Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica e de Computação

Luiz Desuó Neto

Otimização por Enxame de Partículas do Roteamento com Múltiplas Paradas

São Carlos
2016

Luiz Desuó Neto

Otimização por Enxame de Partículas do Roteamento com Múltiplas Paradas

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Prof. Dr. Carlos Dias Maciel

São Carlos

2016

Neto, Luiz Desuó
N469o Otimização por Enxame de Partículas do Roteamento com Múltiplas Paradas / Luiz Desuó Neto; orientador Carlos Dias Maciel. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. DPSO. 2. TSP. 3. Roteamento. 4. Restabelecimento. 5. Sistema de distribuição. I. Título.

FOLHA DE APROVAÇÃO

Nome: Luiz Desuó Neto

Título: "Otimização por enxame de partículas do roteamento com múltiplas paradas"

Trabalho de Conclusão de Curso defendido e aprovado em 22 / 11 / 2016 ,

com NOTA 9,5 (nove , cinco), pela Comissão Julgadora:

Prof. Associado Carlos Dias Maciel - Orientador - SEL/EESC/USP

Mestre Michel Bessani - Doutorando/SEL/EESC/USP

Mestre Tadeu Junior Gross - Doutorando - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

Resumo

O plano de restabelecimento de energia elétrica está diretamente ligado à confiabilidade da rede e ao tempo de restabelecimento de energia. De forma que um plano de restabelecimento ineficiente pode acarretar na interrupção de energia a diversas unidades consumidoras gerando impactos econômicos e sociais. Um plano de restabelecimento consiste em: detecção de falha, isolação e restabelecimento. A detecção ocorre por meio de equipamentos, que desligam o alimentador ou sinalizam a falha. Após o desligamento, o setor é isolado através de chaves de manobra de carga e o alimentador é religado. Por fim, o setor é inspecionado por uma equipe para identificar e reparar a falha. A roteirização dessa equipe por esses pontos visando menores custos pode ser entendida como um caso especial do problema do caixeiro-viajante em que se aplicou o *DPSO*, uma meta-heurística baseada em inteligência coletiva para encontrar soluções otimizadas num espaço de busca discreto. O algoritmo foi desenvolvido na linguagem *Python* 2.7 e foi aplicado em um computador com processador de 64 bits do tipo Intel(R) Core(TM) i3-4005U com *clock* interno de 1,70 GHz, em que foram realizados diversos testes em relação aos parâmetros do algoritmo e às entradas do sistema. Embora o algoritmo não garanta ótimos globais, os resultados obtidos foram satisfatórios quanto a redução do custo. O objetivo desse trabalho é roteirizar uma equipe de inspeção visando menores custos utilizando o *DPSO*.

Palavras chave - *DPSO*, *TSP*, Roteamento, Restabelecimento, Sistema de distribuição.

Abstract

The restoration plan of a distribution system is directly connected to the system reliability and power restoration time. An inefficient restoration plan may result in power interruption to various consumer units and reflects in economic and social impacts. Therefore, a restoration plan consists of: fault detection, isolation and restoration. The detection occurs through equipments, which switch off the power supply or signals failure. After triping, the fault sector is isolated by switching devices and the feeder is switched. Finally, a crew is sent to identify and repair the failure. The routing of this crew through these points aimed at lower costs can be understood as a special case of traveling salesman problem in which case was applied the DPSO, a meta-heuristic based on collective intelligence to find optimal solutions in a discreet search space. The algorithm was developed in Python 2.7 and was applied on a 64-bit processor Intel (R) Core (TM) i3-4005U with internal clock 1.70 GHz in that were conducted several tests on the algorithm parameters and system entries. Although the algorithm does not guarantee the global minimum, the results were satisfactory as reducing the cost. The objective of this work is to route an inspection crew aiming at lower costs using the DPSO.
Index Terms *- DPSO, TSP, Routing, Restoration, Distribution system.*

Lista de abreviaturas e siglas

PSO - *Particle Swarm Optimization*

DPSO - *Discrete Particle Swarm Optimization*

TSP - *Travel Salesman Problem*

PRE - Plano de Restabelecimento de Energia

UC - Unidade Consumidora

DEC - Duração Equivalente de Interrupção por Unidade Consumidora

FEC - Frequência Equivalente de Interrupção por Unidade Consumidora

DIC - Duração de Interrupção Individual por Unidade Consumidora

FIC - Frequência de Interrupção Individual por Unidade Consumidora

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Estatísticas mostram, na maior parte, que falhas na rede de distribuição de energia elétrica ocorrem devido ao clima, desgaste de equipamentos e acidentes (KAVOUSI-FARD ABDOLLAH; NIKNAM, 2014). Mas, antes mesmo de identificar as causas de um desligamento, deve-se identificar o local em que ele ocorreu e restabelecer os clientes o mais rápido possível (ZIDAN ABOELSOOD; KHAIRALLA, 2016), uma vez que a interrupção de energia gera impactos de ordem econômica e social (LINARES PEDRO; REY, 2013).

