Universidade do Minho
Escola de Engenharia

João Carlos Torres Arantes Maia

Desenvolvimento de um pacote para facilitar as tarefas de otimização na ferramenta R

Dissertação de Mestrado

Mestrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efetuado sob a orientação do

Professor Doutor Paulo Alexandre Ribeiro Cortez

Outubro 2016

DECLARAÇÃO

Nome: João Carlos Torres Arantes Maia

Endereço eletrónico: joao.arantes.maia@gmail.com **Telefone:** 916941597

Cartão do Cidadão: 14348251 3 ZY3

Título da dissertação: Desenvolvimento de um pacote para facilitar as tarefas de otimização na ferramenta R

Orientador:

Professor Doutor Paulo Alexandre Ribeiro Cortez

Ano de conclusão: 2016

Mestrado integrado em Engenharia e Gestão de Sistemas de Informação

Universidade do Minho, _____/_____/_____

Assinatura:

AGRADECIMENTOS

No término de uma fase muito importante da minha vida, gostaria sobretudo de agradecer aos meus pais, por me terem proporcionado a oportunidade de chegar a este ponto. Sem eles eu não teria o apoio, motivação, condições e carinho para chegar a este ponto. São sem duvida um exemplo para mim e eles lhe dedico todo o trabalho até agora realizado. Também agradeço à minha irmã Joana, que apesar de todas as nossas divergências enquanto irmãos, esteve sempre presente nos grandes momentos e pela ajuda sempre que ofereceu quando eu necessitava.

Agradeço ao professor Doutor Paulo Cortez, por toda a ajuda prestada neste longo caminho, por estar sempre presente nos momentos em que precisei dele e por todas as orientações dadas que sem dúvida permitiram chegar a este ponto.

Aos meus amigos da Universidade do Minho, que partilharam comigo um percurso de 5 anos, ao qual considero dos melhores anos da minha vida, percurso esse de altos e baixos, mas em que sempre se encontravam presentes em todos os momentos. Agradecimento ao Fábio Silva, Pedro Pereira e Tiago Vilela, pelas noites passadas na biblioteca da Universidade do Minho, na motivação mútua e na ajuda da realização deste trabalho de dissertação. Também gostaria de agradecer a Andreia "Rosa" Costa, Carlos Correia, Telmo Sousa, Ricardo Carvalho, Manuel Silva, e entre outros, que sempre me acompanharam nos trabalhos da Universidade e em tudo a que a vida académica me proporcionou. Agradeço assim a todos os elementos de "A Família" e do *"Workout"*, aos amigos conhecidos na Universidade e com momentos vividos muito bons, e que os levarei para sempre comigo.

Também a Ana Brás, Joana Oliveira, André Moreira e João Cardoso, aquele grupo de amigos que apesar de ter surgido há pouco tempo, me proporcionou bons momentos e apoio.

Aos meus amigos Hugo, Fernando e Mário, que me acompanharam na Escola Oficina e que são aqueles com quem posso contar para tudo, tanto para momentos sérios como para momentos de total animação.

A todos eles muito obrigado.

RESUMO

Hoje em dia, grande parte dos problemas de mundo real podem ser considerados problemas de otimização onde se pretende maximizar ou minimizar um determinado objetivo, sendo estes problemas encontrados em diversas áreas como Agricultura, Economia, Marketing, Produção, entre outras. Com os avanços tecnológicos e com o nível de complexidade dos problemas a aumentar, foi necessário adotar ferramentas computacionais para resolver problemas de otimização utilizando algoritmos de otimização para apoiar no processo de tomada de decisão.

Com isto surgiram diversos algoritmos de otimização que pretendem procurar pela solução ótima tendo em conta um determinado objetivo. Estes algoritmos surgiram implementados em diversas plataformas computacionais, incluindo a ferramenta R.

Esta dissertação consistiu em criar um pacote para a ferramenta R, para facilitar o utilizador no uso destes algoritmos de otimização moderna. Como tal, foi necessário efetuar um levantamento de algoritmos existentes, perceber o seu funcionamento, e de que forma se poderia facilitar a um utilizador com pouca experiencia o uso deste tipo de algoritmos na ferramenta R.

O pacote criado tem o nome de *"R Modern Optimization" (RMO).* Como tal, foi criada a documentação necessária para apoiar o utilizador no uso do pacote. Após a criação do pacote e disponibilização do mesmo no repositório *GitHub,* foi disponibilizado um questionário a utilizadores que tenham testado o pacote para perceber a opinião dos utilizadores em relação ao pacote `rmo`. O *feedback* obtido a partir destes questionários indica que o pacote `rmo` é deveras útil, facilitando a condução de projetos de otimização moderna na ferramenta R.

Palavras chave: Ferramenta R; Pacote; Otimização Moderna

ABSTRACT

Nowadays, there is a vast number of real-world problems that can be considered as optimization tasks, where it is pretended to maximize or minimize a goal. These problems can be found in several areas, such as Agriculture, Finance, Marketing, Production, and many more. With the technological advances and the complexity of these problems getting bigger, it was necessary to adopt computational tools to solve these tasks using optimization methods, to help with the decision making process.

Therefore, several algorithms have been created they main job is to find a solution taking into account a certain goal. Several of these algorithms were implemented in several computational platforms, including the R tool.

In this Master of Science Thesis it is pretended to create a package for the R tool, to facilitate the user when applying modern optimization algorithms. As such, it was needed to study which modern optimization algorithms were available, how these algorithms work and how to make easier their usage with the R tool.

The package that was created was named "R Modern Optimization" (RMO). Also, the `rmo` package was documented, in order to facilitate its usage by optimization analysts. After creating the package, it was published in the GitHub repository, and then a Web questionnaire was provided to several users. The obtained feedback reveals that the `rmo` package is valuable and can make easier the process of executing modern optimization tasks in the R tool.

Keywords: R tool; Package; Modern Optimization

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

Lista de Abreviaturas, Siglas e Acrónimos

ABC – *Artificial Bee Colony*

CRAN – *Comprehensive R Archive Network*

DFEV – *Differential Evolution*

DSRM – *Design Science Research Methodology*

EDA – *Estimation of Distribution Algorithm*

FBS – *Full Blind Search*

GEA – *Genetic and Evolutionary Algorithms*

GRS – *Grid Search*

HLC – *Hill Climbing*

HTML – *HyperText Markup Language*

IDE – *Integrated Development Environment*

MCS – *Monte Carlo Search*

PDF – *Portable Document Format*

PSO – *Particle Swarm Optimization*

RMO – *R Modern Optimization*

SANN – *Simulated Annealing*

TABU – *Tabu Search*

1. INTRODUÇÃO

Este capitulo procura descrever o contexto desta dissertação, bem como a motivação associada à mesma. O que levou ao desenvolvimento desta dissertação foi o facto de não haver um pacote para simplificar o uso de algoritmos de otimização para a ferramenta R, ao qual esta dissertação procurou resolver.

Posteriormente à descrição da motivação, é feita uma descrição dos objetivos que se pretendem atingir e também uma breve descrição da estrutura documental desta dissertação.

1.1 Motivação

A criação deste projeto de dissertação prendeu-se com o facto de não existir um pacote para a ferramenta R que simplifique e facilite o uso de tarefas de otimização moderna. Hoje em dia, um grande número de problemas do mundo real pode ser abordado como uma tarefa de otimização, em que se pretende minimizar ou maximizar um determinado objetivo, estando a otimização presente em diversas áreas como Finanças, Agrícola, Produção, Engenharia e muitas outras.

Nos dias de hoje muitos dos problemas de otimização contêm um nível de complexidade bastante elevado, sendo por vezes esse problema resolvido por humanos. Com a evolução da capacidade computacional, muitos desses problemas passaram a ser resolvidos através de computadores, poupando assim imenso tempo, recursos necessários, permitindo assim uma maior eficiência por parte das organizações.

Os métodos de otimização moderna são úteis para a resolução de problemas complexos onde nenhum algoritmo foi criado para a resolução de um problema específico (Sean, 2012). Em comparação com os métodos clássicos, os métodos modernos não garantem que a solução ótima seja sempre encontrada, no entanto são capazes de obter soluções com maior qualidade requerendo menores recursos computacionais (Michalewicz & Fogel, 2004).

Neste caso, a razão da criação do pacote para ferramenta R é devido a inexistência de pacotes de otimização que sejam tão abrangentes de modo a permitir executar uma vasta gama de métodos de otimização moderna via uma única função. É importante para esta ferramenta a criação de um pacote destes, visto que a área de otimização se encontra em grande expansão e a ferramenta também estar a ser utilizada cada vez por mais pessoas e nas mais diversas áreas, tais como *Data Mining; Machine Learning*; Simulação e outras.

1.2 Objetivos

O principal objetivo desta dissertação é a criação de um pacote que facilite as tarefas de otimização, utilizando métodos de otimização moderna na ferramenta R. Como tal pretende-se um agrupamento de diversos métodos de otimização moderna num único pacote, de forma a facilitar as tarefas dos utilizadores durante a utilização dos mesmos. Para a concretização deste objetivo, é necessário a criação de pequenas metas que complementem este objetivo final, tais como:

- Criação de um suporte de informação sobre métodos de otimização moderna através da revisão de literatura;
- Análise e avaliação de métodos de otimização para a incorporação no pacote;
- Desenvolvimento do pacote seguindo as normas de uma metodologia de desenvolvimento prototipagem rápida;
- Avaliação do pacote desenvolvido via testes em casos práticos de otimização e em questionários efetuados a diversos utilizadores ao qual o pacote foi distribuído; e
- Demonstrar se os objetivos foram cumpridos.

O cumprimento destes objetivos torna-se fulcral para a concretização do objetivo principal que é o desenvolvimento de um pacote para facilitar as tarefas de otimização na ferramenta R.

1.3 Estrutura do Documento

Este documento é o resultado de todo o trabalho desenvolvido. Foi necessário para a concretização desta dissertação, um planeamento adequado, em que foram recolhidas informações de diversas fontes, para se poder expor o problema a ser tratado. Esta dissertação encontra-se organizada em seis capítulos, descritos a seguir:

- **Capítulo 1** – Este capítulo é constituída pela Introdução onde é descrita a motivação para o problema proposto e onde são definidos os objetivos para esta dissertação;
- **Capítulo 2** - Consiste no Estado da Arte, em que é feita uma breve explicação de como foi realizada a pesquisa para o estado da arte bem como uma análise aos métodos de otimização moderna que foram encontrados. Ainda no Estado da Arte, é demonstrado o resultado da pesquisa dos pacotes para a ferramenta R que podem ser relacionados com o pacote a desenvolver:
- **Capítulo 3 -** É descrita a Abordagem Metodológica e a ferramenta R, pelo que são descritas a metodologia de investigação *Design Science Research Methodology* e a metodologia de desenvolvimento Prototipagem Rápida. É realizada também uma descrição sobre as ferramentas utilizadas para o desenvolvimento do pacote;

- **Capítulo 4 –** Neste capítulo é efetuada a descrição detalhada do processo de desenvolvimento do pacote *"R Modern Optimization"* (`rmo`). Este capítulo encontra-se dividido em 7 secções:
    - Analise de requisitos – Onde são descritos os requisitos funcionais e não funcionais pretendidos para o pacote;
    - Desenvolvimento do pacote – Esta é uma das secções com mais descrição de todo o processo de desenvolvimento do pacote `rmo`. De uma forma detalhada é feita a descrição de todo o processo de desenvolvimento, desde testes para obter informações sobre de como é desenvolvimento um pacote, implementação de métodos, entre outros;
    - Documentação – Nesta secção é descrita como foi efetuada a documentação para o pacote `rmo`. Este foi considerado um passo importante no desenvolvimento do pacote;
    - Pacotes necessários – São descritos os pacotes que foram necessários para o desenvolvimento e funcionamento do pacote `rmo`. São considerados, grande parte destes pacotes fulcrais para o funcionamento e uma grande abrangência de métodos de otimização do pacote `rmo`;
    - Instalação – É feita uma descrição do processo de instalação do pacote `rmo`. Também são descritos os passos que levaram a este tipo de instalação;
    - GitHub – É realizada uma descrição do processo de disponibilização do pacote `rmo` para futuros utilizadores; e
    - Funcionalidades disponíveis e Exemplos de Funcionamento – Apesar de apenas ter sido implementada uma função no pacote `rmo`, esta mesma função contém um grande poder de adaptação, pelo que é feita uma descrição de como funciona e são também exemplificadas formas distintas de como pode ser utilizada o pacote `rmo`.
- **Capitulo 5 –** Aqui neste capitulo é feita uma avaliação ao pacote `rmo`. É feita uma descrição de como o pacote `rmo` foi validado e uma análise aos resultados obtidos; e
- **Capitulo 6 –** Onde é feita uma síntese do trabalho realizado e apresenta-se uma discussão sobre o trabalho realizado. Também é realizada uma análise sobre o trabalho futuro.

2. ESTADO DE ARTE

Neste capitulo é descrito o estado de arte que foi realizado para esta dissertação, estando dividido inicialmente por uma descrição da estratégia de pesquisa bibliográfica. Após isso na Secção 2.2 são descritos os métodos de otimização moderna que foram encontrados durante a pesquisa bibliográfica. Finalmente na Secção 2.3, é feita uma relação entre os métodos de otimização moderna e a ferramenta R.

2.1 Estratégia de Pesquisa Bibliográfica

Para o desenvolvimento desta dissertação foi necessário efetuar um levantamento do estado da arte, que consistiu na recolha de informação através de pesquisas bibliográficas. Este processo deveu-se a necessidade de compreender quais os métodos de otimização moderna existentes e a existência de pacotes para a ferramenta R que pudessem ser relacionados com esta área do trabalho. Este processo tornou-se importante para uma definição do trabalho futuro e para clarificar o problema inicialmente proposto.

Inicialmente foi fornecido por parte do orientador, o livro escrito por si chamado *"Modern Optimization With R"* (Cortez, 2014), pelo que este livro é uma base bastante importante para a recolha de informação sobre a existência de métodos de otimização moderna. Também, através de outros livros foram analisados mais métodos de otimização moderna, nomeadamente os livros: "*Adaptive Business Intelligence*" (Michalewicz, Schmidt, Michalewicz, & Chiriac, 2006), "*Clever Algorithms*" (Brownlee, 2011), "*Essentials of Metaheuristics*" (Sean, 2012) e "*Nature Inspired Optimization Algorithms*" (Bash, 2015).

Também foi necessário efetuar uma pesquisa documental em páginas *Web* para complementar informação obtida a partir dos livros analisados, de forma a verificar a existência de mais métodos de otimização moderna. Como tal foi necessário definir algumas palavras-chave para se poder efetuar a pesquisa como por exemplo: "*Metaheuristics*", "*Modern Optimization*", *"Evolutionary Computation", "Nature Inspired Optimization Algorithms", "Optimization Algorithms", "Metaheuristiscs State of The Art".* A pesquisa foi efetuada em páginas *Web*, principalmente no *Google Scholar* (http://scholar.google.pt/), mas também em outras páginas *Web* como a *b-on* (http://www.b-on.pt/), *Scopus* (http://www.scopus.com/) e *Science Direct* (http://www.sciencedirect.com/).

Após esta pesquisa, procedeu-se a pesquisa de pacotes que contivessem os métodos de otimização anteriormente referidos. Este passo tornou-se bastante importante porque com esta pesquisa

a pacotes que podem ser relacionados com o que se pretende desenvolver, é possível retirar ideias do seu funcionamento e também de uma forma bastante importante, utilizar o pacote como dependência para o pacote a ser criado. Esta pesquisa foi efetuada recorrendo a pesquisas no repositório CRAN *(Compreensive R Archive Network)*. Este repositório é bastante importante porque é o repositório oficial da ferramenta R e porque é onde se encontra grande parte dos pacotes para a ferramenta R. Este repositório está espalhado por vários servidores a nível global, que contém os pacotes e a documentação necessária para o seu funcionamento. Também foi efetuada a pesquisa em motores de busca como o *Google* devido a existência de alguns pacotes não se encontrarem no repositório CRAN, mas sim em outros, como o *GitHub.*

No final foi criada uma matriz onde se pode ver a relação de pacotes existentes com métodos de otimização moderna pesquisados e analisados anteriormente, fazendo assim um mapeamento de onde para se poder fazer a verificação da existência ou não de código ou pacote para um determinado método de otimização moderna.

2.2 Métodos de Otimização Moderna

Nesta fase será feita uma análise e descrição de métodos de otimização moderna onde será descrito seu funcionamento e os tipos de métodos de otimização moderna existentes. Hoje em dia muitos problemas de otimização requerem um elevado esforço em termos computacionais para encontrar uma solução satisfatória. Esta é a razão da existência de interesse na criação de algoritmos que sejam capazes de encontrar uma solução num tempo computacional razoável.

Os métodos de otimização moderna são conhecidos também como Metaheurísticas. A palavra Heurísticas tem como significado um conjunto de regras para a pesquisa de uma solução num espaço de procura. Existem dois tipos de Heurísticas, de pesquisa local, onde é feita uma pesquisa num espaço até se encontrar a solução ótima. Ou algoritmos construtivos ou baseados em populações de soluções, onde se juntam várias "peças" até se obter a solução final (Bianchi, Dorigo, Gambardella, & Gutjahr, 2009). A palavra Meta significa "acima de um nível".

Segundo Blum e Roli (2003) o conceito de Metaheurísticas é: "um alto nível de conceitos para explorar espaços de pesquisa utilizando diferentes estratégias. Estas estratégias devem ser escolhidas de uma maneira a criar um balanço entre acumulação de experiencia na pesquisa e a exploração do espaço de pesquisa. Este balanço é necessário por um lado para identificar regiões com soluções de alta qualidade no espaço de pesquisa e por outro lado para não perder muito tempo em regiões em que o

espaço de pesquisa já foi explorado e não consegue providenciar soluções de qualidade" (C. Blum & Roli, 2003).

Os métodos de otimização podem ser classificados de três formas que são:

- Tipo de procura;
- Solução determinística ou estocástica e
- Inspirado naturalmente.

