os dois schedulles
IF DG< dg_arg # se a distância for inferior que o parâmetro recebido, então aplica-se a
mutação a p1
child = Mutation(p1)
ELSE # Senão aplica-se a mutação a p1 e o crossover a p2
child = Crossover(p1,p2)
END IF
child' = Scheduller.getFittestElement(list_os_scheduller,child) # procurar na
vizinhança o elemento com melhor resultado da função de fitness
worst_schedulle = Scheduller.getLowestMakspan(list_os_scheduller) # encontrar qual o
elemento da lista com Makespan mais alto
```

(continua)

```
IF makespan(child')<worst_schedulle # Se o makespan de child for menor que o pior makespan dos elementos da lista
list_os_scheduller.replace(Scheduller.getLowestMakspanMember(list_os_scheduller), child')
# substituir o pior por child'
ELSE IF Scheduller.containsMembersWithMakespan(makespan(child')) # se existirem membros com o mesmo makespan
list_os_scheduller.replaceAll(Scheduller.getMembersWithMakespan(makespan(child')),child')
# substituir todos por child'
END IF
current_ current_iter=current_iter+1;
END WHILE
END FUNCTION

                    (Fim)
```

Figura 0.4 – Pseudo-código Algoritmo Genético

6.1.5. Programação Linear

```
FUNCTION LINEAR_PROGRAMMING()
        resources = problem.resources;
        tasks = problem.tasks
        #builds and initializes the threads needed to start processing
        FOR índex ; índex < NUM_THREADS ; índex++
                new ThreadWorker(stackWithCombination, queueWithResults)
        END FOR
        new ThreadResults(queueWithResults)
        #builds the first node:
        #       all resources are free
        #       all tasks are free
        #       current instant 0: the problema is starting
        node = new LinearProgrammingNode(tasks,resources)
        # adds the node to the stack
        stack.push(node)
        waitForThreadsToEnd()
END FUNCTION
THREAD_WORKER(stackWithCombinations, queueWithResults)
FUNCTION RUN()
        #waits for a signal for the thread to stop working or
# works until there are no more combinations possible
WHILE (working)
                node = stackWithCombinations.pop();
                # Check if node is valid: current node allocated time is not superior to
problem
                # time spam. If i tis not valid, discard the node
                IF NOT node.valid
                        continue;
                END IF
```

(continua)

```
#if the node is finished, add it to the results queue, add wait for the next
#node to process
                    IF node.finished
                            queueWithResults.push(node)
                            continue;
                    END IF
                    # list of nodes to add to the stack
                    nodes[]=[];
                    # gets  the resources that are free in the current node
                    freeResources = node.freeResources;
