UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

BRUNO ARAUJO MUNIZ

ALGORITMOS EVOLUTIVOS HIBRIDIZADOS COM TÉCNICAS DE MINERAÇÃO DE DADOS PARA O PROBLEMA DE OTIMIZAÇÃO CONTINUA

SÃO LUÍS/MA

2021

BRUNO ARAUJO MUNIZ

ALGORITMOS EVOLUTIVOS HIBRIDIZADOS COM TÉCNICAS DE MINERAÇÃO DE DADOS PARA O PROBLEMA DE OTIMIZAÇÃO CONTINUA

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Alexandre Cesar Muniz de Oliveira

São Luís/MA

2021

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Araujo Muniz, Bruno.
ALGORITMOS EVOLUTIVOS HIBRIDIZADOS COM TÉCNICAS DE MINERAÇÃO DE DADOS PARA O PROBLEMA DE OTIMIZAÇÃO CONTINUA
/ Bruno Araujo Muniz. - 2021.
45 f.
Orientador(a): Alexandre Cesar Muniz de Oliveira.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São luis - MA, 2021.
1. Algoritmo genetico. 2. Benchmark. 3. COCO. 4. Hibridismo. 5. Otimização. I. Cesar Muniz de Oliveira,
Alexandre. II. Título.

BRUNO ARAUJO MUNIZ

Algoritmos evolutivos hibridizados com técnicas de mineração de dados para o problema de otimização continua.

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado em São Luís – MA, de__Setembro de 2021

_____

Prof. Dr. Alexandre Cesar Muniz de Oliveira
(Orientador) Universidade Federal do Maranhão

_____

Prof. Me. Carlos Eduardo Portela Serra de Castro
Examinador 1 Universidade Federal do Maranhão

_____

Prof. Dr. Francisco Glaubos Nunes Climaco
Examinador 2 Universidade Federal do Maranhão

São Luís/MA

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, pois me deu forças para continuar até aqui, fazendo com que eu perseverasse e não me deixasse abater com os obstáculos que apareceram pelo caminho.

Gostaria de agradecer meus pais Edvan e Ana Paula pois sem eles eu não teria nem sequer conseguido viver até esse momento.

Gostaria de agradecer minha irmã Sara e meus amigos Clayton, Davi, Weked e Luiza, que sempre me deram forças e estiveram comigo nos momentos em que as coisas pareciam não ter solução.

Gostaria de agradecer a minha namorada e companheira da vida Lorraine, que sempre esteve ao meu lado, sabendo no íntimo de todas frustrações, alegrias e tristezas, pelo ombro amigo nas noites frias e por todas as soluções brilhantes para problemas que se referiram a escrita de trabalhos.

Gostaria de agradecer à célula sementes na qual eu fiz parte, pelos amigos que lá fiz, que sempre me receberam de braços abertos.

Gostaria de agradecer ao corpo docente desta instituição, onde pude aprender muito, e abrir a mente para coisas que antes eu nem fazia ideia que existiam.

E por fim Gostaria de agradecer ao meu orientador Alexandre que tanto fez por mim nessa reta final de curso, pelos conselhos e dicas e por sua paciência comigo.

"A persistência é o caminho do êxito."

(Charles Chaplin)

RESUMO

Os algoritmos híbridos são muito utilizados para resolver diversos problemas de otimização, com isso diversas variações e combinações de algoritmos evolutivos com outras técnicas surgem a todo momento, e são apresentados em eventos que acontecem anualmente, portanto é importante entender como esses algoritmos podem ser classificados, para facilitar o seu entendimento e ajudar na implementação de novos algoritmos híbridos, aplicando hibridização entre metaheurísticas, programação matemática, programação com restrição e aprendizado de maquina com técnicas de mineração de dados, também são apresentados alguns exemplos de algoritmos híbridos. É apresentado também uma ferramenta de avaliação de algoritmos evolutivos em caixa preta o COCO, onde é apresentada a sua arquitetura, interface programável e uma amostra de comparação entre um algoritmo evolutivo de codificação real RealCode-GA e CMA-ES um algoritmo híbrido que é o estado da arte dos algoritmos evolutivos de codificação real.

Palavras Chave: COCO, MachineLearnig, Otimização, GA.

ABSTRACT

Hybrid algorithms are widely used to solve several optimization problems, with this, several variations and combinations of evolutionary algorithms with other techniques arise all the time, and are presented in events that take place annually, so it is important to understand how these algorithms can be classified, to facilitate your understanding and help in the implementation of new hybrid algorithms, applying hybridization between metaheuristics, mathematical programming, constraint programming and machine learning with data mining techniques. It is also presented a tool for evaluating evolutionary algorithms in black box, the COCO, where its architecture, programmable interface and a comparison sample between an evolutionary algorithm of real coding RealCode-GA and CMA-ES a hybrid algorithm that is the state of the art of evolutionary real coding algorithms

Keywords: COCO, machinelearnig, optimization , GA.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AE	Algoritmo evolutivo
Bnc	Branch-and-Cut
CEC	Congress on Evolutionary Computation
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
COCO	Comparing Continuous Optimizers
GECCO	Genetic and Evolutionary Computation Conference
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers)
LACMOR	Laboratório de Aprendizagem Computacional, Métodos de Otimização e Robótica
Metaheurística-P	Metaheurística populacional
Metaheurística-S	Metaheurística de solução unica
MIP	Mixed-Integer Programming
PR	Programação com restrição
UFMA	Universidade Federal do Maranhão

SUMÁRIO

1 INTRODUÇÃO

A necessidade de otimização em aplicações utilizadas no meio científico, financeiro, industrial e de gerenciamento ocorre de maneira frequente, além desses meios mencionados temos também várias situações do cotidiano que precisam ser observadas, onde máquinas com grande poder de processamento não são capazes de lidar com os problemas que surgem em tempo aceitável utilizando métodos exatos. Os métodos exatos são capazes de encontrar o resultado perfeito para o problema de otimização, mas demanda uma complexidade maior em sua modelagem, fazendo assim com que seu uso prático seja limitado (BARBOSA, 2017).

Grande parte dos problemas de otimização vêm sendo resolvidos por abordagens evolutivas, pelos chamados algoritmos evolutivos (AEs), por sua adaptabilidade aos mais diversos tipos de problemas, abrindo mão de uma solução exata por diversas soluções aproximadas que resolvem o problema fornecido em um tempo computacional muito mais aceitável (YANG, 2010).

Os AEs são baseados nos acontecimentos evolutivos que ocorrem na natureza, onde alguns critérios devem ser observados, como: a existência de uma população que irá disputar recursos limitados, a habilidade que cada indivíduo terá de se adaptar ao meio em que está contido, a noção de que a população será redefinida com a morte e nascimento de novos indivíduos, e as condições de variedade entre os indivíduos e a capacidade de passar seus genes adiante (HANSEN e OSTERMEIER, 2001).

Mas na realidade tecnológica atual não basta termos apenas um AEs precisamos das melhores ferramentas possíveis quando o assunto é resolver um problema, isso é possível de se obter com o cruzamento de diferentes áreas do conhecimento, como a modelagem matemática, pesquisa operacional, estatísticas e outros campos. A hibridização não se restringe apenas a combinação de heurísticas e meta-heurísticas, abrange também a combinação com programação matemática, dinâmica, programação de restrições e aprendizado de máquina com técnicas de mineração de dados (BLUM e RAIDL, 2016).

Anualmente acontecem dois grandes eventos de computação evolutiva que é o Genetic and Evolutionary Computation Conference (GECCO) e o IEEE (Institute of Electrical and Electronics Engineers) Congress on Evolutionary Computation (CEC), onde são trazidos vários algoritmos que passam por uma série de avaliações de

benchmarking através do framework COCO. Um dos algoritmos mais famosos nascido na GECCO e o Covariance Matrix Adaptation Evolution Strategy (CMA-ES) representa o estado da arte em algoritmos de otimização evolutivos com codificação real. (Corrêa, 2019).

O COCO é um *framework* de código aberto para comparação de algoritmos de otimização contínua, em um estilo de caixa preta que tem como objetivo minimizar a tarefa repetitiva de benchmarking, onde é possível realizar no mesmo *framework* medições em algoritmos determinísticos e estocásticos para otimização simples e multiobjectivo (Hansen, 2016).

O objetivo deste trabalho é trazer o conceito de hibridização em algoritmos evolutivos, ver quais são as técnicas mais utilizadas e o que mais vem dando certo com o passar dos anos, trazendo diferentes abordagens de hibridização que podem ser avaliadas através da ferramenta de benchmarking COCO.

1.1 Objetivo

1.1.1 Objetivo Geral

Este trabalho tem como objetivo observar alguns modelos hibridos presentes na literatura.

1.1.2 Objetivos Específicos

- Apresentar uma taxonomia presente na literatura que classifique de forma organzada para fins de comparação os algoritmos hibridos.
- Desenvolvimento de algoritmos de otimização adaptados para execução no framework COCO.
- Realização de experimentos abrangentes com qualidade técnica para eventualmente serem submetidos a fóruns técnico-científicos especializados em algoritmos para otimização de funções

2 Justificativa

Na última década o uso de metaheurísticas híbridas aumentou de maneira substancial no campo da otimização. As melhores soluções obtidas para problemas de otimização clássicas ou da vida real foram alcançados graças aos algoritmos híbridos, como em (SILVA, 2017) onde é utilizado um algoritmo estocástico-determinístico na

solução do problema de direção automotiva, esse algoritmo é composto por outros dois algoritmos que são o *Luus Jaakola* (LUUS, 1999) e o *Hooke Jeeves* (LONG, 2014).

Em (OLIVEIRA, 2017) onde se busca identificar novos equipamentos a serem incluídos em uma rede de transmissão temos uma mescla de mais dois algoritmos para a resolução desse problema, que é o *Harmony Search* (OMRAN, 2008) e o *Branch and Bound* em serie (HANSEN, 1992).

Muitos outros exemplos podem ser citados nas mais diversas áreas do conhecimento como em (FERREIRA et al., 2015) com a combinação de metaheurísticas e em (NWANA, 2005) onde o algoritmo *branch-and-bound* é combinado com um algoritmo estocástico, por isso é de extrema importância entender o conceito de hibridismo, as diferentes técnicas, que podem ser mais exploradas além da fusão de duas metaheurísticas.

Entender abordagens mais complexas que pode abrir um leque ainda maior de possibilidades como a combinação de metaheurísticas com métodos exatos de programação matemática (muito utilizado em pesquisa operacional), métodos de programação com restrição (utilizado em inteligência artificial) e com técnicas de aprendizado de máquina e mineração de dados.

3 FUNDAMENTAÇÃO TEÓRICA

Para um entendimento claro dos assuntos que estão por vir, se faz necessário a apresentação de uma fundamentação teórica, pois serão abordadas técnicas de hibridização de alguns nichos diferentes, então será abordado os princípios teóricos de metaheurísticas sobre algoritmos evolutivos, programação matemática, programação com restrição e técnicas de aprendizado de máquina e mineração de dados.

3.1 Algoritmos Evolutivos

3.1.1 Inspiração Biológica

A evolução biológica realiza um estudo sobre um espaço de dimensão e complexidade muito abrangente, formado pelo conjunto de todas as combinações genéticas possíveis, onde algumas dessas combinações serão criaturas viáveis para sobreviver no meio ambiente em que está contido (GASPAR et al., 2012).

De acordo com Darwin em 1859 em seu trabalho *"On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life",* o processo de evolução natural ocorre por meio de dois mecanismos básicos, a seleção e a reprodução com mutação.

A seleção certifica que os indivíduos que melhor se adaptam ao ambiente sobrevivam e tenham uma taxa de reprodução maior do que aqueles que não se adaptaram bem, assim conseguindo manter suas características nas próximas gerações. Com isso, junto ao processo de mutação à medida que os indivíduos vão se reproduzindo, variações são obtidas e essas variações vão sendo passadas adiante obedecendo ao processo de seleção. O ciclo gerado por esses princípios é que garante a gradual evolução.

No início do século XX foi formulada a teoria genética da hereditariedade que permitiu introduzir o conceito de gene no processo de evolução, fazendo com que a maneira que as alterações são introduzidas na população fosse entendida de maneira mais intuitiva. A fusão desses dois conceitos trazidos por Darwin e Mendel é o que conhecemos hoje por Neodarwinismo (GASPAR et al., 2012).