Para se perceber a classe de cada um dos métodos de otimização e ao grupo a que pertencem é apresentado um mapeamento na Figura 1, exemplo retirado do livro *"Modern Optimization with R"* (Cortez, 2014):

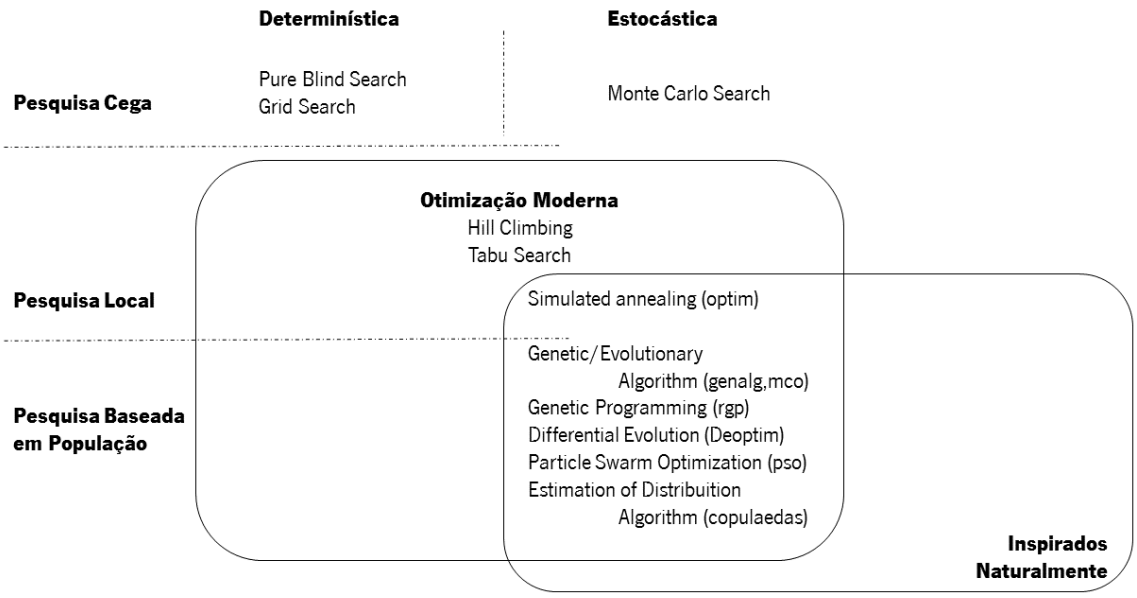


Figura 1 - Classificação dos métodos de otimização a serem apresentados nesta dissertação, adaptado e traduzido de "*Modern Optimization with R*".

Na procura cega, são analisadas todas as soluções num espaço determinado espaço onde a solução anterior não afeta a nova solução, até que por fim é selecionada a solução ótima. Como durante a procura é analisado todo o espaço disponível para se procurar por uma solução, sendo que é sempre encontrada a solução ótima, sendo assim determinístico de procura. Esta procura pode ser feita através de uma criação de uma matriz em que é feita a procura da solução ótima, ou através de uma forma recursiva utilizando árvores de procura.

No funcionamento procura guiada, as novas soluções são geradas por influência de soluções anteriores. Na procura guiada esta pode ser local ou pode ser global. Na procura local é feita uma pesquisa na vizinhança de um espaço até se encontrar a solução ótima. A abordagem mais comum

neste tipo de procura é baseada na escolha aleatória de um ponto inicial e depois efetuar novas pesquisas através de procuras próximas desse ponto inicial.

Na procura global, o seu funcionamento é através de uma pesquisa envolvendo um conjunto de populações. Este tipo de procura é efetuada através de um conjunto de soluções candidatas em vez de apenas uma solução. Esta forma de procura tende a requerer mais esforço computacional visto que é feita uma procura global. É uma procura mais abrangente porque tende a explorar regiões mais distintas do espaço de procura. Grande parte dos algoritmos de otimização baseados em população são inspirados naturalmente (Sean, 2012). A diferença deste tipo de métodos de procura global em relação a outros iguais é em termos de como a solução é representada; que atributos estão a ser guardados por cada procura; como novas soluções são criadas; e como a solução ótima de cada população é selecionada.

Durante a implementação de um método de otimização moderna o utilizador deve ter em conta os seguintes aspetos: representação da solução e a função objetivo (Michalewicz et al., 2006). Para a representação da solução o utilizador tem que decidir como a vai implementar, porque esta indica o espaço de procura e o seu tamanho, sendo que também influencia a forma como novas soluções são pesquisadas. Na representação de uma solução podem ser usados vários tipos de representações por exemplo: binário, inteiro, caracteres, matrizes e árvores. Muitas das vezes, o tipo de representação depende do tipo de método de otimização, isto é, nos Algoritmos Genéticos utiliza-se representações binárias, nos Algoritmos evolucionários utiliza-se representações em valores reais (Cortez, 2014).

Outro fator importante é a função objetivo, que serve para traduzir se o objetivo é para minimizar ou para maximizar. Esta função permite comparar diferentes soluções através de um ranking ou através de métricas de qualidade (Michalewicz et al., 2006). Em vários casos alguns métodos de otimização funcionam apenas para minimizar um determinado objetivo. Nesses casos a solução passa por transformar a função de maximização para minimização da seguinte forma: $f'(s) = - f(s)$. Onde *s* representa a solução. Em muitas áreas é necessário otimizar não um, mas sim vários objetivos. Utilizando algoritmos de otimização que funcionam só com uma solução pode levar a que um objetivo seja otimizado, mas os outros sejam prejudicados, levando a resultados inesperados e inaceitáveis. Nesses casos utiliza-se uma otimização multiobjectivo (Cortez, 2014).

Durante o processo de otimização é normal encontrarem-se restrições que levam a que se influenciem o resultado final de otimização. Existem dois tipos de restrições (Michalewicz, 2008): *hard* e *soft*. As restrições *hard* não podem ser violadas, mais por fatores como leis ou restrições físicas. As restrições *soft* podem ser adaptadas utilizando otimizações multiobjectivo (Cortez, 2014). Para evitar que as restrições pesadas podem gerar soluções pouco viáveis são aplicados procedimentos no procedimento

de otimização, para tratar esse tipo de problemas há vários métodos que podem ser aplicados (Michalewicz et al., 2006):

- ***Death Penalty*** – Nesta situação tal como o nome diz é aplicar uma "pena de morte", mas na realidade o que acontece é que é aplicado um valor demasiado alto na pesquisa que faz com que essa solução na pesquisa seja descartada. No entanto não é muito eficiente, pois atribui uma elevada penalização numa solução inválida, descartando assim soluções que podem ter alguma informação útil;
- ***Penalty Weights*** – O objetivo desta solução é aplicar um "peso" a solução ótima, por exemplo, numa determinada função aplicar seria da seguinte forma: $f = P1\ x\ Lucro(s) - P2\ x\ Custo(s)$. Sendo que $P1$ e $P2$ são os pesos a aplicar a solução final. No entanto esta solução que é fácil de implementar, torna-se difícil encontrar os "pesos" ótimos, especialmente quando existem várias restrições;
- ***Repair*** – Esta abordagem faz com que uma solução que não é viável se torne viável. Isto faz-se utilizando informação dependente do domínio ou através da procura de novas soluções na vizinhança; e
- ***Only Generate Feasible Solutions*** – Com este método pretende-se que só sejam geradas soluções viáveis e para tal há várias soluções, com a utilização de descodificadores que apenas apresentam soluções de espaços de procura viáveis. Outra solução é com operadores que obtém conhecimento através de soluções anteriores e com esse conhecimento geram novas soluções.

Neste capítulo, será dado um maior destaque aos métodos de otimização descritos no livro *"Modern Optimization with R"* de Paulo Cortez (2014).

2.2.1 *Full Blind Search*

Este método de otimização conhecido também por procura cega, efetua uma procura por todas as alternativas possíveis, não sendo influenciado pelo resultado da solução anterior. Sendo este um método em que todo o espaço de soluções é testado, a solução ótima é sempre encontrada, sendo assim um método determinístico porque apresenta sempre a mesma solução (Cortez, 2014). Devido a procura da solução ótima em todo o espaço disponível, este pode causar uma carga computacional incomportável. Durante a sua execução, a geração de uma solução não é influenciada pela solução anterior, sendo que as soluções são independentes.

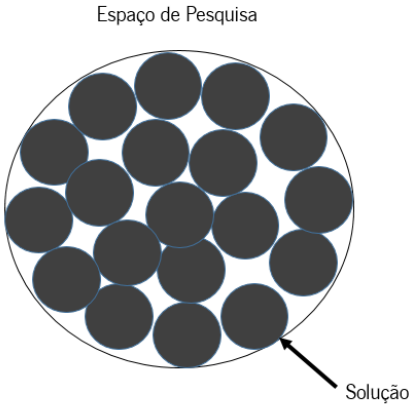


Figura 2 - Exemplo do método *Full Blind Search*, adaptado e traduzido de "*Modern Optimization with R*"

2.2.2 *Grid Search*

Este método que é de procura em grelha, pelo que permite que se possa reduzir o espaço de procura utilizando pesquisas com grelhas dimensionais. Este algoritmo é utilizado em particular para otimização de híper parâmetros em algoritmos de *machine learning*, como redes neuronais ou *support vector machines.*

Existem várias variantes da procura em grelha, como a variante de procura uniforme (Huang, et al., 2007), em que é semelhante ao método original de procura em grelha, mas este utiliza um diferente tipo de grelha com menos pontos de procura. Outra forma de procura em grelha é de forma agrupada onde é feita uma procura de forma mais abrangente, onde é selecionada a melhor solução e depois é efetuada uma nova procura a outro nível de grelha com um tamanho mais pequeno e assim sucessivamente até encontrar a solução final.

Este método de forma agrupada é mais rápido que a procura cega. Também dependendo do número de níveis e o salto que dá entre grelhas, esta abordagem pode-se tornar mais rápida que a procura em grelha original (Cortez, 2014).

Pode-se verificar na Figura 3 como é realizada uma procura em grelha com duas dimensões.

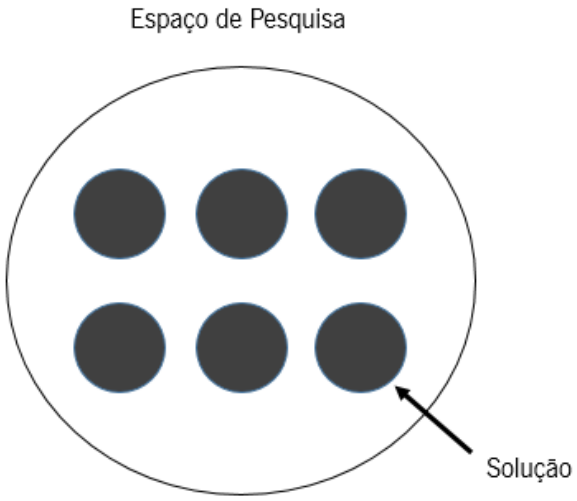


Figura 3 - Estratégia de procura em grelha traduzido e adaptado de *"Modern Optimization with R*'

2.2.3 *Monte Carlo Search*

Este é um método de fácil implementação e um método estocástico, pelo que não se consegue obter sempre a solução ótima de uma forma repetitiva durante a sua utilização. O seu funcionamento consiste em obter um número de soluções aleatórias utilizando uma distribuição de probabilidade, o que pode provocar uma solução longe do ideal (Cortez, 2014).

2.2.4 *Hill Climbing*

O método *Hill Climbing* é um método de procura local, isto significa que as suas soluções dependem sempre da solução anteriormente encontrada, isto é, tem sempre em atenção a vizinhança de uma determinada solução. Assumindo que o tipo de funções é de maximização, o seu funcionamento consistem em subir a "colina" até ao encontro da solução ótima (Cortez, 2014). De uma forma iterativa, vai procurando sempre novas soluções na vizinhança e caso essa seja melhor escolhe essa como a nova, caso contrário é escolhido um outro vizinho e é verificado se a nova solução é a melhor. A execução deste algoritmo termina quando se atinge um critério de paragem, tal como não se encontrar uma melhoria na melhor solução ao fim de algumas iterações ou quando passa um dado tempo limite de execução (Michalewicz et al., 2006).

2.2.5 *Simulated Annealing*

O *Simulated Annealing (SA),* ou em português Arrefecimento Simulado é um dos primeiros algoritmos de otimização moderna (C. Blum & Roli, 2003). É um método que é semelhante ao *Hill Climbing,* sendo este inspirado no arrefecimento de metais, onde se assume que um metal é inicialmente aquecido e depois é realizado um arrefecimento de uma forma controlada até a obtenção de um metal num estado solido (Cortez, 2014).

O seu funcionamento consiste em escolher um valor alto para a variável *T* (frequentemente referido como temperatura) e uma solução aleatória inicial Em cada iteração, faz-se uma procura de *Hill Climbing* mas com um resultado probabilístico que depende da temperatura. Assim, a escolha da solução a adotar é inicialmente mais aleatória para elevadas temperaturas mas à medida que o processo evoluí, e a temperatura arrefece, o processo tende a ser mais determinístico (Michalewicz et al., 2006).

Na Figura 4 é possível ver o funcionamento do algoritmo, figura retirada de "*A survey on optimization metaheuristics"* (Bianchi et al., 2009).

SA

```
1  Choose, at random, an initial solution s for the system to be optimized
2  Initialize the temperature T
3  while the stopping criterion is not satisfied do
4  |  repeat
5  |  |  Randomly select s' ∈ N(s)
6  |  |  if f(s') ≤ f(s) then
7  |  |  |  s ← s'
8  |  |  else
9  |  |  |  s ← s' with a probability p(T, f(s'), f(s))
10 |  |  end
11 |  until the "thermodynamic equilibrium" of the system is reached
12 |  Decrease T
13 end
14 return the best solution met
```

Figura 4 - Algoritmo para o método *Simulated Annealing*

2.2.6 *Tabu Search*

O método de otimização *Tabu Search* foi formalizado em 1986 por Glover, sendo baseado no conceito de memória, isto é, a procura é realizada em novas áreas do espaço de procura de forma a escapar a solução anterior (Michalewicz et al., 2006).

A procura pela solução deste método é equivalente ao método de otimização *Hill Climbing,* visto que procura por soluções na vizinhança da solução corrente.

Durante a execução deste método, é guardado um histórico das soluções mais recentes que são consideradas "tabu". Quando se escolhe uma nova solução, esta vai para a lista que contém o histórico de soluções, impedindo assim que essas soluções sejam procuradas de novo. Ao impedir que a procura retorne a valores que se encontram na lista, faz com que a procura não entre num ciclo infinito (Boussaïd, et al., 2013). Se esta lista de histórico de soluções estiver demasiado grande, o elemento mais antigo da lista é retirado (Sean, 2012).

A paragem da execução deste algoritmo é consoante o número de iterações escolhidas no início da execução.

Na Figura 5 é possível ver o funcionamento do algoritmo, figura retirada de "*A survey on optimization metaheuristics"* (Bianchi et al., 2009).

TS

1 Choose, at random, an initial solution s in the search space
2 $TabuList \leftarrow \varnothing$
3 **while** the stopping criterion is not satisfied **do**
4 | Select the best solution $s' \in N(s) \setminus TabuList$
5 | $s \leftarrow s'$
6 | Update $TabuList$
7 **end**
8 **return** the best solution met

Figura 5 - Funcionamento do método *Tabu Search*

2.2.7 *Genetic and Evolutionary Algorithms*

Este algoritmo pertencente ao conjunto de algoritmos baseados em população, em que individuos competem entre si procurando a melhor solução, por outras palavras, este algoritmo simula o processo evolutivo de competição e seleção natural (Michalewicz et al., 2006). Os algoritmos genéticos foram propostos por Holland (1975).

O seu funcionamento é através da geração inicial de uma população de candidatos ao contrário de outros métodos que geram uma só solução inicial. Após a geração da população é analisada a qualidade desta seguindo métricas de qualidade, sendo que depois é analisada a vizinhança para a procura de novas soluções. As soluções são geradas por operadores genéticos, a partir de um conjunto de soluções progenitoras, que geram assim soluções descendentes. Tal pode ser realizado via operadores de cruzamento ou de mutação (Michalewicz et al., 2006).

2.2.8 *Differential Evolution*

Differential Evolution é um método de otimização estocástico e de otimização global, pertencendo ao grupo de computação evolucionária (Brownlee, 2011). Tal como o algoritmo *Genetic and Evolutionary Algorithms,* este método envolve uma população de soluções. A diferença principal em comparação ao anterior algoritmo é que o *Differential Evolution* utiliza operadores aritméticos para a geração de novas soluções (Cortez, 2014).

Este algoritmo envolve que a população de soluções esteja envolvida em iterações que de recombinação, avaliação e seleção (Brownlee, 2011).

Na Figura 6 pode-se verificar o funcionamento do algoritmo, figura retirada do livro *"Clever Algorithms"* (Brownlee, 2011).

Algorithm 3.5.1: Pseudocode for Differential Evolution.

```
Input: Population_size, Problem_size, Weighting_factor,
       Crossover_rate
Output: S_best
Population ← InitializePopulation(Population_size,
Problem_size);
EvaluatePopulation(Population);
S_best ← GetBestSolution(Population);
while ¬ StopCondition() do
    NewPopulation ← ∅;
    foreach P_i ∈ Population do
        S_i ← NewSample(P_i, Population, Problem_size,
        Weighting_factor, Crossover_rate);
        if Cost(S_i) ≤ Cost(P_i) then
            NewPopulation ← S_i;
        else
            NewPopulation ← P_i;
        end
    end
    Population ← NewPopulation;
    EvaluatePopulation(Population);
    S_best ← GetBestSolution(Population);
end
return S_best;
```

Figura 6 – Pseudo código do algoritmo *Differential Evolution*

2.2.9 *Particle Swarm Optimization*

O funcionamento deste algoritmo é proveniente de um conjunto de algoritmos que é inspirado no comportamento de enxames, isto é, inspirado no comportamento do conjunto de vários animais (exemplo: pássaros, abelhas, formigas). Este algoritmo foi criado por Kennedy e Eberhart (1995) para otimização numérica, sendo um dos mais utilizados devido a sua flexibilidade e simplicidade (Bash, 2015).

O seu funcionamento é baseado num conjunto de partículas (soluções), sendo que o enxame corresponde a uma população de partículas. Cada partícula tem uma velocidade e direção, sendo que o

sucesso (via função objetivo) de uma dada partícula influencia a trajetória de outras partículas, que tendem a ir de encontro à sua vizinhança (Cortez, 2014).

Na Figura 7 é possível ver o funcionamento do algoritmo, figura retirada de "*A survey on optimization metaheuristics"* (Bianchi et al., 2009).

PSO

Initialize a population of particles with random positions and velocities on D dimensions in the search space
while Terminating condition is not reached **do**
  for each particle i **do**
    Adapt velocity of the particle using Equation (10)
    Update the position of the particle using Equation (11)
    Evaluate the fitness $f(\vec{X}_i)$
    if $(f(\vec{X}_i) < f(\vec{P}_i))$ **then**
      $\vec{P}_i \leftarrow \vec{X}_i$
    end
    if $(f(\vec{X}_i) < f(\vec{P}_g))$ **then**
      $\vec{P}_g \leftarrow \vec{X}_i$
    end
  end
end

Figura 7 - Funcionamento do algoritmo *Particle Swarm Optimization*

2.2.10 *Estimation of Distribution Algorithm*

Estimation of Distribution Algorithm (Larrañaga, 2002) é o método de otimização que combina ideias de computação evolutiva, *machine learning* e estatística. Enquanto outros métodos de otimização baseados em populações criam novos indivíduos utilizando uma função de distribuição, este cria novos indivíduos utilizando uma função de distribuição de probabilidades (Cortez, 2014). Isto leva a que se possa fazer uma procura mais efetiva. As novas soluções são incorporadas na solução original, sendo substituídas as soluções antigas (Boussaïd et al., 2013).