Sem um bom plano de restabelecimento de energia, a confiabilidade da rede de distribuição é reduzida já que a falha de um equipamento pode ocasionar a interrupção de várias UCs. Portanto, é essencial um plano de restabelecimento de energia eficaz, que consiste em: detecção de falha, isolamento e restabelecimento (COELHO A.; RODRIGUES, 2004).

Ocorrida a detecção de falha por sinalizadores e disjuntores, o alimentador é desligado. Após o desligamento, a menor parte possível que contém a falha é isolada por chaves de manobra e o alimentador é religado para suprir às demais UCs. Diferentes métodos são usados para localizar as irregularidades com a rede, incluindo o envio de uma equipe, que além de inspecionar o local realiza o reparo (VASCO JOHN; RAMLACHAN, 2008).

A roteirização visando o menor custo, com múltiplas paradas, dessa equipe de inspeção é semelhante ao problema do caixeiro-viajante com algumas ressalvas: a equipe não deve retornar ao ponto inicial, o mapa não consiste em um grafo completo e os pontos não possuem restrição de serem visitados apenas uma vez. Tratando-se de um problema de otimização combinatória, não há heurística com tempo polinomial que encontre resultado ótimo (GERACE IVAN; GRECO, 2008).

Desta forma, a proposta desse trabalho é utilizar a otimização por enxame de partículas (*PSO* - do inglês *Particle Swarm Optimization*), uma meta-heurística bioinspirada no comportamento de um bando de pássaros voando a procura de matrizes alimentícias, para encontrar percursos com custo reduzido. Metaforicamente, os pássaros representam as soluções do sistema, que trafegam pelo espaço de busca e utilizam-se de inteligências cognitiva e coletiva para atuar no próximo deslocamento a procura de soluções otimizadas (KENNEDY J.; EBERHART, 1995).

Pelo fato do *PSO* ser aplicado à funções contínuas, utilizou-se do *DPSO*, uma versão discreta da otimização por enxame de partículas (WANG ; LAN HUANG, 2003). Embora esse algoritmo não garanta ótimo global, foi possível obter resultados satisfatórios.

No Capítulo 2 temos uma contextualização do estado da arte nos planos de restabelecimento de energia. Bem como uma explicação de como é o funcionamento do *PSO* e como é composto o deslocamento das partículas. A explicação de como o *DPSO* foi adaptado do *PSO* e de como foram realizados os testes constam no Capítulo 3, já os resultados desses testes no Capítulo 4. Por fim, no Capítulo 5 temos uma discussão acerca dos resultados, conclusão e diretivas de continuidade do trabalho.

2 Desenvolvimento Teórico

2.1 Plano de restabelecimento de energia

2.1.1 Introdução

O sistema de distribuição de energia elétrica, cujas falhas compõe a maioria dos casos de falta de energia (ZIDAN ABOELSOOD; KHAIRALLA, 2016), sem um plano de restabelecimento de energia (PRE) eficaz, possui baixa confiabilidade, já que a falha de um componente pode ocasionar a interrupção de várias UCs à jusante da região de falha (COELHO A.; RODRIGUES, 2004). A detecção das falhas deve ocorrer mesmo antes de se investigar as causas, já que a interrupção da energia pode causar vários impactos de ordem econômica como: perda de produção, danos de equipamentos, deterioração de matéria prima e custos de reinício de produção. Além disso a interrupção de energia pode trazer impactos sociais como riscos à saúde e segurança (LINARES PEDRO; REY, 2013).
É necessário que o PRE seja eficiente para aumentar a confiabilidade do sistema, além de maximizar o número de unidades consumidoras restauradas e diminuir os indicadores de continuidade coletivos, duração equivalente de interrupção por unidade consumidora (DEC) e frequência equivalente de interrupção por unidade consumidora (FEC) (DISTRIBUIÇÃO, 2010). Podendo ser imprescindível para clientes críticos como hospitais, aeroportos e indústrias.

2.1.2 Indicadores de continuidade do serviço de distribuição de energia elétrica

Segundo (SALES CLAUDIO; MONTEIRO, 2014), a confiabilidade do sistema de distribuição de energia elétrica é avaliada pelos indicadores de continuidade, que podem ser divididos em duas famílias:

- Frequência de interrupções durante o período de apuração: associada às condições físicas dos ativos da concessionária. Como a configuração da rede, o grau de redundância e a idade e qualidade de manutenção dos equipamentos. Mais conhecido no Brasil como FEC;
- Duração cumulativa das interrupções ocorridas durante um determinado intervalo de tempo: associada aos recursos humanos e materiais disponíveis para a recomposição e reparos. Corresponde ao indicador DEC.