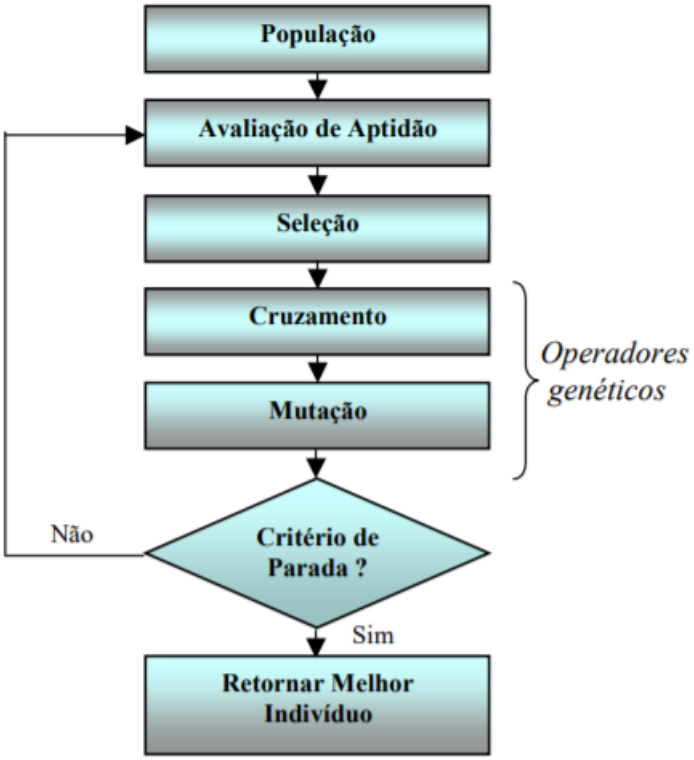
Baseados nos conceitos básicos da evolução biológica foram desenvolvidos algoritmos em ambientes computacionais voltados para as mais diversas áreas, como a matemática, engenharia, biologia etc. Onde o surgimento desses algoritmos é trazido de forma clara por John Holand em seu trabalho intitulado "*Como transformar as ideias de Darwin em algoritmos?".*

3.1.2 Algoritmo Genético

A tentativa de imitar a evolução natural através dos conceitos do neodarwinismo teve início entre os anos 50 e 60, onde vários biólogos davam os primeiros passos em direção aos algoritmos genéticos, mas foi John Holland que iniciou o desenvolvimento das primeiras pesquisas sobre o tema (POZO, 2005) com o objetivo de formalizar matematicamente esses conceitos e desenvolver sistemas artificiais que imitam os mecanismos de evolução natural (LIMA, 2006).

Holland (1975) apresentou uma norma que se baseia em desenvolver uma população, usando o critério de seleção natural e os agentes genéticos de crossover (cruzamento) e mutação como pode ser observado na figura 1.

Figura 1 – estrutura básica de um GA.



Fonte: (POZO, 2005).

POPULAÇÃO

A população é um conjunto de possíveis soluções, que serão utilizadas para criar um novo conjunto de indivíduos que serão submetidos a um processo de avaliação. O ponto de partida para a sua criação é definir como a solução será codificada, onde a forma mais apropriada para codificar está diretamente ligada ao problema que se deseja resolver. A forma comumente utilizada é representar cada característica como uma sequência de bits e a solução será o conjunto desses bits (um conjunto de características cria um indivíduo), outras formas de representação podem ser vistas em (HOLLAND, 1975), outra forma de codificação é utilizar números reais ao invés de números binários (RealCode-GA).

O tamanho da população também é um fator importante a ser observado pois tem impacto direto no desempenho de um algoritmo genético, pois se ela for muito pequena pode não ter a diversidade suficiente para alcançar uma solução aceitável para o problema, mas em contrapartida se ela for muito grande pode gerar lentidão no processo de busca pela solução, além de necessitar de mais recursos computacionais para trabalhar.

AVALIAÇÃO DE APTIDÃO

Para avaliar uma população deve ser definido um critério de pontuação, normalmente são utilizadas funções, onde cada indivíduo recebe uma pontuação de acordo com sua proximidade do objetivo desejado.

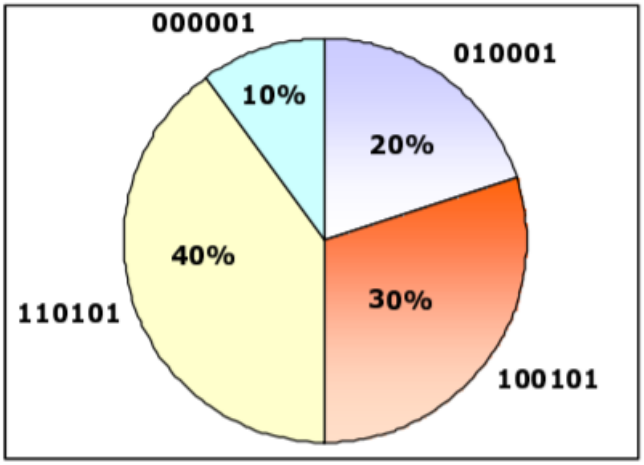
SELEÇÃO

Existem vários métodos para se selecionar os indivíduos para a próxima geração, o algoritmo genético clássico usa o método da roleta que consiste em atribuir a cada solução uma probabilidade de ser sorteado e passar para próxima geração, essa probabilidade é definida de acordo com o fitness (pontuação) da solução, como mostra a figura 2.

O problema desse método é que sempre existe a possibilidade de se perder a melhor solução pois mesmo ela tendo altas chances se passar isso não garante que ela vá passar, uma forma de contornar essa situação é sempre garantindo que o indivíduo com a melhor pontuação vá passar para a próxima geração, esse método é chamado de elitismo (LIMA, 2006).

Outra maneira de realizar a seleção é utilizar o esquema de torneio, onde um número *M* de indivíduos é escolhido aleatoriamente para formar uma população menor temporária, dessa nova divisão o melhor elemento dessa subpopulação é selecionado, assim se escolhe um conjunto de N elementos que serão selecionados, existem ainda vários modelos de seleção na literatura que podem ser achados em (LIMA, 2006).

Figura 2 – divisão das probabilidades de acordo com o fitness de cada indivíduo.



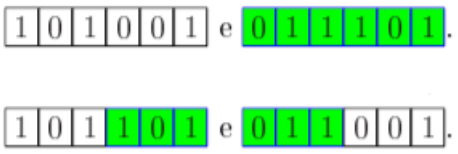
Fonte: (POZO, 2005).

CRUZAMENTO

O cruzamento ou crossover como é mais conhecido é responsável por pegar duas ou mais soluções e combiná-las, com o intuito de gerar um elemento totalmente novo, no algoritmo genético clássico é dada um percentual fixo de ocorrer crossover entre dois elementos, mas para esse cruzamento ocorrer uma taxa de cruzamento deve ser definida ela ditará a probabilidade que uma união irá ocorrer.

O cruzamento mais popular é o de ponto, onde inicialmente são selecionados dois elementos e partir disso é escolhido um ponto de corte no cromossomo (*string* de bits), com o ponto definido, há uma divisão desses elementos onde essas duas soluções são quebradas em quatro e remontadas, gerando assim duas nova soluções candidatas, como mostra a figura 3.

Figura 3 – operação de crossover de ponto.



Fonte: (POZO 2005).

Um outro tipo de cruzamento muito popular é o crossover uniforme, onde cada cromossomo do filho é definido através de sorteio entre dois cromossomos da mesma posição dos seus pais, esses dois são os exemplos mais comuns, é possível encontrar novas formas de cruzamento nos trabalhos (LIMA, 2006) e (POZO, 2005).

MUTAÇÃO

Ao iniciar um AG é definida uma taxa de mutação que deve ser bem baixa para que não atrapalhe a evolução adquirida até então, de tempos em tempos ocorre a mutação em uma das soluções aleatoriamente, tome a codificação clássica apresentada por HOLLAND (1975), que utiliza uma *string* binária como as características do indivíduo como visto na figura 3, um ou mais cromossomos é sorteado para mudar de valor, onde o 0 pode virar 1 e vice versa.

A mutação é necessária para que haja uma maior variedade na população, pois um grupo de soluções com características muito próximas, pode levar a população a convergir prematuramente e ficar estagnada em uma solução pouco satisfatória, esse fenômeno é chamado de ótimo local. Os efeitos de definição de parâmetros das taxas de mutação e crossover, além dos parâmetros de tamanho da população e número de cromossomos de um indivíduo podem ser vistos em um estudo mais aprofundado no (CATARINA, 2008).

Com isso passamos por todos os mecanismos utilizados na implementação clássica proposta por HOLLAND (1975) sobre os algoritmos genéticos, na figura 4 é apresentada uma implementação típica de um AG.

Figura 4 – Pseudocódigo algoritmo genético típico.

Algoritmo 2.1 Pseudo-código de um AG típico.

```
ALGORITMO AG
// inicializa uma população de n indivíduos aleatoriamente
INICIA_POPULACAO(P(t));
// avalia o grau de adequação dos indivíduos de P'
AVALIA(P');
// testa o critério de término (por exemplo, um tempo t máximo ou um nível de adaptação esperado)
ENQUANTO criterio nao atingido FACA
    // obtém uma nova população privilegiando os indivíduos mais adaptados
    P' := SELECIONA_INDIVÍDUOS(P(t));
    // aplica crossover sobre os indivíduos selecionados
    APLICA_CROSSOVER(P');
    // perturba estocasticamente os indivíduos da população que recombinou
    MUTA(P');
    // avalia o grau de adequação dos indivíduos de P'
    AVALIA(P');
    // seleciona os sobreviventes entre os indivíduos de P(t) e P'
    P(t+1) := SOBREVIVENTES(P(t),P');
FIM
```

Fonte: (LIMA, 2006).

3.1.3 OUTROS ALGORITMOS EVOLUTIVOS

Ao decorrer da história da criação dos algoritmos evolutivos, cinco abordagens diferentes foram trabalhadas paralelamente, em cima da mesma premissa de evolução natural, onde os aspectos que as diferenciam são como cada fase de evolução é feita, as estruturas de dados utilizadas na codificação dos indivíduos, como a população inicial é criada e como a mutação é empregada, essas cinco abordagens são (POZO, 2005):

- Algoritmos Genéticos (AG)
- Estratégias Evolutivas (EE)
- Programação evolutiva (PE)
- Sistemas Classificadores (SC)
- Programação Genética (PG)

3.2 Metaheurísticas

As metaheurísticas são derivadas diretamente das heurísticas clássicas. Uma heurística é um conjunto de métodos que visam resolver problemas de otimização sem que haja uma teoria que garanta chegar ao resultado correto, além disso, nem mesmo é possível garantir que a solução obtida estará próxima à solução exata. O que acontece é que o algoritmo busca a melhor opção no meio das alternativas à disposição.

O que difere uma da outra é que as heurísticas eram projetadas para resolverem problemas muito específicos e as metaheurísticas são abordagens mais genéricas, que podem ser utilizadas para resolver uma classe maior de problemas, sendo cabível a maior parte dos problemas combinatórios.

Nos últimos 30 anos as metaheurísticas vem ganhando mais popularidade, pelo seu uso empregado nas resoluções de vários problemas, pois mesmo não sendo capaz de garantir alcançar a melhor solução ela consegue entregar soluções satisfatórias em tempo razoável, alguns exemplos das áreas beneficiadas por esse modelo são:

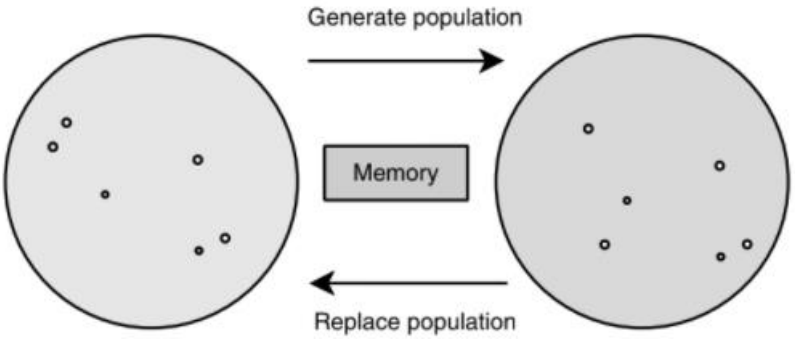
- Aprendizado de máquina, mineração de dados, bioinformática, biologia computacional e finanças.
- Modelagem, simulação, e identificação de sistemas em química, física e biologia, processamento de controle de sinal e imagem.
- Planejamento de problemas de roteamento, planejamento de robôs, problemas de programação e produção, logística e transporte, gerenciamento de cadeia de suprimentos e muitas outras aplicações.