                    # get the tasks that are not yet processed
                    tasksNotEnded = node.tasksNotEnded
                    # gets the allocated tasks
                    tasksAllocated = node.allocatedTasks
                    FOR EACH resource in freeResource
                    #calculates the combinates of paths that can be reached
                            # from the current node, and adds them to the nodes list
                    calculateCurrentNodeCombinations(resource, freeResources,
tasksNotEnded, nodes, allocatedTasks)
END FOR EACH
        # adds nodes to the stack
        stack.push(nodes)
        END WHILE
END RUN
FUNCTION calculateCurrentNodeCombinations(resource, freeResources,
tasksNotEnded, nodes, allocatedTasks)
        FOR EACH task in tasksNotEnded
                # checks if the current
                # validates if task can be allocated
                IF validateAllocation(task,freeResources, resource)
```

(continua)

```
#removes task from list of tasks to do and its resources
                        removeTaskResource(task,freeResources)
                        # adds task to allocated tasks
                        allocatedTask.add(task)
                        FOR EACH resource in freeResource
                        #calculates the combinates of paths that can be reached
                                # from the current node, and adds them to the nodes list
                                calculateCurrentNodeCombinations(resource,
                                freeResources, tasksNotEnded, nodes, allocatedTasks);
                        END FOR EACH
                        node = new LinearProgrammingNode(tasks,resources)
                        # adds the node to the list of nodes to be processed
                        nodes.add(node)
        END IF
        END FOR EACH
END FUNCTION
END THREAD
THREAD RESULT
RUN(resultQueue)
        WHILE (working)
                # takes a valid node from thr queue
                node = queueWithResult.pop()
                IF (node <minTime)
                        # sets the node as the result
                        setResult(node)
                END IF
        END WHILE
END RUN
END THREAD
                                        (Fim)
```

Figura 0.5 – Pseudo-código Programação Linear

6.1.6. Makespan

```
FUNCTION MAKESPAN(problem)
        allocation _score =  utility.allocation_weight * problem.getAverageAlloca-tion()
        end_instant_score = utility.end_time _weight * (problem.getEndInstant() /
maximum_instant)
        dropped_tasks_score  =  utility.dropped_tasks_weight  *  (problem.get-
DroppedTasks()  / tasks.size)
        # calcula o fitness ponderado
        fitness = allocation _score  - end_instant_score  - dropped_tasks_score
        # normaliza o valor de fitness no intervalo configurado, por omissão (0, 100)