2.1.2.1 Indicadores de continuidade individuais

A duração de interrupção individual por unidade consumidora ou por ponto de conexão (DIC) é definida pela Equação 1, expressa em horas e centésimos de hora, e a frequência de interrupção individual por unidade consumidora ou por ponto de conexão (FIC) pela Equação 2, expressa o número da interrupções de por UC durante o período de apuração. Em que n é o número de interrupções da UC considerada; i corresponde ao índice de interrupções da UC, variando de 1 a n; $t(i)$ é tempo de duração da i-ésima interrupção da UC considerada ou ponto de conexão. A

apuração das interrupções ocorre mensalmente (DISTRIBUIÇÃO, 2010).

$$DIC = \sum_{i=1}^{n} t(i) \tag{1}$$

$$FIC = n \tag{2}$$

2.1.2.2 Indicadores de continuidade de conjunto de unidades consumidoras

Os indicadores de continuidade de conjunto de unidades consumidoras ou coletivos DEC, expressa em horas e centésimos de hora, e FEC, expressa em número de interrupções e centésimos do número de interrupções, podem ser definidos de acordo com as Equações 3 e 4 e indicam, respectivamente, a média de horas e o número de interrupções médio de um determinado conjunto de consumidores no período de apuração. Os índices $DIC(i)$ e $FIC(i)$ são os mesmos definidos pelas Equações 1 e 2, respectivamente, para i-ésima UC atendida em baixa ou média tensão faturadas do conjunto; Cc corresponde ao número total de UCs faturadas do conjunto no período de apuração, atendidas em baixa ou média tensão. O período de apuração das interrupções corresponde aos períodos de definição civil: mensal, trimestral e anual (DISTRIBUIÇÃO, 2010).

$$DEC = \frac{\sum_{i=1}^{Cc} DIC(i)}{Cc} \tag{3}$$

$$FEC = \frac{\sum_{i=1}^{Cc} FIC(i)}{Cc} \tag{4}$$

2.1.3 Detecção de falha

As falhas são detectadas por alarmes baseados em altas correntes e baixas tensões elétricas, que em conjunto com disjuntores sinalizam imediatamente que ocorreu uma falha através dos *links* de comunicação para o sistema de automação de redes de distribuição (KAZEMI SHAHRAM; MILLAR, 2014). Esses disjuntores desarmam quando a corrente do alimentador supera uma carga pré-determinada, com uma resposta no tempo que depende do valor medido de corrente (MILIOUDIS APOSTOLOS N.; ANDREOU, 2012). Após o desligamento do alimentador é necessário localizar o ponto em que houve falha.

Diversos métodos são utilizados para determinar o local de falha, dentre eles: a medição de impedância aparente (KEZUNOVIC, 2011), análise de circuito trifásico (YANG, 2007), inteligência artificial integrada na análise de qualidade de potência (SOUZA J.C.S.; RODRIGUES, 2001) (TEO C.Y.; GOOI, 1998) (MOMOH J.A.; DIAS, 1997), além de envio de equipes de inspeção para localizar e reparar a falha (VASCO JOHN; RAMLACHAN, 2008), sendo este último o objeto de estudo deste trabalho. Desses métodos, a medição de impedância aparente e a análise de circuito trifásico possuem o inconveniente de várias estimativas para o local de falha. A inteligência artificial requer dados elevados para treinamento, além de novos treinamentos a cada mudança na configuração do sistema (ZIDAN ABOELSOOD; KHAIRALLA, 2016).

2.1.4 Isolação

Após localizada, a menor parte possível do sistema que contém a falha é isolada por chaves de manobra de carga, ou seja, as chaves seccionadoras ou chaves remotamente controladas são abertas.
Assim que isolada a área de falha, o disjuntor é fechado para que as demais UCs sejam supridas pelo alimentador. Além disso, é realizada uma análise das cargas dos alimentadores interligados para restabelecer a maior quantidade de UCs possível (ZIDAN ABOELSOOD; KHAIRALLA, 2016).

2.1.5 Recomposição

Uma vez que isolado o setor de falha, inicia-se a restauração das unidades consumidoras sem energia visando a menor quantidade de manobras de chaves, maior número de unidades restabelecidas e em menor tempo possível (LI JUAN; MA, 2014). Define-se por setores os trechos entre chaves como por exemplo na Figura 1 que ilustra um grafo de rede de um setor, em que a chave GM é uma chave estratégica; a chave SU_2 é uma chave seccionadora unipolar; B_1, B_2, B_3, B_4 e B_5 são postes; T_4 é um transformador de distribuição. Realiza-se uma análise de carga para checar se os clientes serão parcial ou totalmente transferidos para outro alimentador interligado.
Há duas maneiras de se realizar a inspeção e recomposição (FANUCCHI; CAMILLO, 2016). Em uma delas é feita a inspeção total dos trechos entre as chaves e caso não encontrada a falha, os trechos são religados. Na outra há a recomposição trecho a trecho, em que cada setor é avaliado e caso não apresente falha é religado partindo seguidamente para o próximo trecho desempenhando o mesmo papel e assim sucessivamente (FANUCCHI; CAMILLO, 2016). Portanto, torna-se interessante estudar formas de reduzir o tempo de inspeção das equipes. Como esse roteamento se encaixa no tipo de problema combinatório, utiliza-se de heurísticas e meta-heurísticas para otimizar a solução, dentre elas o DPSO (do inglês *Discrete Particle Swarm Optimization*).