Onde se precisa de otimização, as metaheurísticas costumam estar presentes. Ao se projetar uma metaheurística é necessário realizar dois tipos diferentes de exploração, a exploração das soluções possíveis (diversificação), e o estudo em cima das soluções já encontradas (intensificação), ao analisar as soluções já obtidas durante o processo de intensificação é possível verificar se a região onde ela se encontra tem potencial para apresentar resultados ainda melhores.

Durante o processo de diversificação novas áreas ainda não exploradas pelo processo de busca são verificadas, assim com a combinação dessas duas condições o conjunto solução é melhor explorado. As metaheurísticas ainda são divididas em duas subclasses que são as metaheurísticas populacionistas (metaheurística-P) e metaheurísticas de solução única (metaheurística-S) (TALBI 2009),

3.2.1 Metaheurística-P

Figura 5 – principal princípio da metaheurística-P



Fonte: livro base

As metaheurística-P dividem vários conceitos em comum. Pois basicamente realizam iterações sobre uma população e a cada iteração realizada novos indivíduos podem ser criados e a população é atualizada como mostra a figura 5. A iteração é interrompida quando determinada condição pré-estabelecida é cumprida.

A maioria dos algoritmos baseados nessa classificação de metaheurística são inspirados na natureza, como já vimos anteriormente como exemplo de metaheurística-P temos os algoritmos genéticos, mas além deles também temos muitos algoritmos famosos como colônia de formigas, exame de partículas, colônia de abelhas e sistemas 6imunológicos artificiais. Uma implementação de alto nível pode ser observada na figura 6 (TALBI, 2009).

Figura 6 – pseudocódigo metaheurística-P

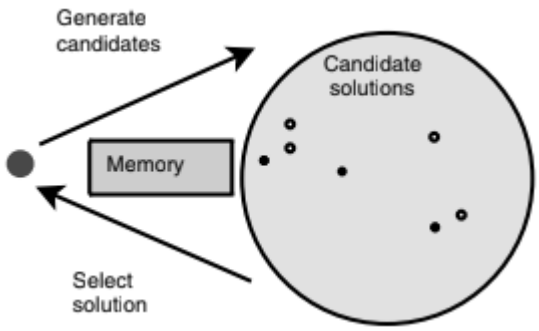
Algorithm High-level template of P-metaheuristics.

$P = P_0$; /* Generation of the initial population */
$t = 0$;
Repeat
  Generate(P'_t); /* Generation a new population */
  P_{t+1} = Select-Population($P_t \cup P'_t$); /* Select new population */
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution(s) found.

Fonte: (TALBI, 2009).

3.2.2 Metaheurística-S

Figura 7: geração de solução de uma metaheurística-S.



Fonte: (TALBI, 2009).

metaheurísticas *single-solution* trabalham na busca de uma única solução para a resolução do problema, na fase de busca de uma solução um conjunto de soluções é gerado a partir da solução atual como ilustrado na figura 7, realizando algumas perturbações para a criação se soluções novas, é realizado um tipo de busca local em soluções próximas da já obtida, alguns exemplos de algoritmos que seguem esse modelo são busca local, busca tabu e recozimento simulado, na figura 8 é possível ver uma implementação de alto nível desse modelo (TALBI, 2009).

Figura 8: pseudocódigo de uma metaheurísticas-S.

Algorithm High-level template of S-metaheuristics.

Input: Initial solution s_0.
$t = 0$;
Repeat
  /* Generate candidate solutions (partial or complete neighborhood) from s_t */
  Generate($C(s_t)$) ;
  /* Select a solution from $C(s)$ to replace the current solution s_t */
  s_{t+1} = Select($C(s_t)$) ;
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution found.

Fonte: (TALBI, 2009).

3.3 Programação matemática

Algoritmos que utilizam programação matemática versam sobre a busca do resultado exato na busca de soluções, a maioria deles não podem ser aplicados para problemas com entradas muitos grandes, as principais abordagens de programação matemática são classificadas em algoritmos enumerativos, métodos de relaxamento,

plano de corte.

- Algoritmos enumerativos: incluem algoritmos como *branch and bound.* programação dinâmica e outros algoritmos de busca em árvore, esses tipos de algoritmos buscam a solução passando por todas as soluções possíveis implicitamente, a técnica aplicada é explorar o conjunto solução através de uma construção dinâmica de uma árvore, que é dividida em outras subárvores, esse conceito é conhecido como dividir pra conquistar onde um problema e dividido em vários outros problemas mais simples.

- Métodos de relaxamento e decomposição: Esse nicho de algoritmos exatos tem como base técnicas como relaxação Lagrangiana (FEO, 1995), que é voltado a relaxar um requisito de um problema, esse problema pode ser simplesmente eliminado ou substituído por outro que seja mais fácil de ser resolvido.

- Plano de corte: proposto em 1958 por GOMORY (2008, 77-103), resumidamente essa técnica consiste em atribuir algumas restrições ao relaxamento de programação linear (LP) do problema.

3.4 Programação com restrição

A fundamentação clássica de uma programação com restrição básica consiste em:

- fazer a declaração de variáveis e domínio
- impor restrições e buscar soluções

O espaço de solução é formado pelas variáveis e o domínio onde podem atuar, as restrições estabelecem regras que devem ser seguidas para a geração de soluções assim reduzindo o espaço de busca das soluções possíveis do domínio.

À medida que novas restrições são inseridas a propagação dessas restrições devem ser realizadas para todas as variáveis, esses conceitos são mais fáceis de compreender através de um exemplo, tome como variáveis X e Y e os valores que eles podem assumir está no intervalo [1, 10], ou seja, o intervalo definido é o domínio, a restrição seria que X<Y, logo o conjunto solução de Dx = [1, 9] e Dy [2, 10].

Isso significa que o conjunto de soluções que X pode assumir está no intervalo [1, 9] não podendo ir até 10 pois se isso fosse possível feriria a restrição, pois não teria um Y que fosse maior que 10, essa restrição faz com que o domínio de X e Y mudem, de acordo com que novas restrições são aplicadas, elas devem ser propagadas por todas as variáveis para a atualização de seus domínios. Após definida as restrições um algoritmo

de busca é realizado respeitando as restrições, esse algoritmo é baseado em uma técnica de busca em árvore (YUNES, 2000).

3.5 Aprendizado de máquina

O aprendizado de máquina ou *machine learning* é um dos ramos da IA (inteligência artificial) que tem como objetivo o aprimoramento de técnicas sobre o aprendizado, assim como a modelagem e criação de programas capazes de adquirir conhecimento de forma automatizada, esse tipo de programa observa soluções obtidas de problemas anteriores e a partir disso busca uma generalização para problemas similares que possam aparecer (MONARD, 2003, 89-144).

O *machine learning* é dividido em aprendizado supervisionado e não supervisionado. No aprendizado supervisionado o algoritmo recebe um conjunto de dados, onde cada indivíduo está relacionado com sua solução, basicamente o problema está codificado como um vetor de características e um rotulo que identifica a qual classe ele pertence como na figura 9, onde o objetivo é construir um classificador, onde dado novos elementos ele possa classificar de maneira correta.

Figura 9 – mostra uma tabela com o id do jogador de futebol e suas habilidades, cada jogados possui um rotulo.

Id	Idade	Altura	Técnica	Passe	Chute	Força	Velocidade	Drible	Classe
1	17	177	72	65	72	60	84	81	Atacante
2	18	188	63	65	55	70	72	60	Defensor
3	18	190	63	65	67	70	72	66	Atacante
4	19	165	65	62	71	62	70	67	Atacante
5	19	174	67	66	69	64	76	74	Atacante
6	19	184	64	68	47	73	71	63	Defensor
7	19	184	64	62	70	75	73	65	Atacante
8	19	186	68	66	52	76	72	63	Defensor
9	20	170	68	65	64	62	79	75	Atacante
10	20	175	67	66	69	62	82	74	Atacante
11	20	180	66	68	70	70	66	68	Atacante
12	20	184	73	65	79	79	77	71	Atacante
13	20	187	64	69	50	72	62	62	Defensor
14	20	188	62	64	47	70	72	64	Defensor
15	20	188	65	64	45	79	72	58	Defensor
16	20	188	72	69	45	82	74	63	Defensor
17	20	190	64	69	58	70	62	61	Defensor
18	20	191	63	63	45	74	73	60	Defensor
19	21	170	64	65	70	66	70	67	Atacante
20	21	172	67	67	66	68	72	71	Atacante

Fonte: trabalho final da disciplina IA UFMA.

O aprendizado não supervisionado observa os dados fornecidos (nesse caso

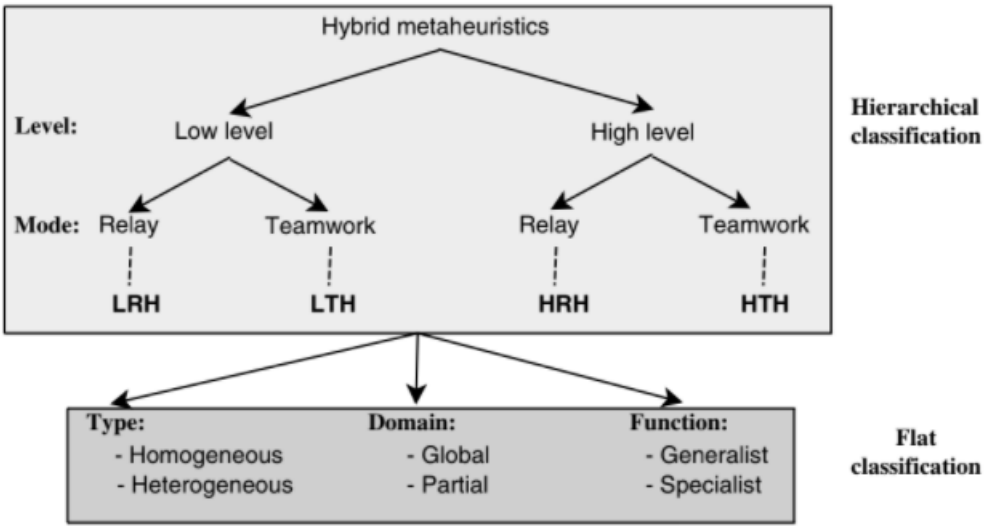
nenhum dado é previamente rotulado) e tenta agrupa-los de acordo com as suas características formando assim agrupamentos ou *clusters* (MONARD, 2003, 89-144), após os dados serem agrupados deve ocorrer uma análise para determinar o que cada agrupamento representa.

4 ALGORITMOS HIBRIDIZADOS

Em 2009 o professor El-Ghazali Talbi mestre e doutor em ciência da computação em seu trabalho intitulado "*Metaheuristics from design to implementation*", trata de como projetar metaheurísticas híbridas, mostra um sistema de classificação de algoritmos híbridos a fim de fornecer ferramentas de comparação entre algoritmos de maneira eficiente.

Onde são apresentas quatro maneiras diferentes de se projetar algoritmos híbridos utilizando metaheurísticas, combinando metaheurísticas-P com metaheurísticas-S, metaheurísticas com métodos de programação matemática, metaheurísticas com abordagem de programação com restrição e metaheurísticas com técnicas de aprendizado de máquina e mineração de dados.

Figura 10: como a combinação de metaheurística está classificada.



Fonte: (TALBI, 2009)

A hibridização pode ser classificada hierarquicamente em dois níveis como mostra a figura 10, onde se vê o nível e o modo em que a hibridização é feita, no nível baixo (*low level*) temos uma metaheurística onde parte de seu funcionamento é

substituído por uma outra metaheurística, em hibridizações de alto nível temos metaheurísticas independentes, onde uma não contribui no funcionamento da outra em nível interno.