2.2.11 Outros Algoritmos

Durante a pesquisa por algoritmos de otimização moderna, foi possível verificar a existência de diversos algoritmos (ainda não mencionados) naturalmente inspirados, como é o caso de *"Ant Colony Optimization"* em que é baseado "no modo de vida" das formigas no meio ambiente (Christian Blum, 2005). Outros métodos semelhantes são o *"Bee Colony"* que foi inspirado nas colmeias de abelhas, em que o seu funcionamento é através de uma procura pela vizinhança através de procuras aleatórias (Pham

et al., 2006), ou o algoritmo "Firefly Algorithm" que baseia-se no comportamento dos pirilampos, através da atratividade de pirilampos de sexos diferentes (Yang & He, 2013).

Também existem vários algoritmos de computação evolutiva como o *"Scatter Search",* que procura por novas soluções, seguindo vários pontos de referência que consistem em boas soluções (Laguna, 1997).

A Tabela 1, descrita na secção 2.3, apresenta uma lista de algoritmos de otimização, em que se pode verificar o seu estado, isto é, se já se encontra em pacote para a ferramenta R, ou se existe código para se poder implementar no pacote a ser desenvolvido ou se simplesmente não existe nenhuma implementação em R.

2.2.12 Métodos de Otimização Multiobjectivo

Nas secções anteriores foi descrito sobre métodos de otimização que tinham como objetivo um único objetivo, no entanto grande parte dos problemas de hoje em dia contém vários objetivos que tem de ser atingidos de forma conjunta. Por exemplo, uma organização pretender aumentar as vendas e ao mesmo tempo diminuir os custos de produção, sendo que nesta situação estão presentes dois objetivos.

Para a resolução de um problema de otimização multiobjectivo existem três abordagens: *Weighted-Formula,* Lexicográfica e de Pareto (Freitas, 2004). A primeira abordagem, tem como vantagem a facilidade com que pode ser implementada. O seu funcionamento tem a ver com a aplicação de um "peso" aos objetivos e por fim otimizar a qualidade Q que pode ser usada utilizando adição ou multiplicação (Cortez, 2014):

$$Q = w1 * \ g1 + w2 * g2 + \cdots + \ wn * gn \ \text{, ou}$$
$$Q = g1^{w1} \ * g1^{w2} * \ldots * \ gn^{wn}$$

Esta abordagem tem algumas desvantagens, como a definição do peso ideal a atribuir a cada objetivo, outra desvantagem em relação a resolver o problema de otimização da variável Q, pelo que devido a variação dos pesos, irá levar a uma nova execução do problema de otimização.

Outra abordagem para a otimização multiobjectivo é a Lexicográfica, onde se pretende indicar uma prioridade a diferentes objetivos e com base nesses objetivos, efetuar a otimização com base na prioridade (Freitas, 2004). Quando dois ou mais candidatos são comparados uns com os outros para escolher o melhor, o que é feito é analisar a performance do objetivo com a prioridade mais alta. Se um candidato é melhor que outro, então esse é escolhido, senão é feita uma comparação com o segundo

objetivo mais importante. Este processo é realizado até que se tenha um objetivo que se destaque. Caso não se consiga chegar a uma conclusão, é selecionado aquele com maior prioridade.

Como tal esta abordagem traz vantagens e desvantagens, como quando comparado com a abordagem anterior, a abordagem lexicográfica consegue evitar que se misturem critérios não comensuráveis na mesma fórmula porque trata cada critério de forma individual. Também se pretende comparar várias soluções, a abordagem lexicográfica é mais fácil em comparação com a abordagem de Pareto.

A ideia da abordagem de Pareto é a transformação de um problema de otimização multiobjectivo em um problema de objetivo único e depois resolve-lo utilizando uma pesquisa de objetivo único. A solução $s1$ domina a solução $s2$ se $s1$ é melhor que $s2$ num objetivo e pelo menos boa como $s2$ nos outros objetivos. Uma solução si é não dominada quando não existe uma solução sj que domine si e a frente de Pareto contem todas as soluções não dominadas (Sean, 2012). O seu conceito de funcionamento é retornar um conjunto de soluções não dominadas em vez de apenas uma solução. Esta abordagem em comparação com outras tem diversas vantagens como, a representação de diversas soluções permitindo assim o utilizador fazer a sua escolha. Para obter os pontos de Pareto é necessário apenas uma execução. Como desvantagem é que esta abordagem necessita de um grande espaço de procura para ser explorado. Considerando que os métodos de Pareto necessitam de manter uma população de soluções, os algoritmos evolucionários tornam-se uma solução bastante importante para gerar um conjunto de soluções ótimas de Pareto (Cortez, 2014).

2.3 Otimização Moderna e a Ferramenta R

Esta fase consistiu na pesquisa de implementações de métodos de otimização moderna para a ferramenta R. Esta fase foi bastante importante porque teve impacto nas decisões a serem tomadas durante o desenvolvimento do pacote. A pesquisa de pacotes ou implementações em R foi feita durante as análises a métodos de otimização pelo que no final foi feita uma matriz onde se pode analisar os métodos de otimização e sua correspondência, de forma a perceber se tem código já implementado em R, se existe algum pacote com determinado método ou se simplesmente ainda não existe o método codificado para a linguagem R. Como o objetivo desta dissertação é desenvolver um pacote para a ferramenta R e não desenvolver implementações de algoritmos de otimização para a ferramenta R, quantos mais algoritmos ou pacotes de otimização existirem, mais completo será o pacote a desenvolver.

Na Tabela 1 é possível é possível analisar o estado em que se encontravam os diversos métodos de otimização em termos da sua implementação na ferramenta R.

Tabela 1 - Implementação de métodos de otimização na ferramenta R

Método	**Implementação na Ferramenta R**
Simulated Annealing	Pacote - GenSA
	Função - optim
	Código em R do livro *"Modern Optimization with R"*
Tabu Search	Pacote - tabuSearch
	Pacote - bnlearn
	Código em R do livro *"Modern Optimization with R"*
Genetic and Evolutionary Algorithms	Pacote - GA
	Pacote - mco
	Pacote - genalg
	Código em R do livro *"Modern Optimization with R"*
Full Blind Search	Código em R do livro *"Modern Optimization with R"*
Grid Search	Código em R do livro *"Modern Optimization with R"*
	Pacote - NMOF
	Pacote - optgrid {heR}
	Pacote - pi0
Monte Carlo	Código em R do livro *"Modern Optimization with R"*
	Pacote - mcmc
	Pacote - fme
	Pacote - mc2d
Hill climbing	Código em R do livro *"Modern Optimization with R"*
	Pacote - bnlearn
Diferential Evolution	Código em R do livro *"Modern Optimization with R"*
	Pacote - Deoptim
	Pacote - RcppDE
Particle Swarm Optimization	Pacote - PSO
	Código em R do livro *"Modern Optimization with R"*
	Pacote - hydroPSO
Estimated of Distribution Algorithm	Pacote - copulaedas
	Código em R do livro *"Modern Optimization with R"*
Genetic Programming	Pacote - RGP
	Código em R do livro *"Modern Optimization with R"*
Ant Colony	Pacote - irace
Bee Algorithm	Pacote - ABCoptim
Firefly Algorithm	Não existe pacote ou código em R

Método	**Implementação na Ferramenta R**
Cuckoo Algorithm	Não existe pacote ou código em R
The bat algorithm	Não existe pacote ou código em R
Harmony Search	Não existe pacote ou código em R
The flower Algorithm	Não existe pacote ou código em R
Scatter Search	Não existe pacote ou código em R
Adaptive Random Search	Não existe pacote ou código em R
GRASP	Não existe pacote ou código em R
Grammatical Evolution	Pacote - gramEvol
Gene Expression Programming	Não existe pacote ou código em R
Learning Classifier System	Não existe pacote ou código em R
Non-dominated Storting Genetic Algorithm	Pacote - nsga2R
Extremal Optimization	Pacote - modMax
Cultural Algorithm	Pacote - SubCultCon
Memetic Algorithm	Pacote - Rmalschains
Population Based Incremental Learning	Não existe pacote ou código em R
Univariate Marginal Distribuition Algorithm	Pacote - copulaedas
Bayesian Optimization Algorithm	Não existe pacote ou código em R
Cross Entropy Method	Pacote - Ceoptim
Bacterial Foraging Optimization Algorithm	Não existe pacote ou código em R
Clone Selection Algorithm	Não existe pacote ou código em R
Negative Selection Algorithm	Não existe pacote ou código em R
Artificial Immune Recognition System	Não existe pacote ou código em R
Immune Network Algorithm	Não existe pacote ou código em R
Dendritic Cell Algorithm	Não existe pacote ou código em R

Com este mapeamento foi possível verificar que existe um elevado número de métodos de otimização moderna e que já estão disponíveis para execução na ferramenta R. Contudo estes métodos estão dispersos por diferentes pacotes, funções, implementações e modos de utilização por parte do utilizador. Tal aspeto torna-se a motivação principal deste trabalho de dissertação. De facto, um utilizador que quisesse utilizar diversos métodos de otimização moderna na ferramenta R tinha o seu trabalho dificultado, uma vez que os métodos de otimização que se encontram dispersos por diversas funções, pedaços de código ou pacotes, sendo assim exigida uma maior curva de aprendizagem, bem como um esforço adicional de homogeneização das diferentes implementações. De referir ainda que foram encontrados mais algoritmos que não se encontram no livro *"Modern Optimization with R"* de Paulo Cortez (2014).

3. ABORDAGEM METODOLÓGICA E FERRAMENTAS UTILIZADAS

Este capítulo consiste em descrever as metodologias que serão utilizadas no desenvolvimento deste projeto. Para a investigação na área e devido a esta se encontrar na área de Sistemas de Informação, será utilizada a metodologia Design Science Research Methodology (DSRM) for Information Systems (Vaishnavi & Kuechler, 2004), enquanto na fase de desenvolvimento será utilizada a metodologia de desenvolvimento Prototipagem Rápida.

Este capítulo encontra-se dividido em duas secções, sendo que em cada uma delas se encontram detalhadamente descritas as etapas de desenvolvimento deste projeto.

3.1 *Design Science Research Methodology*

Design Science Research Methodology (DSRM) *for Information Systems* é uma metodologia que oferece técnicas e perspetivas para avaliação e iteração de projetos de investigação na área de Sistemas de Informação. Na Figura 8 pode-se verificar o seu ciclo de vida:

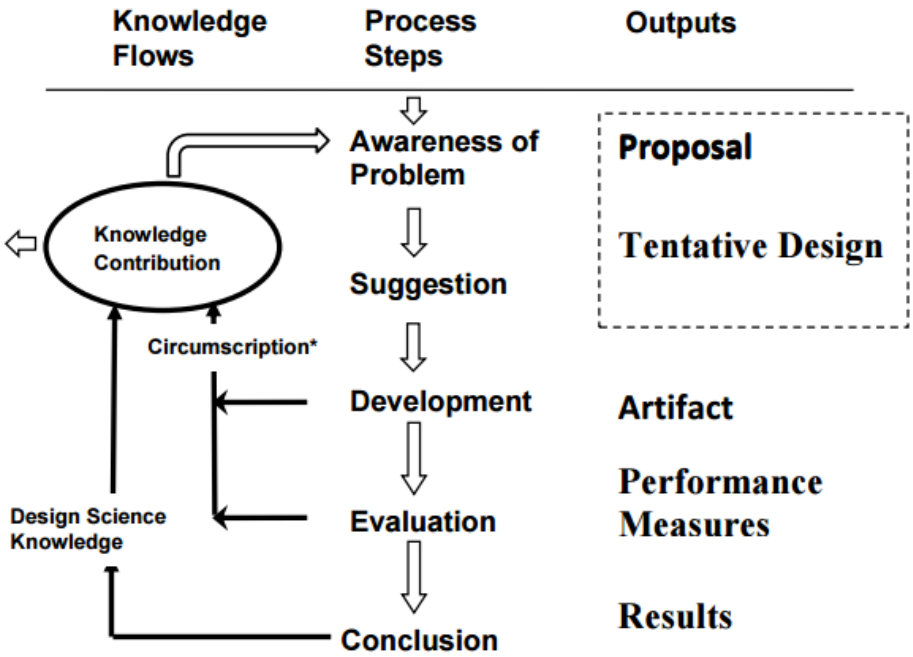


Figura 8- Modelo de processos da Metodologia de *Design Science Research Methodology*

Esta metodologia encontra-se dividida por 5 passos. Passos estes que serão descritos de seguida (Vaishnavi & Kuechler, 2004):

1. *Awareness of Problem* (Reconhecimento do Problema): A origem do problema pode surgir de diversas formas, desde evoluções tecnológicas ou através de uma área

específica. Esta fase vai dar como resultado, a proposta, esta que pode surgir de forma formal ou informal que vai dar origem ao início de uma nova investigação. Esse passo já foi abordado no Capítulo 2, e consistiu na não existência de um único pacote da ferramenta R que integre de modo coerente diversos métodos de otimização moderna.

2. *Suggestion* (Sugestão): A sugestão surge logo após a proposta, que foi onde surgiu a primeira ideia para o protótipo, ideia esta que será melhorada e evoluída na próxima fase.
3. *Development* (Desenvolvimento): nesta fase o protótipo foi desenvolvido e posteriormente implementado. A fase de desenvolvimento e implementação envolveu a metodologia de Prototipagem Rápida. Para esta fase foi necessário o uso de *software* de desenvolvimento, que foi o RStudio. O que resultou desta fase foi um artefacto que neste caso é o pacote `rmo` para a ferramenta R.
4. *Evaluation* (Avaliação): após a criação do protótipo é feita a avaliação deste, seguindo um conjunto de métricas de definidas e de questionários, para verificar se este vai ao encontro dos requisitos impostos na proposta inicial. Será também aqui em que é decidido se o protótipo é aceitável e pode ir ao encontro da resolução do problema detetado ou se será necessário repetir todo o processo.
5. *Conclusion* (Conclusão): esta fase é o final da pesquisa e desenvolvimento. Também consoante os resultados da fase anterior, pode levar a que se volte a fase inicial para corrigir os desvios e falhas detetadas na fase anterior ou é encontrada uma resposta ao problema detetado inicialmente. No final serão formuladas todas as conclusões relativas a este projeto de pesquisa e desenvolvimento.

3.2 Prototipagem Rápida

Para o desenvolvimento deste pacote que tem como objetivo simplificar as tarefas de otimização utilizando a ferramenta R, foi utilizada uma metodologia de desenvolvimento *software*, que serve para garantir que o projeto segue um conjunto de práticas que o conduzem a um desenvolvimento constante e correto do pacote. A metodologia escolhida chamada de Prototipagem Rápida ou em Inglês, *Rapid Prototyping*. A escolha desta metodologia de desenvolvimento foi sugerida pelo orientador.

Esta metodologia consiste na criação de um protótipo que vai sendo evoluído constantemente, sendo que nunca se sabe quando irá haver a versão final do produto, apenas a versão atual. Durante o desenvolvimento, esta metodologia permite um rápido desenvolvimento do produto, permitindo ao programador e utilizadores finais, efetuar testes de um modo mais rápido, de forma a perceber se o que está a ser feito vai ao encontro do estabelecido nos requisitos (Gordon & Bieman, 1995). As evoluções consistem em modificações das versões anteriores ou mesmo na criação de uma nova implementação.

Como tal esta metodologia trás vantagens como:

- Envolvimento com o utilizador final;
- Os utilizadores têm uma melhor perceção de como o trabalho está a ser executado;
- *Feedback* constante por parte dos utilizadores;
- Facilidade na mudança de requisitos e
- Erros podem ser detetados muito mais cedo.

Na Figura 9 pode-se analisar o ciclo de vida do processo de criação do pacote para a ferramenta R. Traduzido e adaptado: (http://www.freetutes.com/systemanalysis/sa2-prototyping-model.html):

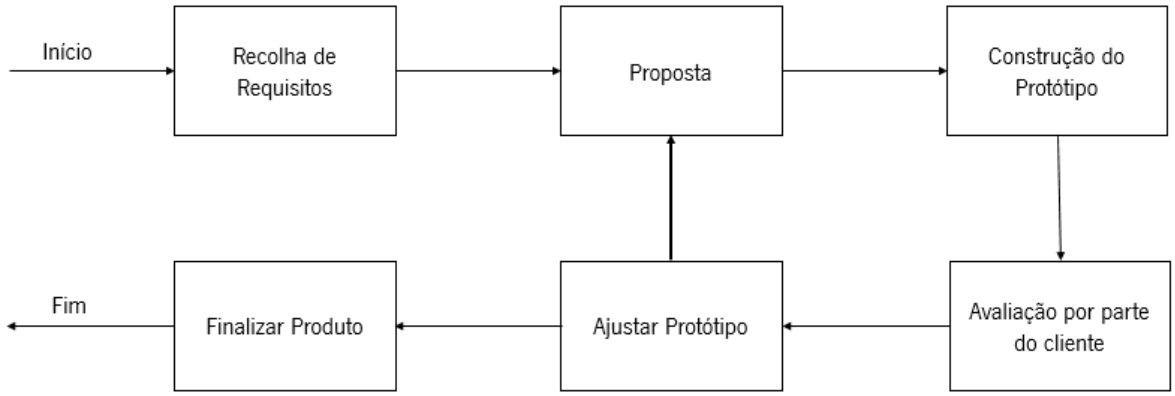


Figura 9 - Ciclo de vida da metodologia Prototipagem Rápida

3.3 Ferramenta R

A ferramenta R (R Core Team, 2015) é uma linguagem de programação para análise estatística de dados e elaboração de gráficos. Esta linguagem apresenta diversas características uteis tais como (Venables & Smith, 2011):

- Gestão de dados e armazenamento;
- Cálculos de *arrays* e em particular matrizes;
- Várias ferramentas para análises de dados;
- Facilidade de análise de dados através de gráficos;
- Versão melhorada da linguagem de programação "S", onde inclui ciclos, condições, funções recursivas e facilidade de inserção e exportação de dados;

Sendo esta uma linguagem *open-source,* esta encontra-se disponível para diversos sistemas operativos, como o *Windows, MacOS, Linux*, entre outros. A utilização desta ferramenta torna-se bastante simples por não ser necessário compilação de código durante a sua utilização, sendo assim uma linguagem interpretada e não compilada e dando uma resposta mais imediata ao seu utilizador.

Normalmente o uso desta ferramenta é feito através de uma consola, o que por vezes requere uma maior curva de aprendizagem. Esta ferramenta está se a tornar o veículo para novos métodos de análise de dados. (Venables & Smith, 2011).

O uso desta ferramenta pode ser expandido através da instalação de packages. De base, a ferramenta é mais orientada para um uso estatístico. Contudo, a ferramenta R permite a criação e instalação de packages que possam ser para orientados para determinadas outras áreas não convencionais de análise de dados, tais com Otimização, *Data Mining* e Simulação.

Estes packages são criados por utilizadores, que depois são colocados no repositório CRAN, que já contem cerca de 9000 *packages* disponíveis para *download*.

