

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Otimização Competitiva Imperialista para
Previsão Espacial Urbana

Marcel Mendonça Grilo

Itajubá, fevereiro de 2017

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Marcel Mendonça Grilo

Otimização Competitiva Imperialista para
Previsão Espacial Urbana

Dissertação submetida ao Programa de Pós-Graduação em
Ciência e Tecnologia da Computação como parte dos requisitos
para obtenção do Título de Mestre em Ciência e Tecnologia da
Computação

Área de Concentração: Matemática da Computação

Orientador: Prof. Dr. Carlos Henrique Valério de Moraes

fevereiro de 2017
Itajubá - MG

Agradecimentos

Ao meu professor e orientador Prof. Carlos Henrique Valério de Moraes, pelo apoio, sugestões, críticas construtivas e excelente exemplo profissional.

Em especial aos meus queridos pais, meus irmãos e família, os quais foram responsáveis pela minha formação e educação, sempre presentes na minha vida.

À minha esposa pelo apoio, suporte e companhia durante este período.

Aos meus excelentes amigos, com os quais sempre pude contar.

À CAPES pelo financiamento através da bolsa de estudos.

There are no limits when you are surrounded by people who believe in you, or by people whose expectations are not set by the short-sighted attitudes of society, or by people who help to open doors of opportunity, not close them

Neil deGrasse Tyson, The Sky is Not the Limit: Adventures of an Urban Astrophysicist

Resumo

A previsão espacial de crescimento de densidades em regiões urbanas apresenta um grande desafio por ser muito utilizada em planejamentos de expansão de áreas urbanas, sendo estas densidades relacionadas ao tipo de planejamento de expansão que se deseja fazer, carga elétrica, populacional, hídrico, ou qualquer outro. Apresenta-se então uma metodologia para previsão de áreas urbanas em alta resolução, a qual melhora a visualização, análise e inferência das informações de densidade para o planejamento em um futuro próximo. Tal metodologia converte os dados de entrada em mapas de quadrículas de alta resolução, que por sua vez são divididos em regiões maiores, onde cada uma destas regiões terá seu crescimento esperado definido por matrizes de convolução e fatores de ponderação, que buscam por características históricas de cada região. A obtenção das melhores características de crescimento das regiões vem através do processamento do algoritmo imperialista competitivo que busca o melhor conjunto de parâmetros, onde tais parâmetros são compostos pelos valores das matrizes de convolução e dos fatores de ponderação e serão utilizados para prever o crescimento da região. A natureza dos dados utilizados nos testes direciona a previsão para densidade de carga elétrica, assim é possível obter uma previsão espacial de crescimento em alta resolução e maior precisão que, neste caso, pode ser usado como um importante fator no planejamento de expansão de carga, podendo até mesmo ser usado no planejamento e aplicações dos conceitos de *smartgrids* (redes elétricas inteligentes) na área urbana definida.

Abstract

The spatial forecasting of density growth in urban regions presents a great challenge because it is widely used in expansion planning of urban areas, these densities can be related to the type of expansion planning that is desired, electric, population, water, or any other. It is presented a methodology for forecasting urban areas growth in high resolution, which improves the visualization, analysis and inference of density information for planning in the near future. Such methodology converts the input data into high-resolution grid maps, which in turn are divided into larger regions, where each of these regions will have its expected growth defined by convolution matrices and weighting factors, which search for historical characteristics of each region. Obtaining the best growth characteristics of the regions comes through the processing of the imperialist competitive algorithm (ICA) that searches for the best set of parameters, where these parameters are composed by the values of the convolution matrices and the weighting factors and will be used to forecast the growth of the region. The nature of the data used in the tests directs the forecasting for electric load density, so it is possible to obtain a spatial forecast of growth in high resolution and greater precision that, in this case, can be used as an important factor in the planning of load expansion, and can even be used in the planning and application of the concepts of smartgrids in the defined urban area.

Listas de ilustrações

Figura 1 – <i>GPWv4: Population Density - 2015</i> com foco na cidade de Taubaté	15
Figura 2 – Regiões e Quadrículas	17
Figura 3 – Distribuições distintas do mesmo erro global, detalhado em (ARANGO; LAMBERT-TORRES, 2000) e definido por (WILLIS; ENGEL; BURI, 1995)	19
Figura 4 – Esferóide oblato	21
Figura 5 – Fluxograma ICA canônico	27
Figura 6 – Movimento da colônia para seu imperialista	29
Figura 7 – Comparação $\beta < 1$ e $\beta > 1$	29
Figura 8 – Movimento da colônia para seu imperialista com desvio de direção	30
Figura 9 – Tomada de império por colonia	31
Figura 10 – Filtragem por convolução	35
Figura 11 – Paradigmas de programação	41
Figura 12 – Classe Country	43
Figura 13 – Interface IFitness	44
Figura 14 – Classe Abstrata StopCondition	47
Figura 15 – Implementação das condições de parada	48
Figura 16 – Classe ImperialistCompetition	49
Figura 17 – Diagrama de classes resumido do ICA	50
Figura 18 – Fluxograma da etapa de inicialização no método Run	53
Figura 19 – Fluxograma sobre a distribuição de colônias entre os impérios	54
Figura 20 – Fluxograma ICA detalhado	55
Figura 21 – Fluxograma sobre as condições de parada	56
Figura 22 – Fluxograma competição imperialista	60
Figura 23 – Diagrama de classes simplificado do problema G1	61
Figura 24 – Vetor de atributos dos países do problema G1	62
Figura 25 – Diagrama de classes simplificado do problema G2	64
Figura 26 – Vetor de atributos dos países do problema G2	64
Figura 27 – Exemplo Problema G2	65
Figura 28 – Resultados do problema G2	66
Figura 29 – Movimento Linear	68
Figura 30 – Movimento Original	70
Figura 31 – Fluxograma Movimento Original ICA	72
Figura 32 – Random Triangular	74
Figura 33 – Testes com movimento refinado	75
Figura 34 – Testes Visão Imperial Distorcida	78

Figura 35 – Movimento combinado	79
Figura 36 – Ilustração da montagem do mapa de quadrículas.	86
Figura 37 – Ilustração do mapa de quadrículas particionado.	89
Figura 38 – Processamento dos mapas de previsão parte 1.	95
Figura 39 – Processamento dos mapas de previsão parte 2.	96
Figura 40 – Vetor de atributos do país.	99
Figura 41 – Exemplo de ponderação - MapaMatriz T0.	104
Figura 42 – Exemplo de ponderação - 1 ponderação	104
Figura 43 – Exemplo de ponderação - 2 ponderações	105
Figura 44 – Exemplo de ponderação - 3 ponderações	106
Figura 45 – Imagem da previsão estourada	111
Figura 46 – Convolução aperiódica versus truncada	112
Figura 47 – Apresentação dos mapas reais de 2013, 2014 e 2015 comparando com a previsão por linha de tendência e com o método proposto para matrizes de convolução de ordem 3, 5 e 7	117

Lista de tabelas

Tabela 1 – Erros de previsão 116

List of Algorithms

1	Algoritmo de convolução de imagens	36
2	Pseudocódigo método <i>Eval</i> do problema G1.	62
3	Pseudocódigo método <i>Eval</i> do problema G1.	65
4	Cálculo das distâncias máximas para Visão Imperial Distorcida.	76
5	Algoritmo Transformação do vetor de atributos do país.	100
6	Avaliação do país.	101
7	Algoritmo função de previsão	108
8	Algoritmo condição de parada para o ICA	113

Glossário

AG	<i>Algoritmo Genético</i>
DAACs	<i>Distributed Active Archive Centers</i>
EOSDIS	<i>Earth Observing System Data and Information System</i>
GA	<i>Genetic Algorithm</i>
GPWv4	<i>Gridded Population of the World, Version 4</i>
ICA	<i>Imperialist competitive algorithm</i>
LINQ	<i>Language Integrated Query</i>
NASA	<i>National Aeronautics and Space Administration</i>
PSO	<i>Particle Swarm Optimization</i>
SEDAC	<i>Socioeconomic Data and Applications Center</i>

Sumário

Lista de ilustrações	6
Lista de tabelas	8
Sumário	11
1 INTRODUÇÃO	13
1.1 Contexto	13
1.2 Objetivos	16
1.3 Revisão Bibliográfica	16
1.4 Estrutura da Dissertação	24
2 METODOLOGIA	25
2.1 Algoritmo Imperialista Competitivo	25
2.2 A operação de convolução e convolução de imagens	33
3 DESENVOLVIMENTO	38
3.1 ICA Orientado a Objetos	38
3.1.1 Modelagem e desenvolvimento	40
3.1.2 Exemplos de aplicação do ICA genérico	61
3.2 Análise da operação de movimento do ICA	67
3.2.1 Movimento refinado	71
3.2.2 Visão imperial distorcida	75
3.3 Paralelização do ICA	80
3.4 Formulação do ambiente	82
3.5 Inicialização do ICA	91
3.5.1 Avaliação dos países	93
3.5.2 A função de avaliação	96
3.5.2.1 Atributos dos Países	98
3.5.3 Implementando a Função de Avaliação	99
3.5.4 A ponderação das matrizes de convolução	103
3.5.5 A Etapa de Previsão	106
4 EXPERIMENTOS E RESULTADOS	109
4.0.1 Modelagem do ambiente	109
4.0.2 Testes sobre cada regiões	110
4.0.3 Testes sobre todas regiões	111

5	CONCLUSÃO	118
5.1	Colaborações do Trabalho	120
5.2	Trabalhos Futuros	120
	REFERÊNCIAS	122

1 Introdução

1.1 Contexto

As metodologias de previsão espacial dividem a área de estudo em pequenas subáreas, que podem ter tamanhos e formas distintas, ou podem ser subáreas homogêneas e de lados iguais (quadrículas) e quadradas. As quadrículas são usadas para formar uma grade, ou um mapa de quadrículas, que apresenta vantagens em relação às outras formas de divisão, pela possibilidade de se aumentar a resolução, dependendo apenas da entrada de dados.

Tal divisão em mapa de quadrículas (grade) tem sido aplicada em diversos métodos de previsão espacial de carga (WILLIS, 2002), (MELO; CARRENO; PADILHA-FELTRIN, 2012), (ARANGO; LAMBERT-TORRES, 2004). Assim é possível obter uma representação gráfica, em forma de uma imagem, da distribuição de densidade de cargas, de modo a determinar um crescimento, seja de carga ou de qualquer outra forma de densidade em um ponto da grade, que seja esperado para aquela pequena área.

Para se fazer o planejamento espacial de áreas urbanas é necessário a criação de uma metodologia que forneça informações bem definidas para o planejamento de expansão destas regiões urbanas, tendo em vista que toda informação regional possa ser usada para definir melhores localizações e minimizar os custos de implantação de sistemas e equipamentos. Por exemplo, no caso de densidade de carga, os sistemas elétricos de distribuição podem ter ser remanejados ou instalados em regiões urbanas de forma a minimizar os custos destes tipos de atividades (WILLIS; ROMERO, 2007).

Alguns métodos de previsão espacial executam simulações (ARANGO; LAMBERT-TORRES, 2004), (CARRENO; ROCHA; PADILHA-FELTRIN, 2011) de crescimento partindo de dados diferenciados no tempo, de modo que o crescimento de cada quadrícula não dependa de outros valores, resultando assim, em metodologias capazes de identificar curvas de crescimento para regiões seguindo heurísticas que usam apenas destes dados diferenciados.

Outras técnicas de previsão de regiões urbanas consideram a taxa de crescimento das pequenas regiões (quadrícula) como sendo a mesma da taxa de habitantes naquela região, provindos de mapas de uso da terra (WU; LU, 2002). Outras variações ainda trazem métodos que avaliam a chance de uma região crescer, fazendo cálculos probabilísticos (MELO; PADILHA-FELTRIN; CARRENO, 2015), (ARANGO; LAMBERT-TORRES, 2004) para definir se uma determinada região irá ou não se desenvolver.

Geralmente as técnicas de previsão espacial são aplicadas à densidade de carga,

e englobamos dados de toda uma área urbana(WILLIS; ROMERO, 2007), de modo que algumas regiões, mais externas ou distantes são ignoradas para não gerar ruído na previsão. Porém, todas as regiões de uma determinada área devem ser consideradas por terem a capacidade de influenciar no crescimento de carga das áreas mais centrais (WILLIS, 2002).

As quadrículas são regiões quadradas e que devem representar o mundo real em posição, sendo que se pode atribuir mais valores àquela região, como densidade de carga. O tamanho da quadrícula define se uma grade de quadrículas (ou mapa de quadrículas) estará em alta resolução ou em baixa resolução. SEDAC o Centro de Dados e Aplicações Socioeconômicas, é um dos Centros de Arquivos Ativos Distribuídos (DAACs) no Sistema de Dados e Informação do Sistema de Observação da Terra (EOSDIS) da Administração Nacional de Aeronáutica e Espaço dos Estados Unidos (NASA). Com foco nas interações humanas no meio ambiente, o SEDAC tem a missão de desenvolver e operar aplicações que suportem a integração de dados socioeconômicos e de ciências da terra e serve como um "Portal de Informação" entre ciências da terra e ciências sociais.

Neste centro de dados apresentam-se diversos mapas em quadrículas em baixa resolução (por exemplo, o mapa de densidade populacional *GPWv4: Population Density - 2015* (GRIDDED..., 2016) como pode ser visto na figura 1) sobre diversos tópicos, tendo suas quadrículas com o área aproximada de 1 milha². Em baixa resolução é possível identificar grosseiramente grandes regiões de crescimento, porém vários detalhes são deixados de lado. Neste trabalho busca-se um refinamento na previsão, usando quadrículas de áreas muito menores, porém existem alguns problemas em se usar uma alta resolução nos estudos de previsão, relacionados (LONGLEY; BATTY, 1996), como a demanda por uma alta quantidade de informações para a caracterização da grade, grande esforço computacional no processamento dos dados, que geralmente depende do número da entrada de dados e da técnica de previsão usada. Porém, permite a criação em alta resolução do mapa de quadrículas, melhorando a visualização, análise e diversos outros fatores para a previsão.

O método de efetuar uma previsão espacial de alta resolução deve considerar como entrada diversos fatores (WILLIS, 2002), requer uma grande quantidade de dados, e neste caso, leva em consideração o crescimento local através da obtenção de características regionais usando matrizes de convolução e fatores de ponderação para se obter o crescimento de cada pequena região (quadrícula) de modo que não se ignore nenhuma região e também que tais regiões não ignoradas não venham a causar erros na previsão das demais regiões. Assim, divide-se o mapa de quadrículas em sub regiões compostas por quadrículas, de modo que a previsão ocorra naquela região independente das demais, evitando assim, que regiões mais distantes prejudiquem o processamento de previsão das demais regiões.

A previsão espacial de carga de cada região é feita através do uso de uma função de previsão que tem como entrada, um mapa base (WILLIS, 2002) e valores otimizados pelo

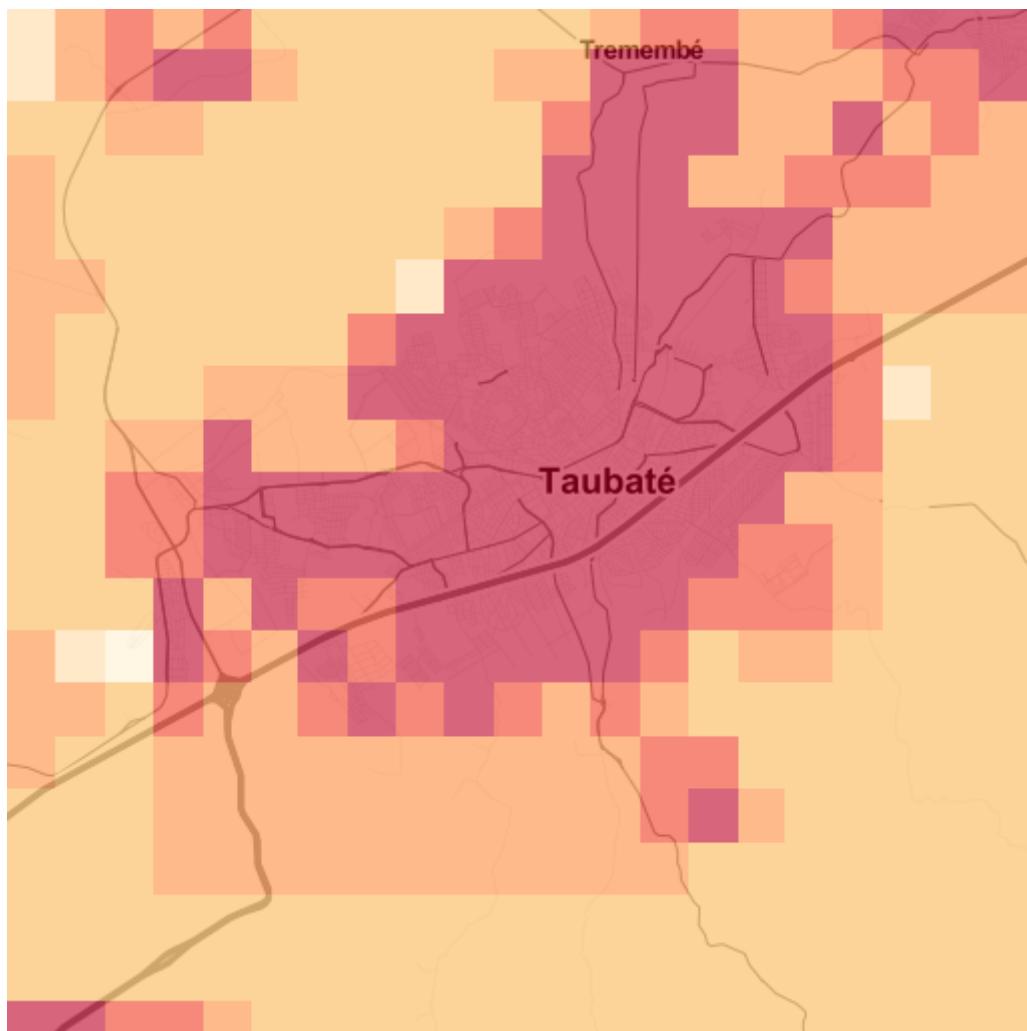


Figura 1 – GPWv4: Population Density - 2015 com foco na cidade de Taubaté

algoritmo competitivo imperialista (ATASHPAZ-GARGARI; LUCAS, 2007) modificado (ROCHE et al., 2011) e otimizado (com alterações propostas neste artigo), que por sua vez possui uma função semelhante à função de previsão, porém que possui rotinas para buscar quais valores são os ideais a serem usados para previsão de períodos à frente do mapa base.

São construídos diversos mapas da mesma região, durante diversos períodos diferentes, espaçados igualmente no tempo, para que se possa ser feita uma previsão espacial como uma série temporal de mapas de quadrículas. Algoritmos evolutivos já vem sendo usados para calcular parâmetros de entrada para funções de previsão de séries temporais (WHITEHEAD; CHOATE, 1996). No caso de WHITEHEAD; CHOATE, é criado um algoritmo genético (GA) (MITCHELL, 1998) cooperativo-competitivo capaz de evoluir os centros e larguras de uma rede neural RBF (REN; JIANG; SUN, 2006) para ser aplicado na previsão de uma série temporal. Neste trabalho portanto, usa-se o algoritmo competitivo imperialista (ICA) para evoluir valores de uma função proposta capaz de fazer uma "regressão", usando fatores de ponderação e matrizes de convolução para a previsão de

crescimento densidade espacial em mapas bidimensionais.

Em resumo, apresenta-se uma técnica capaz de fazer previsão espacial de densidades, sejam elas de quaisquer tipo (carga, populacional, etc.), em alta resolução, que considera diversos fatores na construção dos mapas de quadrículas e que ainda não deixa de ignorar nenhuma área mais afastada da região central urbana, podendo assim melhorar inferência dos resultados obtidos dos métodos de previsão espacial de baseados em quadrículas.

1.2 Objetivos

O objetivo deste trabalho é efetuar a previsão de densidade espacial genérica (qualquer que seja o parâmetro analisado, densidade de carga, densidade populacional, etc.) sobre um período curto de tempo (de 6 meses a 5 anos), com capacidade de previsão sobre dados dispostos em alta resolução.

Para isto, desenvolve-se uma metodologia capaz de traduzir um conjunto de dados para um histórico de mapas de uma localização que então serão processados por um algoritmo evolutivo, que irá gerar indivíduos capazes de gerar previsões de 1 ou mais períodos a frente com vantagem sobre os modelos estatísticos convencionais.

Mais especificamente, este trabalho aborda os seguintes tópicos:

- Modelagem e normalização dos dados para utilização dentro da aplicação.
- Transformação dos dados iniciais em mapas da mesma localização, durante diferentes períodos, de acordo com o foco da previsão.
- Regionalização dos mapas para previsão.
- Definição de uma função de avaliação capaz de fazer previsão espacial sobre regiões.
- Aplicação do processo evolutivo para definir a melhor curva de crescimento para cada pequena área de uma região.
- Utilização do resultado do processamento evolutivo para criar as previsões desejadas.
- Analisar as diferenças entre as previsões geradas pelo modelo proposto com o valores reais e comparar com o modelo de previsão convencional.

1.3 Revisão Bibliográfica

Nesta seção são apresentados, trabalhos que possam servir como base para o desenvolvimento, trabalhos que possuam alguma relação com o objetivo, além de terem suas ideias discutidas para a solução do problema proposto em questão. Alguns conceitos

simples e essenciais para o desenvolvimento da metodologia ou experimentos também podem vir a ser abordados.

Um dos conceitos básicos para se iniciar o desenvolvimento de uma metodologia de previsão espacial é a forma com que se definirá o espaço de trabalho e como os dados serão modelados para que o objetivo seja atingido. A proposta deste trabalho é trazer uma metodologia capaz de resultar em uma previsão de espacial sobre áreas urbanas. Assim, é necessário que se utilize dados que traduzam a realidade para os dados a serem processados, convertendo-os e ajustando-os de modo preciso para que não existam erros durante a aquisição e tradução dos dados reais.

Neste caso, este trabalho explora a filosofia de quadrículas, que diz respeito ao particionamento de uma região em pequenas regiões, para a previsão espacial de cargas aplicadas nestas pequenas áreas. Este modelo é apresentado por WILLIS (WILLIS, 2002) que enfatizam que o planejamento de sistemas de potência deve sempre incluir a previsão de distribuição de carga, pois tais planos de expansão são baseados em diversas proposições, relacionadas ao crescimento futuro da carga, mesmo que este venha ser definido inadequadamente, seja forçado ou implícito. Assim, ao se utilizar este modelo, consegue-se aumentar a precisão da previsão para atender as necessidades do planejamento. No caso deste trabalho fazemos uma generalização da filosofia para que o modelo possa representar quaisquer tipos de dados, sejam eles do setor elétrico ou qualquer outro.

Atualmente, este método, conhecido também como previsão de pequenas áreas (*small area forecast*) (WILLIS; ENGEL; BURI, 1995), é amplamente usado para previsão espacial do crescimento do consumo de carga. Não existe um critério fixo para a separação da região em pequenas regiões, porém, a mais comumente utilizada é a separação em regiões não uniformes, tais que cada uma represente uma região correspondente ao mapeamento elétrico da área de distribuição, sendo esta divisão definida por critérios como área de distribuição de subestações, ou áreas de atuação de concessionárias etc.. Outra forma, é a separação em regiões uniformes, quadradas que se encaixam em diversas situações, por manter a toda região homogênea e normalizada, denominada a filosofia de quadrículas ou quadriculamento da região. A Figura 2 mostra como pode ser dividido um mapa de uma região usando ambos os modelos citados no parágrafo anterior.

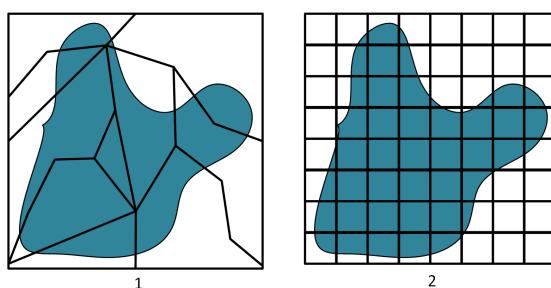


Figura 2 – Regiões e Quadrículas

Ambas possuem vantagens e desvantagens segundo WILLIS (WILLIS, 2002), como por exemplo, a separação por quadrículas, ilustrada no item 2 da figura, pode produzir melhores resultados, porém pode ser mais complexo obter e processar os dados para formar as regiões. E a separação por regiões apesar de possuir sua separação formada por dados que classificam a região de alguma forma, é mais complexa de ser modelada computacionalmente para análise homogênea, pois suas área são de tamanhos e formas diferentes.

Existem muitos métodos de previsão de pequenas áreas, portanto eles sempre se encaixam em uma das duas categorias (definido por WILLIS (WILLIS, 2002) e detalhado por ARANGO; LAMBERT-TORRES (ARANGO; LAMBERT-TORRES, 2000), sendo:

- Métodos de ajuste, que considera a curva de tendência do crescimento de períodos passados, ajustando-a através de métodos como mínimos quadrados, média móvel, etc. de modo que os períodos futuros possam ser extrapolados a partir desta curva ajustada.
- Métodos baseados no uso da terra, que faz as previsões futuras baseando-se na análise de fatores de uso da terra, como planos diretores, zoneamentos, entre outros, os quais possam ser retiradas informações de como os crescimentos de carga (de regiões residenciais, industriais ou comerciais) irão crescer durante os períodos futuros.

No caso deste trabalho, o método desenvolvido se aplica principalmente ao primeiro modelo de métodos, apesar de processar os dados obtidos de modo que eles sejam transformados em informações de crescimento de carga residencial durante os períodos, ao processar tais dados, leva-se em consideração como estes dados crescem ao longo dos períodos, de modo que se possa gerar uma curva de tendência e extrapolar uma previsão de períodos futuros. O segundo método é usado apenas inicialmente, ao se ajustar os dados para que eles possam ser normalizado e traduzidos para um mapa de quadrículas, porém não se faz nenhum tipo de previsão ou geração de regras de previsão durante esta etapa de tradução.

Um modelo dinâmico de expansão de cargas é proposto por ARANGO; LAMBERT-TORRES, e baseia-se em conceitos de análise locacional, que aplica a teoria dos polos urbanos e se encaixa mais nos métodos baseados no uso da terra (segundo item). Deste modo, ele cria um simulador de expansão de cargas que evolui de forma dinâmica e continua as condições para uma nova unidade de carga. Tal modelo consegue representar a expansão de carga, usando uma adaptação do processo de recozimento simulado que fora especificamente desenvolvido para solucionar o problema. Este trabalho traz uma solução mais genérica, para a previsão de expansão de densidades, seja ela de cargas, populacional, entre outras.

Uma análise interessante sobre o erro de previsão espacial é desenvolvida por ARANGO; LAMBERT-TORRES, que avalia o erro de previsão espacial considerando as diferenças entre os valores reais e estimados para cada subárea urbana, e traz um exemplo muito bom, que demonstra a preocupação para com a avaliação da imprecisão dos modelos de previsão espacial. No exemplo ele imagina duas regiões, as quais tenham uma magnitude total de cargas iguais, porém distribuídas diferentemente, como mostra a Figura 3.

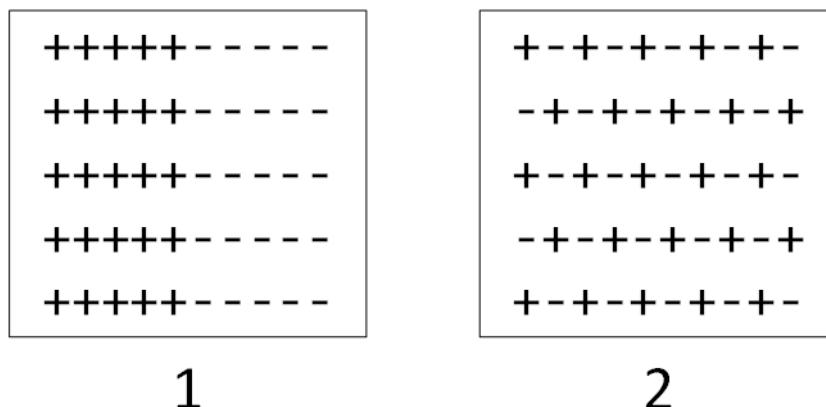


Figura 3 – Distribuições distintas do mesmo erro global, detalhado em (ARANGO; LAMBERT-TORRES, 2000) e definido por (WILLIS; ENGEL; BURI, 1995)

A estimativa para ambas as áreas serão completamente distintas, sendo que na primeira existe uma tendência bem direcionada e na segunda ocorrerá uma estimativa muito mais homogênea em relação a região.

Ao se fazer uma análise estatística entre a uma região estimada e sua região real, comparando o erro absoluto de cada pequena área (quadrícula). É possível ocorrer que uma região seja estimada de forma absolutamente correta, assim como também pode ocorrer que uma região seja estimada de forma totalmente incorreta. Neste trabalho trabalha-se para que tais erros sejam minimizados de modo a fazer com que toda região tenha uma estimativa com o menor erro possível, seguindo as taxas de crescimento tanto das pequenas áreas, quanto da região como um todo, e ainda, também agrupa o crescimento das pequenas áreas presentes nas redondezas da pequena área que esteja sendo analisada.

Para o desenvolvimento da aplicação foi necessária inicialmente a modelagem dos dados iniciais de modo que estes fossem convertidos com a menor ou nenhuma quantidade de erro residual. Pois neste caso, os dados estão representados em coordenadas geográficas, e são utilizadas diversos cálculos de distância para que eles sejam adequadamente identificados em sua pequena região. Para se calcular a distância entre dois pontos, dispostos em um mapa terrestre em coordenadas cartesianas, pode-se aplicar o teorema de Pitágoras:

$$d = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

Sendo $(X1, X2)$ e $(Y1, Y2)$ os pontos iniciais e finais de longitude e latitude respectivamente.

Esta fórmula de cálculo de distância, quando aplicada a pontos cuja distância é menor que 20 km, produz erros dependentes de suas posições reais dispostas no globo, sendo que tais erros variam como:

- De 0 a 30 metros para latitudes menores que 70 graus.
- De 0 a 20 metros para latitudes menores que 50 graus.
- De 0 a 9 metros para latitudes menores que 30 graus.

Tal gestão de erros é desnecessária quando se usa a distância calculada pela fórmula de Haversine (SHUMAKER; SINNOTT, 1984), que calcula a distância entre dois pontos dispostos na superfície de uma esfera, relacionando os lados aos ângulos de uma esfera triangular. Assim, fazendo o uso de um mapa terrestre esférico, onde tal esfera possui raio R , e tendo a representação dos pontos que se deseja calcular a distância como vetores compostos na forma (Longitude, Latitude), temos que o $ponto1 = (lon1, lat1)$ e o $ponto2 = (lon2, lat2)$, calculamos a distância como mostra a expressão:

$$\begin{aligned} dlon &= lon2 - lon1 \\ dlat &= lat2 - lat1 \\ a &= \sin^2\left(\frac{dlat}{2}\right) + \cos(lat1) \cdot \cos(lat2) \cdot \sin^2\left(\frac{dlon}{2}\right) \\ c &= 2 \cdot \arcsin(\min(1, \sqrt{a})) \\ d &= R \cdot c \end{aligned}$$

A distância calculada resulta em um valor exato, tanto computacionalmente quanto matematicamente, onde c é a distância do caminho mais curto entre os dois pontos, denominada ortodromia, representada em radianos, que multiplica R , convertendo o resultado de d para a mesma unidade do Raio R .

A função \min protege contra possíveis erros de arredondamento que podem sabotar a computação do arco-seno caso os dois pontos estejam em lados opostos da terra. Sob estas condições, a Fórmula Haversine é mal condicionada, mas o erro, que talvez pareça grande, que chega a ser de aproximadamente 2 km, está no contexto de uma distância próxima 20,000 km, sendo que este erro é de aproximadamente 0.00001% da distância total.

No ambiente computacional, a maioria das funções ou componentes trigonométricos são expressados em radianos. Uma vez que os dados de coordenadas (Longitude e Latitude) geralmente estão expressos na forma Graus, Minutos e Segundos, é necessário convertê-los para graus decimais:

$$\begin{aligned} coordDec = & \ coordGMS.Graus + \\ & \left(\frac{CoordGMS.Minutos}{60} \right) + \\ & \left(\frac{CoordGMS.Segundos}{3600} \right) \end{aligned}$$

Com as coordenadas convertidas para graus decimais, ainda é necessário convertê-las para radianos, que é simplesmente multiplicar a coordenada em graus decimais por $\frac{\pi}{180}$:

$$coordRad = \frac{coordDec \cdot \pi}{180}$$

Por fim, a definição de R como sendo o raio da esfera, deve ser levado em consideração que o planeta Terra não é uma esfera perfeita, mas sim, possui uma forma chamada esferóide oblato (Figura 4), e portanto, deve-se considerar diferentes raios para diferentes posições de latitude, uma vez que o achatamento ocorre nos polos.

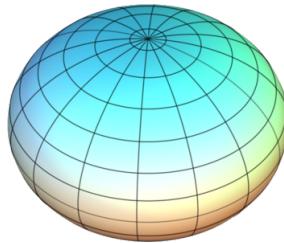


Figura 4 – Esferóide oblato

A circunferência oficial terrestre é de 40003.2 km, que provém da definição de uma milha náutica, que historicamente é definida como 1 arco de minuto medido, à superfície média do mar, ao longo de um qualquer grande círculo da Terra. Como a terra não é uma esfera, tal definição é ambígua. Então, o valor aceito internacionalmente(SI) para uma milha náutica é de exatamente 1.852 km. Deste modo, define-se como circunferência terrestre, o valor de 40003.2 km:

$$40003.2(km) = 360(grau) * 60\left(\frac{minuto}{grau}\right) * 1.852\left(\frac{km}{minuto}\right)$$

Implicando então que o raio R seja de:

$$R = \frac{40003.2}{(2 \cdot \pi)} = 6367(km)$$

Como o planeta Terra não é representado por uma esfera, mas sim por um esferóide oblato, e possui um Raio Polar de 6357 km e um raio equatorial de 6378 km, uma aproximação satisfatória seria como mostra a expressão:

$$R = 6378 - 21 \cdot \sin(lat)$$

Porém, tal aproximação é dependente apenas do valor de latitude, e o raio da curvatura é dependente não só do valor de latitude, mas também da direção, de acordo com (SNYDER, 1987). Então, usa-se:

$$R = \frac{a \cdot (1 - e^2)}{(1 - e^2 \cdot \sin^2(lat))^{\frac{3}{2}}}$$

Onde a é o raio equatorial, b o raio polar, e e é a excentricidade do elipsóide $= (1 - \frac{b^2}{a^2})^{\frac{1}{2}}$. Então tem-se que $e = 0.0167086$. Assim, haverá um erro mínimo sempre que as distâncias entre dois pontos de coordenadas geográficas forem calculadas utilizando a fórmula de haversine.

Este trabalho utiliza-se muito da fórmula de haversine para converter os dados iniciais, dispostos em coordenadas geográficas, para coordenadas cartesianas, de modo que se crie uma matriz de pequenas áreas (quadrículas), onde estas pequenas áreas possam vir a ser usadas para gerar um mapa cartesiano que represente fielmente os dados que foram convertidos, tanto para o processamento destes dados quanto para a apresentação dos resultados.

A previsão espacial apresentada por este trabalho é efetuada usando-se um algoritmo que classifica-se como evolutivo, isto é, está na mesma categoria que algoritmos como algoritmos genéticos (AG) (MITCHELL, 1998) e otimização por enxame de partículas (PSO). (ATASHPAZ-GARGARI; LUCAS, 2007) apresenta o algoritmo imperialista competitivo (ICA), que atinge resultados tão bem quanto algoritmos genéticos, atingindo o mínimo global geralmente ao mesmo tempo que o algoritmo genético.

A vantagem do ICA sobre o GA se dá em relação média do custo de toda população, onde o GA geralmente fica estagnado em um patamar devido ao fato de que sempre que ocorrem mutações, seleções ou criação de novos indivíduos para complementar a população, estes indivíduos são gerados com valores aleatórios, impossibilitando a convergência de toda a população para a solução ótima. O ICA por sua vez consegue fazer com que toda a população seja convergida para a solução ótima, sem que fique estagnado ou ceia em mínimos locais.

Outra util vantagem que o ICA possui sobre o GA é o fato de seus atributos serem valores numéricos de ponto flutuante, uma vez que no GA em sua versão canônica,

utilizam-se vetores de atributos binários, que, desta forma, seria sempre necessário traduzir o valor do cromossomo para valores numéricos.

Este trabalho utiliza um grande conjunto de números de ponto flutuante para sua função de avaliação, que são traduzidos para estruturas usadas pela função. Caso o vetor de atributos fosse de valores binários, seriam necessária uma tradução dos valores binários para valores numéricos de ponto flutuante, antes mesmo de se gerar as estruturas usadas pela função de avaliação, tornando este processo muito mais complexo.

(ROCHE et al., 2011) traz uma aplicação concreta do ICA voltado para otimizar o uso de ferramentas que definem estrategicamente a alocação de unidades de geração de energia para as usinas de acordo com diversos parâmetros. O ICA é comparado com modelos convencionais, como AG e PSO, tendo uma efetividade maior tanto em previsão quanto em velocidade. Voltando o Foco para a o desenvolvimento do ICA feito por ROCHE et al., ele desenvolveu o ICA na linguagem Java, que pode ser acessado pelo repositório git em github.com/robinroche/jica (ROCHE, 2011), sendo desenvolvido em uma linguagem orientada a objetos, de forma simples. Este trabalho propõe uma melhora na implementação orientada a objetos para o ICA de modo que ele possa ser usado genericamente para qualquer caso ou aplicação, de modo que a modelagem de qualquer problema seja rápida, prática, modular e sem a necessidade de alterar o código fonte principal da lógica do ICA.

Tendo conhecimento que o desenvolvimento de uma metodologia para previsão espacial apresenta uma solução de alta complexidade, exige elevada quantidade de informação e a qual se deve levar em conta diversos fatores de crescimentos aleatórios e/ou imprevisíveis (como desejo político ou planejamento externo), tais problemas apresentam soluções de domínio não lineares. Assim, este trabalho propõe uma implementação diferenciada do ICA de forma que este seja capaz de atender a todas as requisições do problema de forma prática, precisa e rápida.

A metodologia empregada faz uso do ICA para previsão espacial, em um ambiente (neste caso, urbano) bidimensional (2D). Então, para que tal problema possa ser solucionado de modo que múltiplas alterações sejam executadas e testadas afim de se obter os melhores resultados, algumas adaptações foram imprescindíveis ao ICA tanto em relação a otimização de seu desempenho na busca de soluções melhores em um tempo menor quanto a organização e modelagem do conceito, que se deu na forma de uma implementação orientada a objetos (BOOCH, 1982) e (COAD; YOURDON, 1991).

A primeira preocupação durante o desenvolvimento foi encontrar quais seriam as abordagens mais certeiras e inovadoras para que a previsão de densidade espacial urbana fornecesse resultados aceitáveis. Tendo isto em mente, sabia-se que deveriam ocorrer diversos testes com diversas funções de avaliação diferentes, e ainda, que tais funções, devido a complexidade do problema, também teriam de ser diversas vezes ajustadas durante os testes. Assim, a aplicação dos conceitos de orientação a objetos foram utilizados

para a modelagem do ambiente, para a implementação dos conceitos do ICA e de diversas funções de avaliação ou modelos de problemas diferentes.