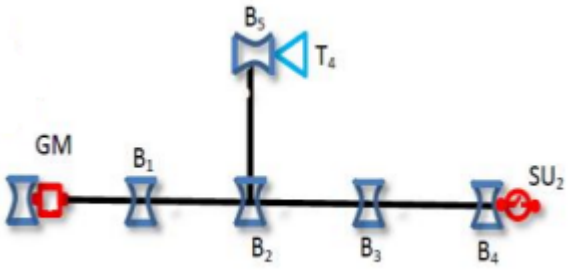

Figura 1 – Exemplo de um grafo de rede de um setor entre as chaves GM e SU_2 para ilustrar a definição de setor. Retirado de (FANUCCHI; CAMILLO, 2016)

2.2 Otimização por enxame de partículas

2.2.1 Introdução

A otimização por enxame de partículas (do inglês *PSO - Particle Swarm Optimization*) é uma meta-heurística bio-inspirada no comportamento de um bando de pássaros a procura de fontes

alimentícias.
As aves realizam buscas baseadas na cognição e na comunicação com as demais do mesmo bando. Desta forma, ao encontrar alimento, as aves disseminam informações influenciando na forma com que são feitas as próximas buscas do bando. Isto é, quando é encontrado um local com abundância de comida, as novas buscas tendem a ocorrer nos arredores daquela área.
Considerando esse comportamento no cenário evolutivo, o bando tende a atingir ótimos locais e até mesmo globais como matrizes alimentícias. Sob esse enfoque, foi elaborado um algorítimo de otimização. Sendo os termos usuais na literatura partículas e enxame para representar respectivamente as aves e o bando.
Primeiramente introduzido por (KENNEDY J.; EBERHART, 1995), o PSO é um algorítimo efetivo para problemas fortemente não lineares. Providas de velocidade, as partículas se deslocam pelo espaço de busca e suas posições são as soluções do sistema. A medida que vão adquirindo posições de interesse, ou seja, soluções otimizadas, o enxame dispõe-se nessas posições (GOMES *et al.*, 2006).

2.2.2 Problemas de otimização

Dada uma função f com domínio S, que intitula-se função de custo ou função-objetivo. Define-se por minimização: encontrar $s^* \in S/f(s^*) \leq f(s), \forall s \in S$, em que s^* é denominado mínimo. A maximização é definida como: encontrar $s^* \in S/f(s^*) \geq f(s), \forall s \in S$, intitula-se s^* como máximo. Entende-se por problema de otimização: minimizar ou maximizar a função-objetivo f e a solução ótima, ou ponto ótimo do problema, corresponde ao mínimo ou máximo, respectivamente. Espaço de busca ou domínio S é o conjunto de todas as soluções viáveis do problema, ou seja, aquelas que obedecem às restrições do problema (BECCENERI, 2008).

2.2.2.1 Ótimo local

O ótimo local s_A^* de uma função f com domínio S, para uma região $A \subset S$ é definido como $s_A^* \in A/f(s_A^*) \leq f(s), \forall s \in A$ ou $s_A^* \in A/f(s_A^*) \geq f(s), \forall s \in A$, para mínimo e máximo, respectivamente. O espaço de busca S pode conter múltiplas regiões $A_i, A_j/A_i \cap A_j = \varnothing, i \neq j$, em que os pontos definidos nessas regiões são únicos, ou seja, $x_{A_i}^* \neq x_{A_j}^*$. Além disso, não há restrições para o valor que f assume como ótimo local, implicando na possibilidade de $f(x_{A_i}^*) = f(x_{A_j}^*)$ (BECCENERI, 2008).

2.2.2.2 Ótimo global

O ótimo global s^* de uma função com espaço de busca S é definido como $s^* \in S/f(s^*) \leq f(s), \forall s \in S$ ou $s^* \in S/f(s^*) \geq f(s), \forall s \in S$, para mínimo e máximo, respectivamente (BECCENERI, 2008).