3.4 RStudio

A utilização do *Integrated Development Environment* (IDE), RStudio (RStudio Team, 2015), foi sugerida pelo livro *"R Packages"* (Wickham, 2015), que foi o livro que se acompanhou para apoiar no desenvolvimento do pacote `rmo`.

Este *software* de desenvolvimento *open-source,* que contém uma interface bastante simples e intuitiva ao contrário da consola R. Além de conter um editor para desenvolver código R, este IDE fornece mais funcionalidades para o utilizador tal como:

- Criação de pacotes;
- Criação de *scripts* em R;
- Criação de aplicações com interface gráfica *Shinny;*
- Integração com a plataforma *GitHub e*
- Facilidade na criação de gráficos.

Como tal considerou-se que este seria o IDE ideal para o desenvolvimento do pacote para a ferramenta R.

4. PACOTE RMO

Neste capitulo, é descrita a analise dos requisitos (funcionais e não funcionais) definidos para o pacote rmo. Após essa descrição dos requisitos, é descrito todo o processo de desenvolvimento do pacote rmo, desde os testes para familiarização com o processo de desenvolvimento e pacotes, bem como a descrição da implementação de cada método de otimização, tal como os problemas encontrados durante a implementação dos mesmos. E feita também uma descrição de como funciona o processo de criação da documentação para o pacote rmo. São também indicados os pacotes que foram e continuam a ser necessários para o bom funcionamento do pacote rmo. De forma a perceber como é efetuada a instalação deste pacote é demonstrada a forma como é feita a instalação do mesmo e também a descrição do repositório GitHub, que é onde o pacote se encontra disponível para os utilizadores o poderem descarregar para o utilizar. Por fim na ultima secção é feita uma descrição das funcionalidades e são demonstrados alguns exemplos de execução do pacote rmo.

4.1 Análise de Requisitos

Para o desenvolvimento do pacote `rmo`, foram estudados os requisitos que se pretendiam com a criação deste pacote. Estes foram sendo alterados durante as iterações da fase de desenvolvimento do pacote utilizando a metodologia de prototipagem rápida.

4.1.1 Requisitos Funcionais

Para requisitos funcionais foram definidos os seguintes:

- Implementação do número máximo de métodos possíveis;
- Possibilidade de resolver problemas de otimização inteira, binária ou real;
- Permitir que o utilizar possa definir os parâmetros de *lower* e *upper* através de um numero real ou através de um vetor de números;
- Criação do parâmetro de controlo que aceita parâmetros originários de pacotes implementados em pacotes dependentes;
- Demonstração do funcionamento do pacote através da invocação da função "`demo("rmo");` *e*
- Disponibilização de um manual de utilização do pacote.

4.1.2 Requisitos Não Funcionais

Para requisitos não funcionais forma definidos os seguintes:

- Usabilidade: o pacote terá de ser de fácil utilização para um utilizador com pouca experiencia na área de otimização moderna;
- Portabilidade: capacidade de executar em várias plataformas (e.g. Windows, Mac, Linux);
- Flexibilidade: o pacote deverá permitir a adição de novas funcionalidades que possam vir a necessárias no futuro.

4.2 Desenvolvimento do pacote

O processo de desenvolvimento deste pacote foi efetuado seguindo o livro "R-Packages" de Hadley Wickham (2015). Este livro forneceu o conhecimento necessário e que era inexistente até a altura de desenvolvimento do pacote. Portanto, foram seguidos os passos indicados no livro que permitiram a criação do pacote `rmo`, seguindo boas práticas de desenvolvimento de pacotes para a ferramenta R.

O desenvolvimento deste pacote foi sustentado num conjunto de pacotes e implementação de métodos de otimização moderna já existentes para a ferramenta R. Como já foi referido em capítulos anteriores, este pacote foi criado para facilitar o uso de métodos de otimização moderna através da agregação de vários métodos de otimização moderna num só pacote e também facilitar o uso desses mesmos métodos.

A criação deste pacote foi feita na plataforma *Microsoft Windows 10* e seguindo as práticas do livro "R-Packages" de Hadley Wickham (2015) que, como tal, para uma fácil criação do pacote, utilizou-se o IDE sugerido, neste caso o RStudio. Este IDE facilitou a criação da estrutura de pastas de pacotes automaticamente através da opção de criação de pacote. A ferramenta RStudio também se tornou bastante útil pelas funcionalidades a seguir descritas:

- Criação de pacotes
- Escrita de código R;
- Realização de verificação de erros de conceção do pacote;
- Possibilidade de fazer o *"build"* do pacote, tarefa esta que consiste na criação de um ficheiro com formato .tar.gz, que permite que se pudesse instalar o pacote em qualquer sistema operativo e
- Integração com a plataforma *GitHub* para disponibilização *on-line* e também para um melhor controlo de versões.

Os métodos escolhidos para implementar no pacote foram aqueles encontrados no livro *"Modern Optimization With R"* de Paulo Cortez (2014), pelo que as razões para tal implementação foi o facto de o Livro ter sido escrito pelo orientador, por já ter código R que facilitou a implementação e por ter sido acordado com o orientador que seriam estes os métodos a serem implementados no pacote `rmo`. Também foi sugerido pelo orientador a implementação do método *"Artificial Bee Colony"* (ABC)

proveniente do pacote ABCoptim que não se encontrava no livro. Alguns métodos de otimização moderna surgiram através de implementações existentes em R e outros métodos surgiram através de pacotes com apenas um método implementado, já disponíveis na ferramenta R.

Aproveitou-se também este IDE para proceder ao conhecimento do funcionamento dos métodos de otimização moderna. Para tal seguiu-se o Livro *"Modern Optimization With R"* de Paulo Cortez (2014), que continha código de funcionamento dos métodos de otimização. Esta fase foi útil porque ficou-se a perceber quais eram os parâmetros de entrada para o funcionamento destes métodos, ajudando assim a chegar a um consenso para saber quais os parâmetros de entrada para a função "`rmoptim`" do pacote `rmo`. É importante referir que grande parte dos parâmetros de entrada são iguais aos que serão implementados nas funções a executar. Assim inicialmente ficou definido que os parâmetros necessários para a função "`rmoptim`" seriam os seguintes:

- Método – Parâmetro de texto, onde é indicado o método de otimização;
- D – Parâmetro de valor inteiro, que indica o numero de valores a ser otimizado;
- *Lower* – Parâmetro de valor real, para indicação do valor mínimo para otimização;
- *Upper* – Parâmetro de valor real, para indicação do valor máximo para otimização;
- Tipo – Parâmetro de texto, que serve para indicar o tipo de otimização a realizar;
- Função – Indicação de qual é a função objetivo;

De referir que o parâmetro "Método", é considerado o elemento diferenciador para a escolha do método de otimização a ser executado. Este parâmetro teria diferentes opções tal como:

- "`fbs`" – *Full Blind Search;*
- "`grs`" – *Grid Search;*
- "`msc`" – *Monte Carlo Search;*
- "`sann`" – *Simulated Annealing;*
- "`abc`" – *Artificial Bee Colony;*
- "`tabu`" – *Tabu Search;*
- "`dfev`" – *Differential Evolution;*
- "`pso`" – *Particle Swarm Optimization;*
- "`eda`" – *Estimation of Distribution Algorithm;*
- "`gea`" – *Genetic and Evolutionary Algorithms e*
- "`hlc`" – *Hill Climbing.*

Inicialmente foi criado um pacote para testes, um pacote que desse para perceber como é o seu funcionamento e também para se ter alguma familiarização com o processo de criação de pacotes para a ferramenta R. Portanto no IDE RStudio foi criado um pacote. O pacote inicialmente quando é criado já contém implementado uma função chamada "`helloworld`" que retornava uma mensagem do tipo texto dizendo *"Hello World!"*. O facto de ter, esta função já vir declarada no pacote, deu para perceber o seu

funcionamento e permitiu que pudesse fazer alterações para testar o comportamento do pacote as mudanças efetuadas. Também ofereceu a possibilidade para que se pudesse explorar como é feita a documentação do pacote e proceder à escrita de alguma documentação para verificar como seria o resultado final. Foi verificado também como iria aparecer escrita no ficheiro PDF *(Portable Document Format)* a documentação oficial e também como apareceria no browser quando se executava o comando "`?helloworld`". Sempre que existiu uma alteração no pacote, para testar se a alteração tinha sido bem-sucedida, este era instalado na máquina em que foi desenvolvido o pacote, através do pacote devtools utilizando o comando "`devtools::load_all()`", que permitia que se pudesse efetuar a instalação do mesmo com um esforço mínimo. Também era instalado o pacote numa máquina virtual com o sistema operativo Windows XP, para se poder testar se o pacote funcionava da forma esperada. De salientar também que, nesta máquina virtual, a versão da ferramenta R era diferente da máquina onde foi desenvolvido o pacote. Esta foi uma fase que se tornou importante, para obter algum conhecimento de como é feito o desenvolvimento de pacotes para a ferramenta R.

Finalizado o processo de familiarização de criação de pacotes, procedeu-se à criação do pacote final. Para tal, foi necessário dar um nome ao pacote. Em conjunto com o orientador, ficou decidido que o nome do pacote seria "`rmo`", significando as siglas *"R Modern Optimization".* Após a criação da estrutura de pastas do pacote, procedeu-se ao desenvolvimento do código em R, código este que é fulcral para a utilidade do pacote e para o seu funcionamento. O código R encontra-se na pasta /R da estrutura do pacote. Esta foi, sem duvida, uma das fases que mais tempo requereu, devido a erros que existiram, testes que foram executados, implementação de métodos de otimização no pacote e também novas funcionalidades que com o tempo foram sendo adicionadas.

Agora será efetuada uma descrição detalhada de como foram implementados os métodos de otimização no pacote `rmo`. A ordem pela qual será descrita a implementação é a mesma ordem pela qual foi feita a implementação no pacote `rmo`. Também é importante referir, que para testar o seu funcionamento, foi necessário recorrer ao problema de otimização dos sacos, encontrado no livro *"Modern Optimization With R"* (Cortez, 2014).

4.2.1 *Monte Carlo Search*

A fase de iniciação à escrita de código R para o pacote iniciou-se com a implementação de um método de otimização, neste caso foi o método *"Monte Carlo Search",* porque foi considerado um dos métodos de mais fácil implementação e utilização.

Para a implementação deste método de otimização, foi necessário recorrer a um ficheiro de *script* em código R para a sua execução. Esta *script* continha apenas uma função chamada "`mcsearch`" que permitia a execução do método *"Monte Carlo Search"*. Surgiu então um problema na sua execução.

Numa implementação normal, seria carregada uma *script* usando o comando "`source()`", no entanto, num pacote é diferente a forma como tal é feito. Num pacote não é necessário, porque todas as funções estão disponíveis para a sua execução sem a necessidade de usar o comando "`source()`". Para tal, as funções tiveram que ser declaradas como objetos para poderem ser acessíveis através do pacote.

Após tal alteração no código da *script* do método *"Monte Carlo Search",* já foi possível executar o método sem qualquer problema, portanto foi considerado implementado o método *"Monte Carlo Search".*

4.2.2 *Hill Climbing*

A escolha da implementação deste método, foi por haver uma grande similaridade na forma de implementação com o método *"Monte Carlo Search"*. Como tal, este método também requeria um ficheiro *script* que permitia o seu funcionamento. *Script* esta que continha uma função que era chamada "`hclimbing`". De referir que alguns parâmetros de entrada são semelhantes aos parâmetros de entrada da função "`rmoptim`" e parte deles tiveram que ser inseridos numa lista para se poder efetuar a sua execução. Após esta operação, a implementação encontrava-se realizada com sucesso.

4.2.3 *Particle Swarm Optimization*

Posteriormente à implementação dos dois métodos de otimização anteriormente referidos, foi feita a implementação do método *"Particle Swarm Optimization"*. A escolha para implementação deste método foi por ser considerado um método de fácil utilização e, mais importante ainda, por ser um método que requeria um pacote para o seu funcionamento. O pacote necessário era o pacote pso (Bendtsen, 2015). Para efetuar a chamada do pacote apenas foi necessário colocar a seguinte linha de código:

```
library("pso")
```

Após isso, apenas foi necessário efetuar a chamada do método "`psoptim`". E toda a implementação foi efetuada sem qualquer tipo de problema.

4.2.4 *Full Blind Search*

A implementação deste método requereu algum esforço, pois foi necessário perceber como é o seu funcionamento, que consiste em definir o espaço de soluções. Após a sua compreensão e adaptação, foi efetuada a implementação do método. Para tal também foi necessário um ficheiro *script,* que continha a função "`fsearch`" para o seu funcionamento.

4.2.5 *Genetic and Evolutionary Algoriths*

Para a implementação deste algoritmo foi necessário recorrer ao pacote genalg (Ballings, 2015). Este pacote contem duas funções para proceder à otimização. A primeira função permite otimizar todo o tipo de problemas. A segunda função permite que se possa efetuar a otimização para problemas do tipo binário. Como tal, foi escolhido efetuar a implementação do método de otimização para qualquer tipo de problemas de otimização.

Após a escolha, foi feita a implementação da função "`rbga`". Esta função foi adaptada de um modo simples e sem terem sido encontrados problemas técnicos.

4.2.6 *Simulated Annealing*

Este foi um dos métodos mais fáceis de implementar, visto que não foi necessário nenhum pacote ou implementação em R para proceder a sua implementação porque este já é um método que vem de base com a ferramenta R. A função que contem este método é a função "`optim`", que permite que possam ser executados vários métodos, mas em especial o método SANN (Simulated Annealing). Como exemplo seguido pelo livro *"Modern Optimization with R"* (Cortez, 2014), foi feita a implementação usando uma função chamada "`hchange`" para alterar o comportamento da sua execução.

Também, tal como o método *"Hill Climbing",* foi necessário uma lista com os parâmetros necessários para a sua execução.

4.2.7 *Differential Evolution*

Este método é oriundo do pacote DEoptim (Ardia, Mullen, & Peterson, n.d.), que representa o método de otimização *"Differential Evolution"*. A sua implementação foi bastante fácil de efetuar, porque era semelhante ao método *"Particle Swarm Optimization".*

A utilização deste método por este pacote tinha uma particularidade bastante interessante. Para se adicionar os parâmetros de entrada e configuração da execução era necessário utilizar uma função. A função a ser utilizada era "`DEoptim.control`", que permitia que se pudesse configurar de uma forma muito fácil e organizada a sua execução.

Para se efetuar a otimização propriamente dita, foi necessário utilizar a função "`DEoptim`" e indicar a configuração anteriormente referida.

4.2.8 *Tabu Search*

A implementação deste método foi considerada uma das mais trabalhosas e com um maior grau de dificuldade. O funcionamento do método *"Tabu Search"* requereu o uso do pacote tabuSearch (Katarina, 2015), em que o seu funcionamento é através de valores binários. Ou seja, isto iria limitar bastante o seu uso, pelo que não se poderiam executar problemas de otimização real ou inteira.

Para tal, seguindo a implementação do método *"Tabu Search"* encontrada no livro *"Modern Optimization with R"* (Cortez, 2014), foi necessário a criação de funções auxiliares que permitissem o seu funcionamento para problemas de otimização inteira. Esta necessidade levou a que houvesse uma grande mudança na função "`rmoptim`". Essa mudança foi a implementação de uma variável chamada "`control`", que consistia numa lista em que era colocado um parâmetro chamado "`numb`" que iria ter como possíveis valores "`real`" para otimização real, "`bin`" para otimização binária e "`int`" para otimização inteira. A razão para tal implementação era para evitar que, quando o utilizador tivesse a necessidade de executar problemas de otimização binária, executasse as funções auxiliares para a execução de problemas inteiros. Assim apenas era utilizada a função "`tabuSearch`".

Na necessidade de resolução de um problema de otimização inteira, foi necessário proceder a implementação de duas funções. A primeira era chamada "`bintbin`" que efetuava a conversão de valores binários para valores inteiros, em cada valor inteiro continha 10 bits. Também foi necessário a criação de uma função objetivo auxiliar chamada "`beval`". A razão para a criação desta função, foi para ser inserida como referencia para a execução do método "`tabuSearch`", em que a função "`beval`" recebia os valores gerados pela função "`tabuSearch`" e efetuava uma conversão para inteiros invocando a função "`bintbin`" e após isso era feita a avaliação por parte da função objetivo inicialmente inserida pelo utilizador. De referir que todo este processo era "invisível" ao utilizador que estava a executar a função "`rmoptim`".

4.2.9 *Grid Search*

O método *"Grid Search"* foi de fácil implementação após o surgimento do parâmetro "`control`", inserido no método *"Tabu Search"* porque, devido ao seu tipo de funcionamento, era necessário a indicação do "salto" que a pesquisa iria fazer para se descobrir uma nova solução. Como tal foi definido um novo parâmetro chamado "`step`" em que iria conter um valor do tipo real. Para a execução deste método, foi definido que seria obrigatório para o utilizador definir o parâmetro "`step`", caso contrário a sua execução não iria avante. Este método contem duas funções para a procura da solução, a função "`gsearch`" e a função "`gsearch2`", e ambas são executadas para descobrir a solução ótima.

4.2.10 *Estimation of Distribution Algorithm*

Este método foi de fácil implementação através do pacote copulaedas (Veda, 2015), visto que não tinha muitas alterações a serem feitas para a sua implementação. Importante também referir que, de todos os métodos de otimização, este foi o único que não sofreu alterações na sua implementação para o seu funcionamento. Posteriormente serão descritas as alterações e implementações de funcionalidades que levaram o pacote `rmo` a ser mais completo.

4.2.11 *Artificial Bee Algorithm*

A implementação deste método foi sugerida pelo orientador, visto que este mesmo método não se encontrava no livro *"Modern Optimization with R"* (Cortez, 2014). Este método de otimização é oriundo do pacote ABCoptim (George, Yon, & George, 2015). A sua implementação foi considerada bastante simples por causa da sua similaridade com o pacote pso.

Assim, após a implementação dos métodos anteriormente referidos, considerou-se que foi realizada uma iteração do ciclo de desenvolvimento da metodologia Prototipagem Rápida.