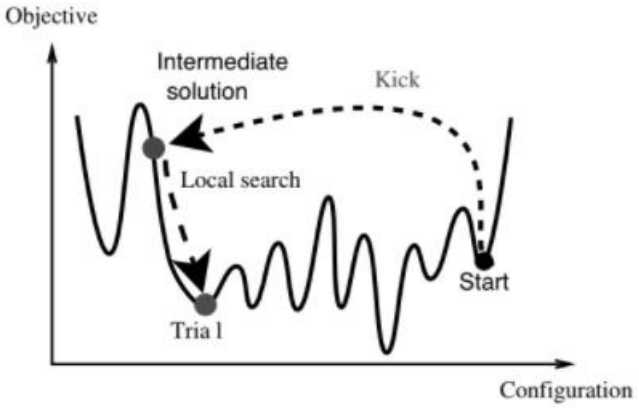
Na hibridização de retransmissão (*Relay*) de alto nível funciona como um pipeline onde a saída de uma metaheurística serve como a entrada da seguinte. Na hibridização de trabalho em equipe (*Teamwork*) de alto nível temos duas ou mais metaheurísticas trabalhando em paralelo para encontrar a melhor solução, dessa hierarquia mostrada na figura 8 são derivadas quatro classes: LHR, LTH, HRH, HTH.

4.1 Combinação de metaheurísticas

4.1.1 Híbrido de retransmissão de baixo nível (LRH)

Esse modelo de implementação é o menos utilizado na hora de se criar uma hibridização, tal modelo consiste em dada uma metaheurística qualquer, existe nela embutida uma metaheurística-S. Um exemplo disso pode ser obtido ao embutir pesquisa local em recozimento simulado o que faz com que a cadeia de Markov relacionada ao recozimento simulado busque apenas por ótimos locais (FERREIRA et al., 2015), a figura 11 apresenta os passos que o algoritmo realiza com essa combinação.

Figura 11 – ilustração do funcionamento da combinação do algoritmo de busca local com recozimento simulado.



Fonte: (TALBI, 2009).

Tome que a solução inicial é um ótimo local sinalizando pelo ponto *start*, um chute (*kick*) é aplicado como uma proposta de nova solução, o novo ponto passa por uma melhoria pelo algoritmo de busca local, que procura por um ótimo local a partir do novo ponto sinalizado na imagem como Tria 1, basta submetê-lo ao teste de aceitação do

algoritmo de recozimento simulado, se o teste for aceito o novo ótimo vira o novo ponto *start*, com isso outro chute deve ser realizado na busca do ótimo global. Tal ciclo é repetido até que o critério de parada seja alcançado.

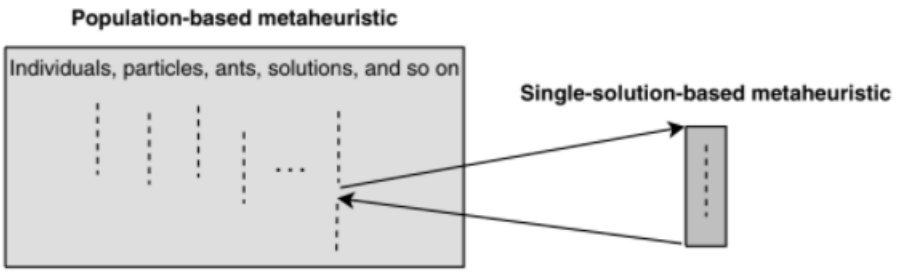
4.1.2 Híbrido de trabalho em equipe de alto nível (LTH)

Como já foi abordado na seção 2.2 uma metaheurística deve ser projetada em cima de duas premissas básicas, a Diversificação e intensidade, onde a diversificação busca explorar novas regiões do espaço de busca que sejam promissoras, já a intensidade explora essas regiões definidas como promissoras passando uma espécie de pente fino sobre esse espaço, com o intuito de encontrar uma solução melhor que a atual.

As metaheurísticas-P são ótimas no quesito diversificação explorando de maneira eficiente o espaço de soluções, já as metaheurísticas-S são mais poderosas no quesito intensidade, ou seja, as duas classes de metaheurísticas se complementam, pois enquanto as metaheurísticas-P tentam encontrar o ótimo globalmente, as metaheurísticas-S tentam encontrar o ótimo localmente, sendo assim metaheurísticas-S foram embutidas em metaheurísticas-P.

Esse modelo de hibridização é muito popular sendo frequentemente aplicado com sucesso em vários problemas de otimização. Tomando um algoritmo evolutivo (metaheurística-P) para fazer a busca global das soluções e inserindo nele um algoritmo de busca local (metaheurística-S) ao invés de utilizar uma busca totalmente cega sem levar em consideração a solução que se tem atualmente, o indivíduo da metaheurística-P é submetido a uma busca local com o objetivo de melhorá-lo, o indivíduo melhorado é devolvido a população do metaheurística-P, como mostrado na figura 12

Figura 12 – mostra um indivíduo sendo retirado da população e sendo devolvido aprimorado por uma metaheurística-S



Fonte: (TALBI, 2009).

Em (DA MATA, 2020) e possível encontrar essa abordagem onde um algoritmo genético é combinado uma busca local, também é adotada em (CRAVO, 2017) com o

algoritmo GRASP que utiliza a técnica de busca local em seus indivíduos.

Na literatura esse tipo de hibridização é conhecido como modelo lamarckiano, um outro modelo também faz uso dessa metodologia de combinar metaheurística-P com metaheurística-S que é o Baldwin que ao invés que usar um tipo de busca local para aprimorar o indivíduo, ele acontece antes do processo de seleção para avaliar a capacidade que o indivíduo tem para evoluir, sem alterar diretamente o genótipo do indivíduo, passando para a próxima geração o indivíduo mais capaz de encontrar uma solução melhor (GUIMARÃES. 2008).

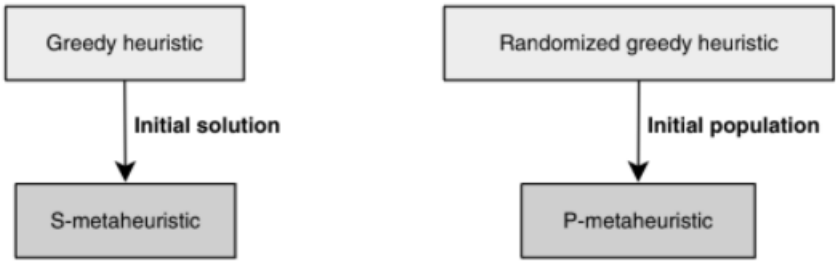
O LTH pode ser usado para aprimorar qualquer abordagem que utilize AGs, seja, colônia de formigas (BERNARDES, 2018) (TAILLARD, 1997), enxame de partículas (O'REILLY, 1995, 573-578), programação genética etc. O problema desse método é o perigo do algoritmo ficar preso em um ótimo local, a opção para resolver esse problema é fazer a hibridização condicional, ou seja, definir uma taxa que controla quando essa técnica seria aplicada, ao invés de realizá-la em todas as gerações.

4.1.3 Híbrido de retransmissão de alto nível (HRH)

A retransmissão de alto nível como já foi dito anteriormente utiliza a saída de um algoritmo de otimização como a entrada para outro, diferente da retransmissão de baixo nível que um algoritmo substitui uma função de uma metaheurísticas.

Um exemplo simples de HTH seria utilizar um algoritmo guloso para encontrar uma solução inicial para um metaheurística-S, ou gerar uma população inicial para um metaheurística-P (CRAVO, 2017), mas com a garantia da diversidade da população como mostra a figura 13.

Figura 13 – geração de soluções iniciais para metaheurística P e S utilizando a saída de um algoritmo guloso.

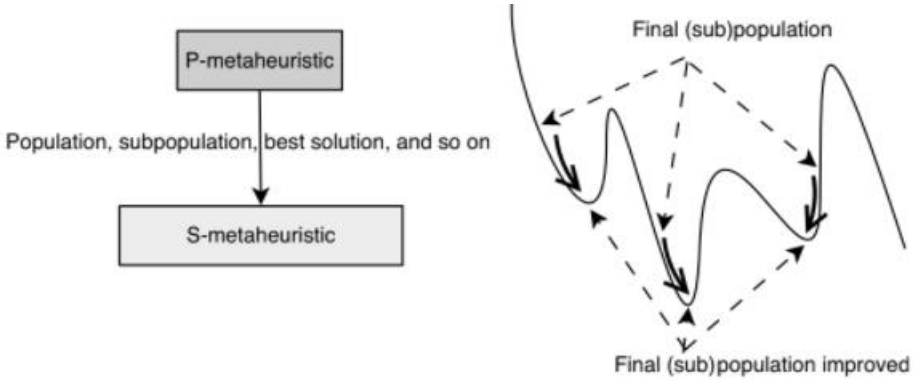


Fonte: (TALBI, 2009).

Combinar Metaheurística-P e Metaheurística-S também é possível nesse modelo, pelo fato de que ao passar das gerações a população de uma metaheurística-P fica menos

variada, o que faz com que a probabilidade de encontrar uma solução melhor que a atual diminua, ou seja, a solução caiu em uma região onde a solução ótima pode estar contida e tenta encontrar o ótimo da região através dos mecanismos de evolução natural, o que pode levar muitas gerações para acontecer. Com isso essa população pode ser usada como entrada de uma metaheurística-S como recozimento simulado ou busca tabu como mostra a figura 14, o que diminuiria o esforço computacional e o tempo que levaria para encontrar a solução.

Figura 14 – mostra uma metaheurística-P servindo como entrada para uma metaheurística-S e como os indivíduos chegam ao ótimo local através dela.



Fonte: (TALBI, 2009).

4.1.4 Híbrido de trabalho em equipe de alto nível (HTH)

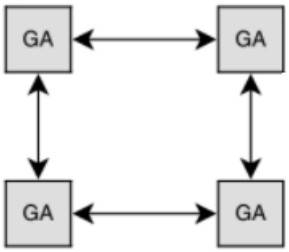
O LHT se trata de um, dois ou mais algoritmos trabalhando em paralelo realizando uma busca sobre o conjunto de possíveis soluções, e à medida que a exploração vai sendo feita eles trocam informações entre si com o intuito de encontrarem um resultado cada vez melhor.

O primeiro algoritmo que proposto seguindo esse modelo foi para a classe dos algoritmos genéticos, na literatura esse paradigma é conhecido como modelo da ilha, analogamente falando seria como pegar uma população de indivíduo e dividir uma porção deles em ilhas diferentes, com o passar do tempo esses indivíduos evoluíram separadamente, e uma vez ou outra um ou dois indivíduos migram para outras ilhas levando diversidade para evolução dos indivíduos daquele lugar como mostrado na figura 15. O modelo da ilha foi utilizado com sucesso em trabalhos como CECHINEL (2020) para alocação de tarefas em sistemas multi-robóticos.

Vários parâmetros são necessários para implementação dessa técnica como a taxa de migração, a conectividade entre as ilhas, o número de imigrantes e se um

indivíduo ao migrar some da sua ilha ou é duplicado como modelo de polinização (NGUYEN, 2019),

Figura 15 – mostra o modelo da ilha de populações evoluído paralelamente e migrando entre si.



Fonte: (TALBI, 2009).

livro "*Metaheuristics from design to implementation"* de Talbi traz conceitos mais detalhados sobre a implementação de combinação entre metaheurísticas.

4.2 Combinação de metaheurísticas com programação matemática

4.2.1 Híbrido de retransmissão de baixo nível (LRH)

A proposta desse modelo aborda o hibridismo entre a classe de algoritmos exatos com alguma metaheurística-S, a metaheurística irá desempenhar alguma função interna do algoritmo exato a fim de melhorar a procura pela solução ótima. O uso desse design pode ajudar a resolver muitos problemas utilizando métodos exatos como a estratégia de limite superior, corte, precificação e vários outros.

No trabalho de MANOUSAKIS (2021) com o objetivo de aprimorar o corte de ramos e minimizar o tempo até encontrar uma solução ótima para o algoritmo *branch-and-cut*, um algoritmo de busca tabu é utilizado para encontrar definir limites de forma heurística para serem usados para geração de soluções de início de MIP para o algoritmo BnC.