2.2.2.3 Heurísticas e meta-heurísticas

Segundo (REEVES, 1993), heurística é uma técnica que busca boas soluções (quase ótimas) a um custo computacional razoável, porém sem ser capaz de garantir que as mesmas sejam ótimas ou admissíveis, podendo não determinar a proximidade da solução admissível. Portanto, a heurística não garante a descoberta da melhor solução, ou seja, a solução ótima para determinado

problema. A flexibilidade e sua busca contínua de utilização mínima de recursos computacionais são seus principais focos. Assim, é possível que se atinja o ótimo global para um determinado problema, mesmo que a heurística não garanta esta responsabilidade e nem indique quão próxima a solução se encontra do ótimo global.

Como as demais heurísticas, a meta-heurística visa encontrar a melhor solução para um problema focando na resposta em questão, utilizando um razoável grau de recursos computacionais e flexibilidade controlada. Se diferenciam das outras heurísticas devido aos seus algoritmos serem aplicáveis a vários tipos de problemas fazendo uso de combinação de uma ou mais heurísticas para explorarem de forma conjunta o espaço de soluções (CURSO,). São agrupadas em dois grandes grupos de acordo com os critérios utilizados para a busca de soluções: busca local e busca populacional, o primeiro se baseia na exploração do espaço de soluções através de movimentos entre seus vizinhos. A cada iteração é gerada uma nova geração até que se consiga atingir uma solução desejável (KIRKPATRICK *et al.*, 1983). Exemplo de um algoritmo qualificado como meta-heurística de busca local são os algoritmos de busca tabu (CURSO,) e *simulated annealing* (KIRKPATRICK *et al.*, 1983). O segundo, consiste na busca da manutenção de um conjunto de boas soluções e, através de combinações entre elas, tentam alcançar soluções melhores. Como exemplo desta classe, pode-se citar os algoritmos genéticos (GOLBERG, 1989). Logo, a evolução da população faz com que a formação dos novos indivíduos caminhem para o ótimo, à medida que aumenta sua função de adaptação (*fitness*). Dentre essas meta-heurísticas, várias propostas de novos procedimentos meta-heurísticos vêm sendo apresentadas. Uma dessas propostas, é o algoritmo conhecido como otimização por enxame de partículas (KENNEDY J.; EBERHART, 1995).

2.2.3 PSO canônico

Divulgado por Kennedy e Eberhart em 1995 (KENNEDY J.; EBERHART, 1995), o PSO canônico ou clássico consiste em um modelo de otimização de funções contínuas não lineares.

As N partículas de um enxame, ou enxame de tamanho N, são as soluções que trafegam pelo espaço de busca de dimensão D. Esse é definido classicamente como um hipercubo da forma $[x_{min}, x_{max}]^D$. Inicialmente, as partículas são distribuídas de maneira aleatória e uniforme no intervalo $[x_{min}, x_{max}]$ para cada dimensão. Essas posições são atribuídas aos vetores p_{best_n}, como valor inicial; sendo esta a melhor posição de cada partícula e $n \in \{1, \ldots, N\}$. As posições são submetidas a uma função de custo, cujo resultado é comparado com os demais a fim de eleger o de menor valor. Uma vez encontrado o menor valor, sua posição correspondente é atribuída ao vetor g_{best}: a melhor posição do enxame.

As velocidades também são iniciadas aleatoriamente, porém na prática sua distribuição é uniforme no intervalo $[(\frac{x_{min}-x_{max}}{2}), (\frac{x_{max}-x_{min}}{2})]$, pois não é desejável que elas tendam a deixar o espaço de busca na primeira iteração.

Em seguida, as partículas têm suas velocidades e posições atualizadas pelas equações de movimento para que novamente sejam submetidas a função de custo e ter seus resultados comparados com os melhores de cada partícula (p_{best_n}) e do enxame (g_{best}), respectivamente. Esse processo ocorre iterativamente até que se alcance a precisão desejada ou o limite de iterações pré-estabelecido.

2.2.4 Teoria dos grafos

Um grafo simples G é formado por um conjunto finito não vazio de vértices, denominado V_G, e um conjunto finito de pares não ordenados de elementos distintos de V_G denominado A_G, que corresponde às arestas. Laços ou lacetes são as arestas que unem um vértice a ele próprio. Dois vértices podem ter diversas arestas unindo-os, denominadas arestas múltiplas. Desta forma, um grafo é dito simples se não possui arestas múltiplas e\ou lacetes. Um grafo em que às arestas se atribuem números não negativos é chamado de grafo com pesos ou grafo ponderado e um número atribuído à aresta a é denominado peso ou ponderação de a. Seja $V_G = \{v_0, \ldots, v_n\}$, $c(v_i, v_j)$ é o comprimento da aresta $a = \{v_i, v_j\}$, em que $v_i, v_j \in V_G$ e $i, j \in \{0, \ldots, n\}$. Em um grafo ponderado, o comprimento das arestas corresponde ao peso (LUCCHESI, 1979).