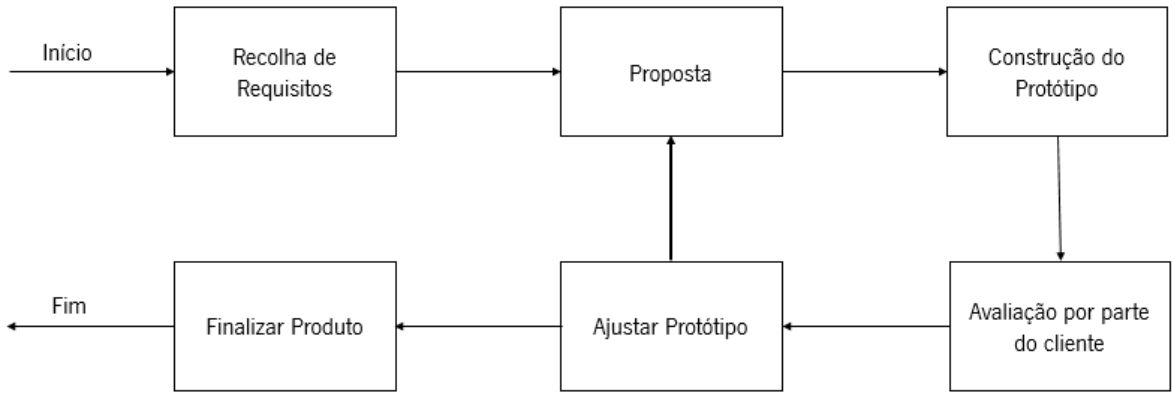


Figure 10 - Ciclo de Vida da metodologia Prototipagem Rápida

De referir também que cada iteração a nível do desenvolvimento do pacote não significava que este tinha erros, mas sim também poderia ser uma sugestão de novas funcionalidades ou alterações no pacote `rmo`.

Como tal, um dos requisitos inicialmente propostos encontrava-se solucionado. Já se encontravam vários métodos de otimização moderna implementados no pacote `rmo` e permitiam uma fácil utilização por parte de um utilizador com pouca experiencia na utilização de métodos de otimização moderna. No entanto, considerou-se que este tipo de execução era pouco completo para o utilizador.

Portanto, foi efetuada uma análise detalhada ao pacote `rmo`, para se perceber quais seriam os problemas existentes quando um utilizador utilizasse o pacote. Nesta análise foram utilizados alguns problemas de otimização também para testar o seu funcionamento e perceber o comportamento do pacote durante o seu funcionamento. Como, tal foram encontrados alguns problemas, que serão descritos de seguida.

O primeiro problema encontrado foi sobre o funcionamento dos parâmetros *lower* e *upper.* O que se sucedia neste parâmetro foi o facto de apenas aceitar valores do tipo real, isto é, para o seu funcionamento, o utilizador apenas indicava o valor real para toda a otimização. O que se pretendia era que o utilizador pudesse indicar qual o valor máximo e o valor mínimo para cada dimensão de valores a ser otimizados. Este permitia um funcionamento mais completo para o utilizador e explorava mais as funcionalidades dos métodos de otimização implementados no pacote, visto que era possível realizar tal coisa com vários métodos implementados no pacote.

O segundo problema encontrado foi o facto de não haver um retorno de uma solução. Isto é, havia situações em que era feita a execução do método, mas nem era retornado qualquer tipo de solução, ou se era retornado, apenas eram soluções do próprio método de otimização. Este também era um problema porque, o pacote retornava soluções umas vezes mais detalhadas outras vezes menos detalhadas.

De seguida, outro problema encontrado foi devido ao tipo de otimização a realizar. Na utilização do parâmetro "`control`" para otimizações do tipo inteiro ou binário, não era feito o arredondamento dos valores que eram retornados pela solução.

Após a identificação dos problemas encontrados procedeu-se a resolução dos mesmos. Iniciou-se pela resolução do segundo problema identificado, por ser algo "estético" e, uma vez que poderia vir a ajudar a perceber se o funcionamento do pacote estava correto, iniciou-se por este. Como tal, foi efetuada uma análise aos métodos que imprimiam na consola a solução encontrada. Após tal análise, foi efetuada a verificação de que tal forma se poderia retirar a mensagem que vinha de forma automática nos métodos de otimização. Chegou-se à conclusão que os métodos de otimização na sua configuração continham

um parâmetro que se chamava "trace", que aceitava valores como "TRUE" ou "FALSE". Assim caso o valor estivesse configurado para "FALSE" o método já não retornava qualquer mensagem. Posto isto, depois foi só efetuar a implementação da funcionalidade em que cada método imprimia uma mensagem igual para qualquer fosse o método que se estivesse a executar, assim permitia uma informação igual para qualquer método. Na mensagem retornada, está descrito os parâmetros otimizados e valor da função de otimização. Importante referir também que caso o utilizador desejasse poderia guardar em variáveis os dados retornados pela lista que contem os parâmetros otimizados e o valor da função de otimização.

Com esta implementação surgiu a ideia de colocar no parâmetro de controlo, na sua lista, um parâmetro chamado "trace" que aceitava valores como "TRUE" ou "FALSE".

No excerto de código seguinte é possível verificar as implementações efetuadas no pacote a nível da representação da solução:

```
# Example of the representation of the solution
> rmoptim("pso",5,1,1000,"max",profit)
Particle Swarm Optimization
Optimum Solution:  414 373 420 393 420  -  43622

# Example of the trace parameter in the control list
# In red, the information provided comes from the pso package
> rmoptim("pso",5,1,1000,"max",profit,list(trace=TRUE))
Particle Swarm Optimization
S=14, K=3, p=0.1993, w0=0.7213, w1=0.7213, c.p=1.193, c.g=1.193
v.max=NA, d=2234, vectorize=FALSE, hybrid=off
It 10: fitness=-42180
It 20: fitness=-42970
It 30: fitness=-43320
It 40: fitness=-43360
It 50: fitness=-43410
It 60: fitness=-43580
It 70: fitness=-43720
It 80: fitness=-43720
It 90: fitness=-43720
It 100: fitness=-43720
Maximal number of iterations reached
Optimum Solution:  401 416 424 393 417  -  43720

# Example of the variable storing the solution
> a=rmoptim("pso",5,1,1000,"max",profit)
Particle Swarm Optimization
Optimum Solution:  414 373 420 393 420  -  43622

> a
$sol
[1] 401 417 408 388 413

$fun
[1] 43622
```

No caso do primeiro problema, iniciou-se a execução de uns testes em métodos de otimização para se perceber se era possível definir de forma personalizada os valores de *lower* e *upper* para diferentes dimensões a serem otimizadas. Esse facto, na realidade, foi possível porque os métodos de otimização implementados no pacote facilitam esse tipo de abordagem à resolução de problemas. Para tal, a solução passou por alterar o pacote para aceitar nos parâmetros de *lower* e *upper* valores do tipo vetor, ao contrário do que acontecia em que apenas aceitava valores reais. Assim já era possível uma melhor personalização na procura pela solução ótima. No seguinte excerto de código será exemplificado uma execução onde é implementado apenas um valor real para os parâmetros de *lower* e *upper* e um exemplo onde serão implementados vetores para uma diferenciação na solução.

```
# Example:
# rmoptim(method,D,lower,upper,type,fun)
# First exemple running with only a real value for
# the lower and upper parameters

rmoptim("pso",5,1,1000,"max",profit)
Particle Swarm Optimization
Optimum Solution:  414 404 403 427 394  -  43622

# Second exemple running with vectors to define the maximum
# and the minimum solution.
# usage:
# “rep(1000,5)” is the same as “c(1000, 1000, 1000, 1000, 1000)”
rmoptim("pso",5,rep(1,5),rep(1000,5),"max",profit)
Particle Swarm Optimization
Optimum Solution:  414 404 403 427 394  -  43622
```

Finalmente, para a resolução do ultimo problema encontrado, que consistia em arredondar os valores da solução final para problemas do tipo binário ou inteiro, foi fácil resolver tal problema. Consistiu apenas em efetuar a chamada da função “`round()`”, que efetuava o arredondamento da variável, na representação da solução final e no *output* da solução após a execução do mesmo. Importante referir que, caso o utilizador não indique como é feito o tipo da representação, este assume sempre valores inteiros, e como tal no final é retornada uma solução inteira. No seguinte excerto de código é possível verificar a forma como esta solução pode ser retornada:

```
# Implementation to return the solution in real numbers
# It can be done by setting the numb parameter as real
> rmoptim("pso",5,1,1000,"max",profit,list(numb="real"))
Particle Swarm Optimization
Optimum Solution:  397.2206 403.5408 440.942 372.7113 372.3438  -  43623
```

```
# Implementation to return the solution in integer values
# It can be done by setting the numb parameter as int
> rmoptim("pso",5,1,1000,"max",profit,list(numb="int"))
Particle Swarm Optimization
Optimum Solution:  413 404 406 392 420  -  43791
```

Assim, após esta fase, foi considerado que foi feita mais uma iteração do ciclo de desenvolvimento da metodologia Prototipagem Rápida. No entanto foram efetuados testes utilizando problemas de otimização real e foi possível verificar que não era possível realizar a otimização no método *"Full Blind Search".* O problema neste método, durante a sua execução, era o retorno de soluções do tipo "NA". A conclusão que se chegou foi que, devido ao tipo de pesquisa que teria de ser efetuada, o espaço de procura iria ser demasiado grande e, como tal, era impossível fazer essa procura. A solução encontrada foi impedir que problemas de otimização real fossem executados usando o método *"Full Blind Search".*

Também foi encontrado um problema na implementação do método *"Grid Search",* em que este retornava um erro quando era feita a resolução de um problema de otimização binária, o espaço de otimização é demasiado elevado quando o numero de parâmetros a otimizar é grande, correspondendo a um *"Full Blind Search"*. Como tal foi discutido com o orientador este erro e optou-se por não permitir a execução do método *"Grid Search"* em problemas binários.

Visto que diversos métodos de otimização que se encontravam já implementados no pacote continham um parâmetro para indicar o numero de iterações para encontrar a solução ótima, procedeu-se à adaptação do código para que na lista "`control`" fosse possível adicionar o parâmetro "`iter`". Este parâmetro apenas aceitava valores do tipo inteiro e representava o numero máximo de iterações efetuadas durante a execução do método para encontrar a solução ótima. No caso de o utilizador não indicar um valor para o numero de iterações a efetuar, este assume sempre o valor de 100 iterações.

Finalmente, considerou-se que o pacote `rmo` estava a funcionar sem qualquer tipo de erros, de uma forma bastante eficaz. Neste momento também se considerou que o pacote estava já pronto para uma disponibilização a vários utilizadores.

No entanto, durante uma das reuniões com o orientador, foi possível verificar que também existia um problema na implementação do método *"Tabu Search"*. Devido ao seu funcionamento, em que eram utilizadas funções auxiliares para a conversão de valores inteiros para valores binários e vice-versa, a função impedia que fossem utilizados métodos de conversão de valores binários para valores reais. Como tal, procedeu-se a uma alteração no modo de funcionamento do método para este conseguir efetuar uma otimização completa de valores reais. Então, a função de otimização começou a trabalhar com

dimensões de oito *bits* cada. Para a execução do problema foi necessário criar uma função chamada "`generateSample`", que simplesmente gerasse a solução inicial em formato, adaptada com o numero de *bits* por dimensão. Após isso, foi necessário também a criação de uma função chamada "bin2real" que efetuasse a divisão de cada dimensão de oito *bits* para um valor único e após isso efetuar a conversão da dimensão extraída para um valor real. Após a conversão de todas as dimensões para tipo real é só inciar a avaliação da mesma solução através da função objetivo. Assim desta forma, já era possível ter o método *"Tabu Search"* a funcionar de forma completa para qualquer tipo de valores.

Após as alterações anteriormente referidas terem sido efetuadas, como consequente uma finalização de uma iteração do ciclo de vida de desenvolvimento da metodologia Prototipagem Rápida, foi feita uma nova análise de requisitos ao pacote `rmo`. Como tal, um novo requisito foi criado, em que permitia que um utilizador mais experiente pudesse alterar os parâmetros de funcionamento dos métodos de otimização que se encontravam implementados no pacote `rmo`. A razão para poder haver a possibilidade de tal alteração é o facto de a execução de um método é sempre feita com parâmetros com valores por defeito.

Para exemplificar de uma forma prática, no método de otimização *"Genetic and Evolutionary Algorithms",* utilizando o pacote genalg é possível definir o tamanho da população alterando o parâmetro "`popsize`" para o valor desejado. Este valor encontrava-se por defeito definido como 200. No seguinte excerto de código é possível verificar como é feita a sua implementação, alterando o valor do "`popsize`" para 300.

```
> rbga(stringMin=rep(1,5), stringMax=rep(1000,5), popSize=300, iters=100,
evalFunc=profit)
```

Para que tal pudesse ser alterado na utilização da função "`rmoptim`", aproveitou-se o facto de a função já conter um parâmetro "`control`", parâmetro este que era uma lista que aceitava vários parâmetros anteriormente implementados (e.g.: "`trace`", "`step`" e "`iter`"). Importante referir que, caso o utilizador deseje alterar algum parâmetro de um método de otimização apenas tem de verificar qual é o nome do parâmetro na documentação oficial do método de otimização, e colocar o parâmetro a ser alterado na lista de "`control`". Assim que o parâmetro se encontre na lista de "`control`", este é aceite durante a execução da função "`rmoptim`" e a sua execução é feita tendo em conta o valor já implementado.

De seguida é mostrado um exemplo de código onde se pode verificar a alteração do valor "`popsize`", utilizando a função "`rmoptim`":

```
# Implementation of a parameter from the original package “genalg”
# The default value was 200 and its being set to 300
> rmoptim("gea",5,1,1000,"max",profit1,list(popsize=300))
Genetic and evolutionary Algorithms
Optimum Solution:  414 390 408 413 394  -  43870
```

Nota que, caso o utilizador deseje alterar o valor do número de iterações, é possível realizá-lo através do parâmetro “`iter`” ou mesmo através do parâmetro oficial do método que está selecionado para efetuar a sua execução.

Após mais uma iteração do ciclo de desenvolvimento do pacote, surgiu mais um requisito de forma a tornar o pacote mais completo e mais robusto. Requisitos estes que eram verificações de tipos nos parâmetros de entrada da função “`rmoptim`”:

- **Método –** Aceitar apenas valores do tipo texto;
- **D –** Este parâmetro aceita apenas valores do tipo inteiro;
- ***Lower e Upper –*** Este paremetro aceita valores do tipo real e vectores;
- **Tipo –** Aceitar apenas valores do tipo texto. E apenas valores a conter “max” ou “min”;
- **Controlo –** Aceitar apenas uma lista neste parâmetro.

No final da implementação de verificações aos parâmetros de entrada da função “`rmoptim`”, foi necessário configurar o ficheiro “NAMESPACE”. Este ficheiro serve para indicar quais são as funções oriundas de pacotes que criam dependência ao pacote `rmo` para a execução destes mesmos métodos. Para a configuração deste ficheiro, só é necessário indicar qual é o pacote e quais são as funções necessárias para o seu funcionamento. Por fim também é necessário indicar qual é a função que está disponível por parte do pacote `rmo`. No seguinte exemplo é demonstrado a configuração do ficheiro “NAMESPACE”, onde é indicado que o pacote pso que importa a função “`psoptim`”. Ex:“`importFrom("pso","psoptim")`”.

No final desta configuração, considerou-se que o pacote se encontrava pronto para ser disponibilizado para ser testado por vários utilizadores.

4.3 Documentação

Na estrutura de um pacote para a ferramenta R, um dos aspetos mais importantes é a sua documentação, pois é nela onde o utilizador vai perceber como pode tirar o máximo partido da utilização do pacote. A documentação também é um aspeto bastante importante para o programador ou futuros

programadores, porque permite que este tenha ao longo do tempo um suporte de informação sobre o que se desenvolveu.

A ferramenta R fornece uma forma padrão para efetuar a documentação das funções do pacote. A escrita da documentação consiste em escrever ficheiros com a extensão *.Rd* que se encontram na pasta *man/* do pacote. A sintaxe para a escrita destes ficheiros é muito próxima da linguagem LaTeX, que posteriormente geram HTML, texto e o PDF para o utilizador poder consultar.

Através da leitura do livro *"R Packages"* (Wickham, 2015), no capitulo de explicação de como criar a documentação para o pacote, este sugeriu a utilização do pacote roxygen2, que utiliza comentários que estão devidamente identificados no código do pacote e utiliza esses comentários para efetuar a criação dos ficheiros *.Rd.* O objetivo na utilização do pacote roxygen2, é simplificar ao máximo a criação de documentação para o pacote. Importante referir que a alteração de código no pacote não implica também uma alteração na documentação. A documentação só era gerada quando era executado o comando "`devtools::document()`", em que o pacote devtools e roxygen2 funcionavam em sintonia para efetuar a geração da documentação.

No exemplo seguinte pode-se verificar um excerto de código em que se pode verificar como é a sintaxe dos comentários para o pacote roxygen2. Este exemplo faz a escrita do campo descrição do método "`rmoptim`":

```
#' R Modern Optimization function
#' @description A vast number of real-world tasks can be viewed as an optimization
#' problem, where the goal is to minimize or maximize a given goal. This package
#' encompasses in a single and simply to use function (rmoptim) several modern
#' optimization algorithms (metaheuristics) that are spread across distinct

#' packages.
#' Examples of implemented methods are: grid search, hill climbing, tabu search,
#' simulated annealing, differential evolution, estimation of distribution
#' algorithms, genetic and evolutionary algorithms, particle swarm optimization
#' and artificial bee colony.
```

Após a escrita da documentação é possível verificar como é que se encontra a geração do código para o ficheiro *.Rd* encontrado na pasta *man/.*

```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/main.R
\name{rmoptim}
\alias{rmoptim}
\title{R Modern Optimization function}
\description{
```

```
A vast number of real-world tasks can be viewed as an optimization problem, where
the goal is to minimize or maximize a given goal. This package encompasses in a
single and simply to use function (rmoptim) several modern optimization algorithms
(metaheuristics) that are spread across distinct packages. Examples of implemented
methods are: grid search, hill climbing, tabu search, simulated annealing,
differential evolution, estimation of distribution algorithms, genetic and
evolutionary algorithms, particle swarm optimization and artificial bee colony.
}
```

E por fim, é possível verificar o resultado final quando se executa o comando "`?rmoptim`":

rmoptim {rmo} R Documentation

R Modern Optimization function

Description

A vast number of real-world tasks can be viewed as an optimization problem, where the goal is to minimize or maximize a given goal. This package encompasses in a single and simply to use function (rmoptim) several modern optimization algorithms (metaheuristics) that are spread across distinct packages. Examples of implemented methods are: grid search, hill climbing, tabu search, simulated annealing, differential evolution, estimation of distribution algorithms, genetic and evolutionary algorithms, particle swarm optimization and artificial bee colony.

Figure 11 - Exemplo do campo descrição do método rmoptim

Também nos comentários para serem gerados pelo pacote roxygen2, haviam informações como os parâmetros para a função "`rmoptim`", exemplos de funcionamento, referências, e descrição sobre valores de retorno. A utilização do pacote roxygen2 foi considerada uma mais valia no desenvolvimento do pacote, porque permitiu que se efetuasse a documentação de uma forma mais rápida e fácil, ao invés de ter que fazer um estudo de como é feita a documentação à "mão" para a ferramenta R e posteriormente ter que ter esse trabalho de escrita.

Todos os pacotes para a ferramenta R contem um ficheiro chamado "DESCRIPTION". Este é um ficheiro bastante importante no pacote porque é onde contêm informação sobre o tipo do pacote, o contacto da pessoa que o criou, a descrição do pacote, versão, a licença para o seu uso, entre outros.