1.4 Estrutura da Dissertação

Este documento está dividido nos capítulos: Introdução, Metodologia, Desenvolvimento, Experimentos e Resultados e Conclusão.

No capítulo introdução é contextualizado o conceito de previsão espacial urbana, que pode ser feita a partir de diversos métodos e técnicas. Também são apresentados os objetivos definidos e a justificativa da realização do trabalho.

Em Metodologia são detalhados os principais conceitos e técnicas pesquisados e aplicados no trabalho, abrangendo principalmente o Algoritmo Imperialista Competitivo e a operação matemática de convolução.

O capítulo Desenvolvimento descreve especificamente como as técnicas escolhidas foram desenvolvidas, modificadas e implementadas focando nos objetivos e na solução do problema.

Já no capítulo Experimentos e Resultados são apresentados como o ambiente foi modelado, como o método proposto se comportou sobre uma e sobre muitas regiões, e apresentação dos resultados sobre dados reais, apresentando os resultados em forma de imagens de alta resolução e uma tabela comparativa.

Por fim, no capítulo Conclusão são considerados os resultados obtidos frente aos objetivos propostos, a colaboração deste trabalho, assim como a indicação de trabalhos futuros para a continuidade da pesquisa.

2 Metodologia

Neste capítulo são apresentados os conceitos do algoritmo imperialista competitivo e a apresentação da operação de convolução e a convolução de imagens, seguidos de uma breve análise sobre estes assuntos, direcionando como estes conceitos serão aplicados na busca da solução do problema proposto.

2.1 Algoritmo Imperialista Competitivo

O algoritmo imperialista competitivo (ATASHPAZ-GARGARI; LUCAS, 2007) (ou ICA - *Imperialist Competitive Algorithm* em inglês) é um algoritmo evolutivo baseado na ideia da disputa entre impérios pela extensão de seus domínios para além de seus territórios. Para que um império aumente seu poder ele deve aplicar sua influência sob mais territórios dominando-os e assimilando-os. Territórios dominados por países imperialistas são chamados de colônias, e um império é denominado pelo conjunto formado por um país imperialista e suas colônias.

As colônias de um império tendem a ter suas características se assemelhando às de seu país imperialista a cada década que passa. Esta tendência pode ser vista como uma forma de movimento, ou seja, uma colônia se move em direção ao seu país imperialista de forma que suas características se pareçam mais com as dele.

A definição do poder dos países, sejam eles imperialistas ou colônias, é baseado na ideia de que um dado país tem um custo para se manter. Se um país é rico em recursos, significa que ele terá um custo muito baixo, e consequentemente poderá usar seus recursos de forma a expandir suas fronteiras através da assimilação de colônias. Assim o poder de um país é definido pelo inverso de seu custo, implicando em quando um país possuir um custo muito próximo de (ou tender a) zero, isto significará que seu poder seria imenso e tenderia a infinito.

Antes que a disputa imperialista comece, todos os países são gerados com características aleatórias. Os custos dos países gerados são calculados, e consequentemente seus poderes são definidos. Uma porcentagem dos países com maior poder se tornam impérios, e o restante se tornam colônias. As colônias são distribuídas aos impérios mais poderosos de uma forma proporcional.

Impérios disputam entre si usando como forma de medição o nível de poder total, que por sua vez é definido pela soma do poder do país imperialista com o poder de suas colônias. Assim, um países imperialistas mais fortes tendem a tomar as colônias de um imperialista mais fraco, e os imperialistas mais fracos tendem a perder suas colônias para

imperialistas mais fortes até que eles não tenham mais nenhuma colônia e sejam eliminados da disputa.

Uma colônia pode se tornar um país imperialista quando seu poder superar o poder de seu atual imperialista, assim, todas as colônias do país imperialista, incluindo ele mesmo, passam a ser colônias da colônia que se tornou o novo país imperialista, alterando assim, a direção do movimento das colônias para a posição deste novo imperialista.

O resultado deste processo competitivo tende para que reste apenas um império na disputa possuindo todas as colônias, de forma que todas as colônias tenham as mesmas características poder de seu país imperialista.

Sendo o ICA um algoritmo evolutivo, seu objetivo é encontrar uma solução dentre diversas outras, tal que esta seja ótima, em termos das variáveis do problema (dimensões) em questão. Como pode ser visto na figura Fluxograma ICA antes de começar o processo evolutivo, deve-se inicializar os países formando os impérios e as suas respectivas colônias. Sendo assim, inicialmente deve-se definir a estrutura básica de um país que, analogamente a um cromossomo presente no algoritmo genético canônico (GA - *Genetic Algorithm*), usa um vetor de valores de dimensão $1 \times N_{var}$ apresentado pela equação 2.1:

$$\text{país} = [p_1, p_2, \dots, p_{N_{var}}] \quad (2.1)$$

Sendo que os elementos contidos no vetor representem as características deste país com valores numéricos de ponto flutuante, diferentemente do GA canônico, que usa uma tripla de valores booleanos que geralmente devem ser traduzidos quando se deseja usar valores numéricos. E define-se N_{var} o número de dimensões do problema a ser otimizado.

O custo de um país deve ser calculado através da aplicação dos valores presentes no país a uma função f de acordo com a equação 2.2:

$$\text{custo} = f(\text{país}) = f(p_1, p_2, \dots, p_{N_{var}}) \quad (2.2)$$

Sendo o custo um valor análogo ao valor de aptidão de um cromossomo no algoritmo genético e f uma função de avaliação.

Um valor inicial N_{pop} define a quantidade de países que serão gerados com valores (características) aleatórios na sua inicialização, antes que se comece a etapa de otimização (competição imperialista). Seleciona-se também uma porção de N_{pop} , denominada N_{imp} , como sendo o número de países imperialistas inicial. A partir disso, são escolhidos como os N_{imp} países imperialistas, os países mais poderosos dentre os N_{pop} países. A quantidade de países restantes é denominada N_{col} , tal como mostra a equação 2.3:

$$N_{col} = N_{pop} - N_{imp} \quad (2.3)$$

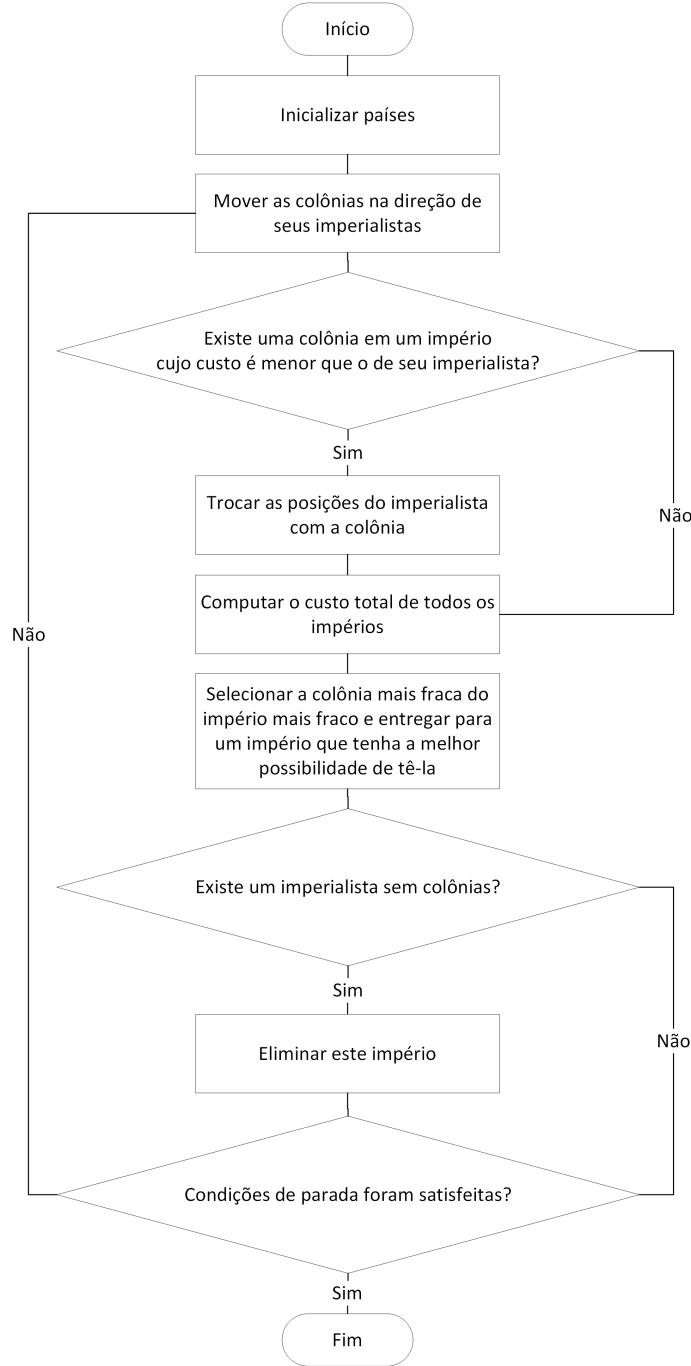


Figura 5 – Fluxograma ICA canônico

E que posteriormente serão distribuídas proporcionalmente a cada um dos imperialistas.

Para a distribuição das N_{col} colônias para os N_{imp} imperialistas ser feita de modo proporcional ao poder de cada um destes países imperialistas, define-se o valor normalizado do custo de um imperialista segundo a equação 2.4

$$C_n = c_n - \max(c_i) \quad (2.4)$$

Onde C_n é o custo normalizado do n-ésimo imperialista, c_n é o custo normalizado deste enésimo imperialista, e a operação $\max_i c_i$ representa o valor máximo dos custos dentre todos os países. Assim, pode-se definir o poder normalizado de cada país imperialista como mostra a equação 2.5:

$$P_n = \left| \frac{C_n}{\sum_{i=1}^{N_{im}} C_i} \right| \quad (2.5)$$

Que é basicamente o custo normalizado do enésimo imperialista dividido pela soma dos custos normalizados dentre todos os imperialistas. Sendo assim, este poder normalizado pode ser entendido como um valor que representa a porção de colônias que devem ser possuídas por aquele imperialista. Então o número de colônias de um império é definido pela equação 2.6:

$$N.C.n = arredondamento(P_n \cdot N_{col}) \quad (2.6)$$

Sendo $N.C.n$ o número inicial de colônias do enésimo imperialista.

Com os impérios já formados, contendo um país imperialista e suas colônias, a sequencia do fluxograma indica que as colônias de um império devem se movimentar em direção ao seu país imperialista, como pode ser visto na Figura6. O movimento de uma colônia até seu imperialista deve ser calculado considerando um fator aleatório e um fator de controle de intensidade, sendo que o fator aleatório pode ser de distribuição uniforme (ou qualquer outra que seja apropriada a situação) e o fator de controle de intensidade deve ser relativo ao império, portanto usa-se a distância entre o imperialista e a colônia. Assim, tem-se que um valor x , representado o movimento que será efetuado pela colônia, seja o produto vetorial entre a distância e o fator aleatório, como mostrado na equação 2.7 a seguir:

$$x = URand(0, \beta \times d) \quad (2.7)$$

Sendo β um valor aleatório maior que 1 e d a distância absoluta entre o país imperialista e a colônia.

Usa-se β maior que 1 por resultar em um efeito elástico no espaço de busca, fazendo que uma colônia se aproxime de seu imperialista explorando ambos os lados como pode ser visto na Figura 7.

Para que se tenha uma busca mais diferenciada, e que explore pontos mais diferenciados ao redor do país imperialista, ainda há a adição de um valor aleatório de desvio na direção do movimento , como mostrado na Figura 8. θ é um outro valor aleatório de

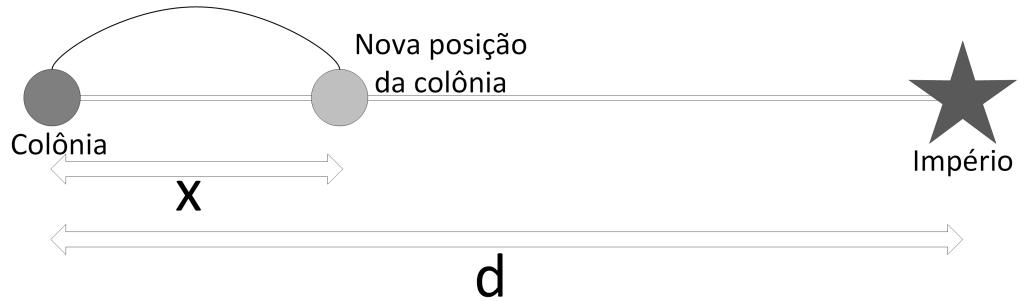
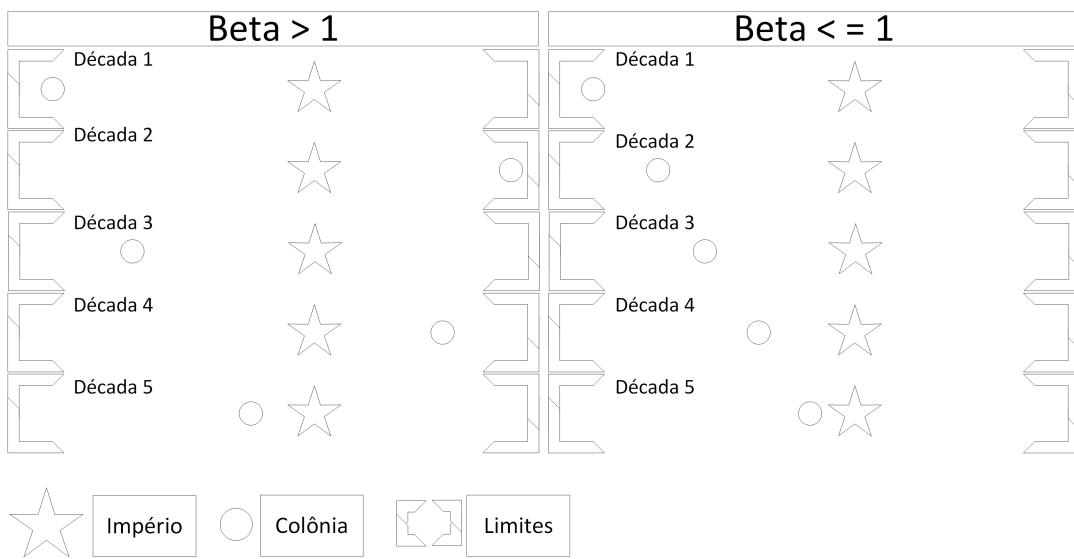


Figura 6 – Movimento da colônia para seu imperialista

Figura 7 – Comparação $\beta < 1$ e $\beta > 1$

distribuição uniforme (ou qualquer outra que seja apropriada a situação), tal que seus limites sejam inversos e de mesma intensidade Gama, como:

$$\theta = URand(-\gamma, \gamma) \quad (2.8)$$

Onde γ é o parâmetro de ajuste de desvio da direção original. Sendo assim, a nova posição da colônia pode ser entendida como um ponto aleatório proporcional a distância entre esta colônia e seu imperialista, que explora as redondezas do imperialista limitando-se na distância e com a adição de um ruído para a exploração mais dinâmica do espaço de busca.

$$\text{Posição}(t+1) = \text{Posição}(t) + x + \theta \quad (2.9)$$

Este processo de movimentação é executado para todas as colônias de cada país imperialista, assim que todas as colônias terminam sua movimentação, ainda é possível

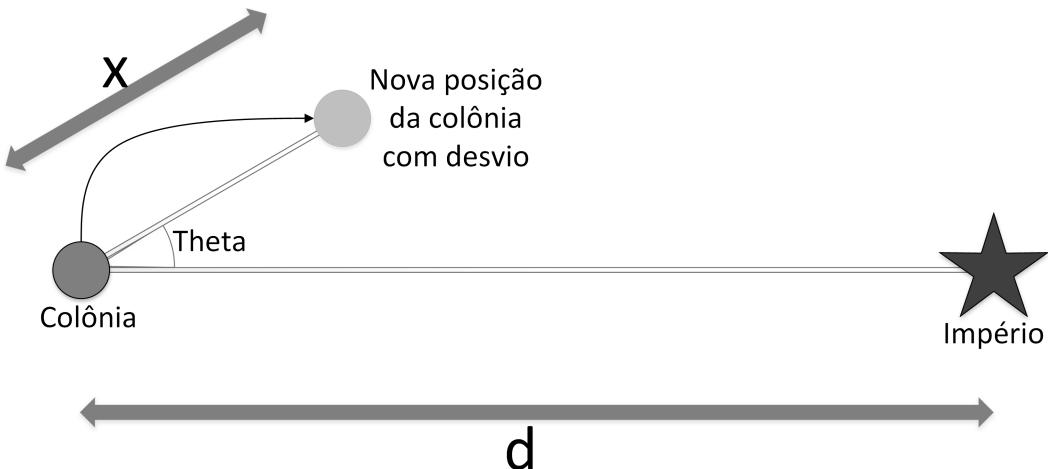


Figura 8 – Movimento da colônia para seu imperialista com desvio de direção

haver o que é chamado de processo de revolução, ou apenas revolução, que é uma operação muito semelhante a operação de mutação no GA, onde todas as características (valores do vetor país) da colônia são geradas novamente de forma aleatória.

A segunda etapa, seguindo o fluxograma, faz a verificação dentre todos os imperialistas se uma de suas colônias possui características melhores que seu país imperialista, assim a colônia passa a ser o país imperialista tomando todas as colônias deste antigo imperialista em questão e inclusive, tornando este país imperialista uma colônia. Situações como esta ocorrem quando a colônia ao mover em direção ao seu país imperialista acaba se posicionando em um local onde suas características passam a apresentar um custo menor que o custo de seu país imperialista, como pode ser visto na Figura 9. Nesta mesma figura, observa-se o movimento de colônias em direção ao seu país imperialista (quadro 1), e quando uma colônia passa por uma posição melhor que a de seu país imperialista (quadro 2), ela altera a sua cor e inicia o processo de tomada do império para si, se tornando o novo império, assimilando as colônias do antigo império e consequentemente transformando este antigo imperialista em uma colônia (quadro 3). Posteriormente, observa-se que as colônias passam a se movimentar em direção a este novo imperialista (quadro 4).

Em seguida, o próximo passo, após as trocas entre impérios e colônias (caso existam), é o cálculo de todos os custos totais dos impérios, que também leva em consideração o valor do custo de suas colônias. Assim, define-se custo total de um império, como sendo a soma entre o custo do país imperialista mais uma porção da média dos custos das colônias pertencentes a este império, definido pela equação 2.10:

$$T.C.n = Custo(Imperialistan) + \epsilon \cdot \text{Média}(Custo(Colônias do império)) \quad (2.10)$$

No qual $T.C.n$ é o custo total do enésimo imperialista, ϵ é o valor que define o

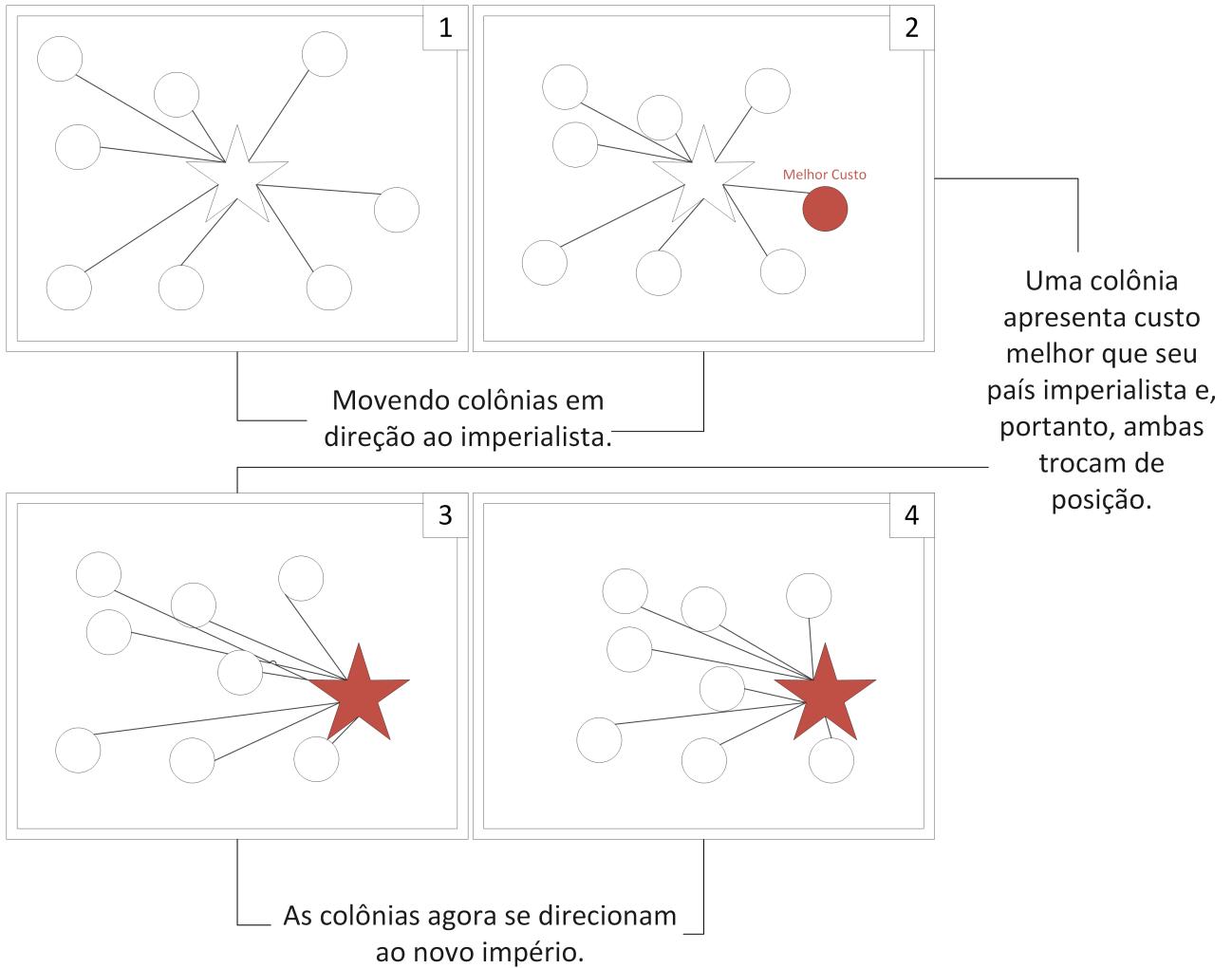


Figura 9 – Tomada de império por colonia

quanto a média dos custos das colônias do enésimo império influenciará no custo total, sendo ϵ menor que 1. O cálculo do custo total de um império é usado para calcular o poder de um império durante a competição imperialista. Assim, quando usa-se um valor muito baixo para ϵ o poder do império será definido basicamente pelo custo do país imperialista, e quando aumenta-se este valor, os custos das colônias passam ter uma maior influência no poder do império, o qual implica diretamente na competição imperialista, ou seja, se o império tem um custo baixo, porém as suas colônias tem custos muito altos, se ϵ for baixo, este império tem maiores chances de ganhar uma competição imperialista do que se este valor de ϵ fosse um valor mais alto.

A partir deste ponto inicia-se a competição imperialista, que se resume em selecionar a colônia mais fraca do império mais fraco e colocá-la sob o comando de um império que tenha a maior afinidade para tê-la. Assim, a competição imperialista é uma forma de aumentar o poder dos melhores impérios e diminuir o poder dos piores impérios. O império com mais chances de tomar uma colônia para si é aquele com maior poder dentre

os impérios competidores, porém não é sempre que o império mais poderoso adquire a colônia em jogo, lembrando que quem adquire a colônia é aquele que tem maior afinidade para tê-la, e não maior poder. Assim, para definir o país com maior afinidade de possessão durante a competição imperialista, usa-se os valores de custo total dos impérios, calculados no passo anterior, para calcular o custo total normalizado, como mostra a equação 2.11:

$$N.T.C.n = T.C.n - \max(T.C.i) \quad (2.11)$$

Sendo $N.T.C.n$ o valor normalizado do custo total do enésimo império, $T.C.n$ o custo total do enésimo império e $\max(T.C.i)$ o máximo valor dentre todos os custos totais. Assim que se calculam todos os custos normalizados, calcula-se as probabilidades de possessão de cada imperialista como mostra a equação 2.12:

$$P_{pn} = \left| \frac{N.T.C.n}{\sum_{i=1}^{N_{imp}} N.T.C.i} \right| \quad (2.12)$$

Com estes valores de probabilidades de possessão para cada império, cria-se o vetor P , apresentado pela equação 2.13:

$$P = [P_{p1}, P_{p2}, \dots, P_{pN_{imp}}] \quad (2.13)$$

Cria-se também o vetor R , com o mesmo tamanho de P porém preenchido com valores aleatórios distribuídos uniformemente e limitados entre 0 e 1, tal como mostra a equação 2.14:

$$R = [r1, r2, \dots, rN_{imp}]; r1, r2, \dots, rN_{imp} \sim U(0, 1) \quad (2.14)$$

Finalmente, para definir qual será o império vencedor, que por sua vez irá tomar a colônia para si, cria-se o vetor de diferenças D de representado pela equação 2.15:

$$D = P - R = [P_{p1} - r1, P_{p2} - r2, \dots, P_{pN_{imp}} - rN_{imp}] \quad (2.15)$$

Os índices dos vetores P , R e D representam os índices dos impérios. Assim, o índice de D que possuir o maior valor dentre os demais valores representará o império que de fato irá tomar a colônia para si. Observa-se ainda, que por meio destes cálculos não necessariamente será o país com maior poder quem irá vencer a competição e tomar a colônia para si. Mas este terá uma chance maior, e de forma proporcionalmente distribuída, de tomá-la.

Assim que a colônia mais fraca do império mais fraco é passada para o vencedor da competição imperialista, é feita uma verificação dentre todos os impérios, que elimina

aqueles impérios que não mais possuem colônias, e portanto não mais participarão das competições imperialistas e entram em colapso. O fato de eliminar o império assim que ele não possua mais colônias é só um critério que define o colapso, porém outros fatores podem fazer um império colapsar, de modo que as colônias deste tenham que ser distribuídas para os demais impérios.

O último passo do Fluxograma é a verificação das condições de parada para decidir se finaliza o algoritmo ou se repete os passos a partir do movimento das colônias para seus imperialistas. As condições de parada podem ser diversas, como número máximo de iterações (ou décadas, contextualizando), valor de custo atingido, etc. Entretanto a condição de parada ideal para convergência do ICA deve ocorrer quando restar apenas um império e todos os demais países serem colônias pertencentes ao país imperialista. Ao considerar um cenário ideal, no qual existe apenas um império, e todas as suas colônias se situam na mesma posição de seu país imperialista, não existirá diferença nenhuma entre as colônias, e a única diferença existente entre as colônias e o império serão seus títulos. Assim, a condição de parada ideal para o ICA, em uma situação normal(não ideal), seria esperar que reste apenas um império, e ainda, se preciso, pode-se também esperar para que todas as colônias se movimentam para a posição de seu país imperialista, refinando ainda mais a solução.

2.2 A operação de convolução e convolução de imagens

A convolução é um operador matemático formal assim como a operação de soma ou subtração. Tal operação é uma operação linear, que é executada sobre duas funções dadas e resulta em uma terceira, a qual será a área super posicionada das mesmas em função do deslocamento existente entre ambas.

Este operador tem sido amplamente utilizado em diversas áreas, sendo utilizado em aplicações como processamento de sinais e imagens, circuitos elétricos, telecomunicações, probabilidade, estatísticas entre outras. No caso deste trabalho aplica-se a convolução para o processamento de imagens(também conhecido como filtragem no domínio do espaço), sobre uma metodologia diferente do convencional, descrita mais adiante, de modo que os resultados da aplicação de um filtro g , que *convoluciona* por toda uma imagem $f(t)$ sejam os mais próximos possíveis de uma previsão futura de expansão, $f'(t + 1)$, tal que $f(t + 1)$ seja uma imagem um período à frente de $f(t)$.

Em a origem e história da convolução (DOMINGUEZ-TORRES, 2010), são apresentadas diversas definições matemáticas sobre a operação de convolução, e neste caso, o que nos interessa é a convolução discreta, que tem seu conceito aplicado no processamento de imagens como:

Seja $\{x_i\}$ e $\{y_i\}$ duas sequências de valores reais ou complexos tal que

$-\infty < i < \infty$. A convolução discreta dessas sequências resulta em uma nova sequência definida pela expressão:

$$\sum_{n=-\infty}^{\infty} x_n \cdot y_{i+n},$$

$-\infty < i < \infty$

Nota-se que se uma sequência, $\{x_i\}$, $i = 0, 1, \dots, n_1 - 1$, tem um número de termos n_1 e a sequência $\{y_i\}$, $i = 0, 1, \dots, n_2 - 1$, tem um número de termos n_2 , então a convolução discreta dessas duas sequências deve ser escrita na forma:

$$\sum_{n=0}^i x_n \cdot y_{i-n},$$

$i = 0, 1, \dots, n_1 - 1;$
 $n = n_1 + n_2 - 1$

A convolução discreta dessas duas sequências finitas que tem n_1 e n_2 termos respectivamente, e resulta em uma nova sequência contendo $n_1 + n_2 - 1$ termos. (DOMINGUEZ-TORRES, 2010)

Este conceito mostra como a convolução é aplicada sobre duas sequências de valores resultando no valor final do somatório, portanto, a técnica aplicada neste trabalho utiliza a convolução para fazer a filtragem espacial de imagens, que gera de uma nova imagem g a partir da convolução dos pontos de uma imagem f sobre uma matriz ou filtro de convolução m .

Esta técnica de filtragem espacial, então, pode ser representada como sendo uma transformação da imagem, de forma ponto a ponto, que dependem do valor de um determinado ponto e do valor dos pontos vizinhos na imagem original. Assim o ponto a ser filtrado terá um valor referente à região que ele se encontra na imagem original.

Assim, a filtragem por convolução pode ser efetuada como mostra a expressão:

$$g = f * m$$

sendo:

- g a imagem resultado da filtragem,
- f a imagem original,
- m a máscara ou filtro de convolução e
- $*$ o operador de convolução.

Para se obter a imagem g , executa-se a operação de convolução de m sobre todos os pontos da imagem f , assim, um ponto (x, y) da imagem g é calculado como:

$$g(x, y) = \sum_{i=\lfloor \frac{O}{2} \rfloor}^{\lfloor \frac{O}{2} \rfloor} \sum_{j=\lfloor \frac{O}{2} \rfloor}^{\lfloor \frac{O}{2} \rfloor} f(x+i, y+j) \cdot m(i, j)$$

Sendo O a ordem da máscara de convolução, que tem altura e largura iguais a O e i e j os iteradores sobre a máscara. Tal operação de filtragem pode ser ilustrada como mostra a Figura 10:

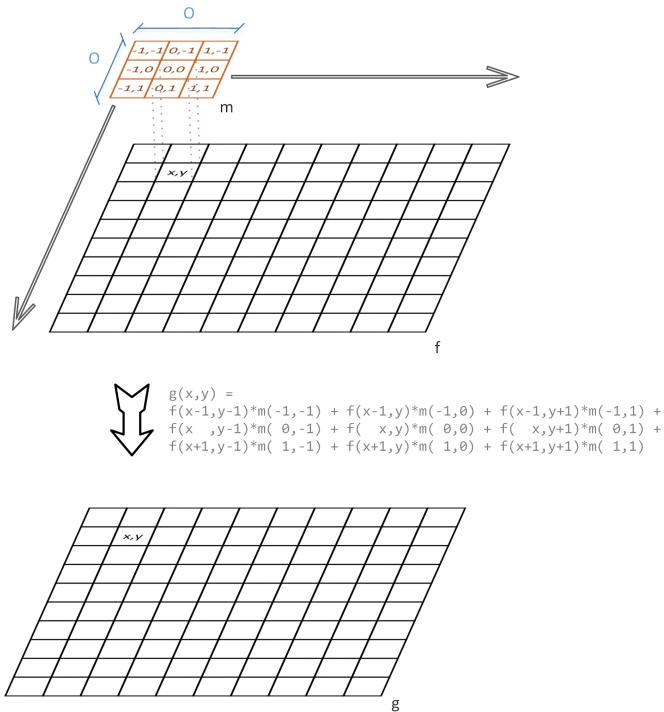


Figura 10 – Filtragem por convolução

Observe que o elemento central da máscara fica posicionado sobre o ponto (x, y) da imagem, para que se possa calcular o ponto (x, y) de g efetuando a convolução de f por m . Observe ainda, que a matriz de convolução, para gerar toda a imagem g , percorrendo todos os pontos de f e aplicando a convolução ponto a ponto. E note também, no cálculo de $g(x, y)$ na figura, que a convolução faz a soma das multiplicações entre os elementos da matriz m com a imagem f , de modo que os índices dos elementos estejam sempre nas mesmas posições, ou seja, o ponto (x, y) de f será multiplicado pelo elemento $(0, 0)$ de m , o ponto $(x+1, y)$ de f será multiplicado pelo elemento $(1, 0)$ de m , e assim por diante.

Com isso pode-se elaborar um algoritmo para processar uma imagem f sob uma matriz de convolução m dadas, ambas na forma de matriz de valores, de modo a se gerar uma imagem resultante g a partir da convolução da matriz m sobre a imagem f , como

mostrado a seguir no algoritmo 1:

Algoritmo 1: Algoritmo de convolução de imagens

Data:

$m \Rightarrow$ Matriz de convolução.

$f \Rightarrow$ Imagem Inicial.

$w \Rightarrow$ Largura da imagem em pixels.

$h \Rightarrow$ Altura da imagem em pixels.

$o \Rightarrow$ Ordem da matriz de convolução.

Result: g - Imagem resultante da convolução de m sobre f

```

1 Inicializar g como uma nova matriz[w][h] com zeros;
2 para  $x \leftarrow 0$  to  $w$  faça
3   para  $y \leftarrow 0$  to  $h$  faça
4     OffsetX =  $x - (\text{int})(o/2)$ ;
5     para  $i \leftarrow 0$  to  $o$  faça
6       OffsetY =  $y - (\text{int})(o/2)$ ;
7       para  $j \leftarrow 0$  to  $o$  faça
8         se ! ( $OffsetX < 0$  OU
9            $OffsetX \geq w$  OU
10           $OffsetY < 0$  OU
11           $OffsetY \geq w$ ) então
12            |    $g[x][y] = g[x][y] + f[OffsetX][OffsetY] * m[i][j]$ ;
13          fim
14          OffsetY++;
15        fim
16        OffsetX++;
17      fim
18    fim
19  fim

```

O algoritmo apresentado é o mais simples possível, e é otimizado para a rápida execução da tarefa, portanto ele não apresenta alguns recursos extras, que geralmente aparecem em algoritmos de convolução de imagens digitais, como a adição de um fator de multiplicação r e de um fator de soma *baías*, que altera a imagem resultante g gerada, de modo que o fator r , geralmente dentro do intervalo $[0, 1]$, multiplica todos os elementos da imagem após o cálculo do ponto, e analogamente, o fator de soma *baías*, que soma todos os elementos calculados da imagem resultante g . Tais fatores, no caso deste trabalho não são necessários, pois a matriz de convolução será gerada a partir de um algoritmo evolutivo, que já integra tais valores nos próprios elementos da matriz de convolução, caso seja necessário. Outro fato interessante a se analisar é que este algoritmo utiliza

muito processamento e é diretamente dependente do tamanho da imagem e da máscara de convolução, pois, tomando como exemplo uma matriz de tamanho 5x5 e uma imagem 100x100, haverão 250000 multiplicações e 250000 somas para gerar a imagem final como resultado da convolução entre a imagem e a matriz.

Os filtros ou matrizes de convolução, quando aplicados sobre imagens, utilizando a técnica mostrada anteriormente tem diversas aplicações, como nas áreas de pré-processamento, remoção de ruídos, segmentação, suavização, detecção de bordas, etc.. No caso deste trabalho, é apresentado uma nova aplicação, dentro de uma metodologia que leva a geração de uma imagem resultante a qual representará uma imagem futura a partir de imagens anteriores.

3 Desenvolvimento

Neste capítulo são detalhados como os conceitos apresentados previamente foram desenvolvidos. Suas melhorias, adaptações e correções são apresentadas de forma a facilitar a organização e desempenho.

Foram propostas modificações no ICA de forma a produzir uma implementação que possui maior desempenho na busca pela solução ideal para o problema proposto, fazendo com que suas operações possam ser processadas em paralelo, corrigindo um problema de valores aleatórios não proporcionais quando se usa dimensões de intervalos variados e a generalização na modelagem dos problemas.

A formulação do ambiente apresenta como os dados são trabalhados, formando os mapas de quadrículas para serem processados pelo ICA, e como o problema de previsão é modelado em uma função de avaliação do ICA, onde todo o processo é descrito na seção 3.5, que usa uma aplicação diferenciada, combinando matrizes de convolução e fatores de ponderação.

3.1 ICA Orientado a Objetos

Esta seção contém todas as melhorias efetuadas no ICA, independente de seu uso para previsão, sendo que tais modificações são genéricas para qualquer aplicação. Então esta seção foca em duas formas de otimização, sendo a primeira, a organizacional, impactando diretamente no desenvolvimento, e a segunda, as alterações para performance.

Assim, o desenvolvimento do ICA de forma orientada a objetos é classificada como uma modificação organizacional, e é focada nos quatro pilares da orientação a objetos:

- Abstração, que é responsável por fazer a separação dos elementos presentes no ICA por identidade, propriedades e métodos, e neste caso criando-se 4 elementos:
- Classe ‘ImperialistCompetition’, que define todo o processo evolucionário da competição imperialista desde sua inicialização até o término da repetição presente no algoritmo, obedecendo as condições de parada. Esta é a principal classe do sistema que faz o uso de todos os outros elementos descritos abaixo, dando sentido a cada um deles.
- Classe ‘Country’, que por sua vez descreve as propriedades e métodos que um país possuirá na competição imperialista, é esta classe que define, após a inicialização se um país é um imperialista ou uma colônia, além de armazenar e gerenciar o vetor país(citado nos conceitos do ICA).

- Interface ‘IFitness’, que traz a definição do problema a ser abordado pelo os países em competição.
- Classe abstrata ‘StopCondition’, que representa uma condição de parada. Na qual após uma competição imperialista, uma ou mais condições de parada podem ser verificadas em sequência, parando a execução do algoritmo. Assim pode-se ter uma condição de parada por número máximo de décadas, ou por número de imperialistas competindo sendo verificadas ao mesmo tempo. A condição que primeiro for verificada como verdadeira irá parar a execução do ICA.
- Encapsulamento, que torna o desenvolvimento mais flexível, de modo que novas implementações sejam mais fáceis de criar ou modificar, pois mantém as partes da implementação separadas, como se fosse blocos ou módulos, os quais apenas precisam ser conectados para que funcionem. Permite também o isolamento de propriedades privadas, sendo estas acesso apenas por métodos específicos *getters* e *setters*. Permitindo a adição de propriedades que facilitam o acesso aos atributos específicos de um elemento, ou validando valores inseridos ou alterados por agentes externos ao escopo do objeto.
- Herança, usada para modelar elementos mais complexos definidos de acordo com o modelo do problema, permitindo a extensão a classe *Country* para adicionar propriedades ou métodos. Além de que ao se usar em combinação com polimorfismo pode alterar o comportamento de métodos através de sobrescrita.
- Polimorfismo, que permite referenciar tipos mais abstratos, os quais apenas definem o comportamento dos elementos concretos que os implementa. Neste caso, a definição da interface *IFitness* e da classe abstrata *StopCondition*, é a aplicação do polimorfismo no ICA, que faz com que diversos problemas possam ser criados separadamente e sem que seja preciso alterar o funcionamento interno do ICA. A sobreescrita de métodos da classe *Country*, quando estendida para alteração de comportamento e/ou funcionalidade, também é uma utilização de polimorfismo, uma vez que o ICA continua usando apenas os métodos e propriedades da classe base *Country* (mesmo se sobreescritos), e não de sua extensão, que provavelmente será usada pela implementação da interface *IFitness* em questão.