4.2.2 Híbrido de trabalho em equipe de alto nível (LTH)

Os paradigmas propostos por essa abordagem trazem a combinação de métodos exatos com metaheurísticas-P, o hibridismo dessas duas técnicas pode ser realizado de duas maneiras, onde uma metaheurística-P está contida dentro de um algoritmo exato ou uma abordagem onde um algoritmo exato está contido dentro de uma metaheurística-P.

- Algoritmo exato em uma metaheurística-P: O CMA-ES é o estado da arte para

algoritmos genéticos de codificação real, o CMA-ES é uma metaheurística-P combinado com programação matemática, onde a matriz de covariância é atualizada utilizando informações da geração anterior para gerar os indivíduos da população atual (MELO, 2019).

- Metaheurística-P em algoritmo exato: O algoritmo ES-AP (*Evolution Strategy with least squares APproximation*) é apresentado no trabalho de DA CRUZ (2017) onde uma estratégia evolutiva é combinada com um algoritmo de aproximação linear e quadrática, na qual, é realizada uma aproximação linear quadrática sobre o conjunto solução, sendo que no centro se encontra a solução pai e para o algoritmo evolutivo é passada uma solução aproximada, denominada solução filho.

4.2.3 Híbrido de retransmissão de alto nível (HRH)

Os algoritmos de alto nível trazem um design onde cada algoritmo age individualmente, ou seja, sem fazer alterações internas como os modelos de baixo nível apresentados até aqui. As características deste modelo apresentam algoritmos que são executados sequencialmente para onde a saída de um serve como entrada para o outro. Isso pode ser interpretado como sendo uma etapa de pré-processamento ou pós-processamento.

Um exemplo desse modelo aplicado em uma hibridização real ocorreu em (RIOS, 2018) onde se combinou o algoritmo de busca local *Adaptive local Search procedure* (ALSP) e uma metaheurística-P chamada Algoritmo Cientifico (ScA), onde a população resultante do ScA serve como entrada do algoritmo de programação matemática ALSP, além dessa combinação mais dois algoritmos seguindo esse modelo surgem em (RIOS, 2019)

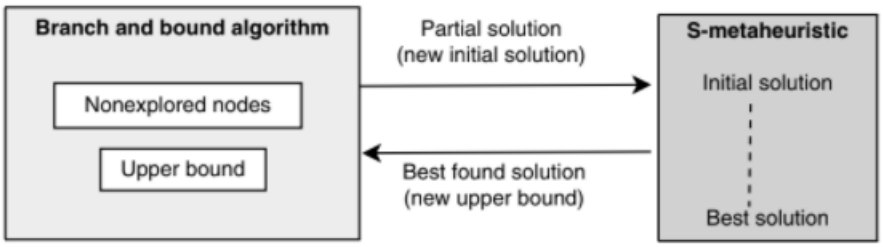
4.2.4 Híbrido de trabalho em equipe de alto nível (HTH)

Um dos maiores problemas nessa abordagem é modelar como a troca de informações entre um algoritmo de programação matemática e uma metaheurística será realizada, com isso a gama de problemas que podem ser resolvidos com essa abordagem não é tão ampla, servindo apenas para resolver problemas bem específicos.

Um exemplo aplicando essa abordagem pode ser observado combinando o algoritmo de *branch and bound* e uma metaheurística-S, uma solução parcial pode ser enviada para a

metaheurística-S onde irá passar pelo teste de aceitação da metaheurística podendo ser aceita ou não, o caminho inverso também pode ser feito, onde uma solução da metaheurística-S é enviada para o algoritmo de *branch and bound* para ser avaliada se é melhor que sua solução atual, o esquema pode ser observado na figura 16, essa troca

Figura 16 – mostra o processo de troca de informação entre um algoritmo de programação matemática é uma metaheurística-S.



Fonte: (TALBI, 2009)

de informações acontece sempre que a melhor solução é atualizada (NWANA, 2005).

4.3 Combinação de metaheurísticas com programação com restrição.

4.3.1 Híbrido de retransmissão de baixo nível (LRH)

A programação com restrição pode ser associada a alguma metaheurística-S, tal combinação traria benefícios na convergência dos algoritmos, pois a programação com restrição tende a propagar as restrições, ou seja, reduzir o espaço de busca do algoritmo.

Por exemplo, tome como variáveis o conjunto [x, y, z] onde a solução está restrita por um limite superior e inferior onde a solução ótima está contida, com o limite superior igual a 5 e o limite inferior igual a -5, para encontrar a solução ótima será usado um algoritmo de busca local, com isso na hora de gerar os indivíduos, o universo de soluções é reduzido para intervalo [-5, 5], fazendo o algoritmo convergir mais rápido para a solução ótima.

4.3.2 Híbrido de trabalho em equipe de baixo nível (LTH)

No LTH temos duas principais abordagens, onde um algoritmo de programação com restrição esta assimilado em um algoritmo metaheurística-P, a outra consiste justo no contrário sendo uma metaheurística-P embutida em um algoritmo de PR (programação com restrição), segue abaixo alguns exemplos de abordagens que podem ser utilizadas onde uma metaheurística está embutida em uma PR:

- Melhoria de um nó: que pode ser aplicado a etapa de pesquisa de uma PR, onde uma metaheurística pode ser aplicada para melhorar ou consertar um nó, pode ser utilizado também algum algoritmo guloso (CASEAU, 1999, 281-303).
- Algoritmos de pesquisa baseados em discrepância: ao percorrer os nós de uma árvore de pesquisa, os caminhos são gerados quase que de maneira gulosa, esse modelo foi usado em alguns trabalhos como CASEAU (1999, 281-303) e GINSBERG (1993, 25-43).
- Ordenação de ramos: utilizar de um algoritmo metaheurístico para definir qual nó da árvore de pesquisa deverá ganhar prioridade na hora da descida, ou seja, qual nó tem maior potencial de encontrar a solução ótima dando preferência a ele (SELLMANN, 2006).

Alguns aspectos das metaheurísticas também podem receber otimizações ao ser combinado com uma PR, isso pode ser na etapa de geração da solução inicial ou durante a fase de pesquisa adotando restrições pré-definidas pode-se conduzir o algoritmo para as regiões mais promissoras mais rapidamente.

4.3.3 Híbrido de retransmissão de alto nível (HRH)

Na forma de hibridização em LHR a saída de uma metaheurística serve como entrada de uma PR e vice-versa, as informações fornecidas por uma metaheurística para uma PR são basicamente o que é fornecido para programação matemática sendo elas limites superiores, soluções incompletas e assim por diante.

Um exemplo da aplicação de uma metodologia onde a PR fornece as informações pode ser visto em (HABET, 2002, 172-184) onde uma busca em árvore com restrições é realizada e ao chegar a certo limite de profundidade um algoritmo de busca local é usado.

4.4 Combinação de metaheurísticas com aprendizado de máquina e técnicas de mineração de dados

4.4.1 Híbrido de retransmissão de baixo nível (HRH)

As metaheurística-S clássicas não utilizam nenhum tipo de memória de iterações anteriores na busca da próxima solução, técnicas de aprendizado de máquina podem ser utilizados para guiar a convergência para boas soluções, como utilizar o algoritmo *Clustering Search* que divide o espaço de busca em clusters, e usa o *Simulated Annealing*

pra buscar dentro dessas regiões, as soluções encontradas são colocadas em clusters para serem exploradas em outro momento (GUERINE et al., 2017).

Outra maneira abordagem de LRH pode ser modificando em tempo de execução parâmetros da metaheurística-S de acordo com o que foi aprendido pelo algoritmo. Os paramentos a serem modificados são, por exemplo, o tamanho da lista tabu ou a temperatura do algoritmo de recozimento simulado.

4.4.2 Híbrido de trabalho em equipe de alto nível (LTH)

Em LTH três abordagens diferentes podem ser usadas na hibridização de algoritmos de aprendizado de máquina e metaheurística-P, onde o aprendizado de máquina pode auxiliar na pesquisa, parâmetros e otimização.

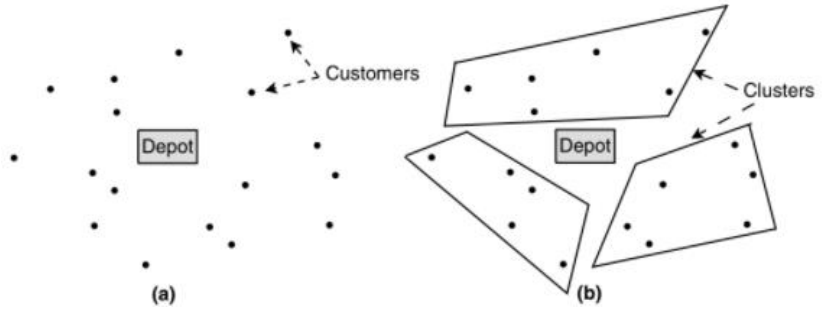
- Pesquisa: Se faz uso de técnicas de mineração de dados para aprimorar a etapa de cruzamento de uma metaheurística-P, onde o conhecimento adquirido é utilizado para melhorar a fase de recombinação dos indivíduos gerando melhores soluções, um exemplo que pode ser dado ocorre com os algoritmos genéticos civilizados, onde eles mantem informações de gerações passadas para não caírem nos mesmos erros, nesse quesito divergindo da teoria darwiniana original (SEBAG, 1997).
- Parâmetros: Qualquer um dos parâmetros de uma metaheurística-P, como a probabilidade de ocorrer mutação, crossover ou outros parâmetros como de algoritmos de colônia de formigas ou enxame de partículas, podem ser alterados nessa abordagem durante a execução do algoritmo, por uma tarefa de mineração de dados, por exemplo, ao modificar a taxa de mutação levando em consideração o progresso que se teve ao aplica-la anteriormente (SCHRIJVER, 1998), esse mesmo critério pode ser utilizado para os outros parâmetros.
- Otimização: funções de otimização usadas nas áreas de engenharia costumam exigir um esforço computacional maior, o que pode a necessidade de um software mais robusto, um modo de melhorar isso é usar o aprendizado de máquina supervisionado, onde o número de classificações realizadas pela função objetivo é reduzido, tome que a função objetivo classifique um número total de mil soluções, um algoritmo como o *k-means* é usado para rotular soluções futuras ainda não classificadas, essa técnica pode ser usada também por algoritmos de agrupamento usando imitação de aptidão (JIN. 2004, 688-699).

4.4.3 Híbrido de retransmissão de alto nível (HRH)

No HRH alguma técnica de aprendizado é previamente usada para extrair informações que podem ser adquiridas através de execuções anteriores de uma metaheurística, ou em cima de uma análise em cima do conjunto solução e assim por diante.

A população inicial de uma metaheurística-P pode ser criada através de técnicas de mineração de dados, optando por inicializar a população de maneira guiada através do conhecimento adquirido ao invés do modelo clássico usando aleatoriedade. Uma outra abordagem é executar uma mesma metaheurística várias vezes alterando apenas os paramentos, e identificar qual o conjunto de paramentos que oferece o melhor rendimento.

Figura 17 – mostra algoritmo de agrupamento sendo aplicado.



Fonte: (TALBI, 2009).

No modelo HRH metaheurísticas também podem ser otimizadas utilizando técnicas de mineração de dados, para problemas de que tratam de distância euclidiana, onde é utilizado um algoritmo de agrupamento para dividir o espaço de busca em N subespaços e para cada subespaço uma metaheurística fica responsável por achar o ótimo do grupo, e assim o ótimo global pode ser encontrado, como mostrado na figura 17 (OLIVEIRA 2004).

4.4.4 Híbrido de trabalho em equipe de alto nível (HTH)

No hibrido de trabalho de alto nível os dois algoritmos são executados de forma paralela, a metaheurística busca as soluções enquanto envia informações para um algoritmo de aprendizagem que por usa vez pega informações uteis que podem ser usadas para explorar o espaço de busca de forma mais eficiente.