2.2.4.1 Grafos direcionados

A modelagem de certos problemas necessita de um sentido para as arestas, como por exemplo em mapas de vias publicas, em que existem vias cujo tráfego é permitido em uma única direção. Um grafo direcionado (digrafo) D é constituído por um conjunto finito não vazio de vértices V_D e em um conjunto finito de arestas orientadas A_D, também conhecidas como aros (LUCCHESI, 1979).

2.2.4.2 Grafos completos

Seja $a = \{v_1, v_2\}$ a aresta de um grafo, denomina-se que a incide em v_1 e em v_2. O grau de um vértice v é dado pelo número de arestas incidentes em v e denota-se por $g(v)$. Um grafo simples cujos vértices possuem o mesmo grau r é intitulado grafo regular de grau r. Um grafo regular com p vértices em que todos possuam o mesmo grau $p - 1$ é denominado grafo completo K_p (LUCCHESI, 1979).

2.2.4.3 Caminho e ciclo

Em um grafo G, caminho é uma sequência como: $v_0, a_1, v_1, a_2, v_2, \ldots, v_{m-1}, a_m, v_m$. Em que $v_i \in V_G$, $i \in \{0, \ldots, m\}$ e $a_j \in A_G / a_j = \{v_{j-1}, v_j\}$, $j \in \{1, \ldots, m\}$. Se $v_0 = v_m$ chama-se caminho fechado, caso contrário caminho aberto. Se todas as arestas forem distintas, classifica-se como caminho sem repetição de arestas. Analogamente, se todos os vértices forem distintos, excetuando-se v_0 e v_m caso caminho seja fechado, categoriza-se como caminho sem repetição de vértices.
Um caminho fechado sem repetição de vértices com o número de arestas $m \geq 1$ é chamado de ciclo. Qualquer par de arestas múltiplas ou lacete também é um ciclo. Caminho hamiltoniano é um caminho sem repetição de vértices que permite passar por todos os vértices de um grafo, caso esse caminho descreva um ciclo, este é denominado ciclo hamiltoniano (LUCCHESI, 1979).

2.2.4.4 Árvore

Um grafo é conexo caso não possa ser expresso como união de dois grafos e desconexo caso contrário. Portanto, um grafo desconexo pode ser expresso como a união de dois grafos conexos. Defini-se por árvore um grafo simples, conexo e sem ciclos (LUCCHESI, 1979).

2.2.4.5 Problema do menor caminho

Dado um grafo ponderado G, em que $V_G = \{v_0, \ldots, v_n\}$ é o conjunto de vértices e A_G o de arestas. O problema do menor caminho consiste em encontrar um caminho no conjunto de caminhos P de v_i para v_j, em que $v_i, v_j \in V_G$ e $i \in \{0, \ldots, j\}, j \in \{1, \ldots, n\}$, tal que $f(p)$ seja mínimo dentre todos $p \in P$. $f(p)$ é a função de peso do caminho ou função de custo, dado pela Equação 5 (LUCCHESI, 1979).

$$f(p) = \sum_{k=i}^{j} c(v_k, v_{k+1}) \tag{5}$$

2.2.5 PSO com grafo de influência

Na forma clássica todas as partículas são informadas pelas demais quando há uma posição de interesse. Esse tipo de dispersão de informação traz ao enxame um comportamento uniforme, que acelera a convergência se a função possui apenas ótimo global. Porém, quando a função possui ótimos locais a busca pelo ótimo global fica prejudicada, uma vez que o foco do enxame pode se fixar em um ótimo local, deixando de explorar os demais.

Com base nessa característica é interessante controlar o tráfego de informação. Para (CLERC, 2006a), é tradicional que modelemos uma rede de informação entre indivíduos por grafo, chamado de grafo de influência. Cada nó do grafo representa um indivíduo e cada aresta, um *link* de informação, entre dois indivíduos A e B significa "A informa B". Esses *links* são redefinidos estocasticamente a cada iteração, em que cada partícula informa outras K partículas escolhidas aleatoriamente com reposição. Isso implica que o grupo de informantes por partícula tem uma média menor que K, pois uma mesma partícula receptora de informação pode ser selecionada diversas vezes.