Uma vez definido como será o desenvolvimento no aspecto de organização, aplicando os conceitos de orientação a objetos, os próximos passos devem focar no aspecto de otimização da aplicação para a obtenção das respostas em um período de tempo aceitável. Assim, foram definidas duas alterações, na qual uma delas não implica diretamente na alteração do funcionamento do ICA, sendo estas, a implementação do ICA de forma que este processe as funções de aptidão de forma paralelizada, e uma outra modificação que

altera como as colônias se relacionam com seus países imperialistas durante o processo evolutivo.

Dentro da segunda categoria de modificações, relacionadas à performance, são propostas algumas alterações no funcionamento de como algumas etapas do ICA são efetuadas e também são propostas algumas funcionalidades adicionais que podem ocorrer entre as etapas básicas do ICA.

Uma modificação que resultou em um bom aumento de precisão do ICA para soluções complexas é a alteração da forma com que uma colônia se movimenta para seu país imperialista, a qual foi denominada de “movimento refinado”, e que é uma otimização feita no algoritmo do ICA para que ele explore o espaço de busca de uma forma mais homogênea, controlando os ruídos (durante a movimentação) e mantendo a velocidade e precisão de convergência. Outra abordagem, denominada “visão imperial distorcida”, também foi utilizada com o mesmo intuito, buscando outra abordagem de movimento aleatorizado, apresentando bons resultados resultados e com uma capacidade de ser combinado com o primeiro.

As funcionalidades de Revolução Colonial e União Imperial são duas adições mencionadas por (ROCHE et al., 2011) que implicam na alteração do comportamento dos países e impérios, responsáveis por aumentar a velocidade de convergência do ICA para uma solução ótima. São mais sutis que

A Revolução Colonial é uma funcionalidade muito semelhante a mutação presente no algoritmo genético canônico, e por sua vez, neste contexto, faz com que um país que seja colônia de um país imperialista, tenha todos os seus atributos aleatorizados de forma que o ICA se torne capaz de evitar máximos/mínimos locais. A revolução colonial tem uma chance de ocorrer, podendo esta chance cair ou aumentar ao longo do passar das décadas.

A União Imperial é uma funcionalidade adicionada para aumentar a velocidade de assimilação de um império por outro e evitar também para evitar que dois impérios ocupem as mesmas posições. Assim, deve existir um limiar de união, que quando um império chegar próximo o suficiente de outro, inicia-se o processo de união, onde o império mais fraco é englobado pelo império mais forte.

3.1.1 Modelagem e desenvolvimento

O desenvolvimento de problemas que venham a utilizar de técnicas de inteligência artificial são implementados a partir de seu modelo matemático, que geralmente não são otimizados computacionalmente, ou ainda, que não são nem modelados apropriadamente para serem ótimos em determinado ambiente computacional. No ambiente em questão existem diversos tipos de linguagens de programação, as quais possuem diversas características,

que por sua vez, definem as qualidades e desvantagens desta linguagem. As linguagens de programação e suas características principais são diferenciadas em sua essência pelos paradigmas de programação, onde os quatro principais paradigmas são bem caracterizados por (NORMARK, 2013). Tais paradigmas são uma forma de classificar determinada linguagem de acordo com seu estilo de programação e classificam suas características principais como apresentado na Figura 11.

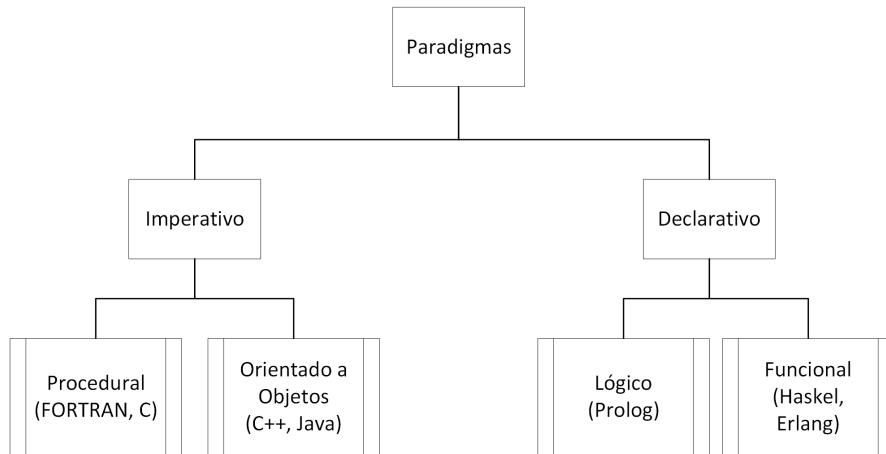


Figura 11 – Paradigmas de programação

Para a modelagem de problemas de inteligência artificial, geralmente são escolhidas linguagens que pertencem aos paradigmas funcionais, porém, tais linguagens possuem mais limitações que as pertencentes ao paradigma imperativo, seja este limite em relação ao controle sobre o sistema ou aos dados. Este é um dos motivos de se ter escolhido uma linguagem pertencente ao paradigma imperativo (em sua essência) para o desenvolvimento deste trabalho. Como pode-se ver na Figura 11 o paradigma imperativo divide-se em dois principais sub paradigmas, o procedural e o orientado a objetos (paradigma escolhido como melhoria para o desenvolvimento).

A linguagem de programação escolhida para o desenvolvimento do ICA pertence ao paradigma de programação imperativo orientado a objetos em sua maior parte, porém algumas de suas características se encaixam nas funcionalidades do paradigma declarativo funcional, que são funcionalidades como delegados (*Delegate* - os quais permitem que funções sejam tratadas como objetos de primeira ordem, o que nada mais é do que um tipo seguro de ponteiro de função), inferência de tipo (que faz dedução automática de um tipo de dados), funções anônimas (ou abstrações lambdas), linguagem de consulta integrada LINQ, entre outros. Estes recursos são capazes de otimizar e imitar algumas das características utilizadas pelo paradigma declarativo funcional dentro de uma linguagem imperativa orientada a objetos. Assim, pode-se dizer que a linguagem é multi paradigmas, pois possui características de orientação a objetos, que permite tratar os campos de dados como objetos manipuláveis através de métodos pré-definidos, e permitindo também que a implementação faça o uso dos seus quatro conceitos mais importantes: abstração,

encapsulamento, polimorfismo e herança, além de usufruir dos conceitos que não fazem parte paradigma imperativo, que também estão presentes na linguagem.

O algoritmo do ICA foi desenvolvido neste trabalho utilizando a linguagem de programação C#. Como descrito acima, esta linguagem multi paradigmas permite o uso dos conceitos de orientação a objetos, que possibilitam uma forma diferente de organizar o problema. Tendo em mente que para se desenvolver uma solução que resolva o problema proposto, será necessário efetuar diversos testes com diversos modelos. A aplicação dos conceitos de orientação a objetos será imprescindível para organizar e manter o desenvolvimento de forma ágil e resiliente. A modelagem do ICA orientado a objetos foi particionado em 4 etapas básicas, na qual cada uma referencia a um dos 4 pilares da Orientação a Objetos.

Primeiramente, foi pensado em como separar os elementos básicos do ICA em objetos de forma que tais objetos pudessem ser caracterizados de forma a ter uma identidade única, seus próprios métodos e propriedades dentro do escopo em questão, abstraindo a concepção do ICA para algo mais tangível como na separação em objetos que podem ser descritos por classes (inclusive interfaces e classes abstratas). Assim, foram criadas duas classes, uma classe abstrata e uma interface, sendo elas:

- Classe *Country*
- Classe *ImperialistCompetition*
- Interface *IFitness*
- Classe Abstrata *StopCondition*

A classe *Country* (Figura 12) representa um país do ICA, sendo este colônia ou império, e é semelhante a um indivíduo no GA. O país possui o conjunto de atributos que serão usados pela função de avaliação de modo que esta função de avaliação calcule um custo para este país, sendo este valor de custo armazenado na própria classe. Esta classe ainda armazena atributos que definem se o país é um país imperialista ou se ele é uma colônia. Logo, para que se possa manusear melhor o país, independente de sua posição política, existe um valor que representa dois estados, dependendo do que o país possa ser, colônia ou império. Se este país for uma colônia, este valor representará o numero do império a que este país pertence. Se este país for um império, este valor será o índice do vetor de impérios ordenados por custos presente na classe *ImperialistCompetition* descrita logo abaixo.

Durante o desenvolvimento foi pensado que poderiam ser criadas mais duas classes, sendo estas a classe Colônia e a classe Imperialista, que derivariam da classe *Country*, porém não foram criadas por dois motivos: (1) A concepção de uma classe Colônia e uma

classe Imperialista que derivassem da classe *Country* não teriam nem propriedades e nem métodos que as diferenciassem da classe, sendo estas diferentes apenas suas identidades (OBS: a classe imperialista até poderia possuir um vetor para as colônias que controla, mas seria redundante e aumentaria a complexidade no controle das operações efetuadas pelo ICA). (2) O segundo motivo é referente ao impacto da implementação destas duas classes, que necessitaria de muito mais lógica para manuseio dos objetos, sendo que o foco não é implementar uma abstração tão afundo da ideia original, mas sim uma implementação de baixa complexidade, que faça com que o sistema seja simples de ser mantido e testado.

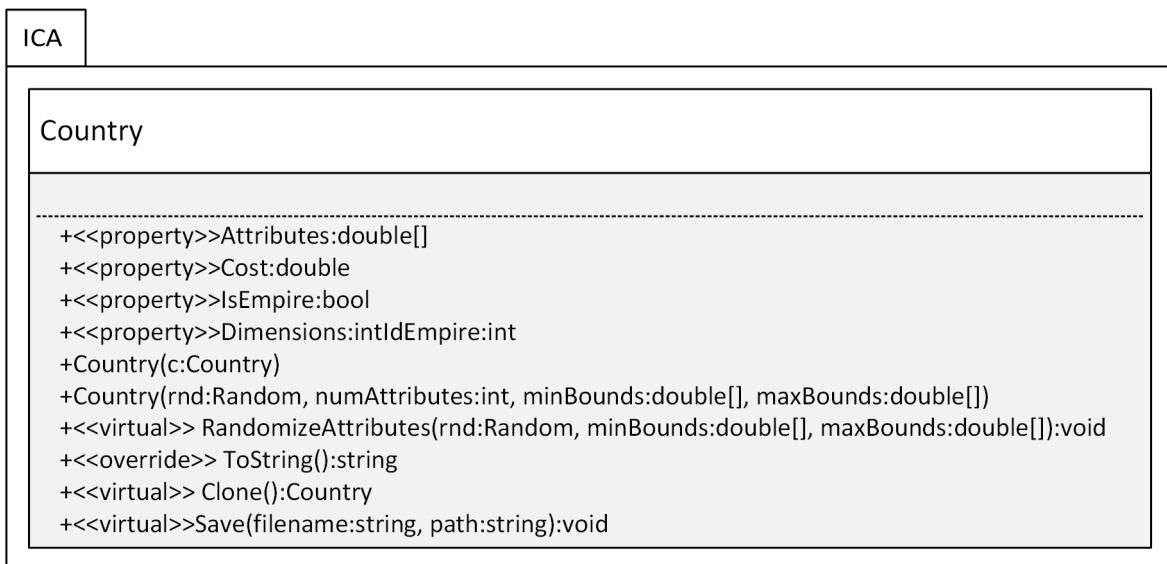


Figura 12 – Classe Country

Para completar a modelagem da classe *Country*, foram adicionados alguns métodos essenciais para o funcionamento genérico do ICA. Dois destes métodos são construtores para a classe, no qual o primeiro recebe como parâmetro um outro objeto do tipo *Country* e inicializa seus valores fazendo uma cópia direta de todos os atributos do parâmetro, o segundo construtor inicializa todos os seus valores com valores padrão da linguagem, porém recebe diversos parâmetros, nos quais são basicamente usados para chamar o método *RandomizeAttributes* logo após a inicialização dos valores da classe. Dentre os quatro métodos restantes o método *ToString* é o mais simples, apenas retornando uma cadeia de caracteres formatada dos valores da classe, o método *Clone* faz uma cópia do objeto em questão usando o primeiro construtor, o método *Save* grava em um arquivo de texto com os dados do objeto, e por fim, o método *RandomizeAttributes* faz uma randomização dentro dos valores *minBounds* e *maxBounds* para todos os atributos do indivíduo.

Nota-se ainda que os métodos *RandomizeAttributes* e *Clone* são virtuais, isto indica que quando a classe *Country* for derivada (usando o conceito de herança), estes métodos poderão ser substituídos por novos métodos com uma lógica diferenciada (OBS: o construtor da classe também pode ser sobreescrito e ter sua lógica alterada em um

cenário de extensão da classe *Country*). O fato de poder se criar uma classe tomando como base a classe *Country* e mesmo assim o ICA continuar funcionando é devido à aplicação do conceito de polimorfismo aplicado na modelagem da aplicação. Neste caso, quando se implementa uma classe *Country* estendida, o ICA não é responsável por instanciar a lista de países, esta responsabilidade é do método *GenerateCountries*, que deve ser implementado por classes derivadas da interface *IFitness* (descrita logo abaixo). Assim, na classe *Country*, foram abordados todos os quatro conceitos essenciais de orientação a objetos, mantendo uma grande flexibilidade para a utilização genérica do ICA.

A classe *ImperialistCompetition* será abordada por último, pois é nela que todos os outros componentes são interligados e é necessário que os demais componentes estejam bem definidos. Então, continuando com a interface *IFitness*, que é responsável por fazer com que os objetos que venha a implementá-la, obrigatoriamente tenham os métodos de interface implementados. Como pode-se observar na Figura 13, de todos os métodos, o mais importante é o método *Eval* e em seguida o método *GenerateCountries*, descritos mais detalhadamente logo abaixo.

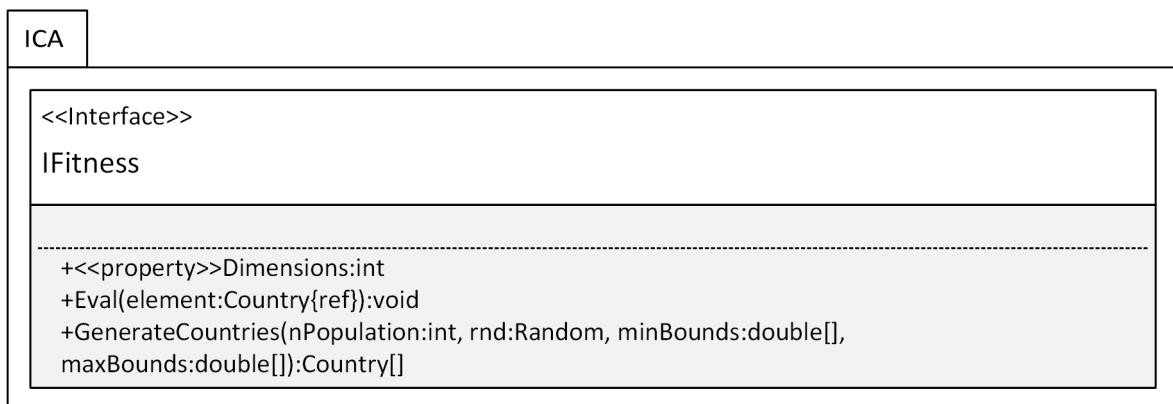


Figura 13 – Interface *IFitness*

O método *Eval* é o mais importante, ele representa a própria função de avaliação do problema em questão e é o principal fator que levou a criação desta interface. É neste método que ocorrem os cálculos para a avaliação de cada país e consequentemente a atribuição do valor do custo calculado a cada país em questão. É essencial que seja passado como parâmetro a referência de um país para que o custo seja calculado a partir dos valores deste, e em seguida já atribuir o valor do custo a este país passado por referência. Assim, qualquer alteração ao país, que ocorra dentro da função *Eval* será persistida para o objeto no escopo da classe *ImperialistCompetition*, vista mais adiante que é a responsável por fazer as chamadas de avaliação dos países, desta forma evita-se múltiplas trocas de mensagens e simplifica-se o funcionamento, tanto para o processamento serial, quanto para paralela.

Na sequência, o segundo método, *GenerateCountries*, é quem define a construção

da lista de países para que o ICA possa iniciar sua execução, a lista de países pode ser iniciada normalmente, gerando-se uma lista e inicializando seus elementos, porém, a grande vantagem de se ter este método de forma explícita e obrigatória para cada problema a ser implementado é referente a liberdade de poder inicializar alguns países (ou todos) em posições privilegiadas, ou seja, se existir algum pré processamento nos dados, pode-se iniciar os países e seus atributos de forma que estes estejam próximos da solução ótima sem que se precise limitar o espaço de busca. Os parâmetros de entrada para este método são:

- *nPopulation* - valor que representa a quantidade de países a serem criados,
- *rnd* - um objeto do tipo *Random* para que a aleatorização inicial seja efetuada sem que ocorram problemas com a paralelização (discutidos no tópico sobre Paralelização),
- *maxBounds* e *minBounds* - que são as duas listas de valores representando os limites superior e inferior respectivamente para cada dimensão do problema.

Nota-se ainda uma propriedade chamada *Dimensions* na interface, sendo que esta propriedade é responsável por definir o número de dimensões do problema em questão. Como visto anteriormente o valor de dimensão é que definirá o tamanho do vetor de atributos dos países, sendo este valor muito usado pelo ICA. A inserção desta propriedade para definir este valor na interface implica que a classe responsável pelo controle do algoritmo não precisa se preocupar em inicializar tal valor (pois, na maioria dos casos, seu valor dependerá da modelagem do problema). Cada problema no ICA deve ser modelado de forma que possa ter suas soluções na forma de um vetor de valores, no caso do ICA tais valores são números de ponto flutuante, sendo o número de elementos deste vetor o número de dimensões do problema.

Por fim, percebe-se que a interface *IFitness* é a responsável por encapsular a lógica do problema a ser otimizado através da implementação de seus membros em uma classe separada para cada problema. Assim, cria-se um nível de abstração que separa o funcionamento do algoritmo da implementação/avaliação do problema, sendo que a classe *ImperialistCompetition* deve obrigatoriamente obter os valores necessários para sua execução (Lista de países, número de dimensões do problema e chamada da função de avaliação de um país) a partir de uma implementação da interface *IFitness*, porém não precisa ter informação alguma sobre como a acontece a geração da lista de países, como avaliação de um dado país ocorre ou como se define o valor de dimensões do problema etc.. Assim, quando se implementa uma função de avaliação para um dado problema, na maioria das vezes existe a necessidade de manter variáveis do problema que não fazem parte diretamente da solução, mas sim do modelo do problema. Estes valores devem estar inseridos na classe que implementa o problema, para que o ICA não precise ter suas

funcionalidades principais alteradas, mantendo a aplicação genérica a qualquer problema deve-se implementar a interface *IFitness* e seus métodos com o foco no problema em questão. Desta forma ICA se mantém genérico a qualquer solução, sendo que qualquer classe que implemente um problema diferente seja polimórfica, ou seja, independente do problema em questão, o ICA apenas utilizará os métodos e propriedades expostos pela interface *IFitness*, não tendo contato algum com os demais elementos da modelagem do problema.

Na metodologia do ICA foi mostrado que podem haver diversas formas de convergir um problema, porém a solução ótima geralmente se apresenta quando existe apenas um país imperialista, e este imperialista possui todas as colônias geradas inicialmente, de forma que tais colônias estejam na mesma posição em que o país imperialista se encontra. Durante a execução de alguns problemas, notou-se que a convergência para apenas um país pode levar muito tempo, ou também que pode-se atingir uma solução aceitável mesmo ela não sendo pertencente ao cenário ótimo de convergência do ICA. Assim, para que o ICA possa gerenciar diversas condições de parada para um dado problema foi pensada uma abstração que permite extrair apenas a lógica das condições de parada para fora do ICA.

A Figura 14 apresenta a classe *StopCondition*, que define como foi modelada a abstração das condições de parada. Uma condição de parada deve implementar esta classe e definir as regras de parada dentro do método *VerifyBreak*, que é chamado sempre que se inicia um ciclo (ou década, contextualizando para o ICA) no algoritmo. Observa-se que diversos parâmetros são passados para este método, assim, as condições de parada mais comuns podem ser implementadas e adicionadas no ICA. Ainda existem duas propriedades na classe abstrata, a primeira, é usada apenas para definir o nome da condição de parada, para fins de análise. Já a segunda, é usada exclusivamente pelo ICA para gerenciar uma lista de diversas condições de parada que possam estar sendo verificadas em sequencia. Note que tal propriedade é pública, porém possui o método *set* como interno (definido pela palavra chave *Internal*), o que significa que apenas a própria instância pode alterar o valor, mas qualquer um pode lê-lo (pois o seu método *get* continua público).

A Figura 15 define as condições de parada:

- Número máximo de décadas - que força parada da evolução do ICA quando um valor de iterações (décadas) for atingido.
- Estagnação por década - que força parada da evolução do ICA quando o valor do melhor custo da solução não se alterar durante um número de ciclos (décadas) estipulado.
- Estagnação por porcentagem de custo - que força a parada da evolução do ICA quando o valor do melhor custo subtraído do melhor custo anterior se manter dentro de uma variação percentual durante um número de ciclos (décadas) estipulado.

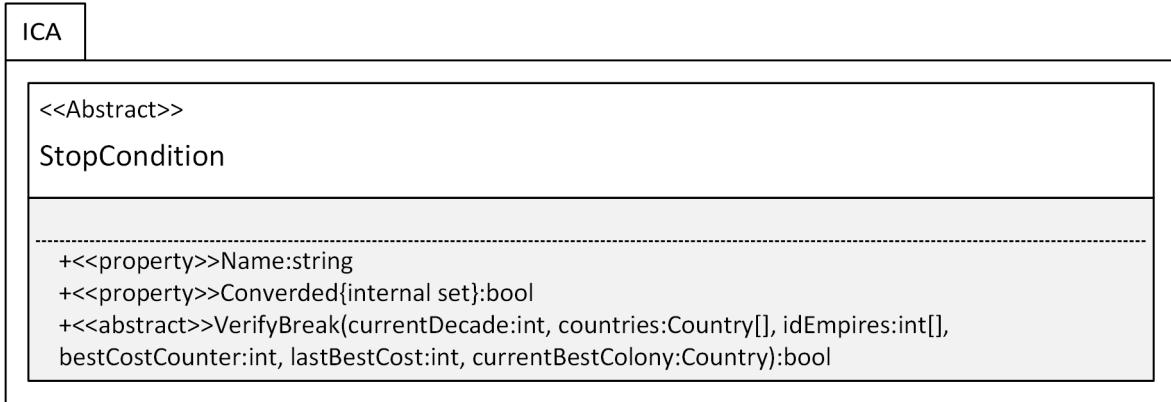


Figura 14 – Classe Abstrata StopCondition

Tais condições de parada foram implementadas e inseridas na rotina de iteração do ICA, para que este possa convergir mais rapidamente para uma solução aceitável. A utilização destas condições de parada é configurável e portanto pode-se utilizar a condição padrão citada na metodologia, que apenas irá parar o algoritmo quando todas as colônias estiverem sob o poder de apenas um império de tal forma que suas posições sejam as mesmas.

Por fim, a classe *ImperialistCompetition* controla toda a competição imperialista, desde a inicialização dos países até o término da competição. Ela é a implementação do fluxograma do ICA apresentado na metodologia, com alterações para adição de novos recursos como processamento paralelo e orientação a objetos, tornando o ICA genérico a qualquer problema. Sendo assim, todos os elementos, lógicas, condições e ideias descritas anteriormente no fluxograma são abstraídas de modo que esta classe processe a competição imperialista idealizada, usando de seus métodos e propriedades em conjunto com os demais elementos modelados (classe *Country*, classe abstrata *StopCondition* e interface *IFitness*).

A Figura 16 representa apenas a classe em questão, destacando em tom mais claros, na área onde se situam os membros da classe, elementos que fazem alguma ligação de relacionamento, seja esta uma associação, agregação ou composição, que podem ser melhor visualizados na Figura 17, que por sua vez representa toda a implementação do ICA orientado a objetos e como os elementos se interagem, porém omite as propriedades e métodos das classes para que melhor visualização. Nota-se também na imagem que descreve a classe *ImperialistCompetition* com suas propriedades e métodos, que para facilitar o entendimento foram adicionados comentários (texto entre '/*' e '*/') que regionalizam as propriedades e métodos usados nesta classe, já que ela é a responsável por implementar, além dos conceitos básicos do ICA, todas as alterações estruturais e de otimização.

Como a classe *ImperialistCompetition* tem uma quantidade muito grande de propriedades e métodos, estes serão melhores detalhados no anexo, e neste texto, apenas serão detalhados os processos principais ao funcionamento da classe e como os elementos são usados para executar o processo evolutivo do ICA.

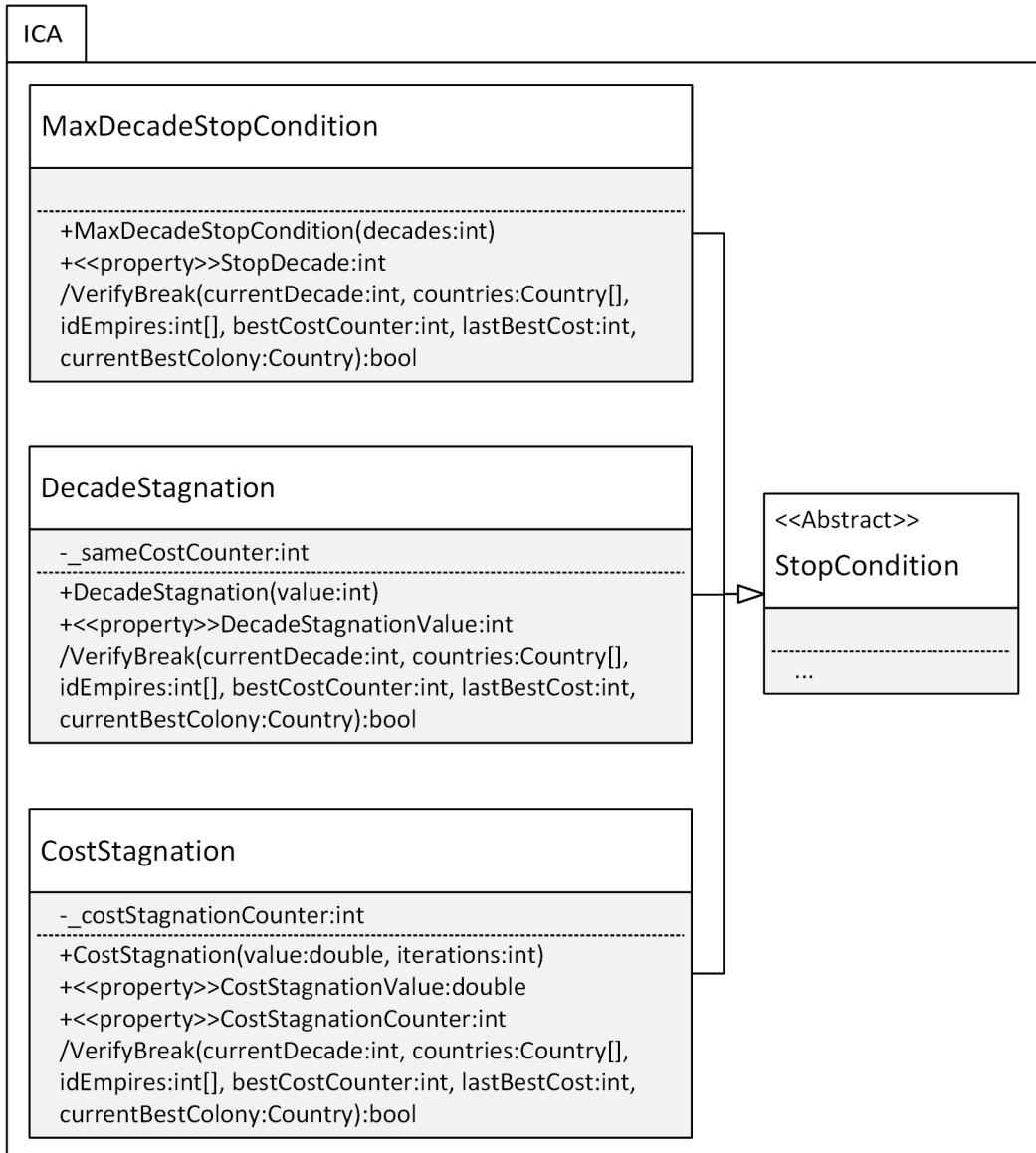


Figura 15 – Implementação das condições de parada

Uma das partes mais complexas do ICA é o preparo para o processo evolutivo, referindo-se a etapa de inicialização dos países presente no fluxograma do ICA apresentado na metodologia pela figura 5, países estes, que por sua vez, devem estar separados em impérios antes que o processo evolutivo de fato comece. Quando se instancia a classe *ImperialistCompetition*, ainda não são efetuadas as tarefas responsáveis por esta etapa de inicialização dos países, pois ainda existirão valores que devem ser configurados, como a adição de um objeto do tipo *IFitness* através da propriedade *Fitness*, e definição dos limites do problema, sejam estes chamados pelo método *InitializeBounds* ou adicionados manualmente nas propriedades *MinBounds* e *MaxBounds*.

A partir do momento em que todos os valores estão configurados apropriadamente, pode-se executar o método *Run()*, que é dividido em duas etapas principais, onde uma delas consiste na etapa de inicialização dos países, e refere-se diretamente à primeira atividade

presente no fluxograma do ICA apresentado na metodologia pela figura 5, e a segunda consiste no conjunto formado por todos os elementos presentes no *loop* principal, definido como processo evolutivo, referentes ao restante das atividades presentes no fluxograma.



Figura 16 – Classe ImperialistCompetition

Então, assim que o método *Run()* é chamado, inicia-se a etapa de inicialização dos países, que por sua vez tem como atividade inicial, instanciar as condições de parada padrões, sendo elas:

- Condição de parada por número máximo de décadas;
- Condição de parada por estagnação de custo idêntico durante as décadas;
- Condição de parada por estagnação de custo em um intervalo durante as décadas, do ICA.

Observe ainda, que o relacionamento apresentado na Figura: Diagrama de classes ICA é de agregação para estas três condições de parada, pois é nesta classe que elas são criadas e mantidas, porém o ciclo de vida destes objetos podem se estender para além do deste objeto, uma vez que existe a propriedade *AchievedStopCondition*, que expõe tais objetos publicamente de forma que estes possam ser referenciados fora desta classe, caso contrário, o relacionamento em questão deveria ser de composição.

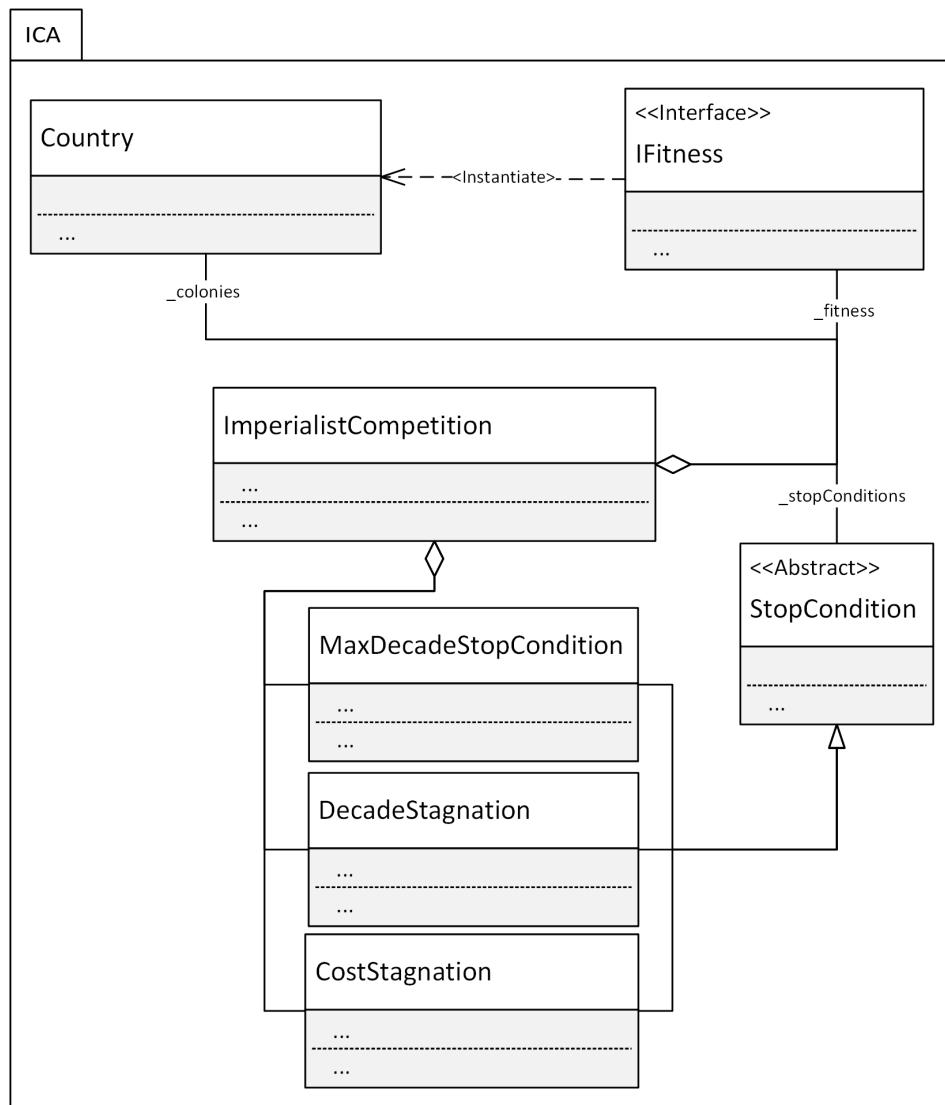


Figura 17 – Diagrama de classes resumido do ICA

Em seguida, inicializam-se os valores constantes ou de cálculo simples, como tamanho do espaço de busca (*SpaceSearchSize*, pelo método *InitializeSpaceSearch*), número

populacional ($nPop$), número de impérios ($nImp$) e número de colônias ($nCol$). Então, com estes valores já estipulados, faz-se uma chamada ao método *GenerateCountries* do objeto *Fitness* para que seja criado o vetor de países que será usado no processo evolutivo.

Mesmo tendo em mão este vetor com os países já inicializados para o problema em questão, não se pode dar a etapa de inicialização como terminada, pois ainda não foram gerados os impérios para que se comece a competição imperialista. Então, tão logo que se crie o vetor de países *Countries*, deve-se fazer a chamada para a avaliação do custo de cada país contido nele. Neste caso, o processamento dos custos de cada país é feito pela chamada do método de avaliação *Eval* do objeto *Fitness* para cada país, que é feito de forma paralela, agilizando o processo de avaliação. Observe que deve-se tomar muito cuidado ao implementar o método de avaliação no objeto *Fitness* para que este não compartilhe recursos globais (somente para escrita, pois a leitura de dados pode ser feita paralelamente), de forma que as diversas chamadas a este método de forma paralela não entrem em conflito, prendendo os recursos de processamento apenas para a tarefa em questão.

Com todos os custos calculados, deve-se definir agora, quais dos países serão os países imperialistas, porém, como foi decidido não criar classes extras apenas para separar imperialistas de colônias, foi criado um vetor de índices, que armazena quais os índices presentes no vetor de países *Countries* representam os países imperialistas. Este vetor foi criado utilizando o recurso da linguagem C# chamado LINQ (Language-Integrated Query), em apenas uma linha de código. Caso este recurso não existisse, seriam necessárias dezenas de linhas de código e chamadas de método para a criação deste vetor de índices. O processo para a criação deste vetor utilizando este recurso é relativamente simples, e envolve a seleção dos índices do vetor de países, ordenado-os e pegando os $nImp$ primeiros valores.

Após a criação deste vetor de índices de países imperialistas, deve-se calcular o poder de cada império para que os países restantes sejam distribuídos como colônias dos países imperialistas formando assim os impérios, e para isto, cada país imperialista deve ter um número de colônias que seja proporcional ao seu poder. Para se obter tal valor, gera-se o vetor de valores inteiros *nOfColonies* de tamanho $nImp$, onde cada índice deste vetor representará a quantidade de colônias que cada império terá.

Para se obter os valores do vetor *nOfColonies* deve-se calcular o poder dos países imperialistas, multiplicá-lo pelo número de colônias $nCol$ e por fim arredonda-lo como proposto na equação. Para se obter o valor de poder pn de cada país imperialista deve-se calcular inicialmente o custo normalizado para todos os países imperialistas e então atribuir como poder pn , de um dado império n , a divisão entre seu custo normalizado $C.N.n$ pela soma de todos os custos normalizados como demonstrado na metodologia.

Originalmente o resultado do número de colônias é o arredondamento da multipli-

cação entre o poder pn calculado e o número de colônias $nCol$, porém computacionalmente existe o problema que arredondamentos na computação na verdade são truncamentos, por este ser um ambiente discreto e não contínuo, então é necessária uma etapa extra para verificar se o número total de colônias está correto, e caso não esteja, corrigir-se o valor até que a soma dos valores do vetor $nOfColonies$ seja igual ao $nCol$, adicionando-se 1 e incrementando o índice do vetor, como pode ser visto na primeira atividade condicional do fluxograma presente na Figura 18.

Antes que se distribua as colônias para os impérios, o vetor contendo os países deve ser misturado, pois as colônias devem ser aleatoriamente distribuídas entre os impérios. Quando se efetua esta mistura, o vetor $IdEmpires$ deve ser redefinido para que se atualize os novos índices dos países imperialistas, pois o vetor de países $Countries$ sempre conterá os países imperialistas e suas colônias, e também nunca terá sua quantidade alterada. O ICA é um algoritmo que não destrói ou constrói países durante seu processo evolutivo, ele apenas altera seus atributos.

A distribuição das colônias para os impérios ocorre iterando-se em i inicialmente pelo vetor de impérios $IdEmpire$ uma única vez, e a cada iteração de i , itera-se em j , $nOfColonies[i]$, efetuando assim a relação império (pelo vetor $IdEmpire$) por número de colônias (pelo vetor $nOfColonies$). Agora basta ir iterando o valor $nCountry$ sempre que se atribuir uma colônia a um império. Esta atividade que distribui as colônias entre os impérios pode ser melhor visualizada pelo fluxograma apresentado na Figura 19.

Após a distribuição das colônias entre os impérios, ocorre uma verificação para certificar-se de que o último império tem pelo menos uma colônia, e caso ele não tenha, a ele é atribuída uma colônia arbitrária de um dos outros impérios. Note que esta etapa traz a impressão de ser semelhante a etapa que verifica se a soma do vetor $nOfColonies$ é igual a $nCol$, porém são coisas distintas, pois a condição $nOfColonies == nCol$ não implica que todos os impérios devam ter pelo menos 1 colônia.