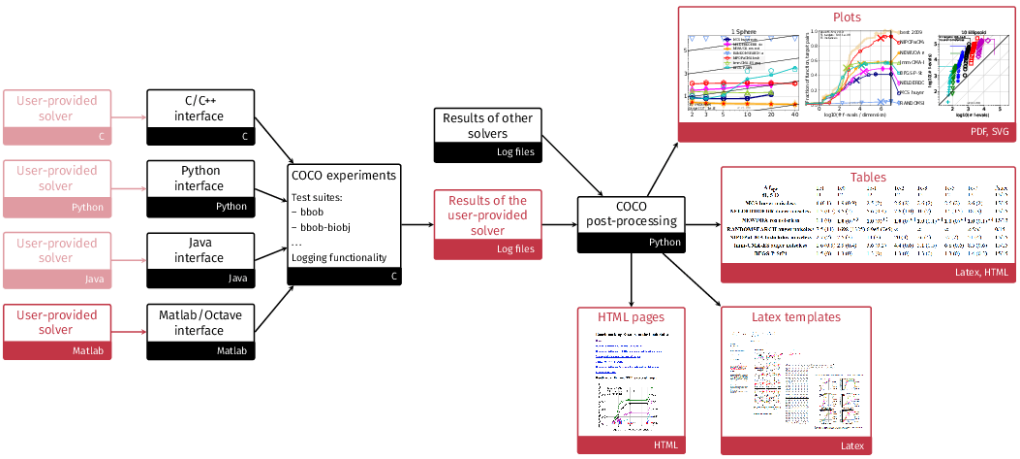
Por exemplo, o algoritmo de mineração pode ser usado para gerenciar a

população de uma metaheurística-P, com o objetivo de melhorar a intensificação e a classificação da população, fazendo com que a metaheurística-P procure por soluções em espaços de busca mais promissores, indivíduos também podem ser criados com base em informações anteriores de soluções geradas de alta qualidade (RIBEIRO, 2006, 23-41).

5 FRAMEWORK DE BENCHMARK COCO

O COCO é uma ferramenta *open source* que é usada para fazer comparações entre algoritmos de otimização em modelo de caixa preta, oferecendo uma interface programável para as linguagens C/C++, Java, *Matlab/Octave* e *Python*, dispões também de um conjunto de testes que avaliaram a eficiência do algoritmo e de tabelas e gráficos organizados em páginas HTML para melhor visualizar o rendimento do algoritmo de otimização.

Figura 18: Estrutura do framework COCO.

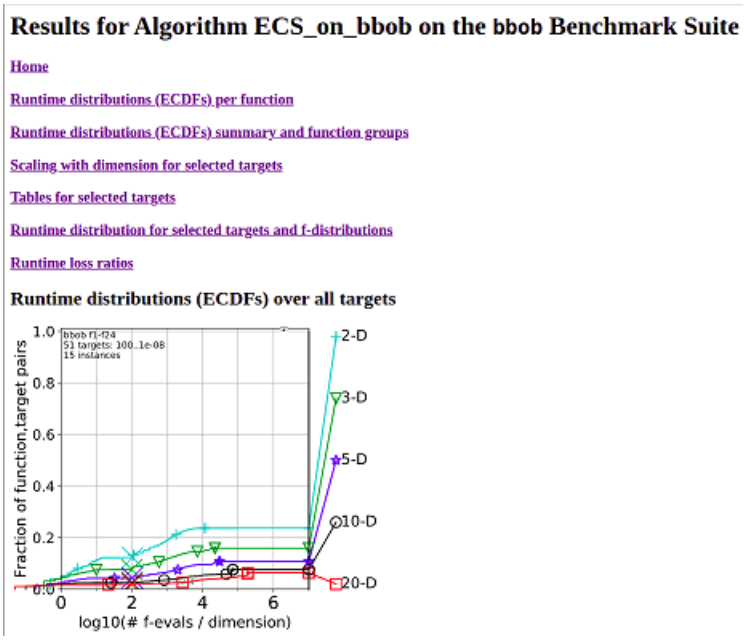


Fonte: (HANSEN, 2021, 144-144)

A figura 18 mostra a estrutura geral do framework, onde os artefatos em vermelho representam entradas do usuário ou saídas gráficas da plataforma, já os artefatos na cor preta indicam o código ou dados fornecidos pela plataforma.

Os quadros mais à esquerda em vermelho representam o algoritmo que o usuário deseja submeter a teste, em seguida temos a interface programável que se liga ao quadro que é responsável por fazer os testes, que por sua vez retornam um conjunto de dados (quadro vermelho após o quadro de testes), que é pós-processado realizado por uma implementação feita em *python*, que organiza em uma página HTML as tabelas e gráficos como mostra a figura 19.

Figura 19 – pós-processamento realizado com os dados de saída do framework COCO.



5.1 INTERFACE PROGRAMAVEL

Ao adaptar um algoritmo ao COCO o programador tem as seguintes informações de entrada que podem ser usadas para inicializar parâmetros dentro do algoritmo que está sendo adaptado:

- A quantidade de variáveis que o algoritmo irá utilizar, ou seja, sua dimensão que pode ser adquirida pela função *coco_problem_get_dimension*.
- A quantidade de objetivos, pois o COCO possui suporte para avaliação de funções multiobjetivo, a função que retorna essa informação é a *coco_problem_get_number_of_objectives*.
- O número de restrições, dada pela função coco_problem_get_number_of_constrain.
- Os limites superior e inferior (*upper bound e lower bound*) onde a região de interesse está dentro desse intervalo, as funções que retornam essas informações são respectivamente *coco_problem_get_largest_values_of_interest* e *coco_problem_get_smallest_values_of_interest*.
- Um método para geração da população inicial, pois para todas as dimensões e funções objetivos as quais o algoritmo será submetido a função *coco_problem_get_initial_solution* deve ser utilizada, para garantir a integridade

das avaliações realizadas, pois todas as funções de avaliação em todas as dimensões devem receber as mesmas instancias.

À medida que o algoritmo passa pela avaliação algumas informações são fornecidas em tempo de execução:

- Como o algoritmo que está sendo avaliado desconhece a função de avaliação que está sendo aplicada, o coco fornece o fitness da solução encontrada para que o algoritmo aplique suas técnicas de seleção, a função que retorna essa informação é a *coco_evaluate_function*.
- A informação se as restrições estão sendo respeitadas através da função *coco_evaluate_constrain.*
- E se a função alcançou a solução ótima através da função *coco_problem_final_target_hit.*

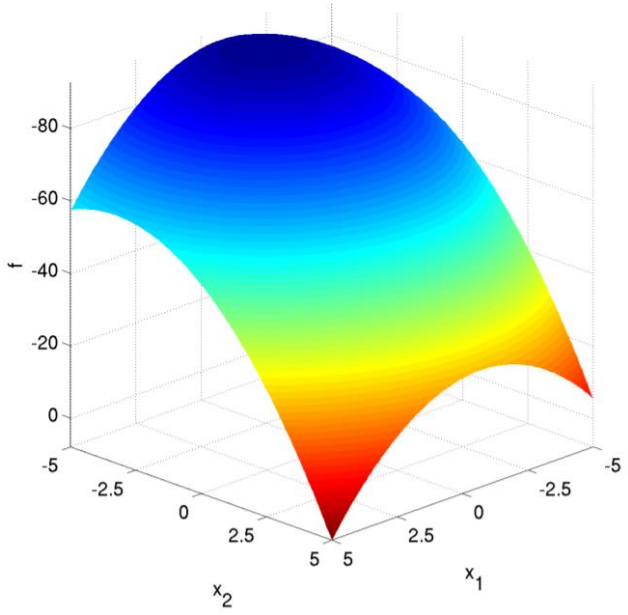
5.2 EXPERIMENTO DO COCO

Atualmente o coco tem suporte para sete tipos de testes diferentes que variam no número de funções de avaliação, se o algoritmo e single ou multiobjetivo, com restrição ou sem restrição, e funções que trabalham com números inteiros ou reais.

Dentre esses sete tipos diferentes abordaremos o *Black-Box Optimization Benchmarking* (BBOB). Que conta com um total de 24 funções de avaliação de objetivo único, o trabalho [artigo 3], mostra todas as funções de avaliação realizadas nesse teste, alguns exemplos de funções que compõe o BBOB são:

- A função da esfera que é dada pela equação $f(x) = ||z||^2 + f_{opt}$ onde temos que $z = x - x^{opt}$ o gráfico da função da esfera é mostrado na figura 19, a função da esfera possui d ótimos locais, mas apenas um ótimo global.

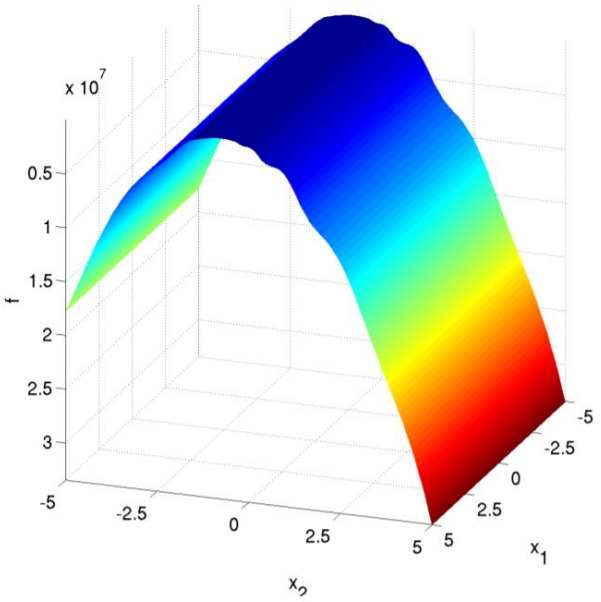
Figura 20 - comportamento da função da esfera.



Fonte: (FINCK, 2010).

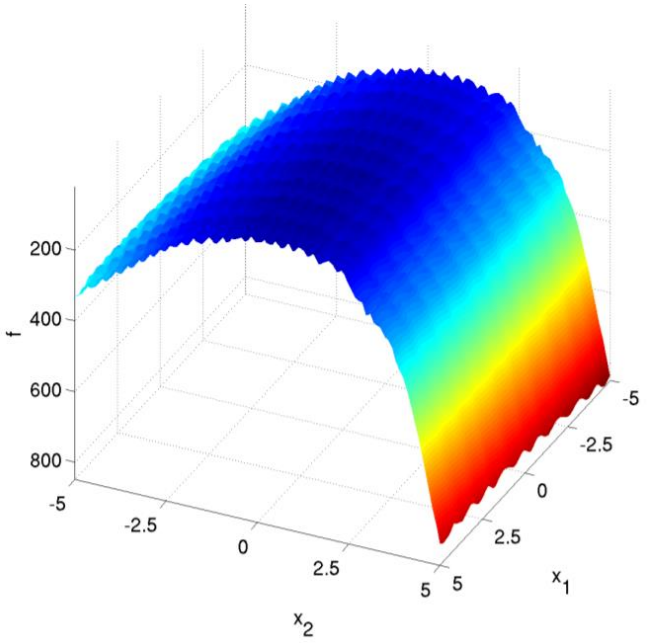
- A função elipsoidal que é dada pela seguinte equação $f(x) = \sum_{i=1}^{d} 10^{6\frac{i-1}{d-1}} * z_i^2 + f_{opt}$ onde $z = T_{osz}(x - x^{opt})$, e o gráfico da sua função pode ser visto na figura 20.
- A função *Rastrigin* que é dada pela seguinte equação $f(x) = 10\,(d - \sum_{i=1}^{d} \cos(2\pi z_i)) + \left||z|\right|^2 + f_{opt}$ onde $z = A^{10} * T_{opt}^{0.2}(T_{opt}(x - x^{opt}))$, e seu gráfico pode ser observado na figura 21.

Figura 21 – comportamento da função elipsoidal.



Fonte: (FINCK, 2010).

Figura 22 – comportamento da função *Rastrigin.*



Fonte: (FINCK, 2010).

5.3 RESULTADOS COCO

Ao finalizar a bateria de testes o COCO retorna em uma pasta um conjunto de dados que é utilizado como entrada para o pós-processamento em *python* que cria em HTML uma página como foi mostrado na figura 19, onde se encontra um conjunto de links onde se mostram gráficos e tabelas, contendo respetivamente:

- A distribuição do tempo de execução por função
- A distribuição de tempo de execução por dimensão
- Número médio de avaliações f para atingir a meta
- Tempo de execução esperado em número de avaliações de função
- Funções de distribuição cumulativa empírica
- Taxas de perda

O algoritmo *Realcode-GA* foi adaptado para rodar no COCO sua implementação se encontra no *github* do LACMOR (Laboratório de Aprendizagem Computacional, Métodos de Otimização e Robótica), onde outros algoritmos como CMA-ES, ECS e C-GRASP também foram adaptados. Alguns dos resultados obtidos da execução do *Realcode-GA* encontram-se nas figuras 23, 24 e 25.