A título ilustrativo, uma partícula escolhe outra aleatoriamente num enxame de tamanho N, incluindo ela mesma, para informar. Logo a probabilidade de um indivíduo ser selecionado durante uma iteração na primeira escolha é de $\frac{1}{N}$ e a probabilidade de não ser informado é $1 - \frac{1}{N}$. Para as demais K escolhas dessa partícula, que são eventos independentes, a probabilidade de um indivíduo não ser selecionado é $q = \left(1 - \frac{1}{N}\right)^K$. E a probabilidade de um indivíduo ser selecionado é $p = q^c = 1 - q = 1 - \left(1 - \frac{1}{N}\right)^K$, ou seja, existem apenas duas possibilidades: a partícula ser informada ou não. Como isso ocorre de forma independente para as N partículas e a probabilidade de ocorrência se mantém constante todas as vezes, tratam-se de ensaios de Bernoulli. Sendo S o número de informantes de uma partícula, essa é uma variável aleatória discreta com $S \sim Bin(N, p)$, pois é o número de sucessos após a realização de N ensaios de Bernoulli. A probabilidade de S assumir o valor de s informantes, em que $s \in \{0, 1, \ldots, N\}$ é dada pela Equação 6. A média do número de informantes por partícula é dada pela esperança matemática na Equação 7, que associada a Equação 6, resulta na Equação 8 (GRIMMETT, 1986). Ao substituir p na Equação 8, concluímos que o número médio de informantes por partícula (K_o) a cada iteração é dado pela Equação 9. Como K é um parâmetro constante, nota-se que a medida que N cresce K_o tende a K, conforme a Demonstração 1. Pela substituição da expressão $1 - \frac{1}{N}$ por X, o limite passa a tender a 1.

$$P(S = s) = \binom{N}{s} p^s q^{N-s} \tag{6}$$

$$\mathbb{E}(S) = \sum_{s=0}^{N} sP(S = s) \tag{7}$$

$$\mathbb{E}(S) = \sum_{s=0}^{N} s\binom{N}{s}p^s q^{N-s} = Np \tag{8}$$

$$K_o = N\left[1 - \left(1 - \frac{1}{N}\right)^K\right] \tag{9}$$

Demonstração 1 $\lim_{N\to\infty} N\left[1-\left(1-\frac{1}{N}\right)^K\right] \overset{X=1-\frac{1}{N}}{=} \lim_{X\to 1}\frac{1-X^K}{1-X} \overset{L'H}{=} \lim_{X\to 1} KX^{K-1}=K$

Supondo que a cada iteração um número K de *links* de informação sejam estabelecidos aleatoriamente por cada partícula, após t iterações teremos difundido uma informação para K^t indivíduos. Generalizando, a probabilidade de um indivíduo não ser informado após t incrementos é de $\left(1-\frac{1}{N}\right)^{K^t}$. Desta forma, a probabilidade de um indivíduo ser atingido pelo menos uma vez é dada por $1-\left(1-\frac{1}{N}\right)^{K^t}$, valor que cresce rapidamente com t. Logo K não precisa ser grande para que a propagação ocorra rapidamente (CLERC, 2006a).

2.2.6 Equações de movimento

O movimento das partículas de um enxame de tamanho N para a k-ésima iteração é dado pelas Equações 10 e 11, que representam, respectivamente, a atualização da velocidade e o deslocamento em cada uma das D dimensões. Os índices n e d se referem ao número da partícula e ao d-ésimo componente de cada vetor, tal que, $n \in \{1,\ldots,N\}$, $d \in \{1,...,D\}$. O coeficiente c_1 é constante e representa a inércia, já os coeficientes c_2 e c_3 são chamados de coeficientes de confiança cognitiva e social, números gerados de forma aleatória em uma distribuição uniforme para cada componente do vetor a cada iteração, ou seja, são variáveis aleatórias contínuas tal que $c_i \sim U(0, C_i)$, $C_i \in \mathbb{R}$, $i = \{2, 3\}$. O vetor p_{best} é a melhor posição de cada partícula e g_{best} a melhor posição atingida pelo enxame. Os vetores x e v são as posições e velocidades de cada partícula. Na Equação 10, $c_1 v^k_{n,d}$ corresponde à componente inercial da velocidade, c_1 caracteriza a confiança da partícula em seu próprio movimento. A parcela $c^k_{2_d}(p_{best_{n,d}} - x^k_{n,d})$ representa a componente cognitiva, em que c_2 indica a confiança na melhor performance da própria partícula. A componente social consta na terceira parcela: $c^k_{3_d}(g_{best_d} - x^k_{n,d})$, em que c_3 determina a confiança no melhor informante.

$$v^{k+1}_{n,d} = \underbrace{c_1 v^k_{n,d}}_{\text{Inercial}} + \underbrace{c^k_{2_d}(p_{best_{n,d}} - x^k_{n,d})}_{\text{Cognitiva}} + \underbrace{c^k_{3_d}(g_{best_d} - x^k_{n,d})}_{\text{Social}} \tag{10}$$

$$x^{k+1}_{n,d} = x^k_{n,d} + v^{k+1}_{n,d} \tag{11}$$

2.2.7 Algoritmo PSO

Algoritmo 1: Otimização por enxame de partículas - PSO

```
1  início
2      para cada partícula ∈ enxame faça
3          iniciar posição;
4          iniciar velocidade;
5      fim
6      repita
7          Gerar links de informação;
8          para cada particula ∈ enxame faça
9              v_{p_best} ← calcular função de custo da partícula;
10             atualizar p_best da partícula;
11             se algum valor ∈ v_{p_best} < v_{g_best} então
12                 g_best ← p_best;
13             fim
14             atualizar velocidade da partícula;
15             atualizar posição da partícula;
16         fim
17     até que chegue a condição de parada;
18 fim
```