Com a etapa de inicialização dos países finalizada, agora existem impérios, os quais possuem colônias que foram atribuídas de forma proporcional e arbitrariamente, e estes impérios e suas colônias estão espalhados por todo espaço de busca aleatoriamente, prontos para iniciar o processo evolutivo do ICA. O processo evolutivo no ICA caracteriza-se por um processo iterativo (*loop*) onde os elementos, que já estão dispostos no espaço de busca, começam a alterar seus atributos para encontrar a melhor solução presente neste espaço de busca. Então, durante o processo evolutivo ocorrem várias operações nas colônias e impérios de forma a atingir a melhor solução, e consequentemente parar este processo iterativo.

Observando novamente o fluxograma do ICA apresentado na metodologia pela figura 5, é possível notar as operações:

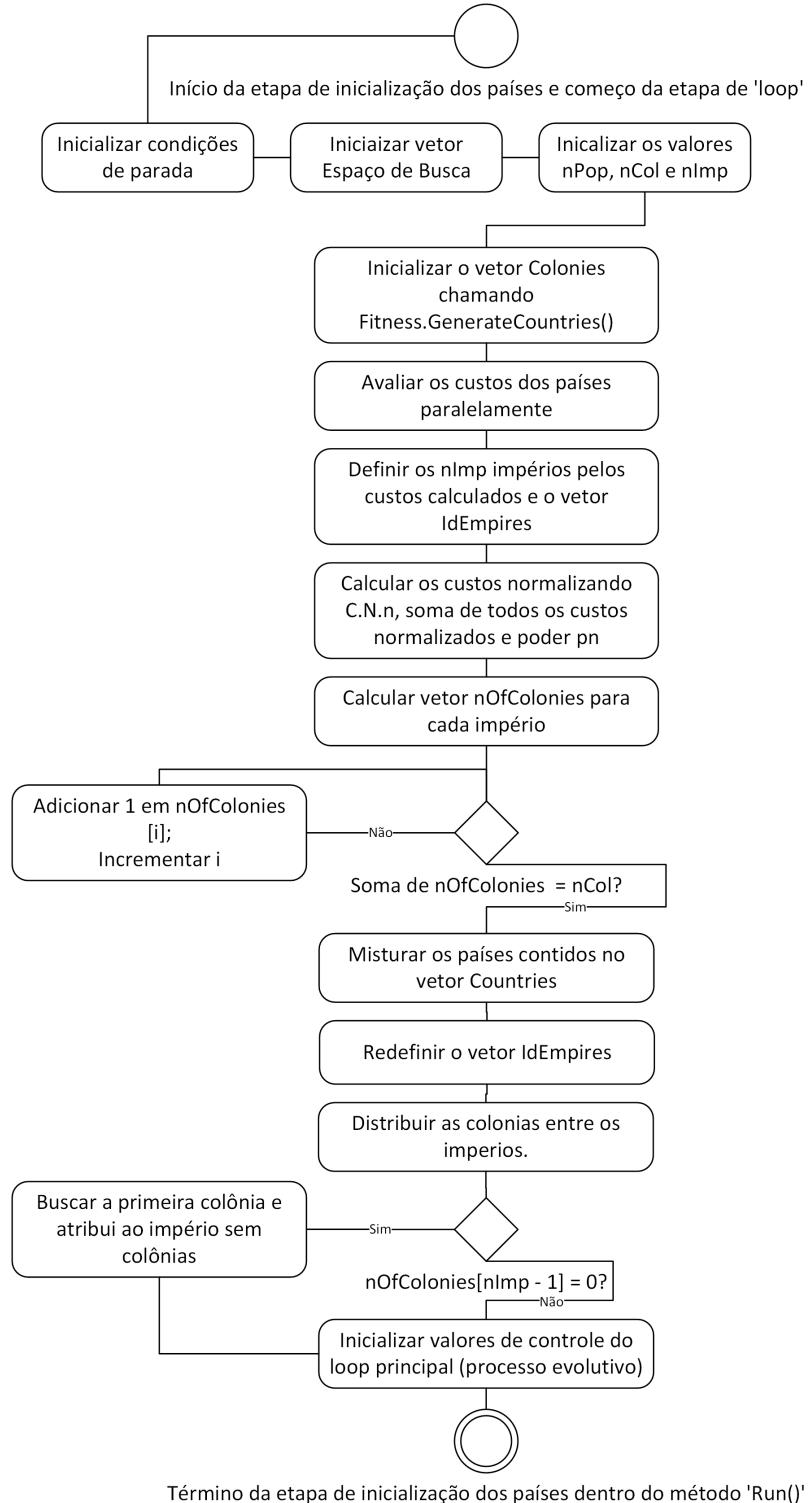


Figura 18 – Fluxograma da etapa de inicialização no método Run

- Assimilação colonial - movimentação da colônia para o império,
- Possessão imperial - colônia mais poderosa toma o império para si, e
- Competição imperialista - a colônia mais fraca do império mais fraco é passada para o império mais propenso a tê-la.

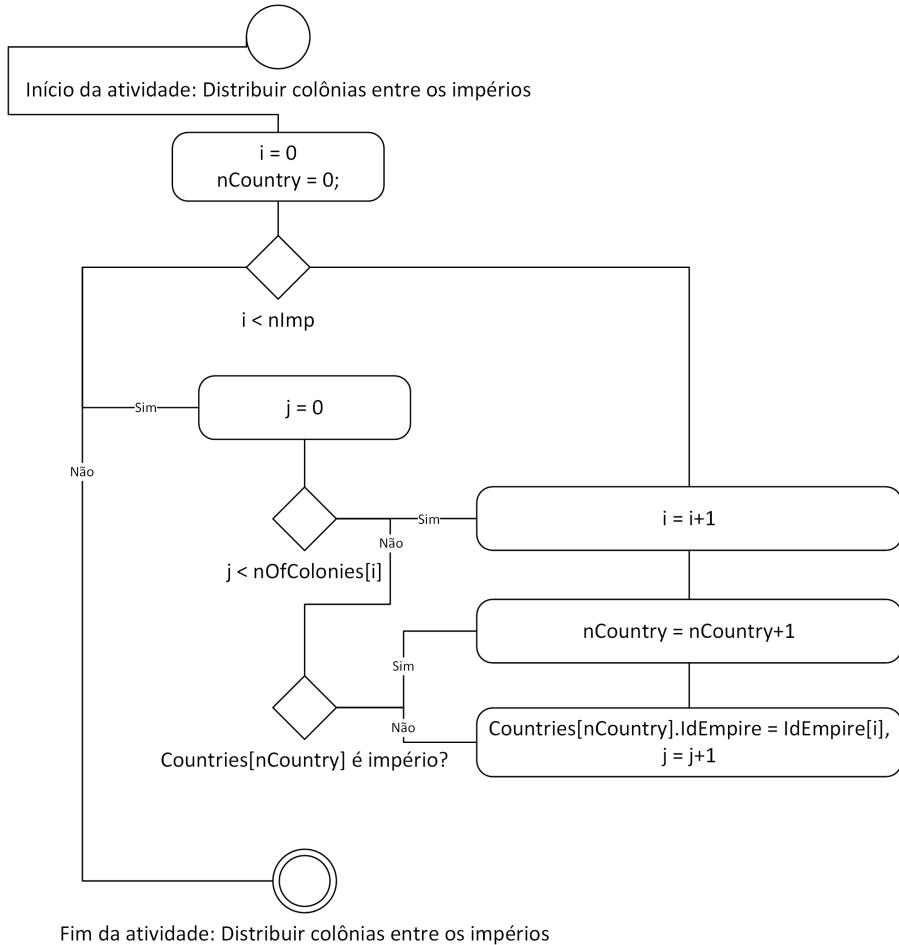


Figura 19 – Fluxograma sobre a distribuição de colônias entre os impérios

A delimitação destes elementos pode ser observada na Figura 20, assim como a separação entre a etapa de inicialização e o processo evolutivo. Estas operações são suficientes para encontrar soluções em qualquer espaço de busca, porém, este trabalho apresenta algumas adições de operações ou melhorias de alguns métodos, a fim de otimizar tanto o tempo de convergência quanto a precisão da busca pela melhor solução, e ainda, de modo que estas operações ocorram de forma agregada ao fluxo anterior. Estas alterações são mencionadas nas subseções seguintes, e aqui nesta seção serão apresentados apenas alguns detalhes referentes ao funcionamento de tais alterações.

A primeira etapa, logo após a entrada no *loop* principal é a de notificação, que envia para todos os elementos registrados nesta classe uma cópia do vetor de países *Countries* e o valor da década atual *CurrentDecade*. Assim, qualquer um que esteja ouvindo pode fazer o rastreio de todos os países e impérios durante as décadas.

Foram efetuadas algumas alterações na ordem de algumas etapas, mas sem alterar o funcionamento do ICA. Assim, para esta próxima etapa foi inserido o bloco responsável por fazer a verificação das condições de parada, que antes de fazer uma verificação pela lista que contém todas estas condições de parada, passa por uma pré verificação, que não

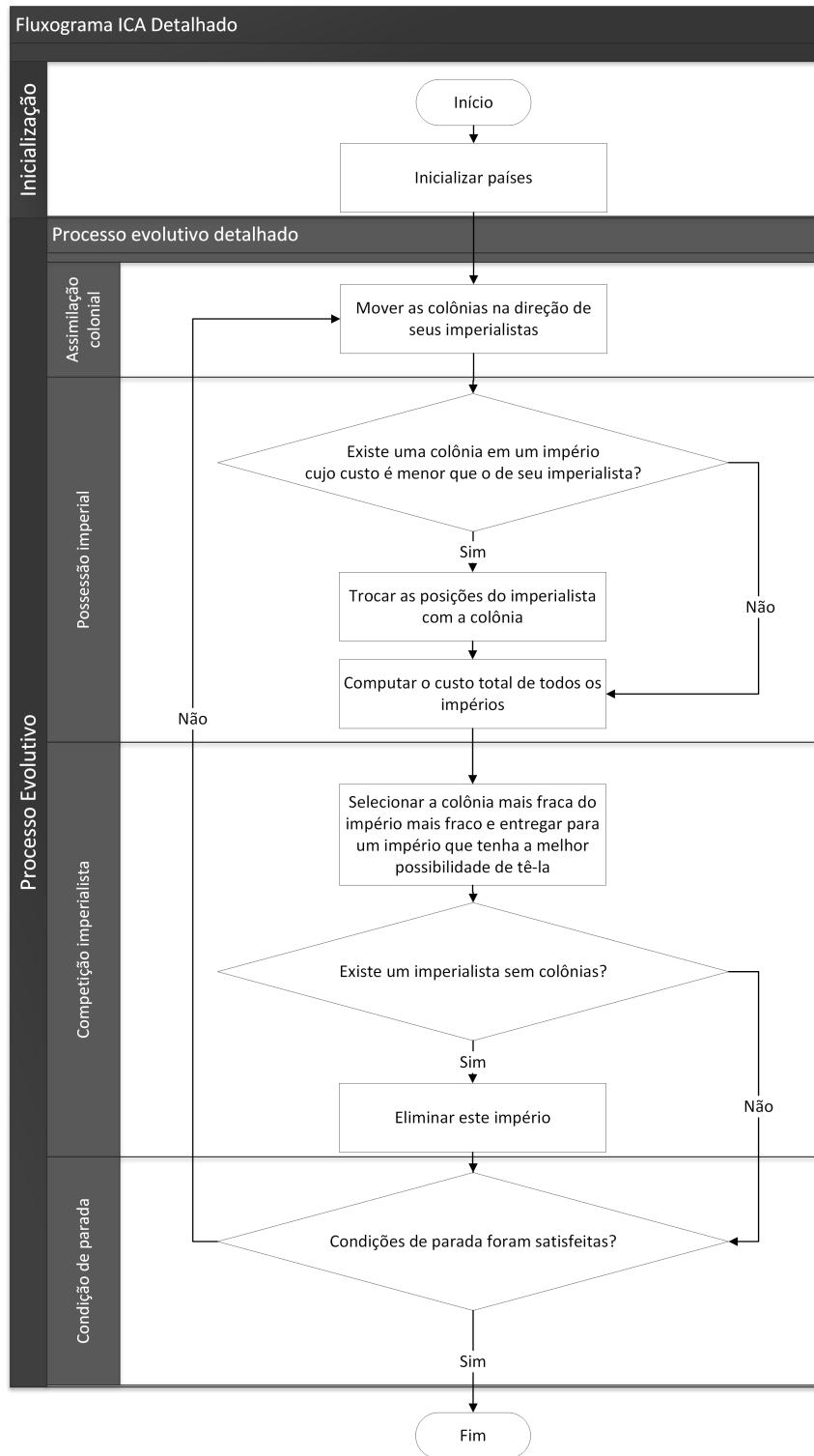


Figura 20 – Fluxograma ICA detalhado

é uma condição de parada, que verifica se o ICA deve convergir para apenas um império (se o número de impérios é 1). Tal lógica de entrada para verificação das condições de parada pode ser vista no diagrama da Figura 21.

Logo após a verificação das condições de parada, atualiza-se o valor da taxa de revolução (Revolução é uma nova operação adicionada, semelhante a operação de mutação

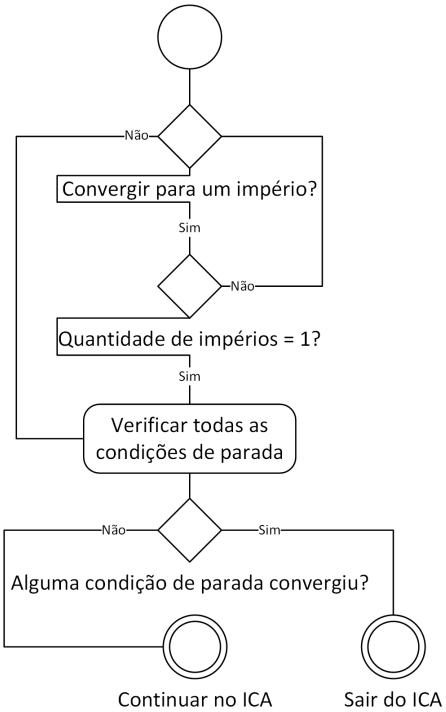


Figura 21 – Fluxograma sobre as condições de parada

do GA canônico e será descrita em um tópico separado), e em seguida efetua uma iteração por todos os impérios, fazendo a chamada para duas operações, sendo a primeira, a operação de assimilação colonial, que move as colônias em direção ao seu país imperialista, e a segunda, a nova operação de revolução, que por sua vez gera novos valores completamente aleatórios para o vetor de atributos de um dado país baseando-se em uma chance de ocorrência relativa a taxa de revolução mencionada anteriormente. Ambas operações são implementadas em métodos que recebem como parâmetro o valor do índice do império no vetor de colônias e uma lista de países contendo apenas as colônias pertencentes a este império.

Assim que a chamada destas duas operações ocorrer para todos os impérios, sabe-se que todas as colônias efetuaram a movimentação em direção ao seu imperialista e que uma porcentagem destas colônias pode ter passado pelo processo de revolução. Como os atributos destes países são alterados, é essencial que se recalcule os custos de todos os países, atualizando-os para que sejam utilizados nas demais operações do ICA. Nota-se também que apenas estas duas operações são responsáveis por alterar os atributos dos países, assim, além de essencial, é imprescindível que a atualização dos custos ocorra.

O cálculo dos custos é feito pela chamada do método `Eval` pertencente ao objeto *Fitness*, que por sua vez é o responsável por avaliar um dado país segundo o problema modelado pelo objeto em questão que implementa os métodos da interface *IFitness*. A avaliação dos países é efetuada de modo paralelo não obstrutivo, ou seja, a avaliação de um dado país não depende dos demais países (Observa-se que a não obstrução no

escopo do ICA é garantida, porém quando se implementa a interface *IFitness* deve-se tomar cuidado para que o compartilhamento de recursos entre as tarefas não faça com que o processamento da avaliação de um país seja interrompido pelo processamento de outra avaliação, que requer o uso do mesmo recurso compartilhado). O fluxo do ICA só é continuado quando a avaliação de todos os países é finalizada, independente de se estar avaliando em paralelo ou em série.

Seguindo o fluxo, a próxima operação verifica se existe uma colônia com poder maior que seu império, e caso positivo, esta colônia se torna o país imperialista, tomando para si as colônias do antigo império. São feitas diversas chamadas a esta operação, de forma iterativa, passando por todos os impérios, para que tal verificação possa ser efetuada através de um método que recebe como parâmetro o valor do índice do império no vetor de países *Countries* e a referência do vetor que contém os índices de todos os impérios *IdEmpires*.

Esta operação executa, inicialmente, uma seleção no vetor *Countries* a fim de filtrar todas as colônias do império em questão, em seguida, é necessário que verifique-se se esta seleção resultou em um número de colônias igual a 0, o significado é que este é um país imperialista sem colônias, e caso isto ocorra, para manter a consistência do ICA para as próximas operações, deve-se eliminar este império. Como mencionado anteriormente, um império quando é eliminado, têm seu país imperialista se tornando uma colônia do império mais forte, assim, esta operação segue os seguintes passos durante sua execução:

- Buscar pelo índice do melhor império e armazena localmente em *bestEmpireId*.
- Transforma o país imperialista sem colônias em uma colônia do país de índice *bestEmpireId*.
- Remove o índice do país imperialista que se tornara colônia, do vetor de índices de impérios *IdEmpires* (sendo este o primeiro motivo de se passar a referência do vetor *IdEmpires*, pois ele será atualizado).

Caso a seleção das colônias seja maior que 0, significa que pode existir uma colônia cujo poder é maior que o de seu império, então se esta verificação for verdadeira, a colônia possuirá seu império, transformando o antigo país imperialista e suas colônias em colônias suas. Para efetuar esta troca executa-se a operação na sequencia:

- Armazenam-se ambos os índices, da colônia que tomara o império e do país imperialista respectivamente em *newEmpireId* e *oldEmpireId*, em cache local;
- Transforma o país de índice *newEmpireId* o novo país imperialista.
- Transforma o país de índice *oldEmpireId* em uma colônia do país de índice *newEmpireId*.

- Passa pela lista de colônias do antigo império alterando o índice das colônias para *oldEmpireId*, obviamente com exceção do país de índice *newEmpireId*, que agora é o novo país imperialista.
- Altera o valor do *IdEmpires*, atualizado-o com o valor do índice do novo império *newEmpireId* (sendo este o segundo motivo para se passar a referência do vetor *IdEmpires* para este método).

Assim termina-se a etapa de possessão imperial, que pode efetuar duas operações no vetor *IdEmpires*, sendo estas, o rebaixamento de um império sem colônias para colônia do império mais poderoso e a própria possessão imperial propriamente dita. Nota-se que a primeira deve sempre ocorrer antes de operações que fazem o remanejamento de colônias e impérios, assim, operações descritas mais adiante, que fazem tal remanejamento, já implicará nesta pré verificação.

A próxima operação, idealizada por (ROCHE et al., 2011), é uma das adições ao fluxo normal do ICA e diz respeito a união de dois impérios próximos. Esta funcionalidade agiliza o processo de convergência, pois quando dois impérios se aproximam e ambos possuem muitas colônias, seriam necessárias diversas décadas até que o império mais forte conseguisse assimilar por completo o império mais fraco, utilizando-se apenas a competição imperialista, tomando colônia por colônia.

Esta operação calcula um valor de limiar para união de dois impérios, sendo que este valor é definido com o valor armazenado em *UnitingThreshold* vezes a norma do vetor que contém os valores dos tamanhos do espaço de busca de cada dimensão, representado por *SpaceSearchSize*, então tem-se que

$$\text{Limiar} = \text{UnitingThreshold} \cdot \text{Norma2}(\text{SpaceSearchSize})$$

Com o limiar já calculado, itera-se por cada império comparando-o com os demais impérios. Para comparar um império com outro, e de fato iniciar o processo de união, deve-se antes calcular a distância entre ambos, e se esta distância for menor que o limiar definido acima, une-se os dois impérios. Assim, a operação de união entre impérios é muito semelhante com a operação de eliminação de um império, apenas adicionando a transferência das colônias do império mais fraco para o mais forte. Nota-se que este método exige a referência do vetor *IdEmpires* para que ele possa ser alterado também no escopo (global) em que este método for chamado.

Por fim, o último método a ser chamado é o responsável por executar a competição imperialista, removendo a pior colônia do pior império e dando esta colônia para o império mais propenso a possuí-la. Nota-se novamente que este método também exige a referência do vetor *IdEmpires* para que ele possa ser alterado também no escopo em que este método

foi chamado, pois assim que uma tomada de colônia ocorrer, pode acontecer de um império ficar sem colônias, então a eliminação deste império ocorrerá logo após a tomada (ou tentativa de tomada) de uma colônia deste império.

Os detalhes do que ocorre dentro deste método podem ser vistos no fluxograma da Imagem 22. Observe que antes mesmo de se iniciar o processo, existem duas verificações sendo a primeira a verificação da chance de ocorrência da competição nesta década, e a segunda é a verificação de certificação por número de impérios (obviamente é necessário que existam no mínimo 2 impérios para que se ocorra a competição imperialista). Logo após as verificações calcula-se o poder total de todos os impérios baseando-se no seu custo e em uma proporção da média dos custos de suas colônias, como pode-se ver no trecho abaixo.

$$\text{possessionProbability}[i] = \frac{\text{power}[i]}{\sum(\text{power})}, i : 0 \text{atén} \text{Imp}$$

Com o poder total de cada império calculado e armazenado no vetor *power* seleciona-se qual é o império mais fraco, e em seguida normaliza-se os custos totais, sobrescrevendo os valores do vetor *power*, e calcula-se o vetor de probabilidade de possessão *possessionProbability*.

A partir do vetor de probabilidade de possessões seleciona-se o império que mais tiver a probabilidade de possuir uma colônia do império mais fraco selecionado anteriormente. Na conceituação do ICA seleciona-se a colônia mais fraca do império mais fraco, mas nesta implementação optou-se por selecionar uma colônia aleatória do império mais fraco com a intenção de acelerar a convergência. Lembrando que a seleção da colônia só acontecerá se o número de colônias do império for maior ou igual a 1. Por fim, verifica se o império que teve a colônia trocada não tem mais colônias, e caso verdadeiro elimina-se este império fazendo com que ele se torne uma colônia do império mais forte.

Após terminar a chamada do método *ImperialistCompetition*, conclui-se o *loop* principal incrementando o valor de década. Ao retornar para o início do *loop*, são verificadas as condições de parada até que alguma delas retorne o valor de parada e o algoritmo finaliza sua execução. Quando o algoritmo termina a execução do método *Run*, a lista de colônias estará posicionada no lugar ótimo para o problema em questão (isto é, seus atributos já estarão otimizados). Os dados podem ser acessados através de suas propriedades públicas pertencentes ao objeto do tipo *ImperialistCompetition* instanciado em memória.

O desenvolvimento do ICA orientado a objetos proporciona principalmente praticidade no que diz respeito a modelagem genérica de problemas, ou seja, qualquer problema que se queira otimizar pode fácil e rapidamente ser implementado para que o ICA consuma os métodos da interface *IFitness* que descreve o problema a ser otimizado. Outra vantagem que a orientação a objetos trouxe foi a organização dos conceitos, de forma que se possa ter

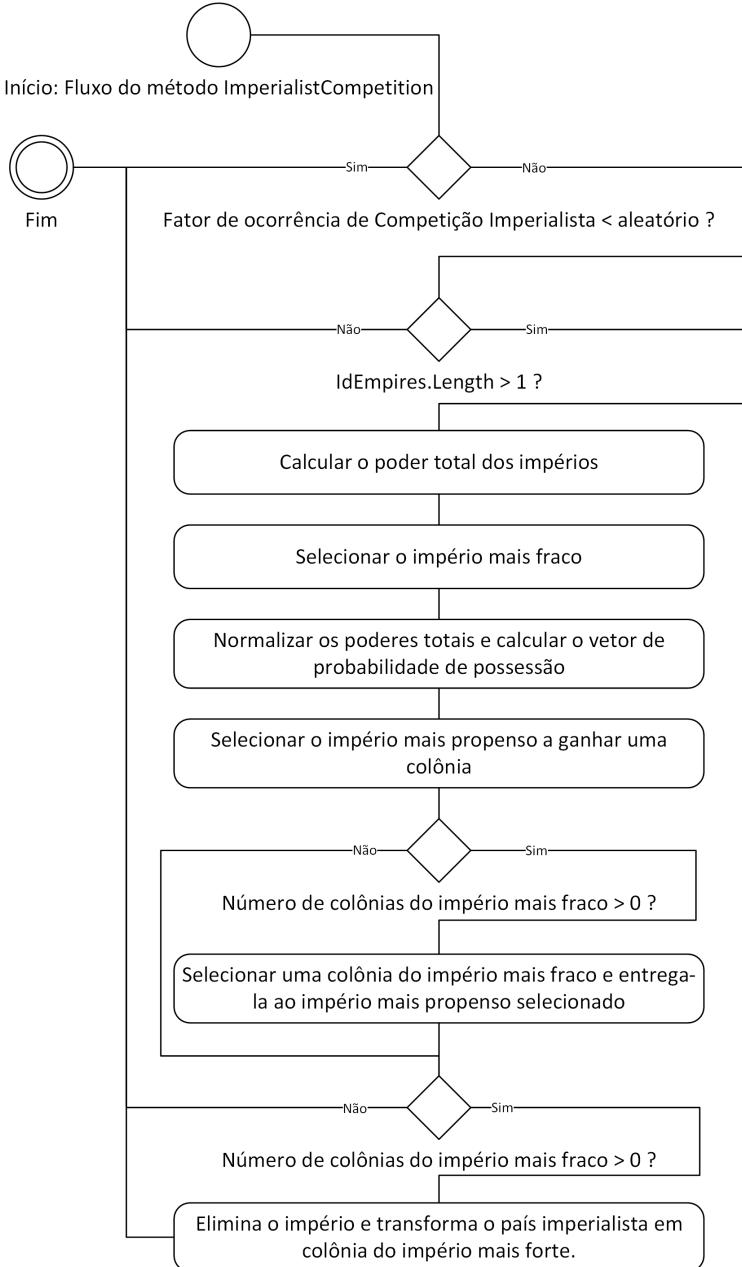


Figura 22 – Fluxograma competição imperialista

um profundo entendimento de como o algoritmo efetua suas operações até que se chegue em soluções ótimas. E por fim, num contexto geral a modelagem do ICA foi pensada para que ele demonstrasse desempenho ótimo, dependendo basicamente do tempo que a função de avaliação leva para executar e nada mais.

Todas estas alterações de otimização do ICA apenas foram possíveis através do desenvolvimento Orientado a Objetos, que permite encapsular as entidades e desenvolver suas lógicas separadas em métodos bem definidos, assim, abstrai-se a solução em objetos capazes de atender e concentrar os aspectos essenciais do contexto (competição imperialista) de uma forma bem definida. Assim, a implementação do ICA permite fácil adaptação a

qualquer abordagem ou ambientação de otimização para um dado problema.

3.1.2 Exemplos de aplicação do ICA genérico

Abaixo apresentam-se dois cenários distintos de implementação do ICA Generalista abordando dois problemas, G1 e G2. O primeiro é uma implementação mais simples, em que a avaliação se dá pela necessidade de minimização da função matemática:

Problema G1:

$$\begin{aligned} F(x, y) &= x \cdot \sin(4 \cdot x) + 1.1 \cdot y \cdot \sin(2 \cdot y), \\ 0 < x, y &< 10, \\ \text{mínimo} : f(9.039, 8.6680) &= -18.5547. \end{aligned} \quad (3.1)$$

A implementação deste problema fica muito simples, pois será um problema de minimização de função matemática dentro de um intervalo fechado, assim, o diagrama de classes simplificado fica como apresentado na Figura 23.

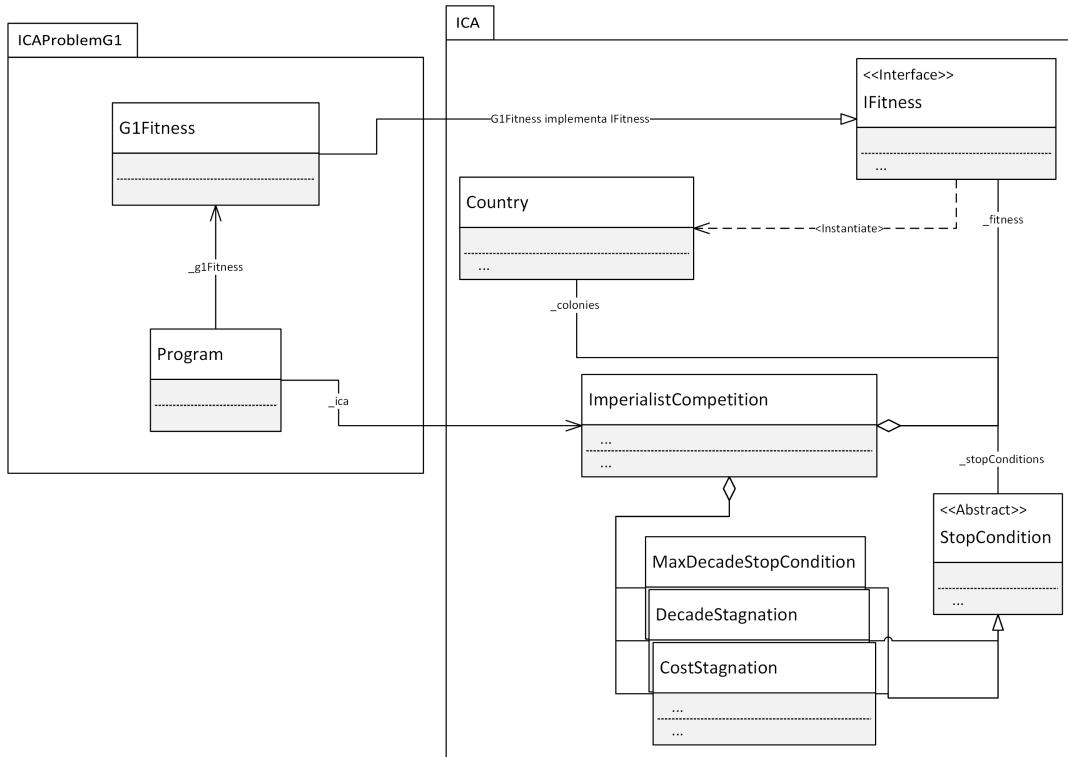


Figura 23 – Diagrama de classes simplificado do problema G1

No diagrama simplificado é possível observar que a estrutura inicial do ICA não foi alterada. No desenvolvimento da solução foi criada uma classe *G1Fitness* implementando a interface *IFitness*, na qual define o problema implementando os métodos obrigatórios da interface *IFitness*.

A modelagem deste problema para o ICA é simples, pois tem 2 dimensões onde seus limites são fechados entre 0 e 10. Assim, tem-se que o vetor de atributos para cada país será como apresentado na Figura 24.



Figura 24 – Vetor de atributos dos países do problema G1

A classe implementada para o problema G1, *G1Fitness*, implementa todos os métodos existentes na interface *IFitness*, sendo assim, a propriedade *Dimensions* apenas retorna o valor 2, pois o problema apresenta duas dimensões(valores de ponto flutuante para x e para y). O método *GenerateCountries* não tem nada de especial, apenas cria instâncias simples da classe *Country*, passando os parâmetros necessários no seu construtor padrão, e agrupa-as em um vetor para o retorno. E por fim o método *Eval* foi implementado de forma que o custo do país seja o valor da função $f(x, y)$ tal que $x = Atributo0$ e $y = Atributo1$, como descrito no algoritmo 2.

Algoritmo 2: Pseudocódigo método *Eval* do problema G1.

Data:

ref element - A referência do país a ser avaliado

Result: nenhum valor de retorno, pois altera-se diretamente na referência de element

- 1 x = element.Attributes[0];
 - 2 y = element.Attributes[1];
 - 3 Fxy = $x \cdot \sin(4.0 \cdot x) + 1.1 \cdot y \cdot \sin(2.0 \cdot y);$
 - 4 element.Cost = Fxy;
-

Para a apresentação dos resultados foi utilizado o próprio console, e os parâmetros de entrada para o ICA foram:

Total de países: 128,

Número de décadas: 512,

Taxa de revolução: 0.9,

Taxa de decaimento de revolução: 0.95,

Coeficiente de assimilação: 0.0001,

Epsilon: 0.025,

Porcentagem inicial de impérios: 0.15

E no término o melhor indivíduo apresentou os resultados:

$$X = 9.0376,$$

$$Y = 8.6725,$$

$$F(X, Y) = -18.55422.$$

Este teste apresentou um erro muito menor que 1%, calculado a seguir:

$$e = \text{Abs}(\text{Valor Esperado} - \text{Valor Obtido}) = \text{Abs}((-18.5547) - (-18.55422)) = 0.00048$$

$$e\% = 0.00258695\%$$

O problema G2 exige uma função de avaliação mais complexa, e se dá no seguinte cenário: Dadas duas imagens, sendo que a segunda (I') é a imagem resultado da passagem de um filtro de convolução (M) qualquer sobre a primeira imagem(I), o objetivo do ICA é encontrar qual foi o filtro (máscara de convolução) utilizado neste processamento. O nível de complexidade deste problema é altíssimo, por este ser um problema de inversão de uma função que quando aplicada pode ocasionar em perda ou geração de informação ou ruído no seu resultado, tornando impossível a descoberta do filtro de uma forma direta. A princípio, nos exemplos, foram utilizados apenas filtros de ordem 3 e imagens em escala de cinza(I e I'), todos representados por matrizes de valores.

A implementação do problema G2 foi mais complexa, e usa quase todas as novas funcionalidades do ICA descritas neste trabalho. Na Figura 25 pode-se observar que não só a classe *IFitness* fora estendida, mas também a classe *Country*. O motivo de se estender a classe *Country* foi para demonstrar duas funcionalidades extras, na qual a primeira tem como objetivo armazenar o resultado obtido pela função de avaliação para que se possa fazer uso posterior sem a necessidade de reprocessamento, já a segunda tem como objetivo alterar o funcionamento da função de randomização de todos os atributos do país, que é chamada sempre que se iniciam os países e também quando ocorre a revolução de uma colônia durante a evolução das décadas.

Para que se alcance um resultado satisfatório a modelagem deste problema deve levar em conta, inicialmente, a ordem do filtro e os valores de cada elemento da matriz que representa este filtro. Então, um dos parâmetros definidos na modelagem do problema deve ser a ordem da matriz de convolução, que define quantos atributos os países terão.

Com a ordem estipulada, tem-se que a propriedade *Dimensions* da classe *G2Fitness* retorna a definição do número de dimensões do problema como sendo $\text{Dimensions} = \text{ordem} * \text{ordem}$. Assim, com o número de dimensões estipulado, para o caso geral de $\text{ordem} = n$, o vetor de atributos deve representar todos os elementos da matriz filtro em uma sequência unidimensional representada conforme a Figura 26.

Com o modelo do problema pronto, antes de implementar os métodos da interface *IFitness* na classe *G2Fitness*, estende-se a classe *Country* para a classe *G2Country*, que por

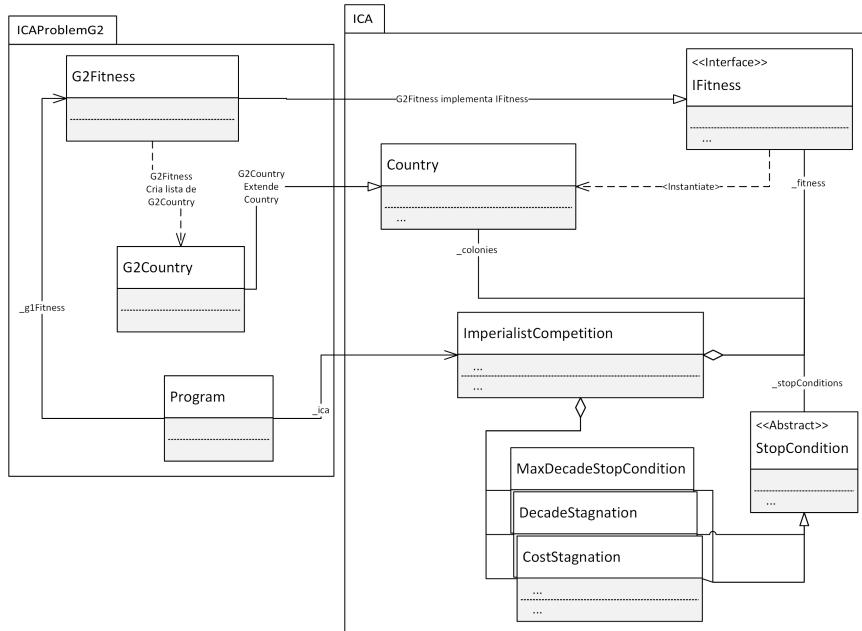


Figura 25 – Diagrama de classes simplificado do problema G2

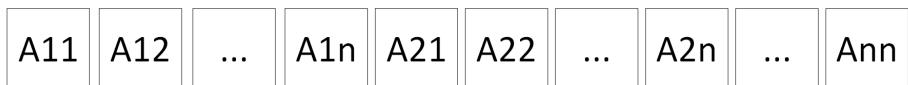


Figura 26 – Vetor de atributos dos países do problema G2

sua vez sobrescreve o método *RandomizeAttributes* para utilizar uma função randômica de distribuição normal (substituindo a distribuição anterior, que era uniforme), que por padrão tem $\mu = 0$ e $\sigma = 0.269$ (de modo que o valor aleatório retornado varie entre 0 e aproximadamente 1). Outra funcionalidade que a classe *G2Country* traz é a adição de uma propriedade para se armazenar o resultado que o método *Eval* da classe *G2Fitness* gera, de forma que se possa salvar esta informação durante ou depois do processamento do ICA.

O método *Eval* da classe *G2Fitness* é implementado para avaliar qual matriz de convolução presente em cada país é a que mais se encaixa na utilizada para gerar a imagem resultado (I') a partir da imagem original (I) conforme o Algoritmo 3

Durante os testes foi utilizada uma mesma imagem para dois filtros diferentes. Os dois filtros utilizados são o desfoque gaussiano e detecção de borda, apresentados respectivamente com sua aplicação na Figura 27.