Figura 23 – Execução da adaptação do Realcode-GA ao COCO, mostra a distribuição do tempo de execução em 6 tipos de funções objetivos diferentes, o tempo de execução é contado om base em quantas vezes a função objetivo foi chamada.

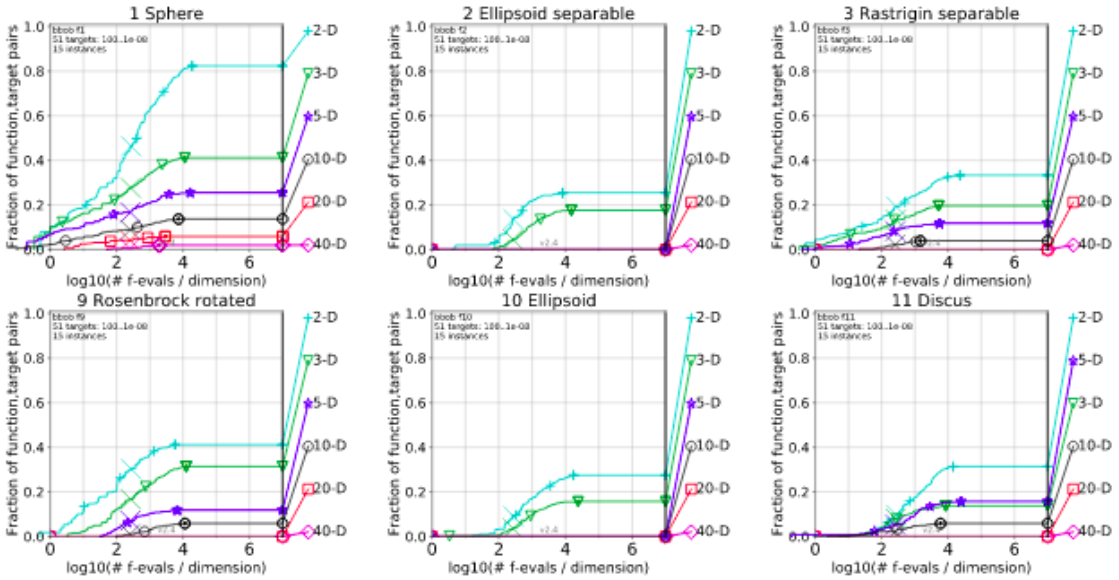


Figura 24 – Execução da adaptação do Realcode-GA ao COCO, mostra o tempo de execução divido por dimensões.

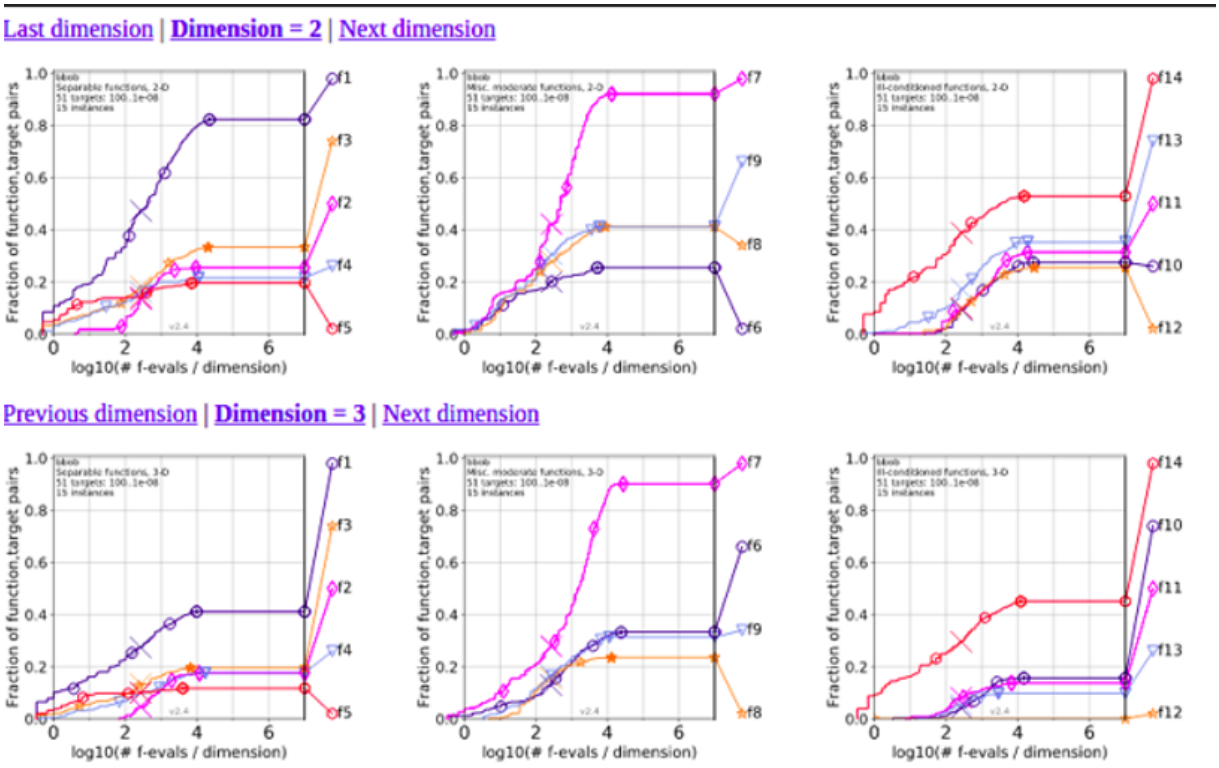
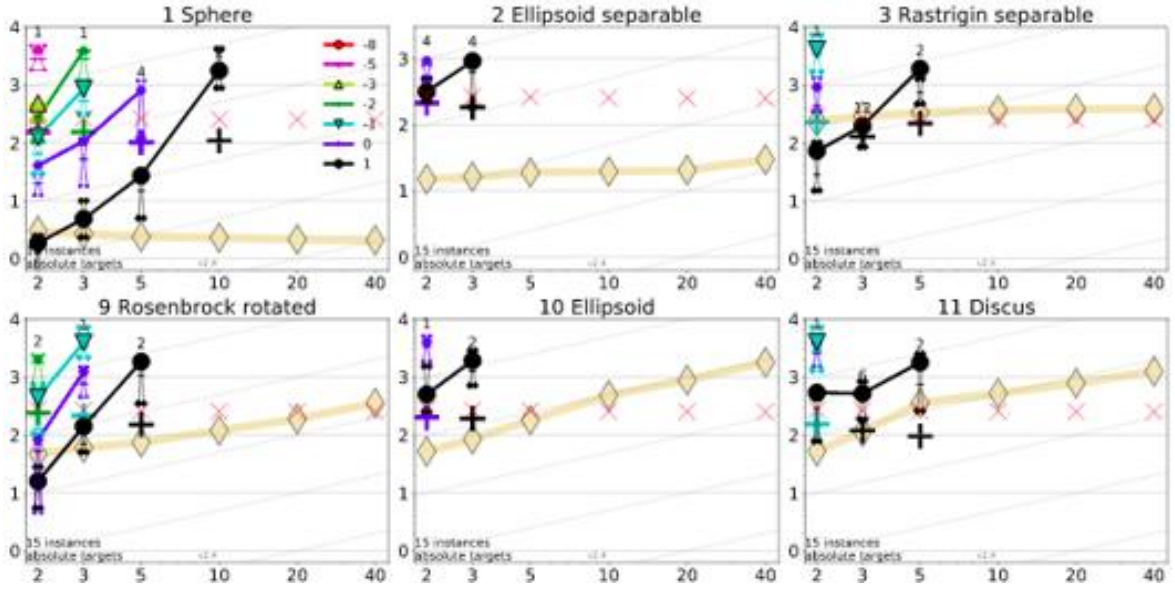


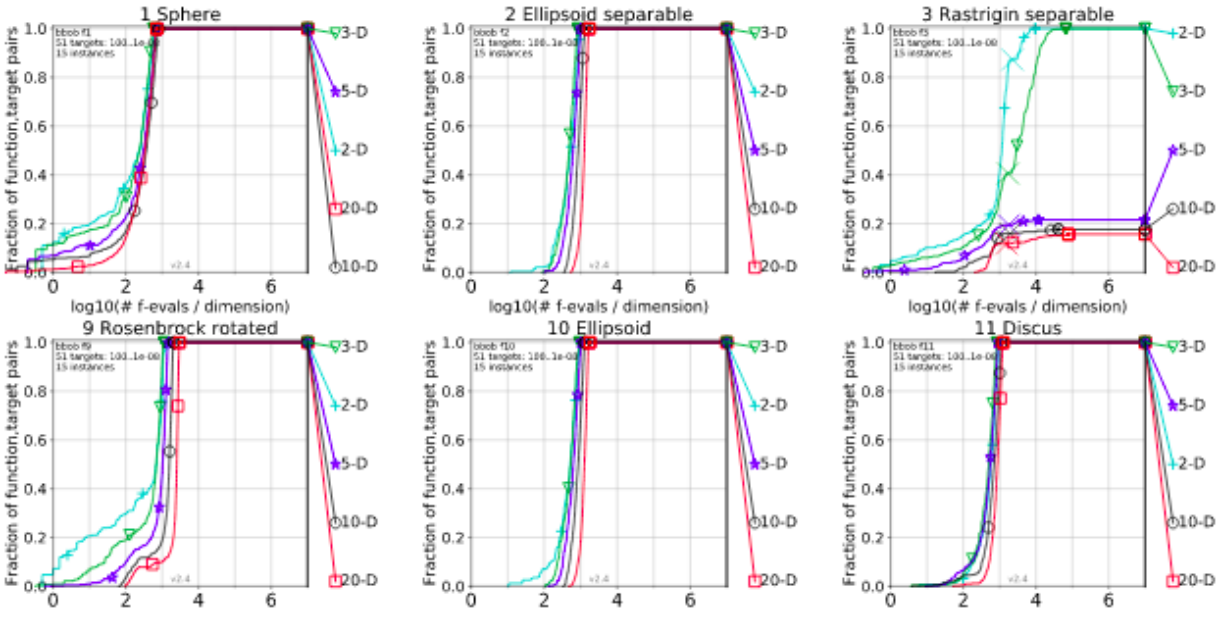
Figura 25 – mostra o número médio que o COCO precisou executar até encontrar o resultado ótimo.



5.3.1 Comparando resultados

Com os resultados do *Realcode-GA* podemos fazer um comparativo com o algoritmo CMA-ES que é o estado da arte para algoritmos genéticos de codificação real, o CMA-ES é uma metaheurística-P combinado com programação matemática, assim pertencendo à classe dos algoritmos LTH.

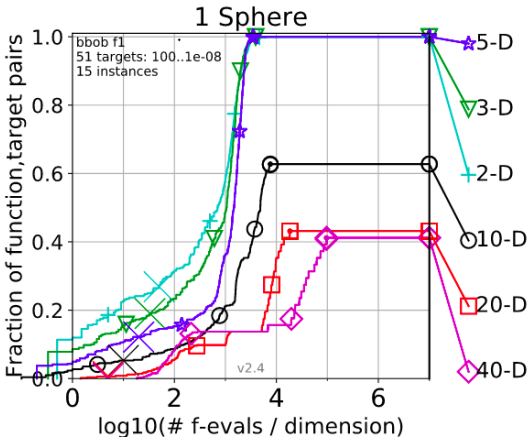
Figura 26 – tempo de execução até encontrar a solução ótima do algoritmo CMA-ES.



Ao compararmos os resultados da figura 23 e 26 é possível verificar que o

algoritmo que ao fazer uso de hibridização houve uma melhora sinificativa se comparado ao algoritmo genetico de codificação real puro. Outro algoritmo observado foi o ECS, no qual, o conjunto solução utiliza trecnicas de mineração de dados, separando o conjunto solução em *clusters*, onde foi possivel encontrar resultados muito promissores para a função de avaliação da esfera com o pode ser visto na figura 27.