No exemplo seguinte é possível verificar estrutura do ficheiro "DESCRIPTION" do pacote `rmo`:

```
Package: rmo
Type: Package
Title: R Modern Optimization Package
Version: 0.1.0
Author: Joao Carlos Torres Arantes Maia
Maintainer: Joao Maia <joao.arantes.maia@gmail.com>
Description:A vast number of real-world tasks can be viewed as an optimization
problem, where the goal is to minimize or maximize a given goal. This package
```

```
encompasses in a single and simply to use function (rmoptim) several modern
optimization algorithms (metaheuristics) that are spread across distinct packages.
Examples of implemented methods are: grid search, hill climbing, tabu search,
simulated annealing, differential evolution, estimation of distribution algorithms,
genetic and evolutionary algorithms, particle swarm optimization and artificial
bee colony.
License: GPL-2
LazyData: TRUE
LazyLoad: Yes
RoxygenNote: 5.0.1
Date: 2016-08-20
Imports: ABCoptim, DEoptim, copulaedas, genalg, pso, tabuSearch, methods
URL: https://github.com/joaoctm123/rmo
BugReports: https://github.com/joaoctm123/rmo/issues
```

4.4 Pacotes necessários

Para um funcionamento completo e uma maior capacidade de escolha por parte do utilizador na altura de escolher métodos de otimização para a sua execução, foi necessário recorrer a dependência de pacotes. Outros também foram necessários para o desenvolvimento e criação de uma forma mais simples da documentação do pacote. Pacotes que são:

- **devtools** – este pacote foi necessário para facilitar o desenvolvimento do pacote. Com este pacote não foi necessário o uso complexo de comandos para o desenvolvimento do pacote. Com uma simples chamada de funções do pacote, permite que se possa, por exemplo, documentar de forma automática o pacote ou mesmo carregar para a memória do computador alterações que tenham sido efetuadas no pacote a ser desenvolvido;
- **roxygen2** – um pacote muitíssimo importante para a criação da documentação. A documentação do pacote é um dos aspetos mais importantes num pacote, porque é este que ajuda o utilizador final a perceber o funcionamento do pacote. Com a utilização deste pacote permitiu que durante, a criação da documentação fosse mais simples, visto que, através da escrita de comentários no código e com a execução de um comando, através do pacote devtools, é gerado automaticamente um código baseado em LaTeX que é escrito num ficheiro com a extensão .Rd que fornece posteriormente a documentação em HTML ou mesmo em PDF para o utilizador;
- **tabuSearch** – Pacote necessário para o uso do método *Tabu Search;*
- **genalg** – Pacote necessário para a utilização do método *Genetic and Evolutionary Algorithms;*
- **Copulaedas** – Pacote necessário para a utilização do método *Estimation of Distribution Algorithm;*
- **DEoptim** – Pacote necessário para o uso do método *Differential Evolution;*
- **ABCoptim** – Pacote necessário para o uso do método *Artificial Bee Algorithm*;
- **pso** – Pacote necessário para o uso do método *Particle Swarm Optimization*.

Estes foram os pacotes instalados para a criação e funcionamento do pacote `rmo`. No entanto muitos dos pacotes anteriormente referidos necessitaram da instalação de pacotes para o seu funcionamento (e.g. o pacote devtools necessita da instalação dos seguintes pacotes: memoise, whisker, rstudioapi).

4.5 Instalação

A instalação do pacote `rmo` foi um aspeto bastante importante a ter em conta durante o desenvolvimento do pacote. O facto do pacote `rmo` ter dependências de outros pacotes criou alguns problemas, visto que o utilizador que iria utilizar o pacote `rmo` poderia não ter as dependências instaladas no seu ambiente R.

A forma como o pacote poderia vir a ser instalado e disponibilizado ao utilizador era através da disponibilização do pacote em formato *"Bundled",* em que seria fornecido um ficheiro com o formato .tar.gz, que possibilita que qualquer utilizador possa instalar o pacote em diferentes plataformas (e.g. Windows, Mac, Linux). O problema deste tipo de instalação é que, para esta ser bem-sucedida, era necessário o utilizador ter os pacotes dependentes já instalados na ferramenta R, caso tal não se verificasse, não era efetuada a instalação.

A outra forma adotada para a disponibilização do pacote `rmo` para os futuros utilizadores foi através da disponibilização no repositório *GitHub.* Esta foi a forma mais fácil para uma disponibilização do pacote para futuros utilizadores, visto que já resolvia o problema da instalação dos pacotes dependentes. A instalação poderia ser feita de uma forma muito simples, utilizando apenas as seguintes linhas de código:

```
install.packages("devtools")
devtools::install_github("joaoctm123/rmo")
library(rmo)
```

Através desta forma de instalação, já é feita a instalação das dependências do pacote `rmo`, resolvendo assim o problema inicial.

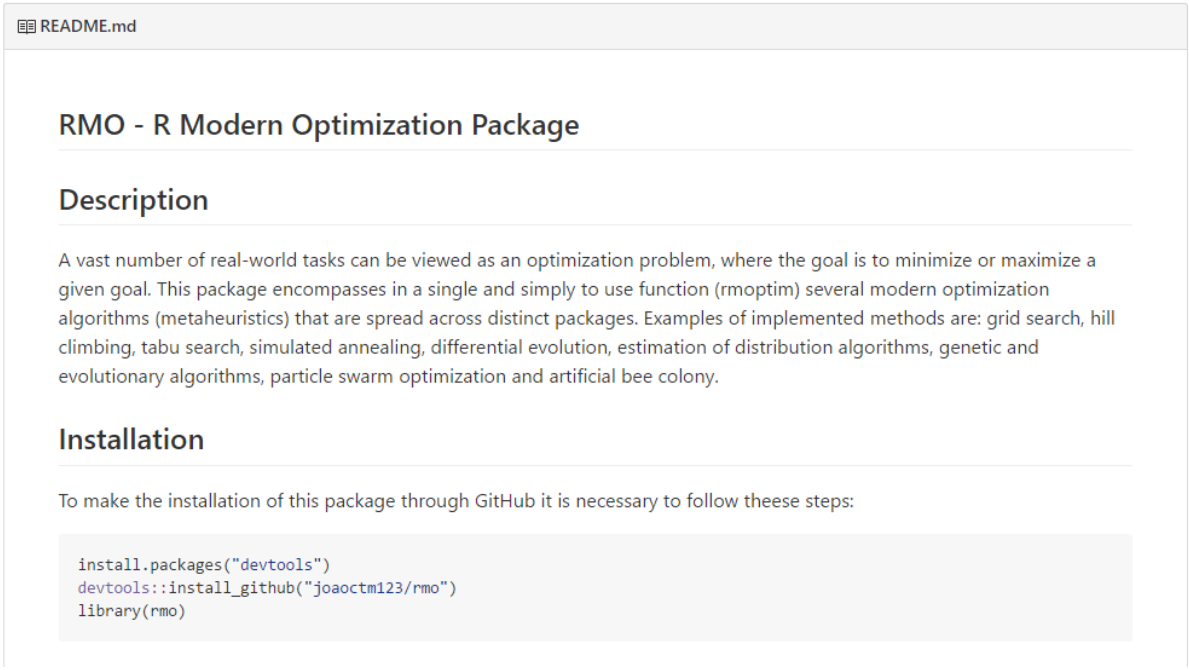
O pacote `rmo` é um pacote que está disponível para ser instalado em qualquer plataforma e foi testado em diferentes ambientes, isto é, testado em diferentes sistemas operativos tal como Windows 10/XP, Mac OS e em diferentes versões do R (3.2.3 - *"Wooden Christmas-Tree"* e 3.3.1 – *"Bug In Your Hair").*

4.6 *GitHub*

Após o desenvolvimento do pacote, foi necessário escolher um locar na Internet para poder disponibilizar o pacote a futuros utilizadores. Foi decidido então que o pacote iria estar disponível no repositório *GitHub.* A plataforma *GitHub* hoje em dia oferece muito mais que um simples local onde o código está guardado, esta plataforma pode ser vista como a página web onde se pode ter o projeto online e disponível para futuros utilizadores. A disponibilização do pacote neste repositório oferece diversas funcionalidades para o utilizador:

- Controlo de versões do pacote desenvolvido;
- Página disponível para o utilizador inserir problemas encontrados no pacote;
- Possibilidade de instalação do pacote de uma forma muito simples;
- Criação de um ficheiro *Readme;*
- Estatísticas sobre visitas no repositório;
- Estatísticas sobre os *downloads* do pacote e
- Criação de uma página *Wiki.*

Para a disponibilização do pacote no repositório, foi necessário efetuar um *commit* de todo o código do pacote para este estar disponível no repositório. *Commit* é quando é criada uma nova versão do que se tenha alterado ou adicionado no pacote. A disponibilização do pacote no repositório foi possível através do IDE, RStudio. Após a disponibilização, foi necessário editar o ficheiro "README.md". Foi colocado texto na Língua Inglesa porque, como este era o local oficial para o pacote `rmo` e este iria ser disponibilizado a várias pessoas de diversas nacionalidades, assim facilitava a sua compreensão para todas as nacionalidades de utilizadores. Após a edição do ficheiro "README.md", este fica logo disponível para leitura quando o utilizador acede a página do repositório do pacote. Na imagem seguinte é possível ver uma amostra do ficheiro "README.md":


README.md

RMO - R Modern Optimization Package

Description

A vast number of real-world tasks can be viewed as an optimization problem, where the goal is to minimize or maximize a given goal. This package encompasses in a single and simply to use function (rmoptim) several modern optimization algorithms (metaheuristics) that are spread across distinct packages. Examples of implemented methods are: grid search, hill climbing, tabu search, simulated annealing, differential evolution, estimation of distribution algorithms, genetic and evolutionary algorithms, particle swarm optimization and artificial bee colony.

Installation

To make the installation of this package through GitHub it is necessary to follow theese steps:

```
install.packages("devtools")
devtools::install_github("joaoctm123/rmo")
library(rmo)
```

Figure 12 - Descrição do pacote rmo na plataforma *GitHub*

De referir que este ficheiro contém diversas informações como a descrição do pacote, forma de instalação do pacote, utilização da função "`rmoptim`", valores retornados pela função "`rmoptim`", endereço para o manual do utilizador e contactos.

4.7 Funcionalidades Disponíveis e Exemplos de Funcionamento

Esta subsecção pretende demonstrar as funcionalidades existentes e o funcionamento do pacote `rmo` na ferramenta R. Serão apresentadas as funcionalidades que existem no pacote `rmo` e em cada funcionalidade explicada, será demonstrada a sua execução na ferramenta R.

O pacote `rmo` contém apenas uma função, a função "`rmoptim`". Durante o desenvolvimento do pacote pretendeu-se que esta fosse uma função poderosa a nível de funcionalidades, tanto para um utilizador iniciado como para um utilizador experiente.

Nos seguintes exemplos serão demonstradas funcionalidades da função "`rmoptim`", utilizando problemas de otimização inteira, real e binária.

O primeiro problema de otimização é um problema de otimização inteira, que se trata de uma fábrica que produz bolsas de senhora e que pretende definir qual o preço ótimo de cada um de cinco tipos diferentes bolsas de forma a poder-se maximizar o lucro. Nos pontos seguintes, são descritos os detalhes deste problema de otimização:

- O preço unitário de cada bolsa é: $u = (\$30, \$25, \$20, \$15, \$10)$;

- O custo de produção de uma bolsa é definido pela seguinte fórmula: $custo(xi) = 100 + ui\ x\ vendas(xi)$, onde i representa um bolsa diferente;
- Número de vendas é dependente do preço de venda (x) e pelo *marketing* aplicado (m). A fórmula é a seguinte: $vendas(xi) = round((1000/ln(xi + 200) - 141)\ x\ mi$;
- O esforço do *marketing* é retornado pela seguinte formula: $m = (2.0, 1.75, 1.5, 1.25, 1.0)$;
- Os preços de cada bolsa variam entre \$1 e \$1000;
- Formula final: $preço\ de\ venda = xi\ x\ vendas(xi) - custo\ (xi)$;

O problema seguinte para uma otimização de valores reais, trata-se da resolução da função *"rastrigin".* Este é um problema de minimização contendo duas dimensões e como valor de *lower* -5.2 e como valor de *upper* 5.2. A sua função é definida da seguinte forma:

$$f_{rastrigin}(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos 2\pi\ x_i + 10)$$

Por fim, para realizar demonstrações de problemas de otimização binária, a sua função consiste em efetuar um somatório dos bits da solução da solução inicialmente encontrada. A sua função é definida da seguinte forma:

$$f_{sum\ of\ bits}(x) = \sum_{i=1}^{D} x_i$$

Mais uma vez importante referir o funcionamento da função "`rmoptim`":

```
rmoptim(method, Dimensions, lower, upper, type, function, control)
```

No seguinte exemplo é possível verificar o funcionamento da função "`rmoptim`", utilizando o método de otimização *"Full Blind Search".*

```
> rmoptim("fbs",5,1,1000,"max",profit)
Full Blind Search
Optimum Solution:  414 404 408 413 395  -  43899
```

O método *"Full Blind Search"* não aceita execuções de problemas de otimização real, portanto será demonstrado um problema de otimização binária. De referir que, devido ao seu funcionamento, onde todo o espaço de soluções é definido, a solução ótima vai ser sempre encontrada, pelo que se pode verificar isso no exemplo seguinte:

```
> rmoptim("fbs",8,0,1,"max",sumbin)
Full Blind Search
Optimum Solution:  1 1 1 1 1 1 1 1  -  8
```

Para a execução do método de otimização *"Grid Search"* é necessário a definição do parâmetro step. Como tal, a sua definição é efetuada dentro da lista do parâmetro "`control`", pelo que tal se pode verificar no exemplo seguinte. O valor do salto nesta execução vai ser de 100.

```
> rmoptim("grs",5,1,1000,"max",profit,list(step=100))
Grid Search
Optimum Solution of gsearch method:  401 401 401 401 501  -  43142
Optimum Solution of gsearch2 method:  401 401 401 401 501  -  43142
```

No exemplo seguinte, será utilizado na mesma o algoritmo de otimização *"Grid Search",* mas desta vez para um problema de otimização real. Referir que foi necessário definir "numb" na lista de controlo como "real", porque no funcionamento do pacote caso o utilizador não indique o tipo de otimização que se está a realizar, este assume que é um problema de otimização inteira.

```
> rmoptim("grs",2,-5.2,5.2,"min",rastrigin,list(numb="real",step=1))
Grid Search
Optimum Solution of gsearch method:  -0.2 -0.2  -  13.89966
Optimum Solution of gsearch2 method:  -0.2 -0.2  -  13.89966
```

Na execução do método *"Monte Carlo Search",* será demonstrada uma funcionalidade que é bastante útil para o utilizador. Será a definição dos valores de *lower* e *upper* para diferentes dimensões a serem otimizadas. No seguinte exemplo de otimização inteira, será exemplificado como se pode definir o preço máximo de $100. A definição do parâmetro upper será efetuada da seguinte forma: "`c(100, 1000, 1000, 1000, 1000)`". E isso é possível verificar no seguinte exemplo:

```
> rmoptim("mcs",5,1,c(100,1000,1000,1000,1000),"max",profit)
Monte Carlo Search
Optimum Solution:  93 353 649 333 333  -  33797
```

Como se pôde verificar, a resolução limitou o preço para o primeiro saco no valor de $93. O mesmo poderia ser efetuado para qualquer dimensão a ser otimizada. Também é demonstrada a execução do problema de otimização real usando a função de otimização "`rastrigin`" e também é efetuada uma definição do numero de iterações através do parâmetro "`iter`" para a procura da solução.

```
> rmoptim("mcs",2,-5.2,5.2,"min",rastrigin,list(numb="real",iter=300))
Monte Carlo Search
Optimum Solution:  -0.01380449 1.015908  -  1.119764
```

O mesmo será demonstrado para a resolução de um problema de otimização binária. De referir que são definidas apenas duas iterações porque, caso fossem definidas mais, a procura iria encontrar a solução máxima e o que se pretende aqui é uma demonstração.

```
> rmoptim("mcs",8,0,1,"max",sumbin,list(numb="bin",iter=2))
Monte Carlo Search
Optimum Solution:  1 0 1 1 0 1 1 0  -  5
```

No exemplo seguinte será demonstrada uma das funcionalidades que é considerada mais importante para um tipo de utilizadores mais experientes no uso de métodos de otimização moderna. Para este exemplo será utilizado o método de otimização *"Particle Swarm Optimization"* pertencente ao pacote pso.

Esta funcionalidade consiste em alterar os parâmetros de funcionamento do método através da inserção desses mesmos parâmetros no parâmetro de controlo da função "`rmoptim`". Para um utilizador saber quais são os parâmetros que pode alterar para um determinado método, apenas tem que seguir a documentação oficial do método que esteja a utilizar. Essa documentação pode ser encontrada no repositório do CRAN, ou através da documentação oficial do pacote `rmo` que indica qual é a documentação oficial de um determinado método.