Para que o ICA possa encontrar estes dois filtros, ambos tiveram como entrada inicial os mesmos parâmetros, diferenciando-se apenas na imagem resultado e nos limites dos atributos. Os parâmetros de entrada que foram configurados na inicialização do ICA foram:

Total de países: 64,

Algoritmo 3: Pseudocódigo método *Eval* do problema G1.**Data:****ref** element - A referência do país a ser avaliado**static** I - Imagem original**static** I' - Imagem resultante**Result:** nenhum valor de retorno, pois altera-se diretamente na referência de element

- 1 Transformar os atributos do país em uma matriz de convolução (M);
- 2 Efetuar a operação de convolução entre a imagem original (I) e a matriz de convolução (M), obtendo a imagem resultante (R) ;
- 3 Calcular a diferença absoluta entre todos os pontos da imagem resultante (R) e a imagem resultado (I'), e armazenar a soma em (d);
- 4 element.Cost = d;
- 5 element.resultado = R;

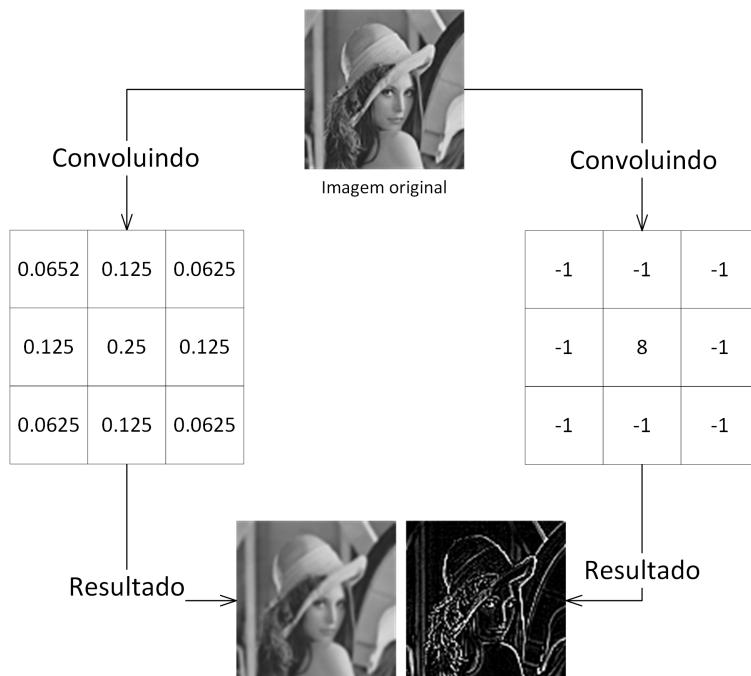


Figura 27 – Exemplo Problema G2

Número máximo de décadas: 2048,

Taxa de revolução: 0.99,

Taxa de decaimento de revolução: 0.9999,

Coeficiente de assimilação: 2,

Epsilon: 0.025,

Porcentagem inicial de impérios: 0.15.

Já os limites, para o filtro de desfoque gaussiano foi escolhido um intervalo entre 0 e 1 para todos os atributos, e para o filtro de detecção de bordas foi escolhido o limite

entre -2 e 10 para todos os atributos.

Os demais valores foram mantidos como padrão e podem ser consultados no Anexo1

Os resultados apresentados pela busca de ambos os filtros foram muito bons, e se aproximaram muito do original, como mostrado a seguir na Figura 28.

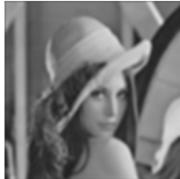
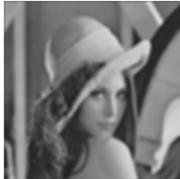
Resultado Esperado	Resultado Obtido
Desfoque Gausiano	
	
0.0652	0,061
0.125	0.127
0.0625	0.062
0.125	0.125
0.0625	0.123
0.0625	0.067
0.125	0.118
0.0625	0.065
Detecção de Borda	
	
-1	-1.033
-1	-0.918
8	-1.040
-1	-0.961
-1	7.882
-1	-0.933
-1	-1.038
-1	-0.924
-1	-1034

Figura 28 – Resultados do problema G2

Observa-se que apesar dos resultados dos filtros não atingirem a exatidão, estes

chegam muito próximo do resultado esperado e geram imagens indistinguíveis a olho nu. As imagens resultado possuem um erro percentual médio de 0.0471% e 0.3342% para as imagens originais geradas a partir do filtro de desfoque gaussiano e de detecção de bordas respectivamente.

Assim concluímos os exemplos demonstrando a praticidade com que a generalização traz para a modelagem de problemas completamente diferentes, mantendo a estrutura principal do algoritmo inalterada, de modo a adicionar tais problemas como se fossem módulos encapsulados os quais são consumidos e processados pelo ICA.

3.2 Análise da operação de movimento do ICA

Antes de apresentar cada alteração na operação de movimentação, foi definido um problema que será usado como referência para ambientalizar e ilustrar o entendimento, tanto da operação de movimento citada originalmente, quanto das operações de movimento alteradas apresentadas neste trabalho. O problema P1 em questão refere-se a minimização da função esfera dentro de um intervalo fechado e bidimensional, como mostra a equação 3.2.

Serão apresentadas várias figuras contendo diversos elementos relacionados ao problema, de modo que a estrela vermelha representa a posição em que o país imperialista se encontra, os pontos azuis ligados por linhas representam uma colônia se movendo em direção ao seu país imperialista durante as décadas e a coloração do fundo da imagem, em escalas de cinza, é uma plotagem da função em questão, neste caso da função esfera, que encontra-se no espaço bidimensional dentro do intervalo $[-10, 10]$ tanto para x quanto para y . Esta função foi escolhida, além de ser bem comum, por ter seu mínimo global bem definido e único para este intervalo.

$$f(x, y) = x \cdot x + y \cdot y, x : [-10, 10], y : [-10, 10] \quad (3.2)$$

Inicialmente é preciso entender o que ocorre durante o processo evolutivo do ICA, então, para estes exemplos a funcionalidade de possessão imperial, a qual ocorre quando uma colônia mais forte que seu país imperialista toma o poder para si, foi desabilitada (exceto alguns casos, detalhados mais adiante), de modo que quando uma colônia passar por uma posição de custo menor ela não tome o império para si, além disso, o valor de β utilizado é menor que 1, a taxa de revolução colonial fora configurada como 0, evitando que países tenham seus atributos sorteados novamente, e, por fim, o espaço de busca será explorado por apenas uma colônia junto de um país imperialista, que irão formar um império contendo apenas dois países. A posição inicial da colônia é definida como $(-8, -5)$ e a posição inicial do país imperialista é definida como $(-2, -2)$, mesmo sabendo que o ponto ótimo será o ponto $(0, 0)$ e o número máximo de décadas é 40.

A forma como as colônias se movem em direção ao seu imperialista é definida pelo resultado de um valor aleatório proporcional a distância entre cada atributo da colônia em relação aos atributos do país imperialista, e de forma independente dos limites de cada dimensão, tal que, este valor aleatório altera cada atributo na mesma proporção. Assim, pode-se dizer que este valor aleatório representa o tamanho do ‘passo’ que uma colônia dará em direção ao seu país imperialista (lembrando que este valor aleatório é gerado uniformemente no intervalo $[0, \beta]$ como mostra a equação 2.7).

A Figura 29 ilustra o movimento linear de uma colônia em direção ao seu país imperialista sem a adição de ruído no movimento. Este ruído no movimento se dá pela alteração de cada atributo da colônia somando um valor aleatório, gerado de forma uniforme, e chamado θ como mostrado na equação 2.9. Observe que não foi encontrado o melhor resultado, o qual estaria na posição $(0, 0)$ para a solução do problema, apenas houve uma convergência para a mesma posição do império, pois o ruído fora desabilitado para este demonstração.

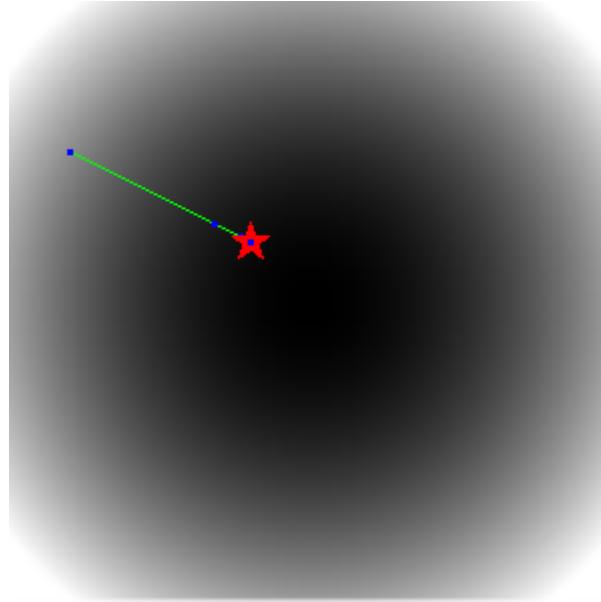


Figura 29 – Movimento Linear

Como dito anteriormente, o movimento de uma colônia em direção ao seu país imperialista tem como objetivo principal explorar o espaço de busca para encontrar soluções melhores. O problema de se mover linearmente uma colônia em direção ao seu país imperialista, sem a adição de um ruído, é o mesmo que limitar a busca por soluções melhores dentro de problemas mais complexos, podendo, ao fim das décadas, não chegar ao mínimo global (solução ótima), como pode ser visto na Figura 29. Porém o fato da solução não convergir para o mínimo global não significa, neste caso, que a solução estagnou-se em um mínimo local, mas sim que o algoritmo é incapaz de evoluir para a melhor solução por não possuir recursos (adição de ruído durante o movimento) para isto. Então é interessante

a adição de um ruído neste movimento, de forma que o espaço de busca seja melhor explorado pela colônia enquanto ela se movimenta em direção ao seu império, como mostra o item *a* da Figura 30. No capítulo 2 este ruído é definido por θ , que define um intervalo para a geração de um número aleatório dentro do intervalo $[-\gamma, \gamma]$ que é adicionado a cada atributo.

O grande problema neste método de inserção de ruídos da forma como é apresentado é a não garantia de proporcionalidade em relação ao espaço de busca de cada atributo, podendo, em um problema mais complexo e com limites diferentes para cada atributo, ter o ruído variando demais em uma dimensão com limites mais apertados, mas ser insignificante em limites de intervalo muito amplo. Além de que com a adição de ruído desta maneira não é possível que a convergência para o cenário ideal (todas as colônias na mesma posição que seu país imperialista) ocorra. Os três primeiros itens da Figura 30 (*a*, *b*, *c*), representam o movimento colonial tendo os ruídos definidos com γ valendo respectivamente $(\frac{\pi}{4})$, como descrito por ATASHPAZ-GARGARI; LUCAS em (ATASHPAZ-GARGARI; LUCAS, 2007), (π) e $(4 \cdot \pi)$. Observa-se que para o item *a*, existe uma quantidade de ruído pequena, que é aceitável, e que poderia estar levando o império em direção ao mínimo global caso o mecanismo de tomada de poder estivesse habilitado. Já os itens *b* e *c* apresentam um significativo aumento no ruído, podendo, também, levar a melhor solução, porém esta variação é muito grande e, consequentemente, faz com que a busca pela solução ótima tenda a ser mais aleatória do que direcionada, podendo fazer com que a solução ótima nunca seja atingida.

O item *d* da Figura 30 apresenta o resultado do mesmo cenário do item *a*, porém com a operação de possessão imperial habilitada. Na imagem, as estrelas menores representam os impérios durante as décadas, e a estrela maior representa a posição do império na última década. Observa-se que com a adição do ruído, houveram diversas trocas pelo controle do império entre os países competidores durante as décadas, até a solução convergir para o mínimo global. Outro fato interessante é que neste caso, o valor do ruído levou a um bom tempo de convergência, porém se a operação de possessão imperial tivesse sido habilitada no item *c*, por exemplo, como mencionado anteriormente, o algoritmo levaria muito mais tempo para encontrar a melhor solução para este problema, pelo fato do ruído estar influenciando demais na definição do passo, que agora passa a ser menos direcionado e mais aleatório.

A movimentação direta da colônia em direção ao seu país imperialista, isto é, sem adição de ruído, apesar de explorar menos o espaço de busca, converge mais rapidamente para seu império, podendo fornecer uma solução próxima da solução ótima. Por outro lado, quando se adiciona o ruído na movimentação, existe uma maior chance de que a colônia encontre melhores soluções do que a apresentada por seu país imperialista, podendo assim, esta colônia se tornar o centro do império. Esta operação ocorrendo diversas vezes leva

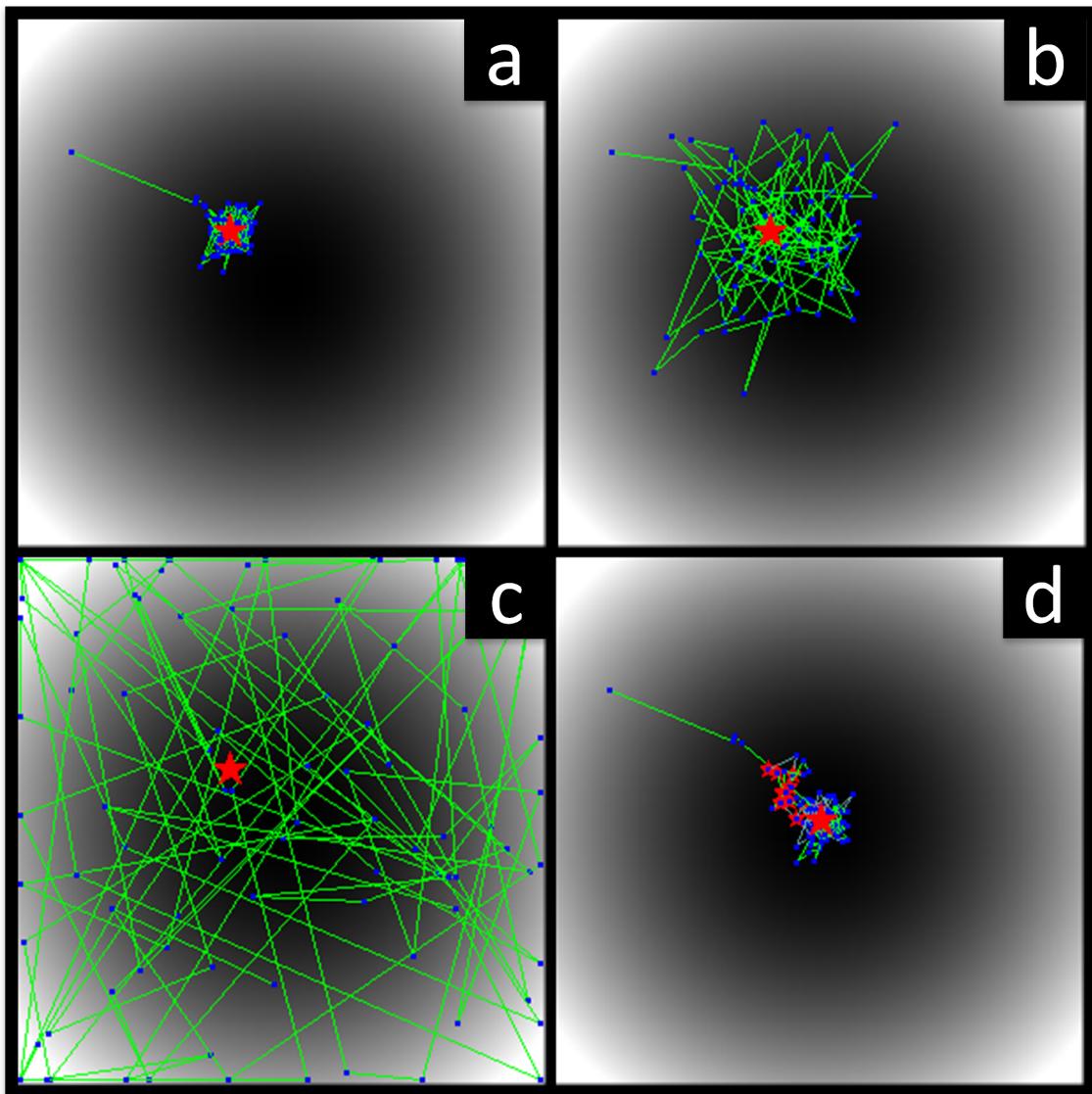


Figura 30 – Movimento Original

os países deste dado império a se moverem para a melhor posição dentro do espaço de busca, de forma que ao fim, ou até mesmo antes do processo evolutivo terminar, se tenha a melhor solução em mãos. Então nota-se uma grande importância na adição de ruído, porém, da forma como é apresentado e calculado, quando um problema apresentar limites de tamanho diferentes, o ruído pode acabar prejudicando a otimização, a velocidade de convergência ou precisão do resultado na busca pela melhor solução.

A partir desta análise, foi possível criar novas formas de movimentação e verificar qual destas seria o melhor modo de mitigar os problemas trazidos com a adição de ruídos na movimentação. Assim foram concebidos os modelos:

- Movimento Refinado,
- Visão imperial distorcida e

- Combinação de ambos.

3.2.1 Movimento refinado

O movimento refinado foi concebido com o intuito de solucionar o problema do movimento Original, o qual tem a geração de ruído fixa, e não proporcional aos limites do problema caso e as dimensões tenham limites diferentes para cada atributo. Então, a proposta é manter o ruído, porém este deve ser proporcional aos limites de cada dimensão. O fluxograma apresentado na Figura 31 Original ICA demonstra a ordem em que ocorrem o cálculo dos valores que irão compor os novos atributos de uma dada colônia para o movimento original descrito anteriormente, sendo, o cálculo tamanho do passo x de cada atributo, calculado com o mesmo valor aleatório $CoeffAssimilação$ para todos os atributos, e quando é multiplicado pela distância, se torna proporcional aquele atributo. E antes de atribuir o novo valor a cada atributo, adiciona-se o valor de ruído aleatório. Assim define-se o valor final de cada atributo como apresentado pela equação 2.9

O método refinado de movimentação então, altera apenas o valor θ de ‘Ruído’, que originalmente é calculado como um valor aleatório distribuído normalmente entre o intervalo $[-\gamma, \gamma]$, que agora passa a ter uma estrutura dependente dos valores que definem os limites mínimo e máximo para uma dada dimensão e consequentemente do tamanho do espaço de busca, e de um valor de intensidade de ruído, que é um valor de controle, usado para definir o quanto de ruído, proporcionalmente ao espaço de busca, se deseja ter. Assim, este método adiciona uma nova propriedade de controle ao ICA, e troca a equação 2.8 que se resume em:

$$\theta = URand(-\gamma, \gamma);$$

pela equação 3.3:

$$\theta = TriangularRand(MinBounds[i], MaxBounds[i], dist) \cdot \left(\frac{dist}{TamanhoEspaçoDeBusca[i]} + p \right); \quad (3.3)$$

Sendo a função $TriangularRand$ uma função que retorna um valor aleatório distribuído triangularmente entre $MinBounds[i]$ e $MaxBounds[i]$, e com moda sendo o valor $dist$ entre o país imperialista e a colônia (observe na figura 30 que não é apenas usado o valor do módulo da distância, mas sim o vetor distância unidimensional, onde o sinal representará o sentido do vetor). O valor p é o que define a proporção do ruído, então se p for definido com o valor de 0.4, o ruído gerado ficará em torno de 40% do tamanho do espaço de busca de cada dimensão para quando a distância entre o império e a colônia for

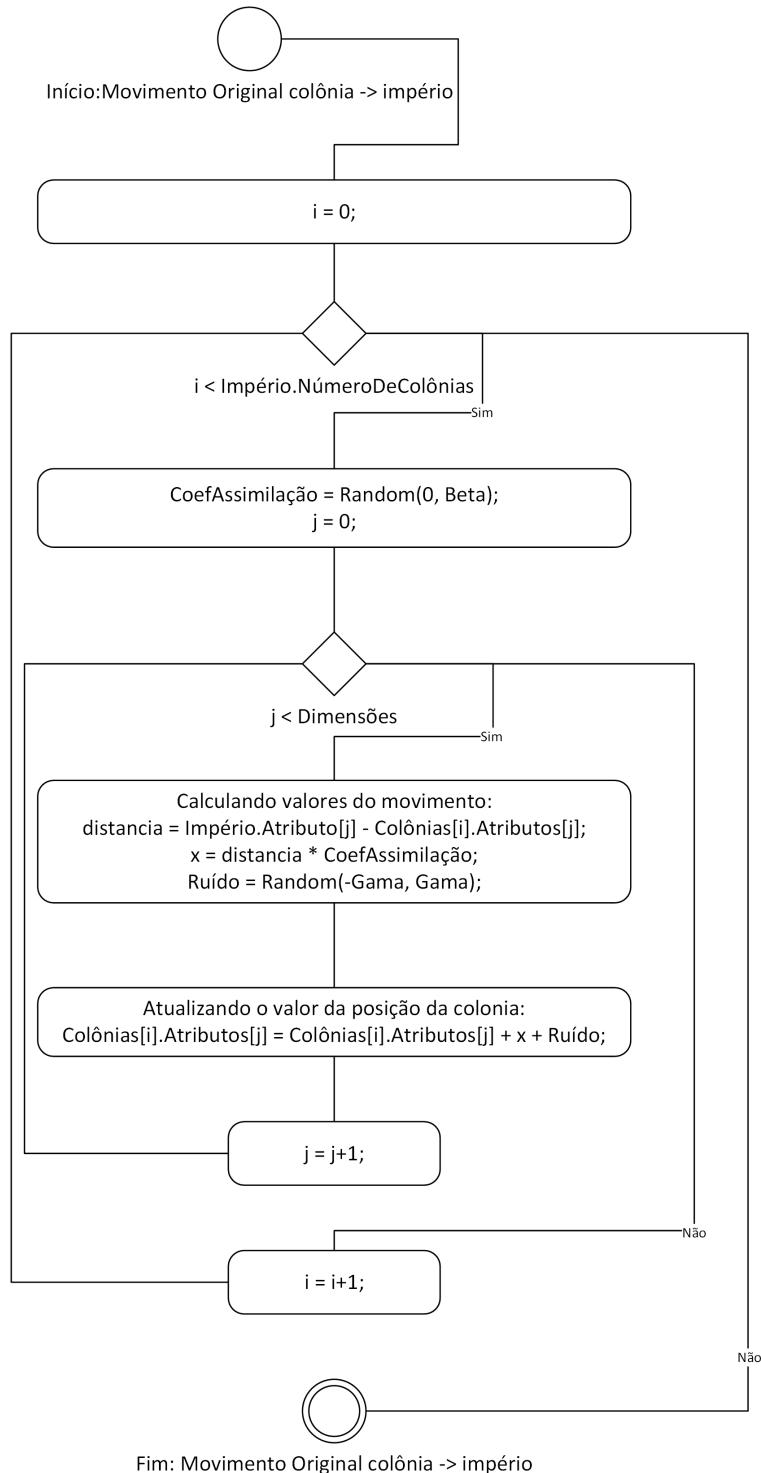


Figura 31 – Fluxograma Movimento Original ICA

próxima ou igual a zero. O fator $\left(\frac{dist}{TamanhoEspaçoDeBusca[i]}\right)$ serve para potencializar o valor de p quando a distância entre a colônia e o império for grande, porém tende a zero quando o império e a colônia se aproximam, o que anularia o ruído, porém soma-se p justamente para manter o ruído na proporção desejada.

A função *TriangularRand* tem um comportamento muito útil para esta alteração no modo de movimento justamente por gerar os valores, primeiro dentro do intervalo desejado e segundo por gerar tais valores aleatórios tendendo para um valor de moda definido, assim, pode-se gerar valores aleatórios dentro dos limites da dimensão e com a moda sendo a distância entre a colônia e o país imperialista, o que implica que a maior parte do ruído é gerado com mais chances de estar mais próximo ao império, aumentando a velocidade de convergência ao mesmo tempo que mantém uma pequena chance de ruídos mais intensos. A Figura32 abaixo representa um histograma de três sequências de 1000 valores gerados por esta função no intervalo $[0, 1]$ e de moda 0, 0.5 e 1 respectivamente. Observa-se então, que existirá uma chance muito maior de que os valores aleatórios gerados pelo ruído estejam mais próximos da distância percorrida, mantendo um controle sobre o ruído gerado e no caso, quando mais a distância diminui menor também será a chance de existirem ruídos mais intensos.

Apenas com o uso da função que gera valores aleatórios de distribuição triangular, já adiciona uma forma de ruído proporcional ao espaço de busca, porém ela ainda gera valores que podem levar a colônia para qualquer posição do espaço de busca aleatoriamente, quando a colônia estiver muito distante do império, fazendo com que a otimização fique muito mais aleatória do que direcionada. Então definiu-se p como sendo um valor definido dentro do intervalo $[0, 1]$ o qual soma-se com o valor proporcional da distância em relação ao espaço de busca $\left(\frac{dist}{TamanhoEspaçoDeBusca[i]}\right)$, que então multiplica o valor gerado aleatoriamente com distribuição triangular entre os limites da dimensão e com moda como sendo a distância entre o atributo da colônia e do império nesta mesma dimensão, tornando este ruído limitado proporcionalmente a uma porção do espaço de busca.

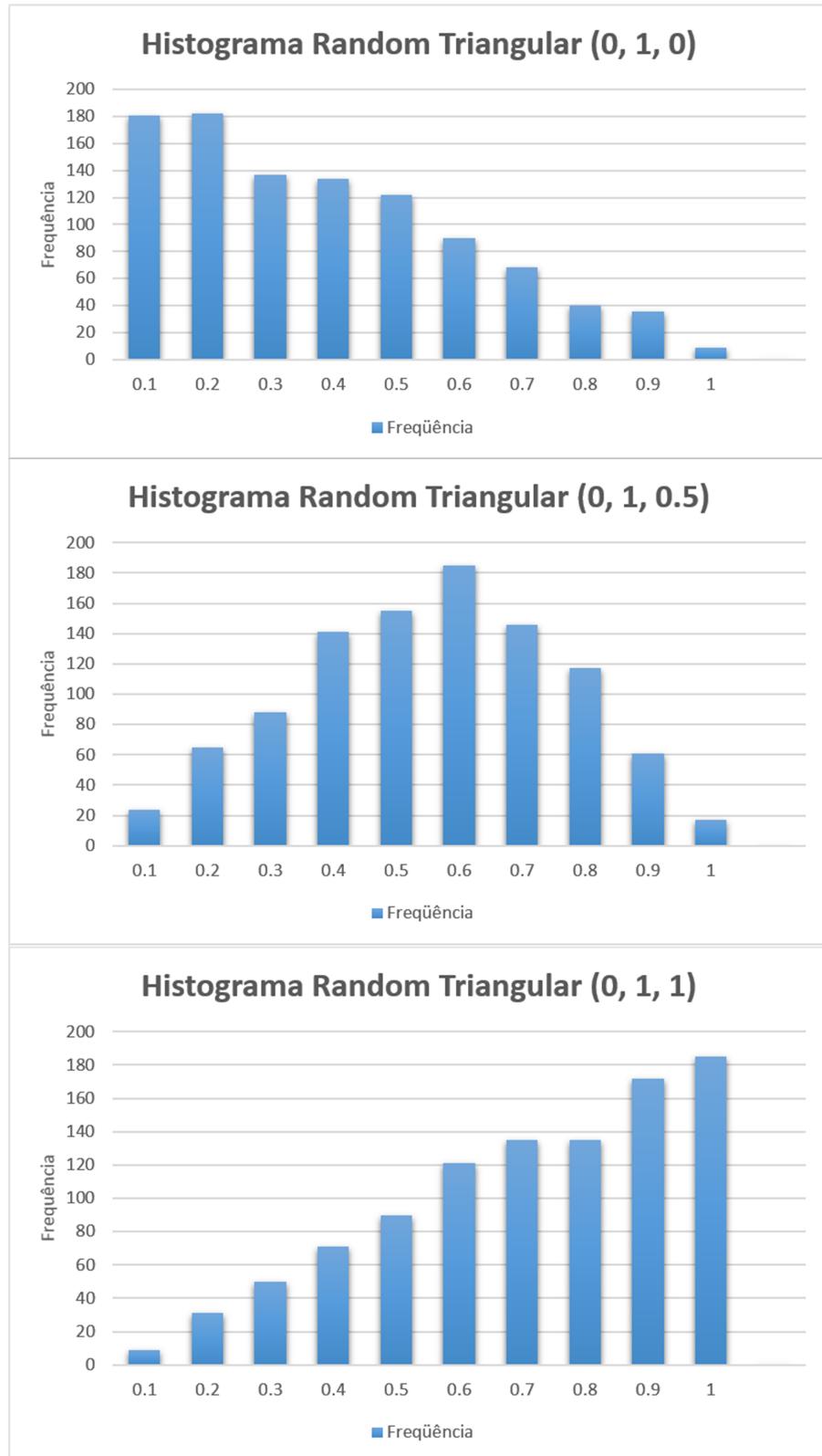


Figura 32 – Random Triangular

A Figura 33, analogamente a Figura 30 apresenta 4 configurações diferentes para a busca da melhor solução, na qual as três primeiras são com a operação de possessão imperial desabilitada e a última tem esta funcionalidade habilitada. Observa-se que as

imagens são muito parecidas, portanto agora existe um controle muito mais preciso sobre a intensidade do ruído a ser gerado, sendo que os valores para p foram de 0.1, 0.4 e 0.8 para os resultados *a*, *b* e *c* respectivamente, e em *d* foi utilizado $p = 0.1$ e a operação de possessão imperial estava habilitada. Em comparação com o modelo original, ao utilizar o método refinado, ocorrem aproximadamente 10% mais possessões imperiais, ou seja, aumentaram-se as chances de uma colônia se mover para uma posição melhor que a de seu país imperialista em 10% durante esta etapa de movimentação.

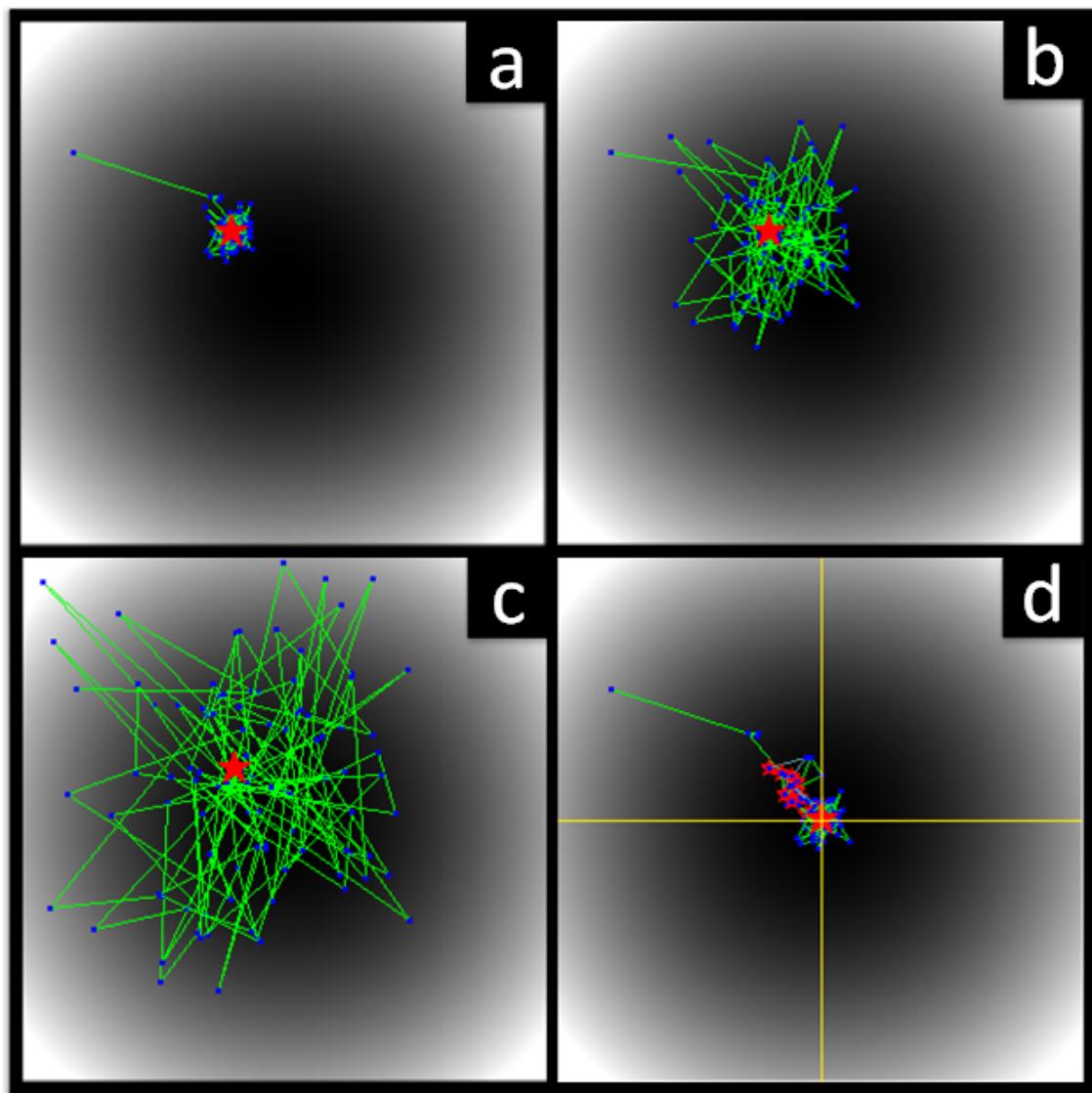


Figura 33 – Testes com movimento refinado

3.2.2 Visão imperial distorcida

A movimentação de uma colônia em direção ao seu país imperialista nada mais é que a tendência da colônia se parecer mais com seu império ao longo do passar de décadas. Para que a colônia se molde de forma a ter suas características mais parecidas com a de seu

império, ela primeiro calcula o quanto distante suas características estão do seu império, e então se movimenta em direção ao seu imperialista de acordo com um valor de assimilação.

Quando uma colônia observa as características de seu império, para então calcular a distância entre cada característica, existe a possibilidade de que esta colônia não localize muito bem a posição ou valor exato da característica do seu império, de modo que tal distorção seja influenciada pelas posições das demais colônias do império. Assim, quando a colônia calcula o quanto distante suas características estão de seu império, ela também observa quais são as características mais distantes ao imperialista, dentre todas as colônias, e adiciona um valor aleatório, distribuído normalmente, na característica do império, para então calcular uma distância com uma visão distorcida destas características do império.

Esta distorção é uma adição de ruído proporcional não ao espaço de busca em geral, mas sim em relação ao posicionamento de todas as colônias do império, isto é, em relação a área ou região do espaço de busca ocupada pelo império. Este tipo de ruído é uma otimização que faz com que a solução seja explorada localmente tal que a melhor solução seja encontrada mais devagar, porém explora localmente de forma aleatória o caminho a ser feito pela colônia, até que esta chegue ao país imperialista. Quanto mais próximas o grupo de colônias estiver de seu império, menor será a região ocupada pelo império e menor também será o ruído gerado. O ruído se torna nulo quando as colônias compartilharem as mesmas posições com seu império. Para se conhecer esta região do império, basta obter as distâncias máximas entre as colônias e o império, isto é, obtém-se os limites do império através das características mais distantes das colônias. Estes valores são armazenados localmente em um vetor *maxDists*, que é gerado segundo o Algoritmo 4

Algoritmo 4: Cálculo das distâncias máximas para Visão Imperial Distorcida.

Data:

Império - o império em questão.

Colônias - o vetor de colônias.

Dimensions - o valor que representa a quantidade de dimensões do problema.

Result:

maxDist - o vetor de distâncias máximas de cada atributo entre o império e todas as suas colônias.

```

1 maxDist = new Double[Dimensions];
2 for i ← 0 to Dimensions do
3     for j ← 0 to Império.NumColônias do
4         localDist = ABS(Império.Attributes[i] - Colônias[j].Attributes[i]);
5         maxDists[i] = MAX(localDist, MaxDist[i]);
6     end
7 end

```

Assim utiliza-se o vetor *maxDists* multiplicado de um valor aleatório distribuído normalmente e soma-o na posição do império, e então subtrai-se o valor da posição da

colônia, para se obter o valor de distância distorcido, como mostra a equação 3.4:

$$dist = (Império.Attributes[i] + GausRand() \cdot maxDists[i]) - Colônia.Attributes[i] \quad (3.4)$$

Após o cálculo da distância distorcida, o algoritmo da movimentação prossegue normalmente no seu fluxo como algo semelhante ao fluxo apresentado pela figura 30, que será então multiplicada pelo coeficiente de assimilação e atribuída ao valor do atributo (ou característica), com a exceção de que agora não se calcula e nem se adiciona o ruído provido pelo valor aleatório baseado em γ .

Este método de adição de ruído é proporcional a distância entre a colônia e império, isto quer dizer que apenas quando o módulo da distância entre ambos for maior que zero é que existirá ruído. Assim quando a colônia se posicionar junto de seu império, não existirá alteração nenhuma na posição da colônia durante a etapa de movimento provinda do ruído. Ao comparar apenas uma colônia com um império, pode-se obter resultados que não sejam ótimos, porém esta alteração funciona muito bem com impérios que possuem várias colônias, uma vez que com muitas colônias tem-se uma exploração maior da região e a tendência será a de tornar o país menos custoso como sendo o centro do império utilizando a operação de possessão imperial.

A Figura 34 mostra em *a* o caso de um país imperialista e uma colônia, com a operação de possessão imperial desativada. Observa-se que quanto mais próxima a colônia está do império, menos desvio ocorre em seu movimento. Em *b* tem-se o mesmo caso de *a* porém com a competição imperialista habilitada. Neste caso, existem algumas trocas entre império e colônia, porém a conversão para a melhor posição não ocorre. O item *c* representa uma configuração diferente, com 1 império e 9 colônias, totalizando 10 países, assim percebe-se como este tipo de movimento é sempre direcionado para o império, e ainda possui uma variação no movimento que decai junto a medida que a colônia chega perto do império. Por fim *d* é uma combinação de imagens que representam o cenário do item *c* durante as décadas de 1 a 5.

Comparando então ambas as alterações, com o método refinado de movimentação é possível controlar o ruído proporcionalmente ao espaço de busca, mesmo que suas dimensões estejam limitadas em valores e escalas completamente diferentes, possibilitando que um país tenha seu vetor de atributos composto por muitos mais elementos sem que o ruído seja um fator que atrapalhe a evolução da solução. Um ponto interessante para se analisar é a quantidade de ruído em relação a precisão da busca pela solução, de modo que tem-se como objetivo principal da movimentação, apenas a transição das características de um indivíduo até as características seu império, pois o algoritmo possui outros recursos, como o aumento do número de países e impérios, operação de revolução imperialista, e

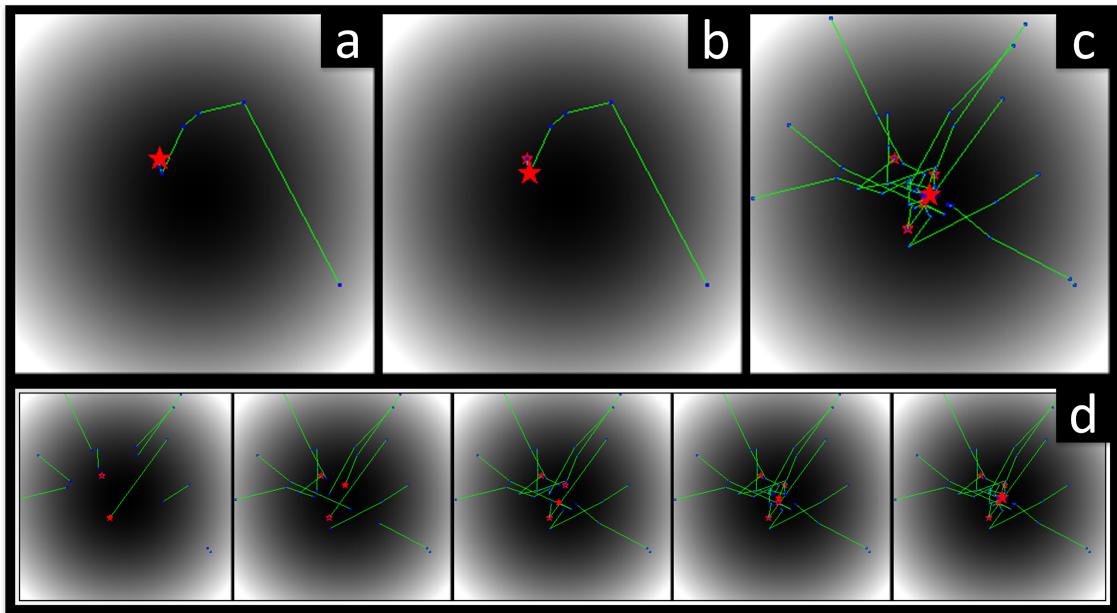


Figura 34 – Testes Visão Imperial Distorcida

até mesmo a competição imperialista em si, fazem com que a adição de ruído pareça redundante. Porém, este não é o caso, o ruído quando usado em uma porcentagem baixa e durante a movimentação, melhora a busca pela melhor solução localmente, fazendo com que o império se move para o vale (ou pico) local mais rapidamente (a movimentação de um império ocorre quando há troca a troca entre império e colônia, através da operação de possessão imperial). E por fim, como mencionado anteriormente, os demais métodos tem o objetivo de evitar que a solução fique estagnada em mínimos (ou máximos) locais, uma vez que possuem apenas o fator aleatório e não são direcionadas, como a movimentação.