        normalized_fitness  =  normalize(fitness,  minimum_fitness,  maximum_fitness)
        RETURN  normalized_fitness
END FUNCTION
```

(Fim)

Figura 0.6 – Pseudo-código Makespan

6.2. Resultados utilizados na bancada de ensaios para os cenários exemplo no ponto 4.2

Apresenta-se os resultados obtidos na bancada de ensaios referente à Função utilidade dos 2 cenários não detalhados no ponto 4.2 (Fu(y) e Fu(z), tempo de execução do plano e plano repetitivo respectivamente.

6.2.1. Função Utilidade Fu(y)

			Tempo (segundos)		
Algoritmos	**Tarefas (TxOp)**	**Valor Médio de Execução (TEy)**	**TC(y)**	**TJ(y)**	**Função Utilidade Fu(y)**
Ganancioso	20	16,17	28.800,00	26.000,00	26.016,17
Linear	20	2,68	28.800,00	24.951,00	24.953,68
Aleatório	20	Inf	Inf	Inf	Inf
Regras (Heurísticas)	20	21,67	28.800,00	27.777,00	27.798,67
Genético	20	1,82	28.800,00	11.918,00	**11.919,82**
Ganancioso	40	22,19	28.800,00	26.519,00	26.541,19
Linear	40	8,75	28.800,00	24.889,00	24.897,75
Aleatório	40	Inf	28.800,00	Inf	Inf
Regras (Heurísticas)	40	21,67	28.800,00	26.519,00	26.540,67
Genético	40	3,90	28.800,00	21.195,00	**21.198,90**
Ganancioso	80	161,07	28.800,00	28.305,00	28.466,07
Linear	80	37,88	28.800,00	27.285,00	27.322,88
Aleatório	80	Inf	28.800,00	Inf	Inf
Regras (Heurísticas)	80	Inf	28.800,00	Inf	Inf
Genético	80	11,03	28.800,00	26.595,00	**26.606,03**
Ganancioso	160	1.271,22	28.800,00	28.575,00	1.123.372,31
Linear	160	64,94	28.800,00	27.554,00	27.618,94
Aleatório	160	Inf	Inf	Inf	Inf
Regras (Heurísticas)	160	Inf	Inf	Inf	Inf
Genético	160	29,71	28.800,00	27.102,00	**27.131,71**

Tabela 0.1 – Demostração de Resultados para cada um dos algoritmos de Fu(y)

6.2.2.Função Utilidade Fu(z)

			Tempo (segundos)		
Algoritmos	**Tarefas (TxOp)**	**TE(z)**	**TC(z)**	**TJ(z)**	**Função Utilidade Fu(z)**
Ganancioso	20	16,17	28.800,00	26.000,00	26.000,00
Linear	20	2,68	28.800,00	24.951,00	24.951,00
Aleatório	20	Inf	28.800,00	Inf	Inf
Regras (Heurísticas)	20	21,67	28.800,00	27.777,00	27.777,00
Genético	20	1,82	28.800,00	11.918,00	**11.918,00**
Ganancioso	40	22,19	28.800,00	26.519,00	26.519,00
Linear	40	8,75	28.800,00	24.889,00	24.889,00
Aleatório	40	Inf	28.800,00	Inf	Inf
Regras (Heurísticas)	40	21,67	28.800,00	26.519,00	26.519,00
Genético	40	3,90	28.800,00	21.195,00	21.195,00
Ganancioso	80	161,07	28.800,00	28.305,00	28.305,00
Linear	80	37,88	28.800,00	27.285,00	27.285,00
Aleatório	80	Inf	28.800,00	Inf	Inf
Regras (Heurísticas)	80	Inf	28.800,00	Inf	Inf
Genético	80	11,03	28.800,00	26.595,00	**26.595,00**
Ganancioso	160	1.271,22	28.800,00	28.575,00	28.575,00
Linear	160	64,94	28.800,00	27.554,00	27.554,00
Aleatório	160	Inf	28.800,00	Inf	Inf
Regras (Heurísticas)	160	Inf	28.800,00	Inf	Inf
Genético	160	29,71	28.800,00	27.102,00	**27.102,00**

Tabela 0.2 – Demostração de Resultados para cada um dos algoritmos de Fu(z)

6.3. Estrutura da BD