No próximo exemplo pretende-se alterar o parâmetro "`trace`" que apresenta mensagens oriundas do método de otimização, que assume valores lógicos, e o parâmetro "`maxit`" que se refere

ao numero de iterações. Também foi implementado o parâmetro "REPORT", que apresenta a mensagem num numero intervalado de iterações. Nota: Foi referido anteriormente a existência do parâmetro "iter" para indicar o numero de iterações que a execução iria ter. Como o parâmetro "maxit" e "iter" efetuam sempre o mesmo, o pacote assume sempre a preferência pelo parâmetro "maxit" no caso de ambos os parâmetros estarem declarados.

```
#First run with a simple configuration
> rmoptim("pso",5,1,1000,"max",profit)
Particle Swarm Optimization
Optimum Solution:  427 377 390 374 346  -  43688

#Second run with an advanced configuration
> rmoptim("pso",5,1,1000,"max",profit,list(maxit=150,trace=TRUE))
Particle Swarm Optimization
S=14, K=3, p=0.1993, w0=0.7213, w1=0.7213, c.p=1.193, c.g=1.193
v.max=NA, d=2234, vectorize=FALSE, hybrid=off
It 10: fitness=-43410
It 20: fitness=-43410
It 30: fitness=-43410
It 40: fitness=-43450
It 50: fitness=-43450
It 60: fitness=-43450
It 70: fitness=-43490
It 80: fitness=-43550
It 90: fitness=-43620
It 100: fitness=-43620
It 110: fitness=-43720
It 120: fitness=-43720
It 130: fitness=-43720
It 140: fitness=-43720
It 150: fitness=-43720
Maximal number of iterations reached
Optimum Solution:  425 377 425 433 418  -  43717

# Third exemple, now with the REPORT parameter
# REPORT set to 50, to show only messages between 50 iterations
> rmoptim("pso",5,1,1000,"max",profit,list(maxit=150,trace=TRUE,REPORT=50))
Particle Swarm Optimization
S=14, K=3, p=0.1993, w0=0.7213, w1=0.7213, c.p=1.193, c.g=1.193
v.max=NA, d=2234, vectorize=FALSE, hybrid=off
It 50: fitness=-43330
It 100: fitness=-43530
It 150: fitness=-43700
Maximal number of iterations reached
Optimum Solution:  401 417 390 409 371  -  43696
```

No seguinte exemplo é possível verificar o funcionamento do método *"Particle Swarm Optimization",* desta vez para problemas de otimização real utilizando a mesma configuração dos exemplos anteriores, usando a função "`rastrigin`". Mais uma vez convém salientar o facto de ser necessário declarar o tipo de otimização a executar porque, esta pode depois não produzir os resultados esperados.

```
>rmoptim("pso",2,-5.2,5.2,"min",rastrigin,list(maxit=150,trace=TRUE,REPORT=50,numb="real"))
Particle Swarm Optimization
S=12, K=3, p=0.2297, w0=0.7213, w1=0.7213, c.p=1.193, c.g=1.193
v.max=NA, d=14.71, vectorize=FALSE, hybrid=off
It 50: fitness=7.699e-05
It 100: fitness=6.34e-09
It 150: fitness=4.263e-14
Maximal number of iterations reached
Optimum Solution:  1.332977e-08 5.116739e-09  -  4.263256e-14
```

O seguinte exemplo pretende demonstrar o uso do método *"Artificial Bee Colony",* com todas as configurações possíveis para a execução da função "`rmoptim`". Irá ser definido a solução inicial para a procura se iniciar, através do parâmetro "`par`", será demonstrado uma atribuição de valores de *lower* e *upper* para diferentes dimensões e também a definição de o numero máximo de iterações, sendo este definido através do parâmetro "`maxcycle`".

```
>rmoptim("abc",5,c(50,1,100,50,50),rep(1000,5),"max",profit1,list(par=rep(NA,5),
maxcycle=30))
Artificial Bee Colony
Optimum Solution:  427 403 391 412 419  -  43801
```

E por fim é demonstrado a utilização do mesmo algoritmo de otimização, mas para os problemas de otimização real e otimização binária.

```
# Solution for the rastrigin problem. Real optimization problem
>rmoptim("abc",2,-5.2,5.2,"min",rastrigin,list(numb="real",maxcycle=30,
par=rep(1,2)))
Artificial Bee Colony
Optimum Solution:  2.566244e-10 -3.177584e-05  -  2.003173e-07

# Solution for the sumbits problem. Binary optimization problem
```

```
> rmoptim("abc",8,0,1,"max",sumbin,list(numb="bin", maxcycle=2,par=rep(0,8)))
Artificial Bee Colony
Optimum Solution:  1 1 1 1 1 1 1 1  -  8
# The previous solution reaches the maximum solution
```

Esta secção demonstrou as funcionalidades existentes na utilização da função "`rmoptim`", sendo que é possível manter o mesmo nível de configuração para diferentes métodos de otimização disponíveis no pacote `rmo`.

Outra funcionalidade disponível no pacote `rmo` é a função "`demo("rmo")`". Esta é uma função nativa na ferramenta R, que demonstra o funcionamento de um pacote que esteja instalado nesta mesma ferramenta. Foi considerado importante adicionar esta funcionalidade ao pacote porque, a par da documentação já existente, iria fornecer um grande suporte de informação para o utilizador sobre a utilização deste pacote. No seguinte exemplo é possível verificar como se invoca o funcionamento da função "`demo("rmo")`":

```
> library("rmo")
> demo("rmo")
       demo(rmo)
       ---- ~~~
Type  <Return> to start :
```

Após isso o utilizador só tem que carregar na tecla "ENTER" para poder continuar a visualizar mais exemplos de funcionamento do pacote `rmo`.

5. AVALIAÇÃO DOS RESULTADOS OBTIDOS

Este capitulo descreve a metodologia aplicada para a avaliação do pacote rmo, bem como os seus resultados da avaliação. Na Secção 5.1 é apresentada a metodologia de avaliação e na Secção 5.2 são apresentados os resultados obtidos da avaliação.

5.1 Metodologia de Avaliação

Após o desenvolvimento do pacote `rmo`, iniciou-se a fase de recolha de opinião sobre o pacote desenvolvido. Para tal, criou-se um breve questionário com 10 perguntas (sendo que este encontra-se no Anexo I). Pretendeu-se que este fosse um questionário breve para permitir um rápido preenchimento por parte dos utilizadores a quem este pacote foi disponibilizado. Após isso, foi disponibilizado o endereço link do repositório *GitHub* onde estava alojado o pacote a alguns contactos do professor Paulo Cortez, a alunos que frequentaram a Unidade Curricular de Sistemas Adaptativos para a Inteligência do Negócio (SAIN) em anos anteriores e também em vários grupos sobre a ferramenta R ou *Data Mining* da rede social *LinkedIn*, para estes testarem o pacote e posteriormente preencherem o questionário. Este questionário esteve disponível aproximadamente 3 semanas, desde 1/10/2016 a 20/10/2016.

Para a criação deste questionário optou-se por utilizar uma escala de *Likert* em grande parte das perguntas, com os valores possíveis de 1- Discordo Totalmente, 2 - Discordo, 3 – Sem Opinião, 4 – Concordo e 5 – Concordo Totalmente. Neste questionário pretendeu-se inicialmente perceber o perfil do utilizador que testou o pacote, após isso pretendeu-se saber qual era a opinião do utilizador sobre o funcionamento e utilidade do pacote `rmo`.

Este questionário foi disponibilizado na língua inglesa, para permitir um fácil preenchimento por parte de utilizadores estrangeiros. Estas foram as questões efetuadas:

1. *How old are you?*
2. *Where are you from?*
3. *What are your qualifications?*
4. *What is your opinion about this sentence: "I am an expert using the R tool"?*
5. *What is your opinion about this sentence: "I am an expert in using R to perform Modern Optimization tasks"?*
6. *What is your opinion about this sentence: "Overall, I believe I that RMO package is an interesting tool to perform Modern Optimization tasks"?*
7. *What is your opinion about this sentence: "The RMO package helps non specialized R users to perform modern optimization tasks"?*
8. *What is your opinion about this sentence: "The RMO package is easier to use than single method packages"?*

9. *Please report bugs/errors that you found when testing the RMO package.*
10. *Please write your suggestions to the RMO package*

Algumas das questões anteriormente apresentadas (e.g. 9 e 10) não podem ser analisadas de modo quantitativo, via análises estatísticas, uma vez que foram inseridas no questionário para se obter uma opinião qualitativa sobre o pacote `rmo`.

5.2 Resultados Obtidos

Nesta secção serão apresentados os resultados obtidos, relativo ás respostas dos utilizadores no questionário anteriormente apresentado.

No período em que o questionário esteve disponível, foi obtido um total de 9 respostas. Trata-se de um número reduzido, mas ainda assim considerado suficiente para a validação deste pacote. Os resultados obtidos estão apresentados na Tabela 2:

Tabela 2 - Frequencia de respostas ao questionário sobre o pacote rmo

Questão	Discordo totalmente	Discordo	Sem opinião	Concordo	Concordo totalmente
4	0	1	3	4	1
5	1	4	0	3	1
6	0	0	0	8	1
7	0	0	0	6	3
8	0	0	1	3	5
Total	1	5	4	23	11

Após uma breve análise aos resultados encontrados, pode-se verificar que em grande parte das perguntas, a resposta do inquirido é "Concordo" e "Concordo Totalmente", pelo que se pode tirar a conclusão de que em geral o *feedback* é bastante positivo. Em particular, realça-se a questão 5, que pergunta se o utilizador é uma pessoa especializada no uso da ferramenta R para executar tarefas de otimização, 55.5% dos inquiridos respondeu dizendo que discorda ou que discorda totalmente e os restantes 45.5% responderam de forma positiva, dizendo que concorda ou que concorda totalmente. Pelo que se pode concluir que talvez os utilizadores não se encontrem tão à vontade no uso deste tipo de ferramentas de otimização por causa da sua dificuldade no uso das mesmas, visto que requerem

algum conhecimento da área da programação. Nas restantes questões (questão 6, 7 e 8) sobre o pacote `rmo`, é possível verificar um *feedback* bastante positivo, o que é considerado um elemento satisfatório de validação do trabalho executado nesta dissertação. No âmbito da realização de tarefas de otimização, 8 dos 9 utilizadores concordam que o pacote `rmo` é bastante interessante para a resolução das mesmas tarefas de otimização. O que leva a crer que os utilizadores possam considerar o seu uso, visto que, também na questão 7, que questiona se o pacote `rmo` ajuda utilizadores não especializados a realizar tarefas de otimização, obtiveram-se respostas bastante positivas, onde 6 dos inquiridos concordam com esta afirmação e 3 concordam totalmente. Também se considera relevante o facto de se terem obtido respostas positivas quando é questionado se o uso do pacote `rmo` é mais fácil em relação a utilização de pacotes específicos para um algoritmo de otimização. Nesta questão obteve-se um *feedback* bastante positivo onde 5 dos questionados concorda totalmente com esta afirmação, 3 concordam e apenas 1 não tem opinião em relação a esta afirmação.

Na Figura 13 é possível verificar a que maior parte dos inquiridos se situa entre os 18 e os 25 anos, o que quer dizer que é possível que tenham sido ex-alunos da Unidade Curricular de SAIN, também devido a 7 dos 9 inquiridos serem de Portugal, o que é possível verificar na Figura 15, e também é possível concluir com base na Figura 14, grande parte dos inquiridos tem o grau de licenciatura, e esta Unidade Curricular pertence ao primeiro ano do Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação.

Em termos globais, considerando as respostas às questões 5, 6, 7 e 8, pode concluir-se que, embora amostra de questionários seja de pequena dimensão, o feedback é bastante positivo e revela que o pacote desenvolvido é pertinente, útil e contribui para facilitar a execução de tarefas de otimização na ferramenta R.

Importante também referir que não se obteve nenhuma resposta para a questão 9 e questão 10, onde não foi reportado nenhum erro ou bug e também não foram dadas quaisquer sugestões para o pacote `rmo`.

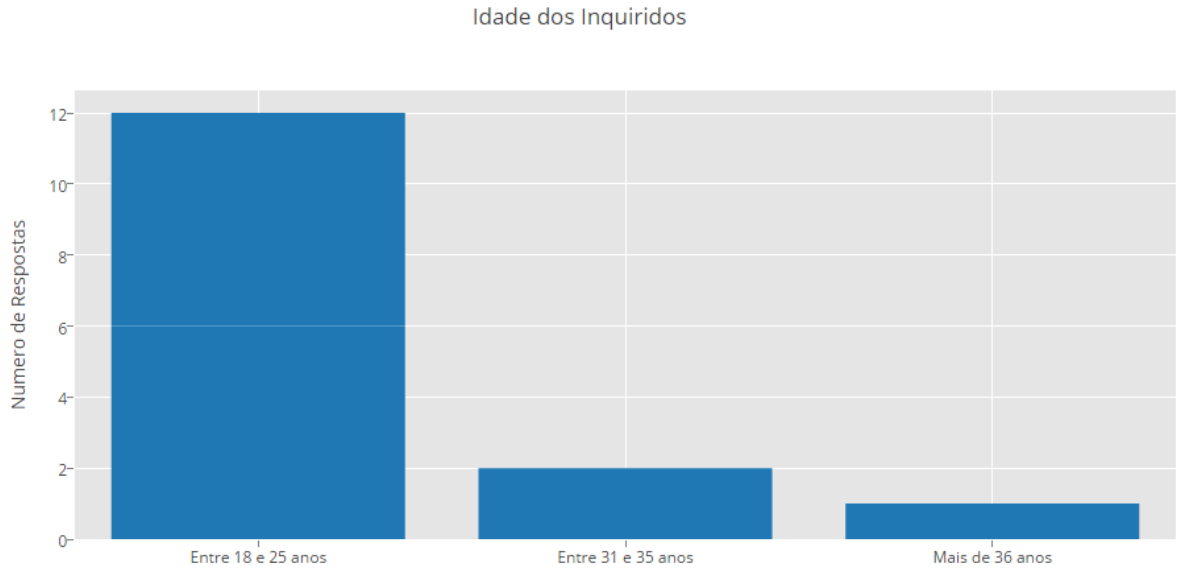


Figure 13 – Histograma com a idade dos inquiridos sobre o pacote rmo

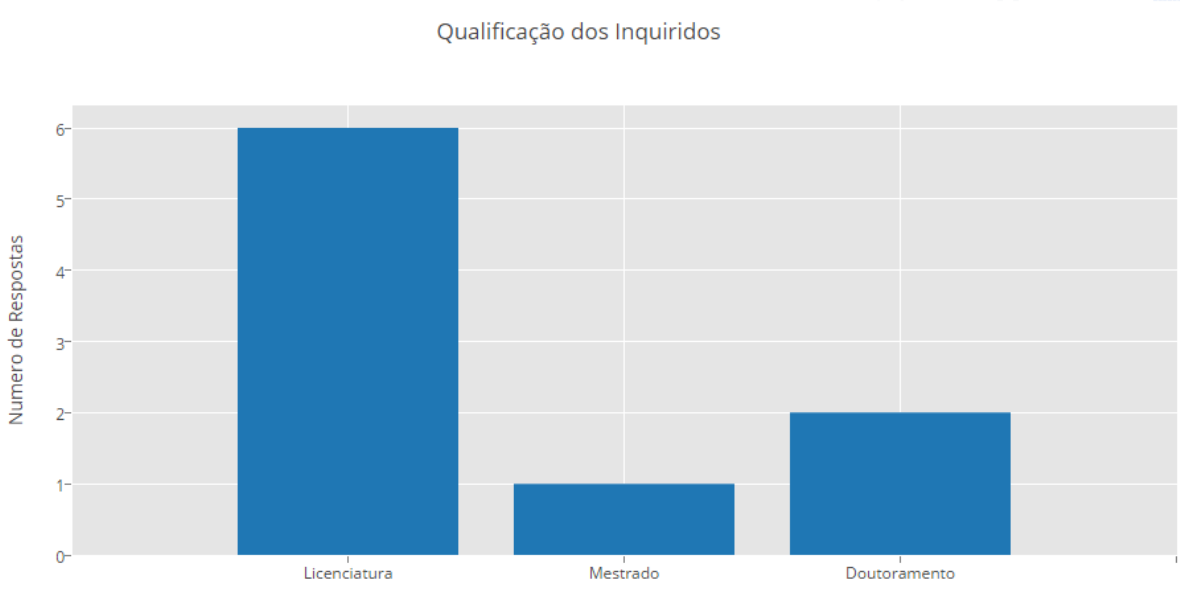


Figure 14 – Histograma com a qualificação dos inquiridos sobre o pacote rmo

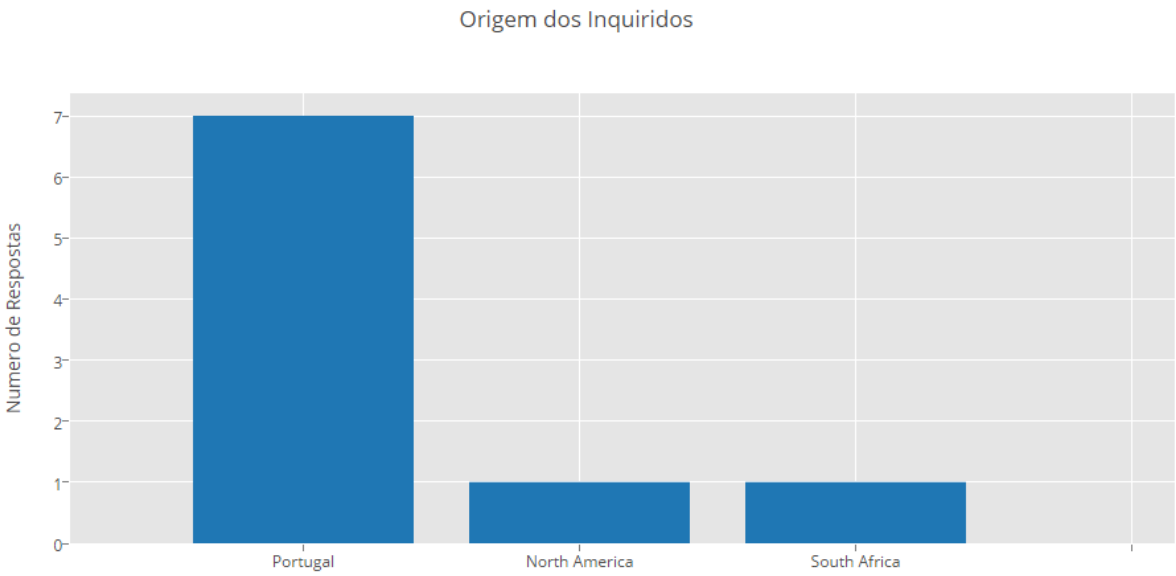


Figure 15 – Histograma com a origem dos Inquiridos

6. CONCLUSÃO

Este capitulo tem como objetivo apresentar uma síntese do trabalho realizado, bem como uma discussão do trabalho realizado e, por fim, uma secção sobre o possível trabalho futuro.

6.1 Síntese

Com o crescimento da capacidade computacional e com o nível de complexidade de problemas de otimização a aumentar, surgiu a necessidade de resolver estes problemas recorrendo a algoritmos de otimização. Sendo que a ferramenta R se encontra em grande expansão, considerou-se importante o desenvolvimento de um pacote para apoiar os utilizadores na utilização de métodos de otimização moderna.

Para o desenvolvimento desta dissertação foi necessário efetuar um levantamento de todos os algoritmos existentes para a ferramenta R, isto foi possível também com a ajuda do livro *"Modern Optimization with R"* (Cortez, 2014) que permitiu perceber o funcionamento dos diversos algoritmos de otimização moderna, bem como os pacotes aos quais diversos destes algoritmos estão dependentes.

Também a leitura do livro *"R Packages"* (Wickham, 2015) tornou-se um elemento importante para o desenvolvimento do pacote. Livro este que veio demonstrar as boas práticas para o desenvolvimento de pacotes na ferramenta R, permitindo que se pudesse poupar tempo do desenvolvimento, através do fornecimento do conhecimento, técnicas e truques para um bom e correto desenvolvimento de pacotes.

Foi necessário proceder a execução de testes para perceber de que forma se poderia facilitar o uso de algoritmos de otimização e perceber o seu funcionamento para facilitar ao máximo o trabalho do utilizador final. Com esta simplificação, pretendeu-se que esta fosse uma ferramenta útil para um utilizador com pouca experiencia, mas também para um utilizador com mais experiencia pudesse tirar o máximo de partido que os algoritmos de otimização são capazes de oferecer. Assim, foram acrescentados argumentos extra, com valores por omissão, à função principal que foi desenvolvida.

Também para apoiar o utilizador foi necessária a criação da documentação para "explicar" ao utilizador como poderia tirar partido do pacote `rmo`.

Por fim foi efetuado um questionário onde se pretendeu perceber qual a opinião de utilizadores reais sobre o pacote `rmo`.