Já o método de movimentação por visão imperial distorcida tem como intuito apenas inserir ruído que seja proporcional às colônias mais distantes do império, trazendo a ideia de que quanto mais distante uma colônia está de seu império mais míope ela é para definir as características de seu império. Assim a exploração do espaço de busca não formará caminhos em linha reta, mas sim caminhos que sejam aleatórios até que a colônia de fato chegue ao império. Esta forma aleatória de se mover pode aumentar o tempo que uma colônia leva para chegar até seu imperialista e lá ficar estagnada, mas por outro lado aumenta a chance do império sair de mínimos (máximos) locais se deslocando pelas décadas através da operação de possessão imperial.

Ambos os métodos ainda possuem um mecanismo de ‘freio’, geralmente relativo ao número máximo de décadas, usado para decrementar gradualmente a quantidade de ruído gerada durante as décadas até que esta quantidade de ruído seja anulada ou se mantenha fixa em um valor. Assim como o decrescimento da operação de revolução colonial decresce, a quantidade de ruído também decresce, porém existe a possibilidade de manter este ruído

até o fim (até que uma condição de parada seja atingida), ou também existe a possibilidade de controlar o decrescimento para que os ruídos sejam decrementados até se anularem durante o processo evolutivo. Quando os ruídos tendem a se anular, a forma de movimento tende ao método de movimento linear, que por sua vez é mais rápido e leva à conversão ideal (onde os países estejam na mesma posição de seu imperialista). Desta forma, pode-se ter uma quantidade grande de ruído nas primeiras décadas, de modo a explorar melhor o espaço de busca, evitando mínimos locais, e gradualmente diminuir o ruído para ajustar a posição do império para a melhor possível e por fim, anular os ruídos para que o processo evolutivo possa convergir para o cenário ideal.

A combinação de ambos os métodos mantém a velocidade de deslocamento do império durante as décadas constante, porém mantém o ruído gerado pela visão imperial distorcida sempre ativo nas colônias que sofreram deslocamento por ruído refinado na década passada, isto é, o ruído gerado pelo método refinado faz com que algumas colônias sempre estejam fora do centro do império e limitados em uma região proporcional ao espaço de busca, então, na próxima década, será aplicado a elas, o ruído gerado pela visão imperial distorcida (pois ele é usado quando se calcula a distância, antes de se aplicar o ruído gerado pelo método de movimento refinado), assim este ruído gerado pela visão imperial distorcida será constante e limitado no espaço definido pelo método de movimento refinado, e como ele é proporcional distância entre colônia e império, também passa a ser proporcional ao limite definido no ruído gerado no movimento refinado. A figura 35 mostra a evolução durante as décadas 1, 4 e 8 respectivamente nos itens *a*, *b* e *c*.

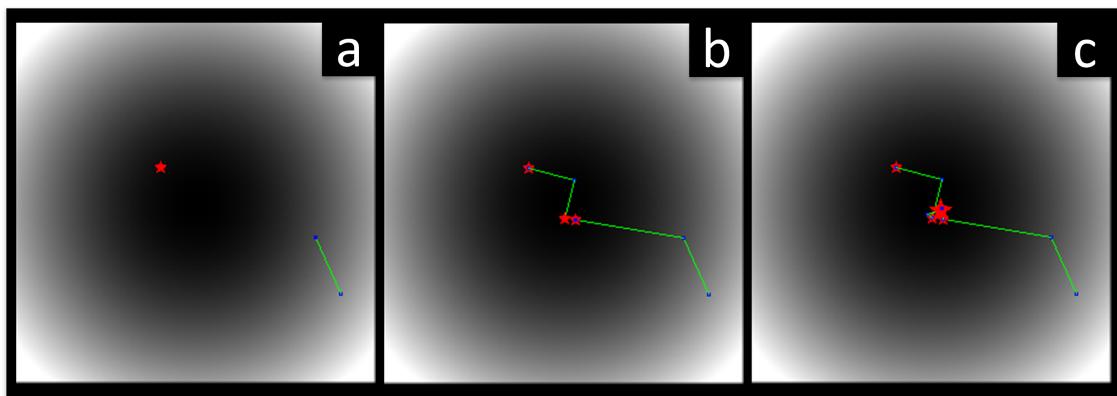


Figura 35 – Movimento combinado

Concluindo esta seção, tem-se agora três (Refinado, Visão imperial distorcida e Combinado) métodos capazes de mover os países, inserindo ruídos proporcionais ao espaço de busca de forma controlada e que ainda é capaz de atingir o cenário ideal de convergência, diferentemente do modelo proposto por ATASHPAZ-GARGARI; LUCAS em (ATASHPAZ-GARGARI; LUCAS, 2007) e ROCHE et al. em (ROCHE et al., 2011), ao adicionar o ruído fixo, o qual levaria a impossibilidade de convergência para o cenário ideal. O método

combinado pode apresentar soluções melhores, mas em contrapartida são necessários mais cálculos e consequentemente mais processamento. Caso a função de aptidão(*Fitness*) seja de rápida avaliação, é ideal usar o modelo refinado, para ruídos independentes da distância entre colônia e império, ou o método de visão distorcida, para ruídos dependentes da distância. já em casos onde o problema apresenta alto custo computacional, o cálculo da forma de movimento não será tão impactante, porém aconselha-se fazer com que os ruídos cessem após uma quantidade de décadas para que a solução possa convergir para o cenário ideal, encontrando o ponto ótimo entre o quanto se deve explorar o espaço de busca de forma mais aleatória e o quanto rápido se deve convergir.

3.3 Paralelização do ICA

Algoritmos evolutivos têm como parte essencial para convergência a função de aptidão, que deve ser bem definida e voltada para o problema. Tal função, deve ser executada durante todas as décadas. No ICA, a função de aptidão proposta foi definida de forma a executar cálculos os quais resultam em um valor de custo para o país, tal que este custo seja utilizado por todos os componentes do ICA, a fim de buscar pela melhor solução. A quantidade de chamadas à função de aptidão é dependente basicamente de dois valores, do número de países np e número de décadas nd .

Sabendo que os países sempre são avaliados, todos de uma só vez, e que em cada década todos os países devem ter seu valor de custo calculado, pode-se dizer que em cada década seus atributos se modificam de acordo com a posição de seus impérios. E ainda, leva-se em conta que inicialmente todos os países são avaliados e têm seus custos atribuídos, antes mesmo da separação entre colônias e impérios. Desta forma, é possível calcular a quantidade total de chamadas da função de aptidão durante uma execução completa do ICA como sendo:

$$npx(nd + 1)$$

Dependendo do tipo de problema, algoritmos evolutivos podem levar diversas épocas, no caso do ICA, décadas, para convergir em uma solução dentro dos critérios de parada (que foram definidas no capítulo x). Em casos cujo problema se expande para o domínio não linear existe a possibilidade de executar tais operações muito mais vezes do que o esperado para se chegar ao resultado.

No problema a ser solucionado por este trabalho, a função de aptidão utilizará muito processamento para o cálculo de custo de cada país. Para que tal solução possa ser executada diversas vezes, em tempo hábil, e também de forma que o problema possa ser testado dentro de diferentes abordagens, é necessário utilizar todos os recursos tecnológicos disponíveis.

Para otimizar a velocidade com que o ICA faça tais cálculos, foi desenvolvida uma adaptação na parte central do ICA, de forma que as chamadas para avaliação das funções de custo dos países sejam executadas paralelamente, já que o cálculo de custo do país são independentes dos demais indivíduos, usando todos os núcleos de processamento possíveis. Assim, a aplicação tem um ganho significativo de tempo e um aumento significativo de processamento, que depende de quantos núcleos estão disponíveis para a execução do ICA

Um fato técnico interessante, é que para se utilizar de forma ótima toda esta alocação de núcleos de processamento, devemos levar em conta que o número de indivíduos deve ser múltiplo do número de núcleos de processamento para que todos sejam utilizados com capacidade máxima para o processamento da função de aptidão. Caso este número seja diferente, também haverá uma melhora, porém um dos núcleos poderá ficar ocioso durante os cálculos, fazendo com que indiretamente se “perca” a capacidade total de processamento em um ou mais núcleos.

A paralelização dos cálculos de custos dos países foi implementada de modo a ser opcional. Por padrão já está ativa, porém ao se configurar os dados iniciais para a solução de um dado problema, fica a critério do desenvolvedor, decidir se utilizará ou não o algoritmo de forma paralelizada.

Algo que pode ser chamado de desvantagem em se paralelizar esta parte do algoritmo, aplica-se apenas no momento em que é necessário executar uma depuração no código que faz o cálculo do custo do país. O que ocorre no caso normal, não paralelo, é que para se depurar uma função, se escolhe uma parte do código, a qual se deseja depurar e então a execução do programa sofrerá uma pausa quando chegar naquele ponto. Quando se utiliza a técnica de paralelização para executar as funções (neste caso as funções são as mesmas, porém apenas a chamada da função é alterada), cada núcleo disponível irá alocar uma tarefa (thread) para executar as funções em paralelo, assim, como foi definido um ponto de depuração dentro da função, toda vez que uma das tarefas atingir o ponto de depuração, ela irá pausar a execução do programa, apresentando os valores a serem depurados para o seu contexto de execução da tarefa, e impedindo que uma depuração por passos possa ser feita sequencialmente apenas em uma tarefa sem que um filtro seja criado em tempo de execução. Tal filtro de depuração para pontos de parada só podem ser adicionados em tempo de execução porque inicialmente não se sabe qual será a identificação da tarefa que irá rodar em paralelo por ela não ter sido criada ainda. É importante fixar que a depuração não é impedida ou resulta em erros, mas tal desvantagem se dá na adição de passos para filtrar e isolar a depuração para apenas uma tarefa.

A melhora no desempenho não é apenas dependente da paralelização do cálculo dos custos dos países, mas também, do que está dentro do algoritmo que faz o cálculo e também da configuração e estado da máquina que executará o processamento. É muito importante que se tome cuidado durante o desenvolvimento para que a função de avaliação,

quando processada em paralelo, não compartilhe recursos entre as tarefas, pois isto pode causar a execução em serial, já que o algoritmo de paralelização ‘trava’ o uso de um recurso apenas para uma tarefa. Outro problema que ocorre com o compartilhamento de recursos é a inconsistência, pois caso haja escrita, uma tarefa pode alterar um recurso que será usado por outra tarefa. Como a função de aptidão do ICA é uma implementação da interface IFitness, estes tópicos devem ser tratados com cuidado durante o desenvolvimento do problema para que a paralelização funcione de forma ótima.

3.4 Formulação do ambiente

Os dados de entrada iniciais utilizados para o desenvolvimento da solução representam e classificam, quando e onde ocorreram instalações na rede de distribuição elétrica de uma dada região. O objetivo é obter a partir destes dados a previsão de expansão espacial urbana, no caso, relativo à densidade de carga elétrica. Assim pode-se aplicar tal previsão no planejamento de cargas, em sistemas inteligentes de energia (smartgrids), gerenciamento de cargas (reposicionamento de transformadores ou subestações para localizações mais apropriadas), dentre outros.

Para o desenvolvimento da metodologia de previsão de densidade espacial urbana é necessária a análise de fatores que caracterizam um ou mais fatores de representação para densidade (estes são fatores que variam no tempo), e de fatores que representam o espaço em que estes fatores de densidade se aplicam. Isto quer dizer que, inicialmente, é necessário o uso de dados concretos baseados diretamente na posição geográfica em que o fator de densidade se encontra e evolui durante o tempo, de modo que a posição geográfica do fator represente, de fato, um ponto no espaço ao qual será prevista a evolução dos fatores de densidade durante um determinado intervalo de tempo.

Entende-se como fatores que representam o espaço em que os fatores de densidade evoluem, como sendo uma composição de fatores que possuem relação direta com o fator de densidade, de forma a representarem este fator de dentro de um plano espacial, que neste caso, são utilizados os pontos de latitude e longitude citados acima para representar a existência deste determinado fator (neste caso, a instalação de ponto da rede elétrica propriamente dita) dentro do espaço em questão. Então tem-se um espaço bidimensional formado por pontos de latitude (representado pelo eixo das ordenadas) e longitude (representado pelo eixo das abcissas), e limitado em latitudes e longitudes, máximas e mínimas, formando uma região retangular que contém tal representação dos fatores de densidade.

Ainda sem se preocupar com os fatores de densidade que se alteram durante o tempo, apenas focando nas posições geográficas que compõem os dados, faz-se o uso da filosofia de quadriculas (WILLIS, 2002), para tornar o espaço homogêneo, pois inicialmente,

este espaço era representado por um conjunto de pontos espaçados desigualmente, de modo que seria muito custoso computacionalmente representar e processar todos os dados de forma a cumprir o objetivo apresentado por este trabalho.

Assim, é possível gerar um mapa de quadrículas homogêneas que representará os dados iniciais na forma de uma matriz, com a altura (número de linhas) sendo o número de quadrículas de norte a sul, e largura (número de colunas) como sendo o número de quadrículas de leste a oeste. Tal matriz pode ser usada para mapear uma região geográfica em uma matriz de dados que possa ser computada sem que seja preciso se preocupar com os fatores de localização geográfica, como por exemplo, converter ou calcular a distância entre 2 pontos geográficos, pois o espaço entre as quadrículas já é normalizado.

Para se fazer previsão de densidade espacial urbana é necessário que se modele um cenário no ambiente computacional, que seja capaz de representar com um certo nível de fidelidade, o que o ambiente real representa. Para criar tal ambiente computacional é preciso primeiro a geração ou obtenção de uma quantidade de dados os quais serão usados na modelagem deste ambiente. Os dados em questão referem-se a densidade de cargas, que é analisado em uma região urbana de forma espacial.

A base de dados utilizada para fazer a previsão de densidade de carga espacial urbana representa uma estimativa das datas de instalação de pontos de rede elétrica durante um período de tempo e de uma determinada região e, portanto são dados que variam no tempo. Os pontos de instalação elétrica foram classificados em residencial, residencial de baixa renda, comercial, industrial, público e rural. Inicialmente imaginou-se que pontes, ruas, rios, altitude, inclinação, etc. poderiam ser úteis como dados de atração, representando pontos que não variam em pequenos intervalos de tempo, e portanto, fixos durante todos os períodos, porém foram removidos para que a análise ficasse mais limpa, pois estavam gerando ruído nos resultados e acumulando muito erro durante a previsão. Então, cada ponto de instalação elétrica possui um município, um grupo, uma data de criação e valores que definem sua localização geográfica, representados por valores numéricos reais de latitude e longitude. Com estes dados separados é possível então criar um banco de dados para armazenar toda informação que é captada, gerada ou obtida, normalizando assim toda entrada de dados para a aplicação apenas nestas 5 características.

Com a tabela possuindo um número grande de dados durante os períodos, é possível separar estes dados em diversos mapas matrizes (mapas de quadrículas) representando a quantidade de carga instalada em cada região e sua evolução durante um período de tempo. A quantidade de mapas é definida pelo intervalo de tempo que se deseja processar. Por exemplo, se inicialmente foram obtidos dados nos períodos entre 2005 até 2015, e o período de tempo desejado é de um ano, geram-se então, 10 mapas matrizes de densidades de carga onde cada mapa matriz possui a densidade de carga para cada período, neste caso, para cada ano.

Antes de gerar os mapas matrizes, ainda é necessário mais um passo, que é a criação de arquivos, a partir do banco de dados, onde cada arquivo representa o período em que se deseja obter os mapas matrizes. Para isto, foram criados diversos filtros no banco de dados para cada período, onde cada filtro seleciona todas as linhas que estejam dentro do intervalo de duas datas, uma inicial e outra final, sendo que destas duas datas, a data inicial é referente sempre a data do período inicial, e a segunda é referente ao período em questão, pois os dados devem representar valores crescentes, e não a diferença entre os períodos. Outra funcionalidade que cada filtro aplica nas linhas, é a adição de um valor de contagem, denominado “Total”, para cada ponto de coordenada geográfica que possa ter sofrido crescimento, sendo este aumento referente a mais instalações de pontos de rede elétrica nesta mesma coordenada, como por exemplo a construção de um prédio, residência ou qualquer outro tipo de adição que possa vir a representar uma forma de crescimento de consumo de carga para aquela coordenada. Assim, cada arquivo deve ser salvo em modo texto, possuindo os nomes seguindo a seguinte forma, $1 < período0 > .csvj, 1 < período1 > .csvj, \dots, 1 < períodoN > .csvj$, e ainda dentro dos arquivos, os dados serão filtrados de modo que transformem a tabela inicial em uma tabela contendo os campos: Longitude, Latitude, Grupo e Total.

Toda a modelagem dos dados de carga citados anteriormente ocorreram fora da aplicação, e foi utilizada a ferramenta *sqlite* para a criação, armazenamento e filtragem dos dados para a utilização destes na aplicação. Já na aplicação, a primeira definição que deve ser feita é sobre a região que se deseja utilizar, sendo esta definida por valores de longitude, leste e oeste, e valores de latitude, norte e sul. Com estes quatro valores formando um setor terrestre quadrangular, onde tal setor representa a região que se deseja processar para obter a previsão espacial de carga. Os dados presentes nos arquivos gerados para cada período que estiverem fora destes intervalos de (norte, sul e leste, oeste) serão ignorados, sendo apenas processados os dados dentro dos intervalos citados.

Os valores presentes nos arquivos de período ainda não estão discretizados, de forma que possam ser processados de forma mais homogênea, então, para utiliza-se a técnica de quadriculamento do setor para discretizar as regiões de forma a fazer com que o mapa seja particionado em tamanhos iguais e, consequentemente cada parte igual do mapa receba um valor de carga de acordo com aquela região. Assim, forma-se um mapa, denominado *MapaMatriz*, para cada período, que representa a densidade de carga espacial em porções homogêneas.

A segunda definição deve ser o tamanho de cada quadrícula, que é definida a partir de apenas um valor numérico, que representa o tamanho dos seus lados em metros. Com estas duas definições iniciais pode-se então criar um mapa de quadrículas que será usado posteriormente como base para a geração dos mapas matrizes.

O mapa de quadrículas utiliza-se destes 5 valores iniciais para ser gerado, e nele

definem-se as quantidades de quadrículas que irão compor o mapa em altura e largura, e consequentemente a quantidade de quadrículas total do mapa, que pode ser calculado multiplicando a altura com a largura. Os valores de altura e largura são calculados em duas etapas, onde a primeira etapa fornece a largura/altura total em metros e a segunda etapa resolve a quantidade de quadrículas para altura e largura, dividindo os valores de altura e largura totais em metros pelo tamanho de cada quadrícula em metros, como pode ser visto nas equações 3.5 e 3.6:

$$\begin{aligned} widthInMeter &= \text{Abs}(\text{DistanceTo}(north, east, north, west)); \\ heightInMeter &= \text{Abs}(\text{DistanceTo}(north, east, south, east)); \end{aligned} \quad (3.5)$$

$$\begin{aligned} mapWidth &= widthInMeter / quadricSizeInMeter; \\ mapHeight &= heightInMeter / quadricSizeInMeter; \end{aligned} \quad (3.6)$$

A chamada função *DistanceTo* calcula a distância em metros entre dois pontos de coordenadas geográficas quaisquer utilizando a fórmula de haversine (SHUMAKER; SINNOTT, 1984), (SNYDER, 1987), onde os dois primeiros valores representam o primeiro ponto e coordenada geográfica e os últimos dois valores representam o segundo ponto de coordenada geográfica. Observa-se que na primeira chamada, mantiveram-se os pontos de latitude fixos em norte, de forma a deslocar-se apenas horizontalmente (de leste para oeste) e na segunda chamada mantiveram-se fixos os pontos de longitude fixos em leste, de forma a deslocar-se apenas verticalmente (de norte para sul), para assim obter o valor transformado de coordenada geográfica para coordenada cartesiana em metros, para largura e altura respectivamente.

Com a quantidade de quadrículas para altura e largura calculados para o mapa de quadrículas, agora é preciso popular cada índice do mapa com uma quadrícula. Cada quadrícula é representada por um ponto de coordenada geográfica, composto por latitude e longitude, podendo este ser convertido em uma distância a partir do ponto inicial ou em um ponto de utm (que é o ponto de coordenada geográfica representado pela *Universal Transversa de Mercator*). Então, para isto, são necessárias mais duas definições, um ponto inicial e uma direção de iteração.

O ponto inicial pode ser definido como qualquer combinação dos valores de latitude e longitude dados inicialmente, então, pode-se ter o ponto inicial como sendo (*Norte, Leste*), (*Norte, Oeste*), (*Sul, Leste*) ou (*Sul, Oeste*), porém deve-se prestar atenção em definir a direção como sendo a oposta ao ponto inicial, uma vez que o processo de inicialização do mapa de quadrículas já possui seus limites definidos e a direção é quem define para qual sentido serão geradas as quadrículas (iterando-se em altura e largura) a partir do ponto inicial. Então, por exemplo, se o ponto inicial for (*Norte, Leste*), a direção de crescimento

deve ser representada pelo texto `isoj`, assim, tem-se uma iteração que gera quadrículas dentro dos limites do mapa de quadrículas, que vai de Norte para Sul e de Leste para Oeste. Caso a direção de crescimento para o mesmo ponto do exemplo seja outra, digamos, `islj`, o mapa terá como ponto inicial os pontos definidos como Norte e Leste, crescerá de norte a sul, porém de Leste a leste, tendo um deslocamento para fora dos limites do mapa de quadrículas esperado, o que pode implicar em um preenchimento incorreto do mapa matriz, que define as cargas, nos próximos passos da aplicação.

Finalizando a criação do mapa de quadrículas, ainda são definidos 5 valores e uma função, sendo estes, o Ponto Final, que representa o ponto mais distante ao ponto inicial, tendo sua distância como a diagonal principal do mapa de quadrículas, e os valores para Latitude Máxima, Longitude Máxima, Latitude Mínima, e Longitude Mínima, definidos a partir dos pontos inicial e final. Estes últimos 4 valores, são usados pela função a ser definida, que, por sua vez, calcula os índices (x, y) do mapa de quadrículas, a partir de um ponto de coordenada geográfica (Latitude, Longitude), isto é, dado um ponto na forma (Latitude, Longitude), a função deve retornar em qual quadrícula do mapa de quadrículas (mapa matriz) este ponto se encontra. Esta é uma função usada para preencher os mapas matrizes a partir dos dados nos arquivos de períodos.

Com o mapa de quadrículas criado, inicializado e seus valores propriamente definidos dentro dos limites estipulados (Norte, Sul, Leste, Oeste), sua estrutura pode ser ilustrada como apresentada na figura 36:

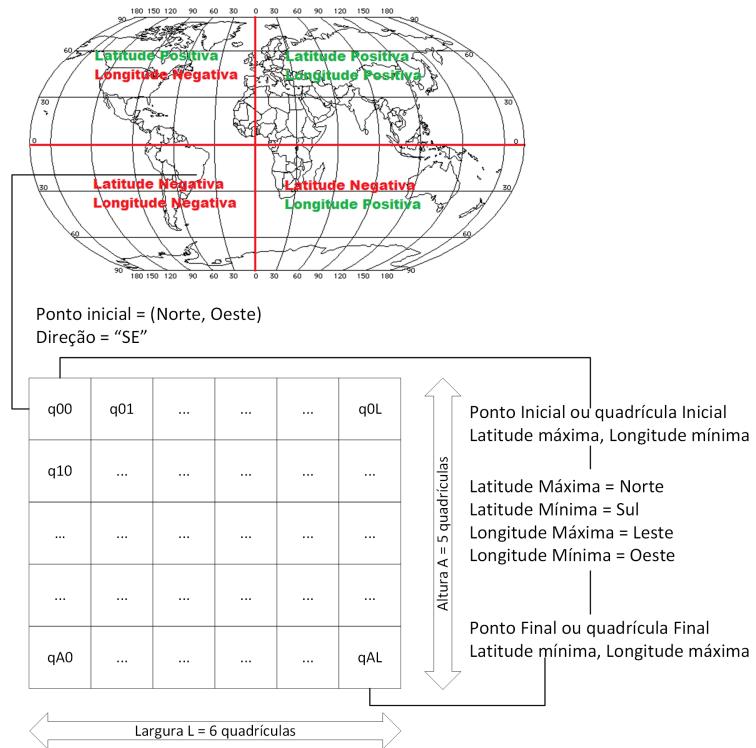


Figura 36 – Ilustração da montagem do mapa de quadrículas.

Observe a região escolhida está no hemisfério sul a esquerda do meridiano de *Greenwich*, o que indica que os valores para latitude e longitude (em graus) sempre serão negativos, e assim, de acordo com a posição inicial e a direção escolhida, os valores máximos para latitude e longitude se resolvem como apresentado.

Passando agora para os mapas matrizes, que nada mais são que uma lista de matrizes com o mesmo tamanho, em altura e largura, do mapa de quadrículas, e tal que a quantidade de mapas matrizes é referente a quantidade de períodos. Cada índice de um dado mapa matriz é diretamente mapeado para uma quadrícula do mapa de quadrículas, podendo assim, este mapa matriz representar um mapa de densidades genérico, sem relação direta ao mapa de quadrículas, porém pode-se usar o mapa de quadrículas para localizar e dar sentido ao mapa matriz sempre que necessário.

Tal encapsulamento dos valores de densidade em uma lista de mapas matrizes, de tamanhos iguais, e de forma que seus índices representem valores de densidade, os quais são crescentes de acordo com o período de cada mapa matriz, faz com que a aplicação seja capaz de processar qualquer mapa de densidades sem se 'preocupar' com a localização geográfica real que este mapa matriz representa. O mapa de quadrículas então, tem duas funcionalidades básicas, que é a de inicializar a região a partir dos dados iniciais, e abstrair as informações geográficas para os mapas matrizes, os quais representam as densidades de carga em cada período.

A partir do mapa de quadrículas gerado e o conceito dos mapas matrizes, pode-se então, escolher qualquer fator para inserir em seus índices, como por exemplo, pode-se criar um mapa de quadrículas para uma determinada região tal que o fator escolhido sejam valores de altitude ou de inclinação. Porém para que uma previsão seja feita é necessário utilizar dados de densidade que variam de acordo com o tempo, como é o caso dos dados citados no início desta seção, os quais podem ser mapeados em vários mapas matrizes cada um representando um período de tempo diferente e diferenciados pelo mesmo intervalo de tempo, como por exemplo, poderia ser gerado um mapa de quadrículas do total de instalações elétricas até 2010, um até 2011 e assim por diante.

Sabe-se que os fatores de densidade devem estar diretamente relacionados com os valores de posição geográfica, por isto podem ser inseridos em qualquer mapa de matriz, e que também, devem variar no tempo, para que possam ser gerados diversos mapas de matrizes, com cada fator representando assim, um período de tempo bem definido e espaçados no tempo uniformemente, tal que possam ser geradas previsões futuras à estes períodos e de mesmo intervalo de tempo.

Neste trabalho, ainda não foi modelada a previsão de múltiplos fatores, mas já foi elaborada uma estrutura que converge vários fatores de carga para apenas um fator de densidade de carga, denominado "multiplicadores de carga".

Para a criação da lista de mapas matrizes, é então, utilizado o mapa de quadrículas já processado, os arquivos de períodos e uma estrutura que representa a densidade de carga de cada tipo de região, denominada “multiplicadores de carga”. Para este trabalho foram considerada as regiões do tipo residenciais ($m = 1$), comerciais ($m = 3$), industriais ($m = 10$), residencial de baixa renda ($m = 0.5$), rural ($m = 0.5$), iluminação pública ($m = 1$), sendo os valores de m o valor que irá multiplicar aquele tipo de carga ao montar o mapa de quadrículas. Assim, os mapas matrizes são preenchidos um a um, de modo que para cada linha do arquivo de período, busca-se qual o índice (x, y) no mapa de quadrículas, através dos valores de latitude e longitude, então, com o índice (x, y) seleciona-se o índice (x, y) do mapa matriz e atribui a este índice, o multiplicador de carga referente ao Grupo da linha multiplicado pelo Total da linha, como mostra a expressão 3.7:

$$\begin{aligned}
 (x, y) &= \text{MapaQuadrículas.BuscaÍndice}(linha.Lat, linha.Long); \\
 \text{MapaMatriz}(x, y) &= \text{MapaMatriz}(x, y) + \\
 &\quad \text{MultiplicadorCarga}[linha.Grupo]* \\
 &\quad \text{linha.Total}; \\
 \end{aligned} \tag{3.7}$$

Este processo ocorre para cada linha de cada arquivo de períodos, e no fim da leitura de cada arquivo tem-se um mapa matriz que representa a região no mesmo período do arquivo. E por fim, todos os mapas matrizes gerados são adicionados em uma lista sequencial referente aos períodos.

Com a lista de mapas matrizes calculada, pode-se então definir um vetor que represente o crescimento absoluto entre cada período, fazendo a diferença entre o mapa matriz do período t com o mapa matriz do período $t + 1$, tendo assim, os valor de crescimento de t para $t + 1$, de $t + 1$ para $t + 2$, de $t + p - 1$ para $t + p$, sendo p o número total de períodos selecionados. Assim o vetor é representado pela expressão:

$$\begin{aligned}
 \text{CrescimentoAbs} &= \\
 &[\\
 &\quad \text{mapaMatriz}[t] - \text{mapaMatriz}[t + 1], \\
 &\quad \text{mapaMatriz}[t + 1] - \text{mapaMatriz}[t + 2], \\
 &\quad \dots, \\
 &\quad \text{mapaMatriz}[t + p - 1] - \text{mapaMatriz}[t + p] \\
 &\quad], \\
 \end{aligned} \tag{3.8}$$

$$\begin{aligned}
 0 &< t < p - 1, \\
 p &= \text{número total de períodos}.
 \end{aligned}$$

Após o cálculo dos crescimentos absolutos entre os anos, particiona-se os mapas matrizes em partes menores, resultando assim em uma lista de mapas matrizes particionados. Tal particionamento se dá porque a operação que fará a otimização das estruturas para a previsão funciona melhor para pequenas porções, uma vez que quando se usa mapas muito grandes, a tendência é se aproximar da média de crescimento geral para aquela região. A figura 37 ilustra a estrutura de partições de mapas para partições de tamanho 2×2 e durante quatro períodos. Observe ainda na figura, que o número de partições não ignora partições incompletas, usando-as assim, como regiões válidas para o processamento.

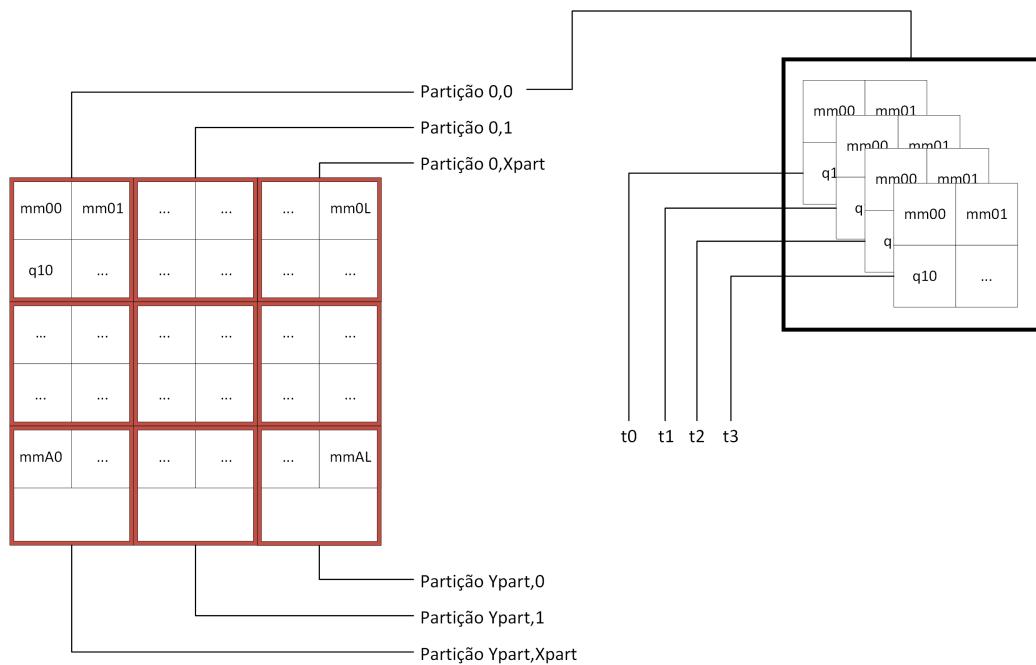


Figura 37 – Ilustração do mapa de quadrículas particionado.

Como mencionado nos parágrafos anteriores, este particionamento é uma forma de fazer uma previsão sobre das pequenas regiões particionadas do mapa original, pois a operação de avaliação utilizada, é uma heurística que leva em consideração a operação de convolução sobre imagens, e neste caso, cada quadrícula está sendo representada por um pixel da imagem (mapa matriz) a ser operada pela convolução para gerar a previsão. Tal técnica será discutida mais adiante, mas já se adianta que a convolução tem a capacidade de transformar uma matriz em outra, passando um filtro ou máscara de convolução sobre da matriz em questão, gerando uma matriz resultado. Caso se tente buscar um filtro de convolução para a matriz completa, englobam-se muitas características das áreas ou regiões mais influentes, e as regiões menos influentes serão ignorados. Isto resulta apenas em um mapa que é a média da distribuição de carga geral por todo o mapa, e com um acúmulo de erro muito grande sobre os períodos, e que neste caso não nos interessa. Por este motivo é que foi feito o particionamento do mapa de modo que cada pequena região tenha destacada suas características principais a fim de fazer uma previsão mais precisa

para todo mapa a partir da previsão das pequenas regiões. Este particionamento pode ser considerado um quadriculamento do mapa de quadrículas, o motivo pelo qual não se usa uma quadrícula maior é que ela é usada segundo WILLIS com seu tamanho original de aproximadamente $\frac{1}{4}$ milha, e para este trabalho, foi usado um tamanho menor, uma vez que se deseja ter mais resolução nos dados para a adição deste a mais, que particiona o mapa de quadrículas em vários mapas menores. Assim cada partição é um conjunto de quadrículas que representará uma imagem a ser processada pelo algoritmo, o qual buscará a melhor previsão desta pequena região.

Então, ao final do processamento não se terá apenas uma solução para previsão total do mapa, mas sim várias soluções boas de previsão, para várias pequenas regiões, as quais irão compor o mapa final de previsão para a quantidade de períodos escolhida.

Para se criar as partições inicialmente define-se apenas dois valores, que representará quantas partições haverão no eixo das ordenadas X_{part} e no eixo das abscissas Y_{part} . Assim, calcula-se o total de partições como sendo $X_{part} \cdot Y_{part}$. No caso da figura 37, X_{part} e Y_{part} são iguais a 3. Assim como para o mapa final, cada partição será processada independente uma da outra, de modo que exista no final, uma previsão para cada partição dos mapas matrizes, as quais são reconstituídas em um único mapa matriz após todo o processamento para a apresentação do resultado final. Observe que como X_{part} e Y_{part} são iguais a 3, cada partição tem o tamanho de 2, pois a regra de cálculo para o tamanho de cada partição a partir de X_{part} , Y_{part} segue a expressão 3.9:

$$\begin{aligned} PartWidth &= FLOOR(mapWidth/X_{part}); \\ PartHeight &= FLOOR(mapHeight/Y_{part}); \end{aligned} \quad (3.9)$$

Calcula-se então os crescimentos absolutos para cada partição a fim de que este valor seja usado também pela função de avaliação que será utilizada no processo evolutivo, e armazena os valores em uma lista de vetores de crescimentos absolutos para cada partição.

Conclui-se então, a modelagem do ambiente, demonstrando como os dados iniciais foram transformados até antes de serem processados para geração da previsão através do processo evolutivo, que neste caso será feita pelo algoritmo imperialista competitivo, com uma função de aptidão ou *fitness* modelada utilizando os dados de uma lista de mapas matrizes particionado e os dados de crescimento absoluto nesta partição que serão descritos a seguir. Neste ponto os dados que aplicação possui carregados em memória são:

1. Mapa de quadrículas, carregado a partir dos arquivos gerados por seleção no banco de dados para cada período.
2. Vetor Multiplicador de Carga, definido de forma constante na aplicação. Para trabalhos futuros pode ser definido um multiplicador de carga para cada região

particionada e seus valores podem ser definidos de acordo com algum algoritmo inteligente que processa os dados

3. Norte, Sul, Leste e Oeste, que são os limites em coordenadas que define a região da previsão.
4. Tamanho da quadrícula, Definido como descrito por WILLIS, com o valor de aproximadamente $\frac{1}{4}$ milha.
5. Vetor de períodos, não foi mencionado no texto mas quando se carrega os dados é criado um vetor contendo o valor de cada período apenas para impressão dos resultados e definição dos períodos de treino e teste descritos posteriormente.
6. Vetor de períodos, não foi mencionado no texto mas quando se carrega os dados é criado um vetor contendo o valor de cada período apenas para impressão dos resultados e definição dos períodos de treino e teste descritos posteriormente.
7. Uma lista de mapas matrizes, onde a quantidade de mapas matrizes presentes na lista é a quantidade de arquivos de períodos lidos no início da aplicação, e cada mapa matriz representa a densidade de carga em uma região durante o período que gerou este mapa matriz.
8. X_{part} e Y_{part} , que definem em quantas partições os mapas matrizes serão quebrados para gerar uma previsão para cada conjunto de mapas matrizes de cada uma destas partições.
9. Uma lista de mapas matrizes particionados, onde a quantidade de mapas matrizes particionados é igual a $X_{part} \cdot Y_{part}$, e, para cada mapa matriz particionado será gerada uma previsão.

3.5 Inicialização do ICA

Para que o ICA seja inicializado apropriadamente, de forma a gerar soluções que possam ser usadas para prever a densidade espacial de carga períodos a frente, é necessário que se modele o problema como foi feito nos problemas ??G1 e G2??, descritos no neste capítulo pelo item 3.1.2. Assim, é necessário que sejam definidos, a função de avaliação, o que cada dimensão representará durante o processo evolutivo, e consequentemente a quantidade de dimensões. Deve-se também antes de iniciar o processo evolutivo definir quais são os limites de cada dimensão definida.