Figura 27 – avaliação do tempo de execução do algoritmo ecs para a função da esfera.



O trabalho de PEREIRA (2018) também trás técnicas de hibridização de algoritmos, que podem ser classificados como combinação de metaheurísticas LTH ao combinar metaheurísticas-P com Metaheurísticas-S para resolver o Job-Shop Scheduling Problem (JSSP) o que mostrou trazer resultados bastante efetivos, em BERNARDINO (2008) também foi possível encontrar soluções eficazes ao combinar um algoritmo genético com algoritmo de sistema imunológico artificial.

6 CONCLUSÃO

A taxonomia proposta por TALBI (2009) consegue formalizar de maneira clara e intuitiva a forma como vemos algoritmos híbridos, entendendo de forma concreta as fases de concepção dos algoritmos trazidos como exemplos à medida que nos aprofundamos na leitura, além do framework COCO que consegue trazer dados comparáveis entre algoritmos o que nos permite verificar a eficiência de cada um.

A princípio houve uma certa dificuldade em se encontrar trabalhos que formalizassem o que é um algoritmo hibrido, pois o comum é encontrar artigos mostrando a hibridização de dois algoritmos, sem obedecer a algum paradigma

predisposto. Com isso a identificação de qual estrutura o algoritmo utiliza fica mais difícil, pois depende de uma leitura mais completa dos trabalhos pesquisados além de um entendimento maior nos casos onde se utilizam conceitos matemáticos mais complexos.

Mas algumas lacunas ainda se encontram abertas para trabalhos futuros, onde o algoritmo COCO pode ser utilizado para verificar a eficiência de novos algoritmos concebidos, ou ainda verificar os algoritmos tratados aqui em outros *frameworks* de *benchmarking*, ou ainda ser realizada pesquisas em cima dos ramos de hibridização não abordados nesse trabalho, como hibridização com técnicas de programação estatística e outros ramos da computação evolutiva, como os algoritmos baseados em redes neurais.

7 REFERÊNCIAS

BARBOSA, Martins Eduardo Carlos. **Algoritmos bio-inspirados para solução de problemas de otimização**. 2017. Monografia (Pós-graduação Computação) – Centro de informática, Universidade Federal de Pernambuco, Pernambuco, Recife, 2017. 10.1162/106365601750190398.

BERNARDES, Lucas Correia et al. Metaheurísticas para o problema de roteamento de veículos capacitados: o algoritmo híbrido de otimização por colônia de formigas e a busca híbrida em grande vizinhança. 2018.

BLUM, C.; RAIDL, G. R. Further hybrids and conclusions. In: Hybrid Metaheuristics. [S.l.]: Springer, 2016. p. 127–136.

BERNARDINO, Heder Soares. Hibridização de algoritmos genéticos e sistemas imunológicos artificiais para problemas de otimização com restrições em engenharia. 2008. Dissertação de Mestrado (Modelagem Computacional). Universidade Federal de Juiz de Fora. 2008

CATARINA, A. S.; BACH, S. L. Study of the effect of the genetic parameters about the optimized solution and time of convergence in genetic algorithms with binary and real codification. Acta Scientiarum. Technology, v. 25, n. 2, p. 147-152, 2008.

CASEAU, Yves; LABURTHE, François. Heuristics for large constrained vehicle routing problems. **Journal of Heuristics**, v. 5, n. 3, p. 281-303, 1999.

CECHINEL, Alan Kunz et al. Arquitetura baseada em algoritmo genético utilizando modelo de ilhas para alocação de tarefas em sistemas multi-robóticos. 2020.

CORRÊA, Martins Raquel Claudia. **Analise e síntese de regras de adaptação em estratégias evolutivas usando funções lyapunov estocásticas**. 2019. Monografia (Pós-graduação Matemática e Computação) – Programa de Pós-graduação em modelagem matemática e computacional, CEFET-MG, Belo Horizonte, Minas Gerais, 2019.

CRAVO, Gildasio Lecchi; AMARAL, André Renato Sales. Um algoritmo grasp com fase de diversificação para problema de layout de facilidades em fila única. In: **Annals of the XLIX SBPO-Simpósio Brasileiro de Pesquisa Operacional, Rio de Janeiro, Brazil**. 2017.

DA CRUZ, Andre Rodrigues. Operadores de busca local baseada em aproximação linear-quadrática para otimização de funções ruidosas. 2017.

DA MATA, Diego A.; BESSANI, Michel. Restabelecimento de Sistemas de Distribuição-Abordagem Hibrida de Algoritmo Genético e Busca local. **Simpósio Brasileiro de Sistemas Elétricos-SBSE**, v. 1, n. 1, 2020.

FEO, Thomas A.; RESENDE, Mauricio GC. Greedy randomized adaptive search procedures. **Journal of global optimization**, v. 6, n. 2, p. 109-133, 1995.

FERREIRA, Kamyla Maria; DE QUEIROZ, Thiago Alvez. Uma Abordagem de Recozimento Simulado com Busca Local para o Problema Integrado de Localizaç ao e Roteamento. 2015

FINCK, Steffen et al. **Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions**. Technical Report 2009/20, Research Center PPE, 2009. Updated February, 2010.

GASPAR-CUNHA, António. et al (coord.). Manual de computação evolutiva e metaheurística. Imprensa da Universidade de Coimbra. Editora da Universidade Federal de Minas Gerais. 2012.

GECCO Workshop on Black-Box Optimization Benchmarking (BBOB 2021) - focus on mixed-integer problems. The BBOB workshop series.19, Abril, 2021. Disponível em:

GINSBERG, Matthew L. Dynamic backtracking. **Journal of artificial intelligence research**, v. 1, p. 25-46, 1993.

GOMORY, Ralph E. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. In: **50 Years of Integer Programming 1958-2008**. Springer, Berlin, Heidelberg, 2010. p. 77-103.

GUERINE, Marcos et al. Heurística Híbrida com Mineração de Dados para o Problema de Agrupamento Capacitado com Centro Geométrico. 2017.

GUIMARAES, Frederico Gadelha. Aprendizagem e busca local em algoritmos meméticos para projeto assistido por computador. 2008. Tese de Doutorado (Engenharia Elétrica). Universidade Federal de Minas Gerais.

HABET, Djamal et al. A hybrid approach for SAT. In: **International Conference on Principles and Practice of Constraint Programming**. Springer, Berlin, Heidelberg, 2002. p. 172-184.

HANSEN, Nikolaus et al. COCO: A platform for comparing continuous optimizers in a black-box setting. **Optimization Methods and Software**, v. 36, n. 1, p. 114-144, 2021.

Hansen N. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. **Cornell University,** 2016. Disponível em: https://arxiv.org/abs/1603.08785v4. Acesso em 07/07/2021.

HANSEN, N. e OSTERMEIER, A. Completely Derandomized Self-Adaptation in Evolution Strategies, in Evolutionary Computation, vol. 9, no. 2, pp. 159-195, June 2001, doi: 10.1162/106365601750190398.

HANSEN, Pierre; JAUMARD, Brigitte; SAVARD, Gilles. New branch-and-bound rules for linear bilevel programming. **SIAM Journal on scientific and Statistical Computing**, v. 13, n. 5, p. 1194-1217, 1992.

HOLLAND, John Henry et al. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. MIT press, 1992.

JIN, Yaochu; SENDHOFF, Bernhard. Reducing fitness evaluations using clustering techniques and neural network ensembles. In: **Genetic and Evolutionary Computation Conference**. Springer, Berlin, Heidelberg, 2004. p. 688-699.

LACMOR, LACMOR-UFMA: coco. Disponivel em: <https://github.com/LACMOR-UFMA/coco> acesso em: 27/09/2021.

LIMA, Telma Woerle de. **Algoritmos evolutivos para predição de estruturas de proteínas**. 2006. Tese de Doutorado (Ciências Matemáticas e de Computação). Universidade de São Paulo. 2006.

LONG, Qiang; WU, Changzhi. A hybrid method combining genetic algorithm and Hooke-Jeeves method for constrained global optimization. **Journal of Industrial & Management Optimization**, v. 10, n. 4, p. 1279, 2014.

LUUS, Rein; HENNESSY, Denis. Optimization of fed-batch reactors by the Luus−Jaakola optimization procedure. **Industrial & engineering chemistry research**, v. 38, n. 5, p. 1948-1955, 1999.

MANOUSAKIS, Eleftherios et al. Improved branch-and-cut for the Inventory Routing Problem based on a two-commodity flow formulation. **European Journal of Operational Research**, v. 290, n. 3, p. 870-885, 2021.

MELO, Mario Gomes de. Adaptação Local da Matriz de Covariância Guiada por Mecanismos de Exploração em Estratégias de Evolução. 2019. Dissertação de Mestrado (Ciência da Computação). Universidade Federal de Pernambuco. 2019

MONARD, Maria Carolina; BARANAUSKAS, José Augusto. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003.

NGUYEN, Trong-The; PAN, Jeng-Shyang; DAO, Thi-Kien. An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network. **Ieee Access**, v. 7, p. 75985-75998, 2019.

NWANA, V.; DARBY-DOWMAN, Ken; MITRA, Gautam. A co-operative parallel heuristic for mixed zero–one linear programming: Combining simulated annealing with branch and bound. **European journal of operational research**, v. 164, n. 1, p. 12-23, 2005..

OLIVEIRA L.E. **Planejamento dinâmico de expansão em sistemas de transmissão de energia elétrica via algoritmos híbridos de otimização.** 2017. 152p. Dissertação(Mestrado em engenharia elétrica) - Faculdade de tecnologia. Universidade de Brasília, Brasília. 2017.

OLIVEIRA, Alexandre CM. Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuos e discretos. **Algoritmos evolutivos híbridos com detecção de regiões promissoras em espaços de busca contínuos e discretos**, 2004.

OMRAN, Mahamed GH; MAHDAVI, Mehrdad. Global-best harmony search. **Applied mathematics and computation**, v. 198, n. 2, p. 643-656, 2008.

O'REILLY, U.-M.; OPPACHER, Franz. Hybridized crossover-based search techniques for program discovery. In: **Proceedings of 1995 IEEE International Conference on Evolutionary Computation**. IEEE, 1995. p. 573-578.

PEREIRA JUNIOR, Geraldo. Algoritmos evolutivos híbridos aplicados no sequenciamento de produção em uma indústria de alimentos. 2018. Dissertação de Mestrado (Informática). Universidade Tecnológica Federal do Paraná.2018

POZO, Aurora et al. Computação evolutiva. **Universidade Federal do Paraná, 61p.(Grupo de Pesquisas em Computação Evolutiva, Departamento de Informática-Universidade Federal do Paraná)**, 2005.

RIBEIRO, Marcos Henrique; PLASTINO, Alexandre; MARTINS, Simone L. Hybridization of GRASP metaheuristic with data mining techniques. **Journal of Mathematical Modelling and Algorithms**, v. 5, n. 1, p. 23-41, 2006.

RIOS, Brenner Humberto Ojeda. **Hibridização de meta-heurísticas com métodos baseados em programação linear para o problema do caixeiro alugador**. 2018. Dissertação de Mestrado. Brasil.

SEBAG, Michèle; SCHOENAUER, Marc; RAVISÉ, Caroline. Toward civilized evolution: Developing inhibitions. 1997.

SELLMANN, Meinolf; ANSÓTEGUI, Carlos. Disco-novo-gogo: Integrating local search and complete search with restarts. In: **proceedings of the 21st national conference on Artificial intelligence-Volume 2**. 2006. p. 1051-1056.

SILVA R.S. Aplicação de um algoritmo híbrido estocástico determinístico na solução de um problema de direção automotiva. **XXXVII encontro nacional de engenharia de produção**. Santa Catarina. Outubro de 2017.

SCHRIJVER, Alexander. **Theory of linear and integer programming**. John Wiley & Sons, 1998.p

TALBI, El-Ghazali. **Metaheuristics: from design to implementation**. John Wiley & Sons, 2009.

TAILLARD, Éric D.; GAMBARDELLA, Luca Maria. **Adaptive memories for the quadratic assignment problem**. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

YANG, X.-S. Nature-inspired metaheuristic algorithms. [S.l.]: Luniver press, 2010.

YUNES, Tallys Hoover et al. Problemas de escalonamento no transporte coletivo: programação por restrições e outras tecnicas. 2000.