6.2 Discussão

Nesta dissertação procedeu-se à criação de um pacote para a ferramenta R que facilitasse as tarefas de otimização, pelo que se considerou que esse objetivo foi cumprido. Foi assumido que a utilização de algoritmos de otimização moderna criava muitas dificuldades para utilizadores com pouca experiencia no uso deste tipo de algoritmos e ainda mais dificuldade àqueles que não possuem o conhecimento na linguagem e ferramenta R.

Com uma análise de requisitos efetuada inicialmente percebeu-se de uma forma mais fácil quais eram os problemas encontrados no uso de algoritmos de otimização, fazendo com que se criasse requisitos que fosse ao encontro da solução do problema. Para o desenvolvimento do pacote, é importante referir que a utilização do livro *"Modern Optimization with R"* (Cortez, 2014), foi um elemento importante para o desenvolvimento e implementação de métodos de otimização moderna no pacote, visto que permitiu a compreensão do funcionamento destes métodos e o código necessário para efetuar a sua implementação.

Para além da criação do pacote foi necessário obter uma opinião de utilizadores reais sobre o pacote `rmo`. De realçar que foram obtidas somente 9 respostas ao questionário que foi disponibilizado na *Internet*. Tal resultado é comum, em geral os utilizadores não estão particularmente interessados em responder a questionários, considerando muitas das vezes esse preenchimento uma perda de tempo. Por exemplo, na Universidade do Minho, a taxa de resposta aos questionários de avaliação ao ensino ministrado tem sido muito baixa nos últimos anos. Seja como for, foi o *feedback* que foi possível de obter durante a execução desta dissertação. De facto, só foi possível terminar o desenvolvimento do pacote em setembro de 2016, pelo que houve pouco tempo para divulgar o pacote, questionário e obter mais respostas. Uma forma de obter mais respostas seria demonstrar o pacote e pedir feedback em tempo real nas aulas de Sistemas Adaptativos para a Inteligência do Negócio. Contudo, esta unidade curricular só decorre no segundo semestre, pelo que fica para trabalho futuro esta possibilidade. Não obstante terem sido obtidas somente 9 respostas, e conforme já descrito no capítulo anterior, considera-se que o feedback obtido é positivo e permite validar de modo satisfatório o pacote rmo desenvolvido.

Deve-se também referir que este pacote foi criado do "zero", ou seja, grande parte do tempo consumido no desenvolvimento desta dissertação foi na criação do pacote `rmo` para a ferramenta R.

6.3 Trabalho Futuro

Como anteriormente indicado, o principal objetivo desta dissertação consistiu no desenvolvimento um pacote que facilitasse as tarefas de otimização para a ferramenta R, pelo que se considera que esse

objetivo foi cumprido. Dados os limites temporais associados a este trabalho, não foi possível abordar diversos pontos relevantes que se deixam aqui para trabalho futuro, nomeadamente:

- Implementação de uma interface gráfica;
- Disponibilização do pacote no repositório CRAN;
- Implementação de um sistema de análise dos resultados da otimização através de gráficos;
- Impressão de mais parâmetros de *output*;
- Obter um maior numero de respostas em questionários por parte de potenciais utilizadores;
- Adaptação do pacote para representar soluções hibridas, isto é, representação de soluções que envolvem uma mistura de vários tipos de solução, tais como Binário com Real, Inteiro com Binário, etc;
- Aplicar o pacote `rmo` em contextos reais; e
- Demonstrar e disponibilizar o pacote `rmo` a futuros alunos da Unidade Curricular de Sistemas Adaptativos para a Inteligência do Negócio.

REFERÊNCIAS BIBLIOGRÁFICAS

Ardia, D., Mullen, K. M., & Peterson, B. G. (n.d.). DEoptim : An R Package for Global Optimization by Differential Evolution, (2006).

Ballings, M. (2015). Package " genalg ."

Bash, E. (2015). *Nature Inspired Optimization Algorithms*. *PhD Proposal* (Vol. 1). http://doi.org/10.1017/CBO9781107415324.004

Bendtsen, C. (2015). Package "pso" Title Particle Swarm Optimization.

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239–287. http://doi.org/10.1007/s11047-008-9098-4

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, *2*(4), 353–373. http://doi.org/10.1016/j.plrev.2005.10.001

Blum, C., & Roli, a. (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 189–213. http://doi.org/10.1007/s10479-005-3971-7

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117. http://doi.org/10.1016/j.ins.2013.02.041

Brownlee, J. (2011). *Clever Algorithms*. *Search*. Retrieved from http://www.cleveralgorithms.com

Cortez, P. (2014). *Modern Optimization with R*. Guimarães: Springer.

Freitas, A. A. (2004). A Critical Review of Multi-objective Optimization in Data Mining: A Position Paper. *SIGKDD Explor. Newsl.*, *6*(2), 77–86. http://doi.org/10.1145/1046456.1046467

George, A., Yon, V., & George, M. (2015). Package "ABCoptim" Title Implementation of Artificial Bee Colony (ABC) Optimization. Retrieved from http://github.com/gvegayon/ABCoptim,

Gordon, V. S., & Bieman, J. M. (1995). Rapid Prototyping: Lessons Learned. *IEEE Software*, *12*(1), 85–95. http://doi.org/10.1109/52.363162

Huang, C., Lee, Y., Lin, D., & Huang, S. (2007). Model selection for support vector machines via uniform design. *Computational Statistics & Data Analysis*, *52*(1), 335–346. http://doi.org/10.1016/j.csda.2007.02.013

Katarina, D. (2015). Package " tabuSearch ," (c), 1–6.

Laguna, M. (n.d.). Chapter 8 SCATTER SEARCH, (1977), 1–13.

Larrañaga, P. (2002). *Estimation of Distribution Algorithms*.

Michalewicz, Z. (2008). Adaptive Business Intelligence, Computer Science Course.

Michalewicz, Z., Schmidt, M., Michalewicz, M., & Chiriac, C. (2006). *Adaptive business intelligence*. *Adaptive Business Intelligence*. http://doi.org/10.1007/978-3-540-32929-9

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm–a novel tool for complex optimisation problems. *Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006)*, 454–459. http://doi.org/http://dx.doi.org/10.1016/B978-008045157-2/50081-X

R Core Team. (2015). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.r-project.org/

RStudio Team. (2015). RStudio: Integrated Development Environment for R. Boston, MA: RStudio, Inc. Retrieved from http://www.rstudio.com/

Sean, L. (George M. U. (2012). *Essentials of Metaheuristics: A Set of Undergraduate Lecture Notes*. *Optimization*.

Vaishnavi, V., & Kuechler, B. (2004). Design Science Research in Information Systems Overview of Design Science Research. *Ais*, 45. http://doi.org/10.1007/978-1-4419-5653-8

Veda, R. C. R. (2015). Package " copulaedas ."

Venables, W., & Smith, D. (2011). An Introduction to R, *3*.
Wickham, H. (2015). *R Packages*. O'Reilly Media.
Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence*, *1*(1), 36. http://doi.org/10.1504/IJSI.2013.055801

ANEXO I – QUESTIONÁRIO DE FEEDBACK DO PACOTE RMO

RMO - Feedback Form

Please fill the attached form (10 questions, only 8 are mandatory).
This form is part of an MSc thesis project where the rmo package was implemented.
All answers are anonymous and will be treated for the purpose of user evaluation of the package.
The goal of the rmo package is to facilitate the use of modern optimization methods with the R tool by using a single function (rmoptim) and it is available at: https://github.com/joaoctm123/rmo

*Required

1 - How old are you? *

- ◯ Between 18 and 25 years old
- ◯ Between 26 and 30 years old
- ◯ Between 31 and 35 years old
- ◯ More than 36 years old

2 - Where are you from? *

- ◯ Portugal
- ◯ Europe
- ◯ Africa
- ◯ Asia
- ◯ North America
- ◯ South America
- ◯ Oceania

3 - What are your qualifications? *

- ○ No University Degree
- ○ Bachelor Course
- ○ Master of Science Degree
- ○ PhD Degree

4 - What is your opinion about this sentence: "I am an expert using the R tool"? *

- ○ Strongly disagree
- ○ Disagree
- ○ Neither agree neither disagree
- ○ Agree
- ○ Strongly agree

5 - What is your opinion about this sentence: "I am an expert in using R to perform Modern Optimization tasks"? *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neither agree neither disagree
- ◯ Agree
- ◯ Strongly agree

6 - What is your opinion about this sentence: "Overall, I believe I that RMO package is an interesting tool to perform Modern Optimization tasks"? *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neither agree neither disagree
- ◯ Agree
- ◯ Strongly agree

7 - What is your opinion about this sentence: "The RMO package helps non specialized R users to perform modern optimization tasks"? *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neither agree neither disagree
- ◯ Agree
- ◯ Strongly agree

8 - What is your opinion about this sentence: "The RMO package is easier to use than single method packages"? *

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neither agree neither disagree
- ◯ Agree
- ◯ Strongly agree

9 - Please report bugs/errors that you found when testing the RMO package.

Your answer

10 - Please write your suggestions to the RMO package

Your answer

ANEXO II – MANUAL DO PACOTE RMO

Package 'rmo'

October 7, 2016

Type Package

Title R Modern Optimization Package

Version 0.1.0

Author Joao Carlos Torres Arantes Maia

Maintainer Joao Maia <joao.arantes.maia@gmail.com>

Description A vast number of real-world tasks can be viewed as an optimization problem, where the goal is to minimize or maximize a given goal. This package encompasses in a single and simply to use function (rmoptim) several modern optimization algorithms (metaheuristics) that are spread across distinct packages. Examples of implemented methods are: grid search, hill climbing, tabu search, simulated annealing, differential evolution, estimation of distribution algorithms, genetic and evolutionary algorithms, particle swarm optimization and artificial bee colony.

License GPL-2

LazyData TRUE

LazyLoad Yes

RoxygenNote 5.0.1

Date 2016-08-20

Imports ABCoptim, DEoptim, copulaedas, genalg, pso, tabuSearch, methods

URL https://github.com/joaoctm123/rmo

BugReports https://github.com/joaoctm123/rmo/issues

R topics documented:

1

rmoptim *R Modern Optimization function*

Description

A vast number of real-world tasks can be viewed as an optimization problem, where the goal is to minimize or maximize a given goal. This package encompasses in a single and simply to use function (rmoptim) several modern optimization algorithms (metaheuristics) that are spread across distinct packages. Examples of implemented methods are: grid search, hill climbing, tabu search, simulated annealing, differential evolution, estimation of distribution algorithms, genetic and evolutionary algorithms, particle swarm optimization and artificial bee colony.

Usage

```
rmoptim(method, D, lb, ub, type, eval, control = list())
```

Arguments

`method`	- type of optimization method (search engine)

- `"fbs"` - Full Blind Search - searches all the possible alternatives. It only works for binary and integer optimization problems.
- `"grs"` - Grid Search - uses a hyper dimensional grid search that is dependent on a step size. It only works on integer and real optimization tasks.
- `"mcs"` - Monte Carlo Search - random search according to a given probability distribution.
- `"hlc"` - Hill Climb - local search method.
- `"sann"` - Simulated Annealing - uses `optim` function https://stat.ethz.ch/R-manual/R-devel/library/stats/html/optim.html.
- `"tabu"` - Tabu Search - uses `tabuSearch`
- `"gea"` - Genetic and Evolutionary Algorithms - population based method that uses `rbga` (or `rbga.bin` for binary tasks).
- `"dfev"` - Differential Evolution - uses `DEoptim`.
- `"pso"` - Particle Swarm Optimization - uses `psoptim`.
- `"eda"` - Estimation of Distribution Algorithm - uses `edaRun`.
- `"abc"` - Artificial Bee Colony - uses `abc_optim`.

`D`	- Dimension, the number of elements, variables or parameters to be optimized
`lb,ub`	- two values or two vectors setting the real lower and upper bounds for each parameter to be optimized (used for integer or real value optimization).
`type`	- type of optimization goal - `"max"` for maximization or `"min"` for minimization.
`eval`	- Evaluation function to be minimized or maximized. This function needs to return a numeric value.
`control`	- A list of control parameters. See "Details"

Details

The control argument is a list that can supply any of the following components:

- **iter** - The maximum number of iterations.
- **step** - Mandatory control parameter to the use of Grid Search (see examples).
- **numb** - Type of data representation of the final solution. `bin` for binary solutions and automatically sets lower and upper bounds to 0 and 1 respectively. `real` for real number solutions or `int` for integer solutions. By default sets `real` for optimization problems.
- **...** - Other parameters. For example, in the `"SANN"` method of the `optim` function the default value for `"temp"` is 10. But if the user wants to use a different value, then she/he just needs to insert the parameter `"temp"` in the control list.

Value

The output of the function `rmoptim` is a list containing the optimized parameters `"$sol"` and the optimum searched value `"$fun"`.

References

- - To check from where was inspired this package and to find some Optimization Problems: Paulo Cortez, Modern Optimization with R, Use R! series, Springer, September 2014.
- - ABCoptim: George, A., Yon, V., & George, M. (2015). Package "ABCoptim" Title Implementation of Artificial Bee Colony (ABC) Optimization. Retrieved from http://github.com/gvegayon/ABCoptim,
- - DEoptim: Ardia, D., Mullen, K. M., & Peterson, B. G. (n.d.). DEoptim : An R Package for Global Optimization by Differential Evolution, (2006).
- - TabuSearch: Katarina, D. (2015). Package tabuSearch (c)
- - Genalg: Willighagen, E., Maintainer, M. B., & Ballings, M. (2015). Package "genalg" Title R Based Genetic Algorithm. Retrieved from https://github.com/egonw/genalg
- - PSO: Bendtsen, C. (2015). Package "pso" Title Particle Swarm Optimization.
- - EDA: Gonzalez-Fernandez, Y., Soto, M., & Maintainer,]. (2015). Title Estimation of Distribution Algorithms Based on Copulas. Retrieved from https://github.com/yasserglez/copulaedas

Examples

```
## Demo function that executes three types of optimization. The objective of this Demo function
 ## is to help the user to understand the parameters and the operation of the package.
## The first problem being presented is an integer optimization problem. This problem can
 ## be found on the book Modern Optimization with R. The objective of the problem is to find
 ## the best price for five different bags that a company produces. The objective of this
 ## function is to maximize the profit.
## The second problem, is a real number optimization problem which is executed a minimization
 ## function only with two dimensions.
 ## Finally the last one, is a binary optimization problem, only with two iterations.
 ## The objective function performs the summation of all dimensions. Per example, if the
 ## dimensions are (0,1,1,0,0,1,1,0), the sum is 4.
 ## Not run:

   demo(rmo)

## End(Not run)

  ## More examples:
```

```
## Real Number Optimization Problem
## Objective Function:

fun <- function(x) {
-cos(x[1])*cos(x[2])*exp(-((x[1] - pi)^2 + (x[2] - pi)^2))
}

## Minimization Function
## Dimension: 2
## Lower: -20
## Upper: 20


## using the Particle Swarm Optimization method
rmoptim("pso",2,-20,20,"min",fun,list(numb="real",restart=2))

## Also in this case can be percetible the use of the restart variable in the control list
## which makes two restarts in the search process.
## The user can set some parametrs in the control list, parameters from the official documentation
## of the PSO package.

## Output:
## Particle Swarm Optimization
## Optimum Solution:  3.141602 3.141586  -  1

## Now the same method but with reporting
## To set the reporting thats necessary to have the trace value set as TRUE
## Then the package PSO, reports in 10 iterations.
rmoptim("pso",2,-20,20,"min",fun,list(numb="real",restart=2,REPORT=10,trace=TRUE))

## Output:
## Particle Swarm Optimization
## S=12, K=3, p=0.2297, w0=0.7213, w1=0.7213, c.p=1.193, c.g=1.193
## v.max=NA, d=56.57, vectorize=FALSE, hybrid=off
## It 10: fitness=-0.5297
## It 20: fitness=-0.9478
## It 30: fitness=-0.9968
## It 40: fitness=-0.9998
## It 50: fitness=-0.9999
## It 60: fitness=-1
## It 70: fitness=-1
## It 80: fitness=-1
## It 90: fitness=-1
## It 100: fitness=-1
## Maximal number of iterations reached
## Optimum Solution:  3.141594 3.141592  -  1

#using the Differential Evolution method
rmoptim("dfev",2,-20,20,"min",fun,list(numb="real"))

## Output:
## Differential Evolution
## Optimum Solution:  3.141593 3.141593  -  1

## By default the package rmo, has in the control list a parameter to set the
## maximum number of iterations. By default: iter=100.
```

```
## In the DEoptim package who runs the Differential Evolution Method
## has the parameter itermax. By default itermax=200.
rmoptim("dfev",2,-20,20,"min",fun,list(numb="real",iter=200))

## Now a example with the itermax parameter
rmoptim("dfev",2,-20,20,"min",fun,list(numb="real",itermax=200))

## With this, is pretended to show the way that can be manipulated



rmoptim("abc",2,-20,20,"min",fun,list(numb="int",maxCycle=100))

## The same optimization problem was used with Artificial Bee Colony, but in this case
## the final representation of the solution is with integer values.
## Also it is set the maxCycle in control list, with the value of 100.
## This means that the maximum number of iterations is 100. The user instead
## could set the value iter, which both does the same.

## Output:
## Artificial Bee Colony
## Optimum Solution:  3 3 -  1


################################################################################
## Now will be represented a binary optimization problem
sumbin <- function(x) sum(x)


## This function represents the sum of the binary values
## Example x=c(0,1,0,1,1,0,0,0)
## sum(x) = 3

## For this example, will run only with two iterations, because if it is set more iterations
## Will be always found the solution c(1,1,1,1,1,1,1,1)

## First it's going to be tested the Full Blind Search method.
## Because the search space is defined on the start and
## all the possible solutions are tested

rmoptim("fbs",8,0,1,"max",sumbin,list(numb="bin"))

## Output:
## Full Blind Search
## Optimum Solution:  1 1 1 1 1 1 1 1  -  8


## This example will show, the method running with the number of iterations in default
rmoptim("pso",8,0,1,"max",sumbin,list(numb="bin"))

## Output:
## Particle Swarm Optimization
## Optimum Solution:  1 1 1 1 1 1 1 1  -  8

## But if the number of iterations sets to 2 iterations
## The final value is going to be different
```

```
rmoptim("pso",8,0,1,"max",sumbin,list(numb="bin",iter=2))

## Output:
## Particle Swarm Optimization
## Optimum Solution:  1 0 0 1 1 1 1 1  -  6

## The execution is the same for every method.
rmoptim("hlc",8,0,1,"max",sumbin,list(numb="bin",iter=2))
## Hill Climbing
## Optimum Solution:  0 0 1 1 1 1 1 1  -  6

## For this execution will be shown the execution of Tabu Search method
## This method works only with binary values, which makes it the best
## for this type of tasks.

rmoptim("tabu",8,0,1,"max",sumbin,list(numb="bin",iter=2))
## Tabu Search
## Optimum Solution:  1 0 0 1 1 0 0 1  -  4
```

Index