A modelagem do problema da previsão da densidade espacial de carga para otimização pelo ICA, se resume em uma implementação da interface *IFitness*, de forma que tal implementação seja capaz de evoluir os valores, ou atributos, dos indivíduos, a fim

de que a melhor solução possa ser aplicada na base de dados para se gerar previsões. Observa-se que para obter uma previsão sobre os dados é necessário utilizar o ICA para evoluir os indivíduos até um período antes do que se deseja gerar a previsão (sobre dados históricos, em forma dos mapas matrizes particionados), e então utilizar os indivíduos evoluídos junto deste último período, passando-os pela função de previsão usada durante a avaliação, a fim de gerar uma previsão para o próximo período. Para fazer a previsão de mais de um período à frente, basta utilizar o resultado obtido da previsão do período anterior e executar novamente a função de previsão, que irá gerar desta vez uma previsão sobre um período já previsto, acumulando assim o erro de previsão inicial (base inicial \Rightarrow previsão1) e o erro de previsão atual (previsão1 \Rightarrow previsão2). Quanto mais períodos forem previstos mais erro acumulado será gerado nas previsões futuras.

A técnica utilizada para fazer com que os elementos do ICA possam evoluir, e fornecer as melhores soluções, tal que quando aplicadas sobre os dados para períodos futuros, devem fornecer previsões espaciais da região selecionada. Assim entende-se que a função avaliação deve ser dependente do tempo, e este tempo, será definido como o número de períodos selecionados para o problema, por serem espaçados igualmente. Porém a intenção não é fazer uma previsão de crescimento para cada quadrícula presente no mapa, uma vez que a intenção é buscar também a influência que os vizinhos aplicam sobre a quadrícula, elaborando assim, uma técnica que leva em consideração uma quantidade de quadrículas, de modo que se defina um padrão de crescimento médio, relativo tanto ao crescimento do ponto quanto à evolução da forma da região em questão. Tendo assim, no final, um mapa que forme regiões de maior possibilidade de ocorrência de aumento de densidade para os períodos futuros.

Tem-se que o mapa é representado como uma imagem através dos mapas matrizes para cada período, então foi elaborada uma técnica semelhante a técnica apresentada no problema G2 ??, a qual buscará pela melhor solução, de forma que se encontrem os melhores filtros ou matrizes de convolução, porém, neste caso, ponderados de acordo com suas relevâncias, tal que estes filtros quando aplicados na matriz de um período t_0 opere gerando uma nova matriz t_1' , que seja mais semelhante possível à matriz do período t_1 , e quando aplicada à matriz do período t_1 , opere gerando uma nova matriz t_2' , que seja mais semelhante possível ao período t_2 , e assim por diante.

Um grande problema, não apenas para este tipo de solução, mas para todas as técnicas de previsão, é a previsão futura que leva em consideração muitos períodos a frente, e que utilizam-se dos períodos anteriores para prever o novo período, é referente ao grande acúmulo de erro durante a previsão para estes períodos futuros mais distantes da origem. Então sabe-se que a previsão tem um bom resultado para os períodos iniciais e acumulará mais erro para os períodos futuros. Entretanto, insere-se um fator na avaliação a fim de fazer com que as matrizes ou filtros de convolução, durante a evolução, levem

em consideração a taxa de crescimento média daquela região, o que ajuda a diminuir um pouco deste erro, e faz com que a solução tenha uma tendência para o crescimento médio desta região.

3.5.1 Avaliação dos países

A avaliação do problema deve fornecer indivíduos capazes de efetuar uma previsão de densidade espacial, sobre a região apresentada, para um ou mais períodos à frente, a partir de um conjunto de mapas matrizes e os índices de crescimento absoluto durante os períodos. Assim, a avaliação de cada indivíduo irá ser resumida nas etapas:

- Tradução do vetor de atributos dos países;
- Processamento dos mapas de previsão;
- Cálculo parcial do erro;
- Avaliação dos erros e atribuição do custo;

A tradução do vetor de atributos basicamente transforma o vetor de atributos do de um dado país, em estruturas mais práticas para o processamento, tal que o acesso ao vetor de atributos seja minimizada e o gerenciamento dos elementos sejam simplificados, gerando assim, um vetor de valores unitários, contendo os valores de ponderação, e um vetor de filtros, ou máscaras, do mesmo tamanho do vetor anterior, contendo matrizes que representam os filtros que serão usados para a convolução.

O processamento dos mapas de previsão é a parte principal da avaliação, e é responsável por reproduzir o processo de previsão sobre os mapas matrizes existentes, o qual será usado para efetuar a previsão real, assim que o ICA terminar a competição imperialista, fornecendo o melhor país como solução. Esta etapa então, itera por todos os mapas matrizes, com exceção do último, utilizando as máscaras de convolução, transformadas do vetor de atributos do país, para gerar mapas que sejam o resultado da convolução entre o mapa matriz do período em questão com as matrizes de convolução provindas de cada ponderação, gerando assim vários mapas matrizes. A partir destes mapas resultantes, gera-se então um mapa ponderado, sendo este a ponderação dos mapas resultantes da convolução, onde os fatores de ponderação são os contidos no vetor de valores unitários de ponderações, traduzido dos atributos dos países. Assim, com este mapa final ponderado dos diversos mapas de convolução, gera-se um mapa final que é o resultado da soma do mapa deste período com o mapa ponderado, sendo este mapa final o mapa previsto para o período seguinte, t'_1 , sendo t_0 o período atual.

Antes mesmo de sair da iteração principal, pelos mapas matrizes, calcula-se a diferença entre o mapa de previsão e o mapa real do próximo período, e tais diferenças são

usadas então, para gerar os erros produzidos em cada iteração. O cálculo dos erros leva em conta ainda o crescimento absoluto que cada período tem para com o seu próximo, então, se a diferença pode ser calculada como mostra expressão 3.10:

$$dif = ABS(Mapa[t_1] - Mapa[t'_1]). \quad (3.10)$$

Para se calcular o erro total gerado durante este período, levando em consideração o mapa de previsão e o crescimento em relação ao período de forma igualitária, fazendo-se a multiplicação da diferença $diff$ por 0.5, e então soma-se com o valor de crescimento absoluto entre os períodos t_x e t_{x+1} , que também é multiplicado por 0.5. Sendo assim, este valor final de erro para o dado período t_x , ou iteração atual, é armazenado em um vetor de erros *Errors* que armazenará todos os erros gerados a durante cada iteração entre os períodos em que se fará a previsão, como mostra a expressão ??:

$$Errors[t_x] = diff \cdot 0.5 + CrescimentoAbsoluto[t_x \Rightarrow t_{x+1}] \cdot 0.5. \quad (3.11)$$

A utilização do crescimento absoluto na função de avaliação poderia ser removida, que o resultado seria algo muito parecido se não o mesmo, porém este fator é inserido para que o processo evolutivo chegue mais rapidamente ao resultado esperado, de modo que tal influência acelere e refine o resultado final do país. Por fim, durante a última e mais simples etapa, somam-se todos os erros e atribui ao valor de custo do país em questão, finalizando assim a avaliação de um indivíduo.

Este processo de avaliação faz com que o país, ao longo das eras tenha seus atributos convergindo para a melhor solução, operando a convolução e ponderando os mapas matrizes tal que o erro gerado pelo mapa de previsão seja o mais próximo do mapa real e também, tal que seu crescimento seja predito de acordo com o crescimento entre os períodos reais.

A ponderação entre os mapas de convolução, ao invés de apenas usar uma operação de convolução diretamente, sem ponderação, é necessária para que o mapa final resulte em uma previsão mais exata. Pois entende-se que ao tentar buscar uma matriz de convolução entre duas imagens, as quais uma é o resultado da passagem da matriz de convolução sobre a outra, como descrito no problema G2 ??, é um problema diferente do objetivo do trabalho, o qual é mais simples, apesar da complexidade da solução, por se saber que existe, de fato, um filtro ou matriz de convolução que leva uma imagem ser transformada em outra, quando operada a convolução. Já o problema proposto, a diferença entre um período e outro não é apenas a busca por uma matriz de convolução, mas sim a previsão, que para isto, são usadas matrizes de convolução com seus resultados ponderados, onde cada uma destas matrizes de convolução agrupa uma ou mais características da região durante um dado período, que são relevadas pelo fator de ponderação. Observa-se que o problema proposto ainda utiliza as mesmas matrizes de convolução para operar por todos

os períodos, então se houverem várias características que possam vir a ser identificadas durante todos os períodos, as matrizes terão mais liberdade de adaptação, sendo que tais características podem variar entre as máscaras de convolução e ser ponderadas de acordo com suas relevâncias. Observa-se ainda, que a intenção não é obter o resultado exato, mas sim uma matriz que gere um mapa semelhante que possa ser usado para previsão futura das regiões.

A figura 38 ilustra uma situação de avaliação para cinco períodos, t_0, t_1, t_2, t_3 e t_4 , tais que máscaras de convolução, $Masks_{1, 2, \dots, n}$, geram mapas matrizes, $t_1'(1, 2, \dots, n)$, $t_2'(1, 2, \dots, n)$, $t_3'(1, 2, \dots, n)$ e $t_4'(1, 2, \dots, n)$, que são ponderados para gerar mapas de previsão, T_1', T_2', T_3' e T_4' .

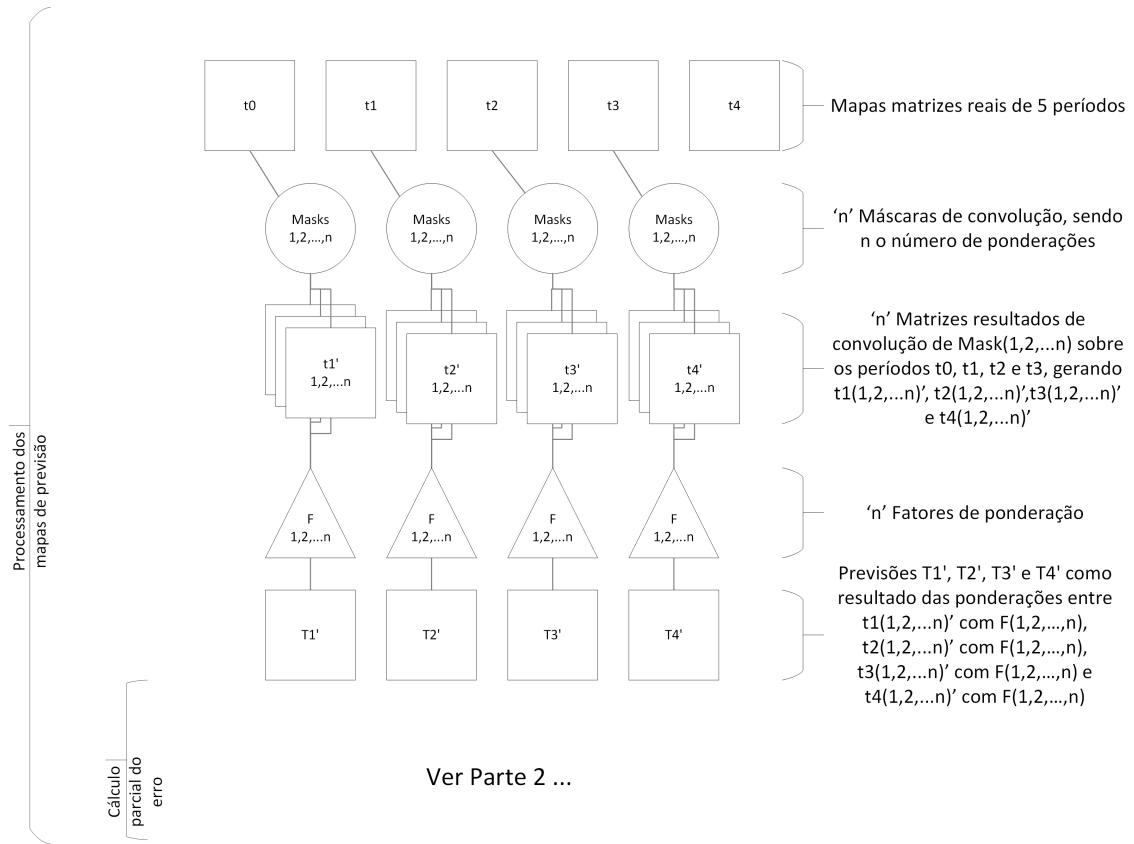


Figura 38 – Processamento dos mapas de previsão parte 1.

Já a figura 39 mostra como os cálculos parciais são executados, sendo que faz-se o uso dos mapas matrizes de previsão e dos mapas matrizes reais tal que se tenha no final, armazenados no vetor *resultValues*, os erros gerados de modo que este vetor possa ser usado para que se atribua o custo do país na próxima etapa. de avaliação dos erros e atribuição do custo.

Assim, resumidamente o que ocorre no processamento dos mapas de previsão durante a evolução, é a utilização de todos os períodos para uma forma de treinamento dos países competidores no ICA tal que estes sejam capazes, quando aplicados a um dado

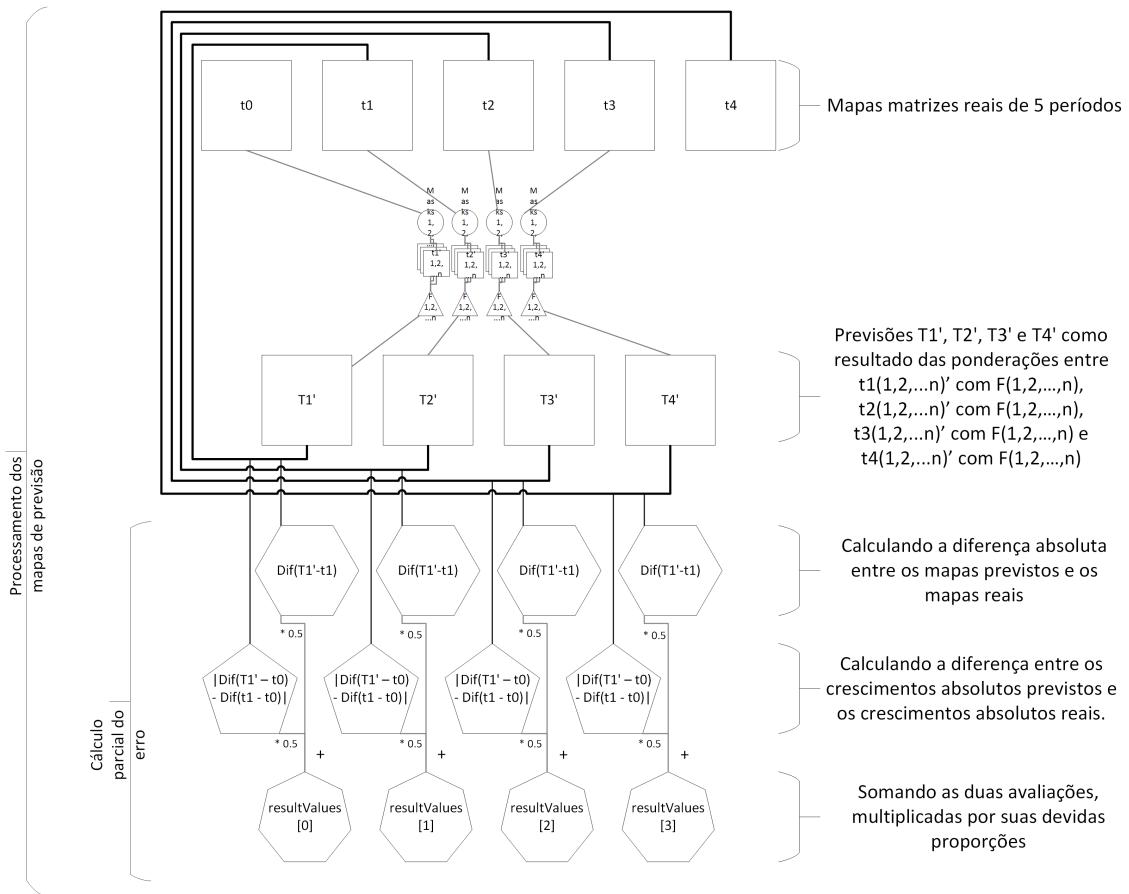


Figura 39 – Processamento dos mapas de previsão parte 2.

período, de produzirem um período à frente do período utilizado, como mostra a expressão:

$$Mapa(t'_1) = F(Mapa(t_0), País) \quad (3.12)$$

De modo que durante as próximas etapas, sejam calculados os erros, comparando o mapa de previsão gerado em $Mapa(t'_1)$ com o mapa real do período $Mapa(t_1)$, tanto em semelhança quanto em crescimento absoluto. Atribuindo assim este erro como sendo o custo do país, a fim de se obter um país, para que o ICA, então, retome seu processo evolutivo, que decidirá através das operações já mencionadas se este país possui atributos melhores que os demais, terminando assim a etapa de avaliação dos países, que ocorre muitas vezes até que se chegue em um resultado otimizado, o qual satisfaça as condições de parada.

3.5.2 A função de avaliação

A primeira etapa da implementação da função de avaliação, que é definida pelo problema em questão, para ser processado pelo ICA, é a definição da interface *IFitness* e consequente implementação de seus métodos e propriedades em uma classe chamada

PonderationFullConvolutionFitness. A implementação desta interface, independente da implementação dos métodos ou propriedades, divide-se nas partes:

- Definição de constantes e inicialização das constantes;
- Definição do método de inicialização dos países;
- Definição da função de avaliação e Teste.

Começando então pela definição dos valores constantes e suas inicializações, para este problema é interessante manter uma cópia da lista de mapas matrizes, denominada *MapsList*, em cache, de modo que esta lista possa ser acessada por múltiplas tarefas em paralelo (provindo de chamadas feitas paralelamente pelo ICA), evitando que haja escrita, pois o modelo utilizado para representar esta lista é seguro e não obstrutivo (*Thread Safe*) apenas para leitura de dados da lista. Outros dois valores constantes que definem o vetor de atributos do país são:

- *PonderationCount*, que define o número de ponderações e consequentemente quantas matrizes de convolução existirão;
- *Order*, que define qual será a ordem das matrizes de convolução.
- *absoluteGrowths*, que define o crescimento absoluto entre os mapas matrizes período a período, sendo este valor calculado assim que se entra com os mapas matrizes na lista de mapas *MapsList*.
- *maskCount*, que representa a quantidade de valores presente em cada máscara, utilizado para a definição do número de dimensões e durante a tradução da lista de atributos.

Observe que estes dois últimos elementos poderiam não existir, porém, para que possa se otimizar a função de avaliação, que pode ser chamada milhares de vezes, calculam-se todos estes elementos independentes, evitando cálculos desnecessários durante as chamadas de avaliação.

O método de inicialização dos países é bem simples, uma vez que não insere dados de entrada nos atributos dos países e o problema não exigiu alteração das funcionalidades básicas do país. Assim, a inicialização dos países se resume na criação de uma lista de países do tamanho *Dimensions*, que gera *nPopulation* países com atributos uniformemente aleatórios, onde cada dimensão está limitada entre *minBounds* e *maxBounds*. Os valores *nPopulation*, *minBounds* e *maxBounds* são parâmetros de entrada da função recebidos diretamente do ICA, os quais podem ser configurados antes de se chamar o método *Run()* do ICA que evolui os países. Já o valor de *Dimensions* é um cálculo que utiliza os valores

definidos anteriormente e é definido a seguir, em 3.5.2.1, mostrando o que cada atributo presente nos países representa no problema e como eles são traduzidos para serem utilizados pela função de avaliação.

3.5.2.1 Atributos dos Países

A partir da ideia gerada durante a modelagem da função de avaliação, definiu-se como deve ser modelado um país para que o ICA processe este problema e quais são os componentes do vetor de atributos a serem evoluídos durante a competição. Então como será feita uma ponderação de matrizes de convolução, em *PonderationCount*, tem-se quantas ponderações devem ser feitas, em *Order*, qual a ordem das matrizes de convolução. Assim, o número de atributos total que os países do ICA terão, definido em *Dimensions*, na implementação do objeto que contém a função de avaliação, pode ser descrito como apresentado na expressão 3.13:

$$\begin{aligned} Dimensions &= PonderationCount \cdot (maskCount + 1), \\ &\quad \text{Onde} \end{aligned} \tag{3.13}$$

$$maskCount = order \cdot order.$$

A definição do valor de dimensões dos países é obrigatória e ocorre durante a inicialização do objeto que implementa a função de avaliação (ou implementação da interface *IFitness*), e faz parte da modelagem do problema. Note que não se define diretamente o valor de dimensões no país, mas sim na função de avaliação, para que a solução fique genérica e dependa apenas da implementação da função de aptidão como demonstrado neste capítulo em 3.1, e ainda, não se insere um valor fixo, mas sim na expressão 3.13 de modo que o número de dimensões ainda varie de acordo com os valores de número de ponderações e ordem da máscara de convolução.

Cada país usa a implementação padrão da classe *Country* do ICA, sem nenhuma alteração, assim, todo indivíduo terá uma distribuição uniforme quando gerar os números aleatórios para seus atributos, tanto em sua inicialização quanto em chamadas que sorteiam estes atributos do indivíduo durante a evolução, causadas pela operação de revolução colonial.

A figura 40 ilustra como os valores são dispostos no vetor de atributos do país. Observe que primeiro vem o valor de ponderação e em seguida vem os valores da máscara de convolução, formando um bloco. O número de blocos é dependente da quantidade de ponderações que se deseja gerar para um mapa, e o tamanho de cada bloco depende da ordem da matriz de convolução.

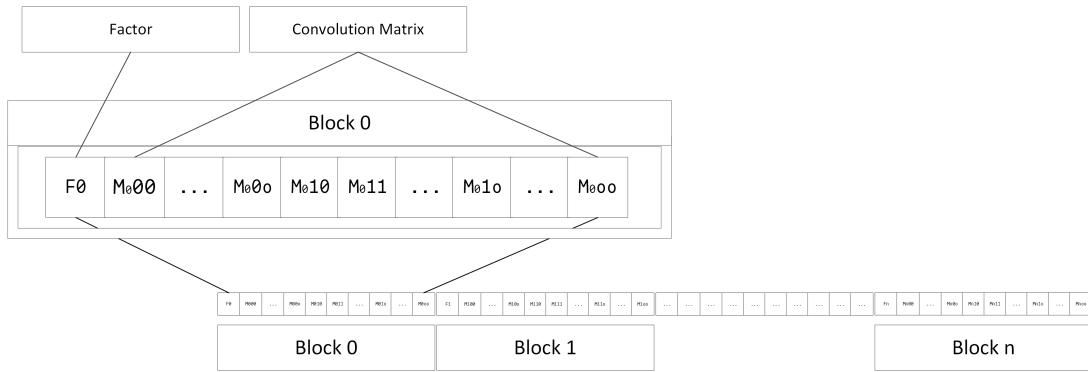


Figura 40 – Vetor de atributos do país.

3.5.3 Implementando a Função de Avaliação

Voltando então para a implementação da função de avaliação, com o número de dimensões e a estrutura do vetor de atributos dos países já definidos, pode-se então, definir a implementação da função de avaliação em três partes como descrita anteriormente, porém agora mais resumidamente e com foco na implementação de cada item:

- Tradução do vetor de atributos dos países;
 - Processamento dos mapas de previsão
 - Cálculo parcial do erro;
 - Avaliação dos erros e atribuição do custo;

A tradução do vetor de atributos deve gerar dois elementos, o vetor de ponderações e o vetor de máscaras. Ambos são obtidos em um mesmo bloco que itera i de 0 até o número de ponderações $PonderationCount$. Assim o vetor de ponderações é preenchido como 3.14:

$ponderations[i] = element.Attributes[ponderationIndex];$ Sendo (3.14)

ponderationIndex = $i \cdot (\text{maskCount} + 1);$

Em 3.14 ocorre uma cópia do valor contido no vetor de atributos no índice *ponderationIndex* para o índice *i* do vetor *ponderations*.

E em seguida, utilizando-se deste índice de ponderações *ponderationIndex*, copiam-se um intervalo de valores do vetor de atributos do país, de modo que esta cópia comece no índice *ponderationIndex + 1* deste valor de atributos, que representa o primeiro elemento da máscara, e copie os valores até se atingir uma quantidade de elementos igual a *maskCount*,

tendo assim, um vetor de elementos que represente a máscara. Ainda não se tem portanto, a máscara no formato de uma matriz, então chama-se uma função externa para converter um vetor unidimensional de valores para uma matriz que represente uma máscara ou filtro de convolução, respeitando a ordem da máscara, como mostra o exemplo contendo o algoritmo 5 usado em tal transformação:

Algoritmo 5: Algoritmo Transformação do vetor de atributos do país.

Data:

Atributos - [1, 2, 3, 4, 5, 6, 7, 8, 9].

Ordem - 3.

Result:

Máscara

[1, 2, 3]

[4, 5, 6]

[7, 8, 9]

```

1 inicializar a matriz masks[Ordem][Ordem] com zeros;
2 for k ← 0 to Ordem * Ordem do
    // índice x é o resto da divisão de k por ordem.
    3   x = k % Ordem;
        // índice y.
    4   y = (k - x) / Ordem;
    5   mask[x][y] = Atributos[k];
6 end

```

Após a obtenção da máscara representada pela estrutura matricial, que é então adicionada em um vetor de matrizes chamado *ponderationMasks*, termina-se a etapa de tradução do vetor de atributos do país para dois vetores, *ponderations* e *ponderationMasks*, contendo os valores de ponderação e as máscaras de convolução usadas para gerar os mapas de ponderação respectivamente.

Seguindo, para a etapa de processamento dos mapas de previsão, diversas inicializações são feitas, na ordem:

1. Inicializar o vetor de valores resultados em *resultValues*.
2. Inicializar o vetor de imagens resultado da operação de ponderação entre as matrizes de convolução em *ponderationImages*.
3. Inicializar o vetor de mapas resultantes da convolução em *resultMaps*
4. Inicializar uma matriz para representar o mapa final gerado dos processos de convolução e ponderação, para posterior comparação e cálculo do erro.

Após estas atualizações, itera-se por todos os mapas com exceção do último, gerando os mapas de convolução de cada período, no vetor *resultMaps*, mapeando o resultado do

cálculo para um mapa ponderado, na matriz *result* , e gerando então, um mapa final, *finalResult*, de previsão do próximo período ao se somar o mapa do período atual. E, por fim, comparando ambos, o mapa real do próximo período e o mapa final de previsão processado, a fim de gerar os erros e armazená-los no vetor *resultValues* para posterior definição de custo do país, conforme mostra o algoritmo:

Algoritmo 6: Avaliação do país.

Data:

PonderationCount - número de ponderações.

TrainingSize - quantidade de períodos para treino.

height - altura em quadrículas da região.

width - largura em quadrículas da região.

MapList - Lista de mapas históricos de quadrículas.

masks - vetor de máscaras de convolução.

ponderations - vetor de fatores de ponderação.

Result:

cost - custo avaliado do país.

```

1 next = 1;
// Iteração de previsão.
2 for now ← 0 to TrainingSize – 1 do
    // Início da função de previsão.
    3 for i ← 0 to PonderationCount do
        | resultMaps[i] = Convolution(MapList[now].MapMatrix, masks[i]);
    5 end
    6 result = new Matrix[width][height];
    7 for x ← 0 to width do
        | for y ← 0 to height do
            | num = 0; den = 0;
            | for i ← 0 to PonderationCount do
                |     num += resultMaps[i].MapMatrix[x][y] * ponderations[i];
                |     den += resultMaps[i].MapMatrix[x][y];
            | end
            | result[x][y] = num / den;
        | end
    16 end
    17 finalResult = Sum(MapList[now].MapMatrix, result);
    // End Of Fp function.
    18 resultValues[now] = Diference(MapList[next].MapMatrix, finalResult) * 0.5 +
        Abs(Diference(MapList[now].MapMatrix, finalResult) - absoluteGrowths[now]) *
        0.5;
    19 next = now + 1;
20 end
21 cost = resultValues.Sum();

```

Na linha 1 inicia-se o valor de iteração para o índice do próximo período em *next*. Na linha 2 inicia-se o processo de treino do país, passando por todos os períodos exceto o

último, de forma que se itere o valor *now*, que representa o período atual, até um período antes do último. Em seguida, nas linhas 3 e 4, geram-se todos os mapas de convolução, referentes a cada ponderação, baseando-se no mapa matriz do período definido por *now*. A linha 6 apenas inicializa um mapa vazio com as mesmas dimensões dos mapas matrizes dos períodos. Então, nas linhas 7 e 8, inicia-se o processo de iteração por todas as quadrículas (ou pixels) do mapa matriz, usando *x* para deslocamento pela largura e *y* pela altura. Em 10, 11, 12 calcula-se a soma dos numeradores e denominadores das ponderações de cada mapa multiplicado pelo valor de ponderação, gerando assim, um valor ponderado para cada pixel dentre cada um dos mapas de ponderações. Deste modo, atribui-se ao pixel (x, y) na matriz resultado, o valor ponderado $result[x][y]$, sendo o numerador dividido pelo denominador calculado no passo anterior usando os valores de ponderação como base de cálculo. Para a obtenção do mapa final de previsão na linha 17, adiciona-se o mapa resultado ao mapa base do período, sendo que tal mapa final represente o mapa futuro como sendo o mapa atual somado com os valores de desvio gerados da ponderação das convoluções de cada pixel. Por fim, na linha 18, ocorre a atribuição do valor de erro gerado pela previsão deste período para o vetor *resultValues*, sendo esta a parte mais crítica, que define de fato se a previsão feita é semelhante ao esperado ou não.

Observa-se que o cálculo dos valores *resultValues* são ponderados com 0.5 e 0.5, o que indica que ambos os valores têm mesmo peso para a avaliação. Caso fosse utilizado 0.75 para a previsão e 0.25 para os valores absolutos, teria-se uma avaliação que 'prefere' que o país tenha suas matrizes de convolução e fatores de ponderação se caracterizando mais nos mapas matrizes reais do que levando em consideração os crescimentos absolutos entre os períodos previstos e reais.

Em seguida, os erros acumulados no vetor *resultValue*, referentes cada um a uma iteração pelos períodos, comparando-os ao período seguinte, devem ser somados e então atribuídos como custo do país, terminando assim a avaliação do país, como:

$$Country.Cost = Sum(resultValues);$$

As funções *Abs*, *Sum*, *Diference* e *Convolution*, são funções, chamadas externamente, que operam valores, vetores ou, matrizes, tal que:

- *Abs* recebe um valor numérico e retorna o valor absoluto deste número.
- *Sum* tem duas sobrecargas, sendo
 - a primeira, tendo como entrada um vetor de valores e retorna a soma entre todos os valores deste vetor.
 - e a segunda, tem como entrada duas matrizes de largura e altura iguais e retorna como resultado uma nova matriz, com altura e largura também iguais

as matrizes de entrada, onde esta matriz resultado representa a soma dos elementos de cada índice das matrizes de entrada.

- *Difference* tem como entrada duas matrizes de altura e largura iguais, onde será calculada e retornada a diferença total e absoluta entre os os elementos de cada índice das matrizes de entrada.
- *Convolution* é uma operação mais complexa, que executa a convolução de um filtro ou matriz de convolução sobre uma matriz, ambos passados como parâmetro, e neste caso, ignora-se os valores de ajuste(*offset*) e divisor (citados nos conceitos de convolução) são mantidos nulos, valendo respectivamente 0 e 1, uma vez que o ICA já se encarrega de configurar apropriadamente os índices da matriz, como se ela já possuísse os valores de ajuste e divisor aplicados a si.

3.5.4 A ponderação das matrizes de convolução

A ponderação se faz importante porque neste caso, as partições dos mapas que representam os períodos subsequentes, não necessariamente são o resultado da convolução de uma máscara pela região do mapa do período anterior. Então, deve-se estimar a média de crescimento da região através de um fator, porém o objetivo não é obter o crescimento homogêneo de toda uma região, mas sim obter um crescimento direcionado não homogêneo e relativo tanto aos valores de cada quadrícula quanto aos valores das quadrículas vizinhas. Assim gera-se no final, um mapa com maior resolução, que sofre influência e aplica influência sobre seus vizinhos, demonstrando o crescimento regional num mapa bidimensional, sem que seja necessário calcular regiões as tendências ponto a ponto, o qual levaria a apenas um crescimento calculado estatisticamente para aquele ponto apenas, sem levar em consideração seus pontos vizinhos.

A figura 41 apresenta um exemplo de como esta técnica de ponderações age sobre uma matriz de valores representando uma região de 10×10 quadrículas (ou *pixels*), que está prestes a ser ponderada. O exemplo abaixo apresenta valores totalmente aleatórios para as máscaras e fatores de ponderação, como se fossem o primeiro cálculo durante o treinamento, tendo como intenção a demonstração detalhada da mecânica deste processo de ponderação. Durante o processo evolutivo espera-se que tanto os fatores quanto as máscaras de convolução tenham seus valores ajustados para o mais similar possível do mapa do próximo período.

Na figura 42 pode-se observar que foi escolhido apenas um fator de ponderação, o que fez com que a operação de convolução se torne completamente inútil, não sendo necessário seu cálculo, pois o fator aplicará como resultado do cálculo de apenas uma ponderação um crescimento homogêneo por quase todo o mapa, por exemplo, para um

Mapa Matriz T0									
3	3	7	1	7	1	1	4	7	4
7	4	7	8	2	4	1	5	0	6
5	6	0	1	7	3	9	0	7	1
6	5	8	8	0	6	9	8	5	7
1	1	1	9	1	1	5	5	3	0
9	0	9	6	7	4	5	7	3	9
3	4	3	5	5	8	7	3	3	3
0	0	0	8	6	4	4	3	5	9
0	0	0	0	1	3	5	5	5	0
0	0	0	1	7	1	2	7	8	1

Figura 41 – Exemplo de ponderação - MapaMatriz T0.

ponto qualquer (x, y) , este deve ser calculado na matriz de ponderação como:

$$Ponderação(x, y) = \sum_{i=0}^{PonderationCount} \left(\frac{Convolution(x, y) \cdot Fator[i]}{Convolution(x, y)} \right) \quad (3.15)$$

Assim o ponto (0,0) seria:

$$\begin{aligned} Pond1(0, 0) &= \frac{Convolution(0, 0) \cdot Fator1}{Convolution(0, 0)}; \\ Pond1(0, 0) &= \frac{141 \cdot 0,5}{141} = 0,5; \end{aligned}$$

Máscara 1			Convolução 1										Ponderação 1										Resultado 1 T1'										
3,2	7,3	5,9	141	227	231	235	164	116	116	137	183	129	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	3,5	3,5	7,5	1,5	7,5	1,5	1,5	4,5	7,5	4,5
6,2	7,1	7	213	281	243	251	249	228	205	228	235	147	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	7,5	4,5	7,5	8,5	2,5	4,5	1,5	5,5	0,5	6,5
6,2	9,5	8,1	250	316	327	274	239	300	319	318	262	192	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	5,5	6,5	0,5	1,5	7,5	3,5	9,5	0,5	7,5	1,5
Fator 1			167	213	258	254	238	259	327	320	251	129	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	6,5	5,5	8,5	8,5	0,5	6,5	9,5	8,5	5,5	7,5
	0,5		173	252	332	331	267	267	338	326	301	190	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1,5	1,5	1,5	9,5	1,5	5,5	5,5	3,5	0,5	
			138	217	263	330	300	311	319	268	237	139	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	9,5	0,5	9,5	6,5	7,5	4,5	5,5	7,5	3,5	9,5
			115	149	248	327	356	326	298	265	297	232	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	3,5	4,5	3,5	5,5	5,5	8,5	7,5	3,5	3,5	
			45	56	120	182	254	285	275	262	245	157	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	8,5	6,5	4,5	4,5	3,5	5,5	9,5
			0	0	55	167	202	204	231	309	293	172	0,0	0,0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,0	0,0	0,5	0,5	1,5	3,5	5,5	5,5	5,5	0,5
			0	0	7	62	88	119	145	200	160	73	0,0	0,0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,0	0,0	0,5	1,5	2,5	7,5	8,5	1,5		

Figura 42 – Exemplo de ponderação - 1 ponderação

Observe que se houver apenas uma ponderação o processo evolutivo tenderá ao crescimento médio da região durante os períodos, que neste caso levaria a uma previsão

similar às apresentadas por técnicas estatísticas que usam a linha de tendência de crescimento aplicadas ponto a ponto, neste caso região por região. Ainda assim, este método demonstra um crescimento que expande ao longo dos períodos, pois no canto inferior esquerdo, houve expansão de 1 quadrícula sobre os valores que antes eram nulos. A matriz Resultado $T1$ é basicamente a soma ponto a ponto da matriz Ponderação 1 com o Mapa Matriz $T0$ (da figura 41).

Ao se adicionar mais um fator de ponderação, como mostra a figura 43, já é possível notar que o mapa de ponderação gerado não é mais homogêneo como o anterior, e que os cálculos de cada ponto do mapa de ponderações leva em consideração o valor de seus vizinhos. E ainda, o mapa *Resultado 2 T1'* apresenta-se muito semelhante ao mapa anterior, crescendo não homogeneamente nesta região. A matriz *Ponderação 2* segue o mesmo cálculo apresentado em 3.15, usando as matrizes *Convolução 1* (da figura 42) e *Convolução 2* (da figura 43)

Máscara 2										
9,2	6,6	9,3								
6,8	7	6,5								
1,9	2,4	9,2								
Fator 2										
		1,7								
Convolução 2										
94	174	151	119	84	96	94	167	90		
190	258	232	319	206	232	147	224	219	149	
217	329	307	223	270	278	280	232	249	121	
175	227	293	215	206	318	303	347	207	160	
121	283	322	299	270	238	347	325	330	140	
123	188	262	290	310	264	256	258	233	124	
106	234	269	350	326	331	303	269	329	160	
57	82	155	211	306	314	286	260	211	154	
0	0	84	182	205	213	250	299	255	157	
0	0	6,5	62	97	143	173	240	189	108	
Ponderação 2										
1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
1,1	1,1	1,1	1,2	1,0	1,1	1,0	1,1	1,1	1,1	1,1
1,1	1,1	1,1	1,0	1,1	1,1	1,1	1,0	1,1	1,0	1,0
1,1	1,1	1,1	1,0	1,1	1,2	1,1	1,1	1,0	1,2	1,2
1,0	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,0	1,0
1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1
1,1	1,2	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,0	1,0
1,2	1,2	1,2	1,1	1,2	1,1	1,1	1,1	1,1	1,1	1,1
0,0	0,0	1,2	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1
0,0	0,0	1,1	1,1	1,1	1,2	1,2	1,2	1,2	1,2	1,2
Resultado 2 T1'										
4,0	4,0	8,0	2,0	8,0	2,0	2,0	5,0	8,1	5,0	
8,1	5,1	8,1	9,2	3,0	5,1	2,0	6,1	1,1	7,1	
6,1	7,1	1,1	2,0	8,1	4,1	10,1	1,0	8,1	2,0	
7,1	6,1	9,1	9,0	1,1	7,2	10,1	9,1	6,0	8,2	
2,0	2,1	2,1	10,1	2,1	2,1	6,1	6,1	4,1	1,0	
10,1	1,1	10,1	7,1	8,1	5,1	6,0	8,1	4,1	10,1	
4,1	5,2	4,1	6,1	6,1	9,1	8,1	4,1	4,1	4,0	
1,2	1,2	1,2	9,1	7,2	5,1	5,1	4,1	6,1	10,1	
0,0	0,0	1,2	1,1	2,1	4,1	6,1	6,1	6,1	1,1	
0,0	0,0	1,1	2,1	8,1	2,2	3,2	8,2	9,2	2,2	

Figura 43 – Exemplo de ponderação - 2 ponderações

Por fim, a figura 44, aumenta-se o número de ponderações para 3, o que consequentemente leva a 3 fatores de ponderação com a necessidade do cálculo de 3 matrizes de convolução e geração aleatória de 3 máscaras de convolução. Neste caso, observa-se que o fator apresenta um valor negativo, que decresce todo o crescimento do mapa mas mantém um padrão de crescimento muito similar ao anterior (apresentado pela figura 43).