Para auxílio na bancada de ensaios foi utilizada uma base de dados MySql para armazenar dados relevantes na utilização do simulador e gerador. Apesar de apresentar uma estrutura simples, a figura 6.7 apresenta o modelo de dados utilizado.

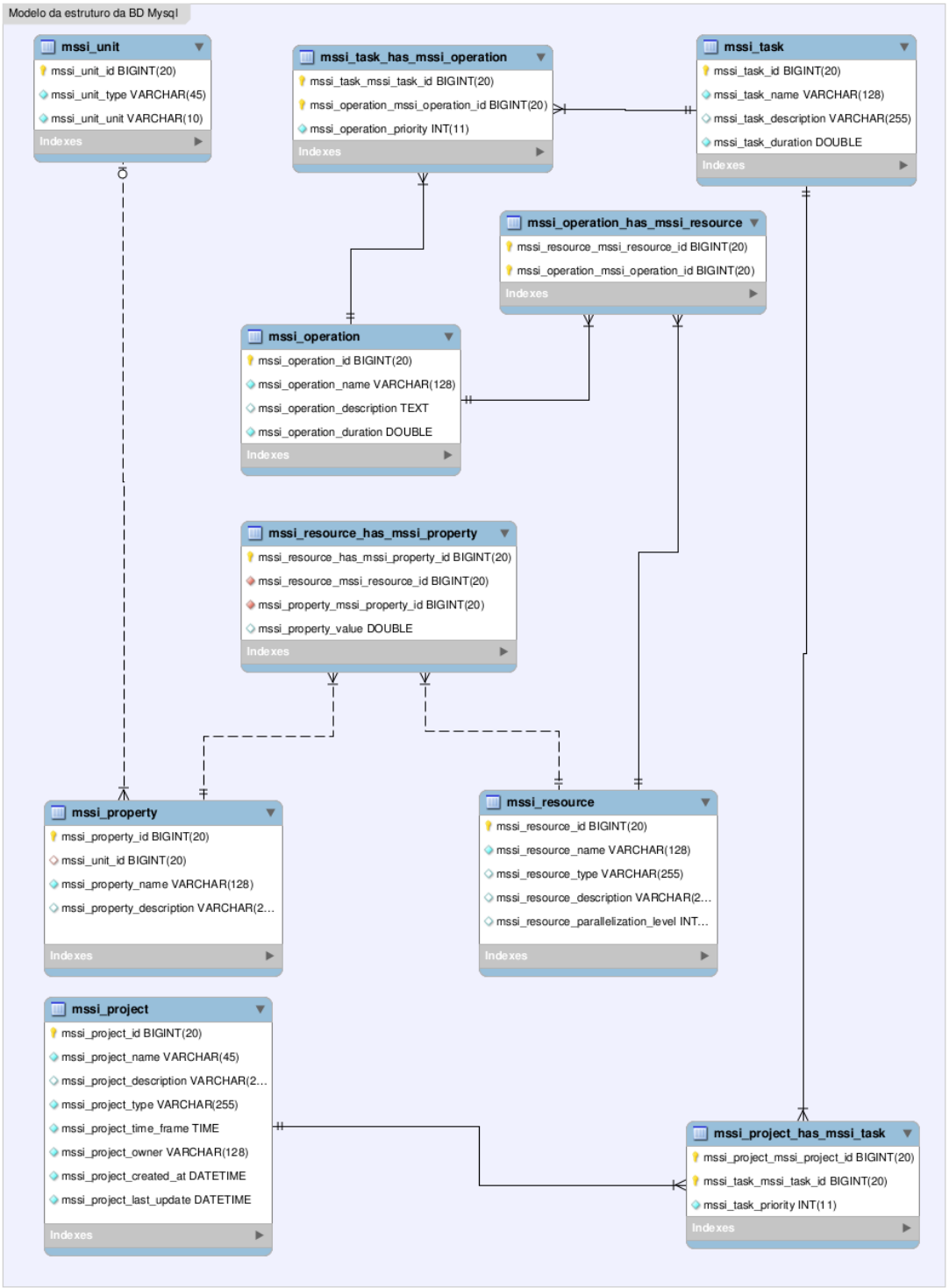

Figura 0.7 – Modelo de Dados da BD MySql

Referências

(Bent, 2004) Russell Bent and Pascal Van Hentenryck, *Regrets Only! Online Stochastic Optimization under Time Constraints, American Association for Artificial Intelligence, 2004*

(Davenport Davenport, A., & Beck, J. C. 2000). *A survey of techniques for scheduling with uncertainty. Tech. rep.*

(Nazarathy, 2001), *Evaluation of On-Line Scheduling Rules for High Volume Job Shop*

Problems, a Simulation Study. University of Haifa. A. in Applied Probability (summa cum laude), Department of Statistics

(Thomas 2007), *Evaluating Online Scheduling Techniques in Uncertain Environments. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference (MISTA 2007)*

(Barnes, 2001), *Introduction; The Definitive Handbook of Business Continuity Planning; Wiley*

(BCPG, 1998) Various Authors, *The Business Continuity Planning Guide, The Office*

of Public Service do Governo do Reino Unido.

(Brown & Nasuti, 2007), *Sarbanes - Oxley and Enterprise Security: IT Governance - What It Takes to Get the Job Done*

(Charters, 2001), *Risk Evaluation and control: II Practical guidelines forrisk assessment - The Definitive Handbook of Business Continuity Planning; Wiley*

(Cornish, 2001), *The Business Continuity Planning Methodology;The Definitive Handbook of Business Continuity Planning; Wiley*

(DeLone & McLean, 2003),

(Fulmer,2000), *Business Continuity Planning, 2000 Edition: A Step-by-Step Guide with Planning Fomrs*

- (Hiles, Barnes, 1999), *The Definitive Handbook of Business Continuity Management", Wiley e Survive, 1999*
- Meredith, (2001), *Business Imapct Analisys - The Definitive Handbook of Business Continuity Planning; Wiley*
- Serrano & Jardim, (2007), *Disaster Recovery, Um paradigma na Gestão da Continuidade*
- SIBS, (2006), *Plano de Continuidade de Negócio*
- Strohl, (2002), *The Business Continuity Planning Guide (2002)*
- Syed, (2004), *Business Continuity Planning Methodology*
- Vancoppenolle,(2001), *The Definitive Handbook of Business Continuity Planning*
- LUNA, (1997) - *Sérgio Vasconcelos. Planeamento de pesquisa: uma introdução. São Paulo: EDUC, 1997.*

Pesquisas efectuadas na internet:

http://www.obitko.com/tutorials/genetic-algorithms/example-function-minimum.php

http://repositorio-iul.iscte.pt/bitstream/10071/169/1/SCH-INT.pdf

http://paginas.fe.up.pt/~eol/PRODEI/mpe1011_files/Intro_MPE.pdf

https://estudogeral.sib.uc.pt/bitstream/10316/8541/3/AntonioFerrolho_PhD.pdf

http://www.obitko.com/tutorials/genetic-algorithms/example-function-minimum.php