Neste caso, com três ponderações, é possível notar que os valores dos fatores de ponderação geram valores para os pontos tal que estes serão diferentes porém com uma tendência à média dos fatores, e também são limitados na soma dos fatores. O que proporciona esta diferenciação entre os valores são as matrizes de convolução, que são usadas como ajuste, de modo que cada máscara gere uma matriz que releve uma ou mais

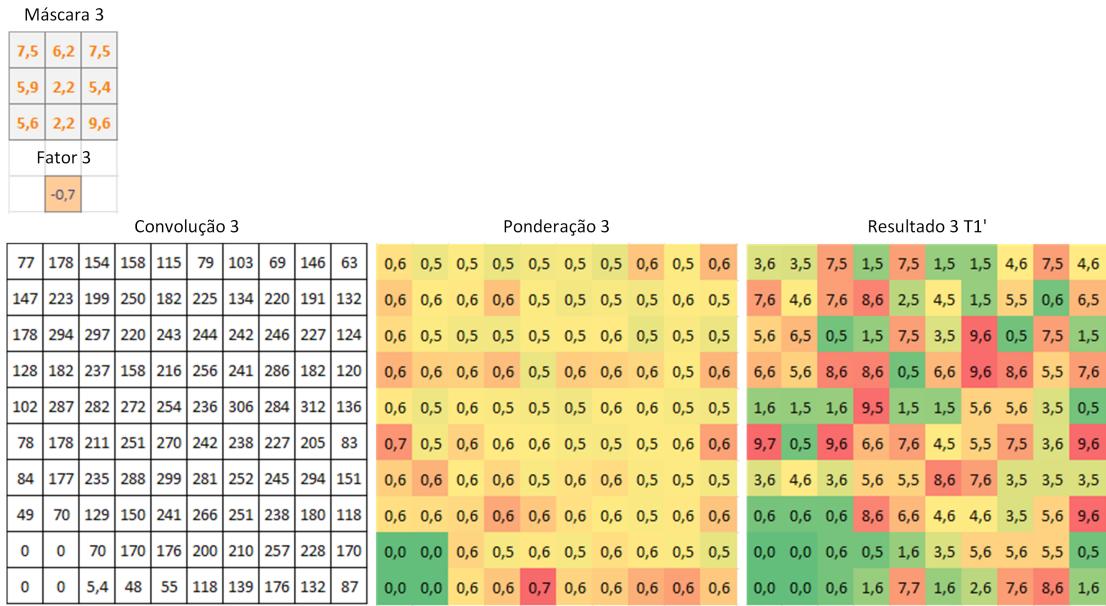


Figura 44 – Exemplo de ponderação - 3 ponderações

características do mapa a ser ponderada, com o fator, neste caso, fazendo o papel de relevar ou não a característica em questão.

Assim, é possível notar, que neste caso totalmente aleatório já existe uma tendência de crescimento geral, onde algumas regiões crescem mais que outras, de forma não homogênea, definida por características do mapa do período base, podendo assim, repetir-se tal processo para que tal tendência seja aplicada novamente sobre o mapa Resultado gerado, a fim de gerar uma previsão futura de um próximo período seguindo esta mesma tendência de crescimento.

Espera-se então, que ao fim do processo evolutivo, ao aplicar esta técnica, que leva como parâmetros o conjunto de valores compostos por máscaras e fatores, seja possível gerar uma previsão tendenciosa, ou seja, que segue a curva de tendência de crescimento para a região, de forma não homogênea, ressaltando pontos de maior e menor crescimento dependentes de características adquiridas pelos ajustes dos valores, principalmente dos valores das máscaras de convolução, e por fim de forma apresentável bidimensionalmente, de modo que tal região venha compor o mapa final.

3.5.5 A Etapa de Previsão

Voltando o foco agora para a mecânica do ICA, a avaliação ocorre para todos os indivíduos na mesma etapa, podendo ser processada em serial ou em paralelo. Observe que a avaliação de cada indivíduo ocorre independente dos demais, de modo que todo recurso compartilhado não restrinja o uso apenas para aquela avaliação, possibilitando assim que o processamento das avaliações dos países possa ser executado em tarefas paralelas paralelas.

Assim que todas as avaliações são finalizadas, independente de terem ocorrido paralela ou serialmente, o ICA passa para as próximas etapas que, então irão alterar todas as características dos países até que se termine a iteração e entre novamente na etapa de avaliação, onde todos os indivíduos são novamente avaliados. A condição de parada padrão geralmente é o número máximo de décadas, que quando atingida, termina a evolução dos indivíduos e consequentemente da competição imperialista.

Assim que a competição termina e o método ‘Run’ acaba sua execução retornando para o escopo da aplicação, é possível então pegar qualquer país usado na competição imperialista, e utilizá-lo para aplicar tal solução ao problema. O interessante é utilizar o país de menor custo, uma vez que esta é a melhor solução buscada pelo ICA. Assim, este melhor indivíduo é usado na etapa de teste, que, neste caso, é aplicado para gerar mapas de previsão de períodos futuros através de um processo muito semelhante ao processo feito pela função de avaliação. No processo de avaliação, iterou-se por diversos períodos gerando um mapa de previsão para cada período. No caso da etapa de testes, não se faz tal iteração, apenas executa-se o processo de geração do mapa de previsão, que basicamente faz os processos:

- Tradução do vetor de atributos do país.
- Processamento de um mapas de previsão a partir do último período usado.

O processo de tradução do vetor de atributos do país já fora detalhado anteriormente e não muda em nada durante esta etapa de testes. O processamento de um mapa de previsão deve sempre ocorrer a partir do último mapa usado na etapa de avaliação, sendo este o ano base para que se gere as demais previsões. Este processo de testes sempre irá gerar um período de previsão a frente do período passado, então para se gerar mais de um período de previsão, é necessário executar este processo tantas vezes quanto se quiser ter previsões de períodos futuros, passando como parâmetro, um mapa para previsão e um país, para se gerar um segundo que é o resultado do processo de previsão por usando o método de convoluções ponderadas, como mostra o algoritmo 7:

Observe que o algoritmo é uma cópia de uma parte da função de avaliação, que gera o mapa de previsão de apenas um período, fazendo a soma de todos os mapas operado-os pela convolução e ponderando-os em um mapa final. Assim, como mencionado anteriormente, é possível gerar várias previsões, utilizando o mapa gerado de uma previsão t_1' para se prever t_2' e assim por diante. Porém, como a previsão de um segundo período é sobre uma previsão já feita, acumulam-se os erros de previsões, aumentando drasticamente a incerteza para previsões mais futuras.

Algoritmo 7: Algoritmo função de previsão

Data:

PonderationCount - número de ponderações.

Map - o mapa base, que terá previsão de 1 período à frente.

height - altura em quadrículas da região.

width - largura em quadrículas da região.

masks - vetor de máscaras de convolução.

ponderations - vetor de fatores de ponderação.

Result:

finalResult - como o mapa de previsão de 1 período à frente de *Map*.

```

1 for i ← 0 to PonderationCount do
2   | resultMaps[i] = Convolution(Map.MapMatrix, masks[i]);
3 end
4 result = new Matrix[width][height];
5 for x ← 0 to width do
6   | for y ← 0 to height do
7     |   num = 0; den = 0;
8     |   for i ← 0 to PonderationCount do
9       |     | num += resultMaps[i].MapMatrix[x][y] * ponderations[i];
10      |     | den += resultMaps[i].MapMatrix[x][y];
11    |   end
12    |   result[x][y] = num / den;
13  | end
14 end
15 finalResult = Sum(Map.MapMatrix, result);

```

4 Experimentos e Resultados

Ao fim do processo evolutivo do ICA, que parou por alguma das condições de parada implementadas, obtém-se os atributos do melhor país para se gerar um mapa de previsão futura, que mantém os aspectos de crescimento médio. E então define-se uma região de provável expansão para o mapa matriz, neste caso, partição do mapa matriz, com a aplicação de uma operação semelhante à utilizada na função de avaliação, removendo as partes dos cálculos de erro, mas mantendo o processo de previsão, que faz as convoluções e ponderações para gerar o mapa de previsão final.

A previsão gerada pelo método proposto pode ser comparada com um modelo estatístico que gera uma previsão sobre a curva de tendência linear também para cada região, de modo que cada região apresente uma taxa de crescimento seguindo a curva de tendência, onde cada quadrícula da região cresce homogeneamente, sem que haja a aplicação de nenhuma heurística, e seja usado apenas o modelo estatístico. Desta forma é possível comparar a precisão do modelo de previsão proposto região por região.

4.0.1 Modelagem do ambiente

Neste trabalho, os dados foram selecionados utilizando a cidade de Taubaté-SP como região base para a criação dos mapas de quadrículas. Os dados contidos no banco de dados são referentes aos períodos de 2008 a 2015. Tais dados são separados de modo que cada período represente 1 ano, assim existirão inicialmente 8 arquivos de entrada para a aplicação, que transformará estes arquivos em mapas de quadrículas, que por sua vez serão particionados em várias pequenas partições para o processamento pelo ICA.

Para que o método seja validado, devem ser usadas apenas uma porção dos dados para se gerar os países, e então usa-se o restante dos dados reais para se comparar com os dados de previsão computados após a etapa de evolução. Então pode-se separar a validação em duas etapas, sendo a primeira a etapa de treino, a qual se usa o ICA para evoluir os países sobre uma porção dos dados, e a segunda sendo referente à comparação entre os dados reais que não participaram do processo de evolução e os dados de previsão, gerados a partir dos países evoluídos.

Assim, são selecionados os 5 primeiros anos, de 2008 a 2012, como *nLearn* períodos de treino para o ICA, restando os 3 últimos anos, sendo 2013, 2014 e 2015, como *nTest*, para testar e validar a metodologia de previsão. Neste caso, assim que se termina a etapa de evolução dos indivíduos, obtém-se o melhor indivíduo para aquela partição, e então geram-se 3 previsões seguidas a partir deste país, as quais serão comparadas com os mapas dos períodos reais restantes. A semelhança entre os mapas reais e previstos, devem seguir

o crescimento definido pela curva de tendência gerada a partir dos períodos de teste, de modo que a previsão tenha seu crescimento seguindo a média padrão de crescimento entre os anos para que tal previsão seja validada e mais próxima possível do real.

Os mapas gerados de diversos períodos da cidade de Taubaté estão limitados nas latitudes *Norte* = -22.963715 e *Sul* = -23.099506 e nas longitudes *Leste* = -45.493534 e *Oeste* = -45.674465 . O tamanho para cada lado das quadrículas é de 100 m , resultando em uma área de 100 m^2 , sendo que tais áreas englobem aproximadamente de 1 a 4 quarteirões residenciais em média. Assim, os mapas ficam com altura igual a 150 quadrículas e largura igual a 185 quadrículas, o qual nos leva a ter que a região representa um retângulo de 18.5 km de largura por 15 km de altura, tendo uma área total de 277.5 km^2 .

Estes mapas ainda são particionados em 15 partes para largura e para altura, gerando 210 partições de 13×10 quadrículas, sendo que as últimas 15, referentes às últimas colunas de cada linha sempre tenham dimensões 17×10 quadrículas para que não haja perda de informação ou geração de ruído em regiões não que estejam fora da definição (*Norte, Sul*) e (*Leste, Oeste*). Como cada quadrícula possui lado igual a 100 m , tem-se que cada partição irá abranger uma área de $1300\text{m} \times 1000\text{m} = 1300000\text{m}^2 = 1.3\text{km}^2$ para as 210 quadrículas e $1700\text{m} \times 1000\text{m} = 1700000\text{m}^2 = 1.7\text{km}^2$ para as 15 quadrículas restantes.

O tamanho das quadrículas foi diminuído, segundo WILLIS, o ideal é usar o tamanho das quadrículas como sendo $\frac{1}{4}\text{milha}$, o que é aproximadamente 400m ($\frac{1}{4}\text{milha} = 402.3360\text{m}$), a fim de se aumentar a resolução de cada partição para que esta possa captar características mais bem definidas sempre que efetuar a metodologia de previsão sobre a partição. Caso fosse utilizado o tamanho da quadrícula original, igual a 400m , cada partição seria composta por aproximadamente 3×2 quadrículas, diminuindo assim a resolução de cada partição. Esta diminuição tem como intuito fazer com que cada quadrícula represente aproximadamente de 1 a 2 quarteirões de tamanho médio.

4.0.2 Testes sobre cada regiões

Não se foi definida uma relação entre o tamanho de cada partição e o tamanho da máscara de convolução, porém espera-se que cada partição seja pelo menos 2 vezes maior que a matriz de convolução tanto em largura quanto em altura para que as máscaras possam gerar mapas que possuam características da região abordada nesta região de uma forma aproximada sem que os valores mais periféricos da máscara fiquem, em sua maior parte durante o processamento, para fora da região a ser analisada.

Sendo assim máscaras de ordem 3, 5 e 7 podem ser usadas para gerar as previsões para este caso, onde tem-se que a menor partição é de 13×10 quadrículas, já as máscaras de ordem 9, estão no limite entre o que se considera previsão com erro aceitável, podendo

apresentar resultados de previsão não aceitáveis, e as máscaras de ordem 11, 13, etc. apresentam um resultado mais agressivo (mesmo aumentando o número de ponderações), já que alguns de seus valores mais periféricos ficarão para fora da imagem por muitos ciclos do processo de convolução, e assim, o ICA é incapaz de evoluir seus valores para alguma característica da região, de modo que a previsão de algumas regiões resulte em uma previsão irreal, *estourando* a previsão de crescimento da região, tal que estes valores que ficam durante muitos ciclos de fora da região sejam em sua maior parte aleatórios, aplicando as características inexistentes na região de uma forma muito intensa em pontos mais distantes, gerando muito ruído no processo de previsão.

A figura 45 mostra uma região real, e a mesma região com a previsão 'estourada'. As cores das regiões em questão foram invertidas para facilitar a visualização das áreas urbanas, neste caso pontos mais claros representam pontos de alta densidade e regiões mais escuras representam pontos de baixa densidade, a cor preta representa uma região com nenhuma residência.



Figura 45 – Imagem da previsão estourada

Uma solução para este problema seria alterar o algoritmo de convolução para efetuar um processo de convolução aperiódica, que é uma forma diferente do padrão(convolução truncada, que centra a máscara com o primeiro pixel da imagem, e atribui zero aos valores inexistentes da imagem), que sempre agirá dentro da imagem porém atribui zero para resultados não calculáveis. A figura 46 abaixo ilustra no seu item (a) a convolução aperiódica, e em (b) a convolução padrão (Truncada).

A opção de utilizar esta forma de convolução não se aplicaria para a previsão, pois estaria-se perdendo informação e aumentando o erro da previsão de uma forma drástica, sendo que os valores das bordas de cada região seriam 0.

4.0.3 Testes sobre todas regiões

Para cada partição, será gerado um país através do processamento no ICA, que por sua vez será usado para fazer a previsão dos *nTeste* períodos desta partição. Foram

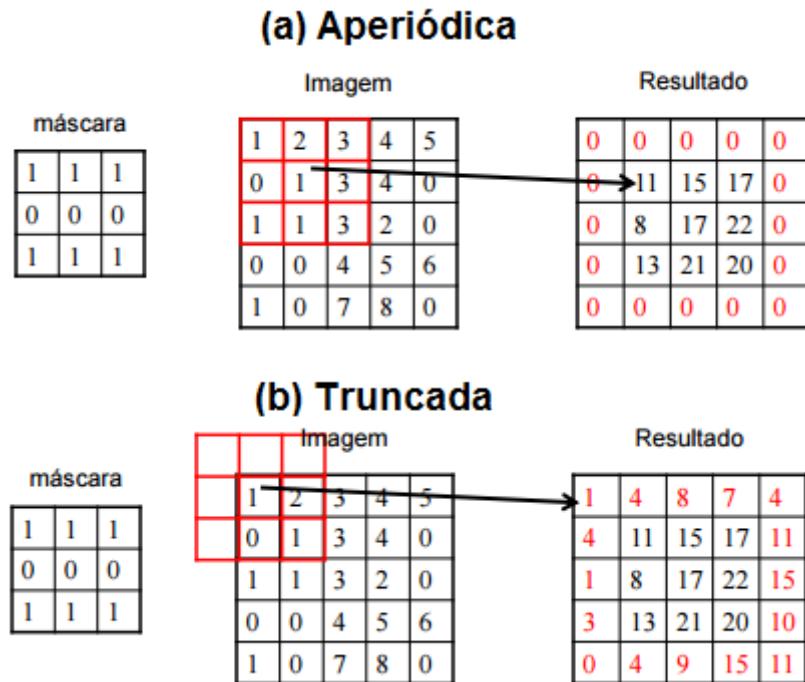


Figura 46 – Convolução aperiódica versus truncada

executadas duas configurações de treino, sendo a primeira com apenas a condição de paradas por máxima quantidade de décadas, com os valores do ICA configurados como:

- Vetor de atributos do país:
 - Os valores de fator estão limitados em [-50,50];
 - Os valores das matrizes de convolução são estendidos para [-100,100];
 - valores das propriedades do ICA:
 - Beta: 2;
 - Taxa de revolução: 0.99;
 - Taxa de decaimento de revolução: 0.999999;
 - Taxa de impérios iniciais: 15%.
 - É Paralelo? sim;
 - População inicial: 64;
 - Máximo de décadas: 2048;
 - Modo de movimento: Combinado(Refinado + Visão Imperial Distorcida);

Para a segunda configuração, foi implementada uma condição de parada que leva em consideração o número máximo de décadas e o custo mínimo a ser atingido. Esta condição de parada segue o padrão de validação segundo o algoritmo 8 apresentado abaixo:

Algoritmo 8: Algoritmo condição de parada para o ICA

Data:

$curD$ - década atual.

$maxD$ - número máximo de décadas.

$curC$ - custo do melhor país.

$minC$ - valor do custo mínimo.

Result:

retorna se a condição de parada foi atingida ou não.

```

1 se  $curD \geq maxD * 10$  então
2   | retorna Condição de parada atingida;
3 fim
4 se  $curD > maxD$  então
5   | se  $curC < minC$  então
6     |   | retorna Condição de parada atingida;
7   | fim
8 fim
9 retorna Condição de parada NÃO atingida;
```

Observa-se no algoritmo da condição de parada que ela só será atingida quando atingir 10 vezes o número de máximo de décadas estipulado ou quando o custo for menor que $minC$ e, inclusivamente, o número de décadas for maior que o número máximo de décadas estipulado. Assim, garante-se que o algoritmo só irá parar segundo estas condições, tal que considera-se importante atingir o valor $maxD$ de décadas mesmo se o custo for menor que o valor definido em $minC$, e ainda tenta-se atingir $minC$ até 10 vezes o valor estipulado em $maxD$. É importante mencionar que se um país chegar ao custo zero, o ICA para imediatamente, por este ser o melhor país (melhor solução a ser encontrada), e ainda, esta condição não é avaliada junto das condições de parada, mas sim antes de começar tal avaliação.

Esta segunda configuração não difere muito da primeira em questão dos parâmetros do ICA, basicamente insere-se esta nova condição de parada, que fica como:

- Vetor de atributos do país:
 - mesmos valores;
- valores das propriedades do ICA:
 - mesmos valores;
 - Adicionar condição de parada padrão por número máximo de décadas: Não;
 - Usar condição de parada por década e Custo: Sim;
 - Condição de parada adicional (por década e custo):
 - * Custo mínimo = 50;

* Máximo de décadas = 2048;

Geralmente, os países não convergem para custo 0, mas geram mapas com uma diferença mínima entre os períodos, de modo que este país quando utilizado para fazer uma previsão futura, seguirá o padrão de crescimento, elevando os valores de cada quadrícula de acordo com a função de testes descrita, que neste caso é a própria função que faz a previsão da expansão urbana, que usa o país como solução ótima aplicada sobre os dados, operando a ponderação das convoluções e gerando um mapa final de previsão.

Esta segunda configuração, apesar de parecer mais rápida, acaba sendo mais demorada por permitir que o número máximo de décadas se estenda para 10 vezes o tamanho configurado inicialmente. Porém, obtém-se uma previsão mais precisa em troca de tempo de processamento.

Em posse dos melhores países para cada partição, inicia-se o processo de geração dos $nTest$ mapas de previsão, também para cada partição. Assim que os mapas de previsão para cada partição são gerados, pode-se compará-los com os mapas particionados reais, tomando a diferença absoluta entre a soma das diferenças das quadrículas (ou *pixels*) dos mapas como sendo o erro de previsão daquela partição.

Ao considerar cada partição como um bloco fechado, que cresce em densidade durante os períodos, tal bloco deve seguir uma linha de crescimento tendenciosa, que define em quantidades absolutas como tal bloco se desenvolverá durante períodos futuros. Mas é impossível saber, dentro deste bloco, onde e como tal crescimento será imposto. A linha de tendência sobre o crescimento do bloco fornece apenas um valor unitário de intensidade sobre como tal bloco se desenvolverá no futuro, de modo que só se pode afirmar, que todos os pontos (*pixels* ou quadrículas) deste bloco irão crescer de um valor x para um valor $x+y$, seguindo especificamente a função de crescimento gerada pela curva de tendência linear.

O modelo proposto, então apresenta uma solução para que se possa obter tal nível de precisão, de modo que os blocos, tenham um crescimento que siga tal linha de tendência, mesmo que a função de avaliação não os considere, e que além disso, apresentem uma solução que leve em consideração os pontos contidos no bloco, suas posições, intensidades e características regionais, dentro deste bloco, e portanto possibilitando a construção de um mapa de quadrículas disposto bidimensionalmente em forma de uma imagem, onde cada quadrícula pode ser representada por um pixel dessa imagem. Assim a análise de previsão desta metodologia utiliza-se de dois parâmetros para comparar a previsão, sendo o crescimento geral da região durante os períodos e o valor absoluto de acerto ponto a ponto.

Dentro deste cenário foram efetuados diversos testes, variando o tamanho da matriz de convolução, e a quantidade de ponderações. Foi definido que o número de ponderações igual a 3 é suficiente para prover suavidade e precisão no mapa de previsão gerado em

relação aos métodos convencionais para geração do mapa de previsão. O uso de apenas um fator e uma matriz de convolução pode resultar algumas vezes, em um mapa de previsão semelhante ao modelo estatístico, que usa a curva de tendência (neste caso qualquer que seja, linear, exponencial, polinomial etc.) referente ao crescimento da região, mas não é tão preciso quando refere-se à constituição (forma e intensidade dos pontos) do mapa, relativo a quantidade de pontos, seja ela percentual ou absoluta, acertados pela previsão. O ponto negativo é que quanto mais ponderações houver, mais aumenta-se o uso de processamento, colocando mais precisamente, a quantidade de convoluções aumenta de acordo com o número de ponderações, sendo que a operação de convolução é o processo mais pesado que ocorre durante a avaliação, que por sua vez aumenta o tempo de execução de todo o processo de avaliação.

O aumento da ordem da matriz de convolução aumenta drasticamente o tempo de processamento, porém traz resultados mais interessantes. Quando seleciona-se o tamanho mínimo de 3 para a ordem da matriz de convolução, os resultados apresentados já são satisfatórios e apresentam uma precisão no acerto sobre o crescimento geral da região e sobre a quantidade de acertos ponto a ponto melhor que a curva de tendência aplicada sobre a mesma região. Ao aumentar a ordem da matriz de convolução para 5, tem-se uma melhora no erro do crescimento da região em relação à matriz de ordem 3, porém aumenta-se sutilmente o erro da previsão ponto a ponto. Aumentando a ordem da matriz para 7, algumas regiões passam a apresentar um comportamento atípico, fazendo com que o resultado final de previsão exploda para valores de erros, tanto no crescimento regional geral quanto no ponto a ponto, muito maiores que os apresentados pelo modelo estatístico, tornando o mapa final de previsão muito ruidoso e impossível de ser usado para fazer qualquer tipo de análise. A figura 47 mostra uma comparação entre os valores reais, os resultados gerados pela metodologia de previsão proposta, usando matrizes de convolução 3x3, 5x5, 7x7, com o modelo estatístico, para os anos de 2013, 2014 e 2015.

Dois parâmetros foram numericamente comparados: o erro sobre a taxa de crescimento da região prevista (Eg), e a precisão da previsão de cada célula da região (Epp). A tabela 1 mostra faz uma comparação entre os 4 métodos (proposto 3x3, proposto 5x5, proposto 7x7 e curva de tendência), onde é possível observar que o método proposto apresenta uma precisão muito semelhante em relação à linha de tendência quando compara-se o crescimento geral da região, e apresenta ainda, na previsão ponto a ponto (sobre cada célula ou *pixel*) valores notavelmente melhores. Na comparação deste segundo parâmetro é possível observar ainda que ao aumentar-se a ordem da matriz de convolução não necessariamente piora-se a previsão, porém é importante notar que este aumento pode vir a extrapolar a previsão e fazer com que a precisão seja invalidada, apresentando uma previsão "estourada" como foi visto na 45. Os valores da matriz 5x5 apresentam-se um pouco piores, devido exatamente à extração na previsão do período de 2015, sendo que na figura 47 é possível observar alguns pontos de crescimento nunca vieram a ocorrer

em determinadas regiões no mapa real ou nos mapas previstos usando as matrizes de convolução de ordem 3 e 7.

Tabela 1 – Erros de previsão

	Ano	Eg	Eg%	Epp	Epp%
Matriz 3x3	2013	3407,36	8,38%	7044,81	2,46%
	2014	9240,26	18,52%	14717,04	5,14%
	2015	15614,48	26,13%	23649,79	8,27%
Matriz 5x5	2013	3640,63	8,84%	7359,92	2,57%
	2014	9574,66	19,25%	15303,43	5,35%
	2015	21038,39	28,86%	29585,02	10,34%
Matriz 7x7	2013	3326,51	8,80%	7306,07	2,55%
	2014	9355,60	18,52%	15325,48	5,36%
	2015	16181,37	26,27%	24992,63	8,73%
Linha de Tendência	2013	3699,5	8,93%	89488,3	31,28%
	2014	9300,8	18,52%	95026,6	33,21%
	2015	15757,1	23,15%	100564,9	35,15%

Nota: Validação dos resultados apresentando os erros de crescimento (Eg e Eg%) e por pixel (Epp e Epp%) entre o método proposto com as matrizes de convolução de ordem 3, 5 e 7 e Linha de Tendência para as previsões de 2013, 2014 e 2015



Figura 47 – Apresentação dos mapas reais de 2013, 2014 e 2015 comparando com a previsão por linha de tendência e com o método proposto para matrizes de convolução de ordem 3, 5 e 7

5 Conclusão

Este trabalho apresenta uma abordagem evolucionaria para previsão espacial de densidades quaisquer, enfatizando no trabalho e em seus exemplos e aplicações a densidade de carga elétrica. Esta abordagem usa dados históricos de pontos de instalação elétrica de diversos tipos (residencial, comercial, industrial, etc.), sendo tais dados podendo ser inseridos de qualquer fonte em uma forma normalizada (coordenada geográfica, valor de densidade, tipo e data). Quando comparado com dados históricos reais de uma cidade de tamanho médio, os resultados apresentados foram de alta qualidade, apresentando um erro espacial de menos de 9 % para uma previsão que acumulou erros durante 3 períodos consecutivos.

São gerados então diversos mapas históricos, isto é, que representam como cada uma dentre todas as suas pequenas regiões se encontrava, quantitativamente em relação à um parâmetro principal (densidade de carga, densidade populacional, etc.) a ser analisado, parâmetro este, que por sua vez, pode ser composto por outros fatores (no caso da densidade de carga: residencial, comercial etc.).

Para que o método proposto tivesse um impacto melhor na previsão e adquirisse mais características de regiões mais próximas do ponto a ser analisado, o mapa de quadrículas foi criado em alta resolução (quadrículas de $100\ m^2$), tal que o processamento e a obtenção das características de crescimento fossem obtidas através da otimização regional, formando assim diversas sub regiões (compostas de aproximadamente 130 quadriculas, algumas maiores que outras devido às bordas não serem proporcionais ao valor escolhido para a divisão das sub regiões), onde cada sub região tem seus próprios parâmetros de previsão de crescimento. Assim cada mapa histórico é composto por sub regiões, que por sua vez são compostas por quadrículas. A partir da observação e análise destes mapas históricos imaginou-se uma função de previsão capaz de executar uma previsão futura de um período t_{x+1} , tal que esta função tenha como parâmetros, um mapa de quadrículas (sub região de quadriculas) de um dado período t_x e um conjunto de parâmetros (otimizados pelo ICA, usados para formar os fatores e matrizes de convolução).

Após a concepção da função de previsão, esta fora adaptada de modo que pudesse ser utilizada como função de avaliação do ICA, tal que seus parâmetros dinâmicos (todos os parâmetros com exceção do mapa de quadrículas) fossem otimizados na forma de uma *regressão espacial*, sendo que tais parâmetros ficasse otimizados pela passagem da função de previsão por todo histórico de mapas anteriores à previsão.

Foram propostas duas otimizações no algoritmo que faz a otimização dos parâmetros da função de previsão, solucionando um problema de dimensionalidade ao se adicionar

ruído durante a exploração do espaço de busca da solução. A combinação dos dois métodos apresenta uma melhora ainda mais significativa na convergência para a melhor solução.

O método proposto é ainda comparado com o método estatístico de previsão por curva de tendência, onde o método proposto obteve vantagem em tal comparação no quesito de comparação ponto a ponto, uma vez regiões não seguem uma função de crescimento linear, e obteve a mesma taxa de acerto em relação ao crescimento da região, o que valida esta abordagem proposta.

Uma possível desvantagem deste método é a aquisição dos dados reais, que podem se fazer complexos, por isto inicialmente resume-se a base de dados em latitude, longitude, tipo, data e valor. A representação do problema com dados reais de carga se fazem importantes para melhorar a qualidade da previsão e executar outras atividades relacionadas ao planejamento e operações de sistemas de distribuição.

Outra desvantagem aparente é o tempo de processamento da função de convolução, sendo esta uma operação muito pesada, levando aproximadamente 2 a 3 horas para o processamento de todas as 225 regiões em um computador AMD-FX8350 3gHz (limitado) e 16GB RAM usando a matriz de convolução de ordem 7 e 3 ponderações e com o processamento paralelo ligado. Como o código do ICA foi desenvolvido de forma a ser ótimo em relação às suas operações internas, a maior parte do tempo se deve ao processamento da função de avaliação, e não à rotinas do ICA. Pois na Função de avaliação ocorrem, neste caso, uma operações de convolução de imagem (em escala de cinza) para ponderação durante cada período a ser analisado na otimização, sendo que o número de operações de convolução que podem ocorrer durante uma

Uma das grandes vantagens deste método sobre os demais é a consideração dos diversos pontos vizinhos durante a etapa de otimização dos atributos para a função de previsão, e devido à operação de ponderação dos resultados de cada pixel calculado de convolução, executada na função de avaliação faz com que os valores tenham uma precisão maior e condizem mais realmente que métodos estatísticos com a realidade.

Os resultados deste tipo de previsão são de grande valor para as concessionárias, por exemplo, em um caso de planejamento de expansão, será possível encontrar os pontos ideais para inserir novos transformadores ou realocar transformadores usados e outros dispositivos para outras partes da cidade economizando dinheiro, podendo diminuir os investimentos em novos equipamento e otimizando o uso de equipamentos sub-utilizados.

Com esta previsão de crescimento de alta definição é possível então fazer o planejamento sustentável, em planejamento de expansão urbana em diversos aspectos, como por exemplo, de sistemas de *smartgrids* em regiões urbanas com custo mínimo de implantação, ou seja, quando se sabe onde e como uma cidade irá crescer é possível baratear os custos de implantação, direcionando investimentos para regiões mais aptas, melhorando o consumo

geral.

Em resumo, este trabalho resultou em uma metodologia de previsão de densidade de áreas urbanas capaz de manter uma alta definição na previsão espacial de forma bidimensional (mapas de quadrículas em 2d), seguindo a taxa de crescimento padrão segundo as metodologias de previsão existentes para regiões urbanas.

5.1 Colaborações do Trabalho

Este trabalho apresenta duas colaborações relevantes, sendo a primeira a resolução do problema da dimensionalidade dos limites dos atributos do ICA, que fora solucionado com duas implementações distintas, podendo elas serem utilizadas separadamente ou de forma combinada, alterando diretamente a forma como os ruídos são inseridos durante a assimilação de colônias por um império, diminuindo o tempo de conversão, em número de décadas, para o mínimo global. Além disso, o ICA foi desenvolvido de forma a ser facilmente utilizado por qualquer aplicação, isolando a sua lógica do processo evolutivo do problema a ser implementado, de modo que, qualquer problema que venha a ser modelado no ICA não precise alterar os componentes relativos ao processo evolutivo.

A segunda colaboração deste trabalho é referente à forma como foram usadas as operações de convolução, sendo estas combinadas em uma função de previsão, que faz ponderações sobre diversas matrizes de convolução, que neste caso são treinadas pelo ICA com o intuito de obter características de crescimento de densidade distintas de uma mesma região, as quais, junto de outros atributos, são usadas para formar um mapa futuro de previsão. Com a operação de convolução, é possível buscar tais características de crescimento para cada ponto de modo que tanto o ponto em questão quanto os pontos vizinhos sejam levados em consideração no momento da definição da característica, durante a etapa de treino. A ponderação entre as diversas matrizes de convolução garante que a previsão de um dado ponto condiz mais com a realidade, dentre diversas características (matrizes de convolução).

5.2 Trabalhos Futuros

Este trabalho aplicou a função de previsão sobre os dados de densidade de carga elétrica. Porém, tal técnica pode ser utilizada para a busca de um mapa de crescimento futuro baseado em qualquer tipo de densidade. Para trabalhos futuros tem-se a intenção de alterar a função de previsão para que esta faça o uso de outra técnica de processamento de imagens ou até mesmo, que venha a utilizar técnicas de visão computacional, como a identificação de padrões através de técnicas do tipo Haar (MITA; KANEKO; Hori, 2005), que geralmente são utilizadas para encontrar padrões faciais, porém podem ser

aplicadas para o treinamento e reconhecimento de qualquer tipo de padrão em imagens, sendo possível a detecção de padrões de crescimento sobre a identificação de padrões na forma e intensidade dos pontos da imagem, diminuindo a quantidade de parâmetros, porém aumentando a quantidade de características de crescimento de uma região.

Referências

- ARANGO, H. G.; LAMBERT-TORRES, G. Modelo de simulação espaço temporal do mercado consumidor de energia elétrica baseado em múltiplos polos urbanos e uso do solo. 2000. Citado 3 vezes nas páginas 6, 18 e 19.
- ARANGO, H. G.; LAMBERT-TORRES, G. Spatial electric load distribution forecasting using simulated annealing. *WSEAS Transactions on Systems, World Scientific and Engineering Society Press, ISSN*, p. 1109–2777, 2004. Citado na página 13.
- ATASHPAZ-GARGARI, E.; LUCAS, C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: IEEE. *2007 IEEE Congress on Evolutionary Computation*. [S.l.], 2007. p. 4661–4667. Citado 5 vezes nas páginas 15, 22, 25, 69 e 79.
- BOOCH, G. Object-oriented design. *ACM SIGAda Ada Letters*, ACM, v. 1, n. 3, p. 64–76, 1982. Citado na página 23.
- CARRENO, E. M.; ROCHA, R. M.; PADILHA-FELTRIN, A. A cellular automaton approach to spatial electric load forecasting. *IEEE Transactions on Power Systems*, IEEE, v. 26, n. 2, p. 532–540, 2011. Citado na página 13.
- COAD, P.; YOURDON, E. *Object-oriented design*. [S.l.]: Yourdon press Englewood Cliffs, NJ, 1991. v. 92. Citado na página 23.
- DOMINGUEZ-TORRES, A. *Origin and history of convolution*. 2010. Citado 2 vezes nas páginas 33 e 34.
- GRIDDED Population of the World, Version 4 (GPWv4): Population Density. NASA Socioeconomic Data and Applications Center (SEDAC), 2016. Disponível em: <<http://dx.doi.org/10.7927/H4NP22DQ>>. Citado na página 14.
- LONGLEY, P. A.; BATTY, M. *Spatial analysis: modelling in a GIS environment*. [S.l.]: John Wiley & Sons, 1996. Citado na página 14.
- MELO, J.; PADILHA-FELTRIN, A.; CARRENO, E. Spatial pattern recognition of urban sprawl using a geographically weighted regression for spatial electric load forecasting. In: IEEE. *Intelligent System Application to Power Systems (ISAP), 2015 18th International Conference on*. [S.l.], 2015. p. 1–5. Citado na página 13.
- MELO, J. D.; CARRENO, E. M.; PADILHA-FELTRIN, A. Multi-agent simulation of urban social dynamics for spatial load forecasting. *IEEE Transactions on Power Systems*, IEEE, v. 27, n. 4, p. 1870–1878, 2012. Citado na página 13.
- MITA, T.; KANEKO, T.; HORI, O. Joint haar-like features for face detection. In: IEEE. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. [S.l.], 2005. v. 2, p. 1619–1626. Citado na página 120.
- MITCHELL, M. *An introduction to genetic algorithms*. [S.l.]: MIT press, 1998. Citado 2 vezes nas páginas 15 e 22.

- NORMARK, K. Overview of the four main programming paradigms. Aalborg University, 2013. Citado na página 41.
- REN, W.; JIANG, X.-h.; SUN, T.-f. Rbfnn-based prediction of networks security situation [j]. *Computer Engineering and Applications*, v. 31, p. 136–139, 2006. Citado na página 15.
- ROCHE, R. *github.com/robinroche/jica Java ICA repository*. 2011. <<https://github.com/robinroche/jica>>. Accessed: 2016-11-30. Citado na página 23.
- ROCHE, R. et al. Imperialist competitive algorithm for dynamic optimization of economic dispatch in power systems. In: SPRINGER. *International Conference on Artificial Evolution (Evolution Artificielle)*. [S.l.], 2011. p. 217–228. Citado 5 vezes nas páginas 15, 23, 40, 58 e 79.
- SHUMAKER, B.; SINNOTT, R. Astronomical computing: 1. computing under the open sky. 2. virtues of the haversine. *Sky and telescope*, v. 68, p. 158–159, 1984. Citado 2 vezes nas páginas 20 e 85.
- SNYDER, J. P. *Map projections–A working manual*. [S.l.]: US Government Printing Office, 1987. v. 1395. Citado 2 vezes nas páginas 22 e 85.
- WHITEHEAD, B. A.; CHOATE, T. D. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE transactions on neural networks*, IEEE, v. 7, n. 4, p. 869–880, 1996. Citado na página 15.
- WILLIS, H. L. *Spatial electric load forecasting*. [S.l.]: CRC Press, 2002. Citado 8 vezes nas páginas 13, 14, 17, 18, 82, 90, 91 e 110.
- WILLIS, H. L.; ENGEL, M. V.; BURI, M. J. Spatial load forecasting. *IEEE Computer Applications in Power*, IEEE, v. 8, n. 2, p. 40–43, 1995. Citado 3 vezes nas páginas 6, 17 e 19.
- WILLIS, H. L.; ROMERO, J. Spatial electric load forecasting methods for electric utilities. *Quanta Technology*, 2007. Citado 2 vezes nas páginas 13 e 14.
- WU, H.-C.; LU, C.-N. A data mining approach for spatial modeling in small area load forecast. *IEEE Transactions on Power Systems*, IEEE, v. 17, n. 2, p. 516–521, 2002. Citado na página 13.