3 Materiais e métodos

3.1 Roteamento das equipes de inspeção de rede e o problema do caixeiro viajante

3.1.1 Problema do caixeiro-viajante

O problema do caixeiro-viajante, mais conhecido como TSP (do inglês *Travel salesman problem*), é representativo de uma gama de problemas de otimização combinatória como controle de cobertura para rede de sensores sem fio (DU *et al.*, 2014), programação de recursos de máquina virtual (HUANG; YANG; DING, 2013), problema do roteamento de veículos que possuem capacidade limitada de carga (KAO; CHEN, 2013), dentre eles a proposta de roteamento de equipes de inspeção de redes de distribuição de energia. Pertencente à classe *NP-Complete*, não há um algoritmo polinomial que encontre o resultado ótimo, este encontrado apenas em tempo fatorial $(N-1)!$.
Na forma original do TSP, um mapa das cidades é dado a um vendedor e ele deve visitar todas apenas uma vez e retornar a cidade de origem, desempenhando o menor trajeto possível. O mapa consiste em um grafo finito completo e o objetivo é encontrar um ciclo hamiltoniano (GERACE IVAN; GRECO, 2008).

3.1.2 Inspeção de rede

Muito semelhante ao TSP, a proposta de roteamento das equipes de inspeção consiste em fiscalizar pontos intermediários às duas chaves de manobra que isolam o setor, de forma que a equipe percorra a menor distância possível. Porém com ressalvas:

- a equipe não deve retornar ao ponto inicial, isto é, deve partir de uma chave de manobra, inspecionar os pontos de interesse e chegar na outra chave que isola o setor;
- o mapa não consiste em um grafo completo, pois as equipes trafegam num grafo de vias urbanas georreferenciado com os pontos de inspeção incorporados. Este último denominado como grafo de rede;
- como o deslocamento é feito por vias urbanas, os pontos de inspeção não terão a restrição de serem visitados apenas uma vez. Visando o menor percurso de um grafo direcionado, pode ser necessário passar novamente por um mesmo ponto.

3.2 DPSO aplicado ao roteamento de equipes de inspeção de rede

Para o caso do roteamento de equipes de inspeção de rede, o algoritmo PSO canônico não deve ser utilizado, pois se tratando de um otimizador de funções contínuas oferece resultados inválidos. Como alternativa para casos discretos há o DPSO. Para manter a mesma estrutura do algoritmo 1, as operações e os elementos das equações 10 e 11 foram redefinidos (CLERC, 2004).

3.2.1 Posição

Seja G um grafo de vias ponderado, já com o grafo de rede incorporado, no qual $G = \{N_G, A_G\}$, em que N_G são os nós e A_G são as arestas. Os M nós são rotulados por números naturais, ou seja, $m \in \{0, 1, \ldots, M-1\}$.
A posição de cada partícula consiste em um vetor cujos D elementos, ou dimensões, são os nós a serem inspecionados.

3.2.1.1 Subtração de posições

A subtração de posições resulta em velocidade, desta forma essa operação detecta todas as trocas de elementos necessárias para levar de uma posição a outra, alojando-as em um vetor velocidade. Exemplificando a operação, dados $x_1 = [5, 9, 14, 20, 19]$, $x_2 = [9, 20, 14, 5, 19]$, temos v, tal que $v = x_1 - x_2 = [(9, 5), (20, 9)]$. Desta forma, entende-se que v é a velocidade que desloca a partícula da posição x_2 para x_1.
A subtração de posições consta no algoritmo 2, em que D é a dimensão dos vetores de posição (posição$_1$ e posição$_2$) e velocidade, definido na subseção 3.2.2, é um vetor cujos elementos são duplas que representam uma troca entre dois nós. Além disso, utilizam-se duas funções: a *AntiSwap*, definida no algoritmo 3, retorna a primeira dupla de troca detectada entre dois vetores de posição e a *Swap*, contida no algoritmo 4, realiza a troca de dois elementos. A dupla do tipo $(\text{enumerar}(\text{posição}), \text{posição})$ (algoritmo 3-linha 2) consiste como primeiro componente na função enumerar aplicada ao vetor posição, essa função retorna o índice de cada elemento do vetor e como segundo componente, o elemento do vetor posição associado ao índice retornado pela função enumerar. Desta forma, a dupla para o primeiro elemento do vetor x_1 do exemplo acima seria $(0, 5)$, para o segundo $(1, 9)$.

Algoritmo 2: Subtração de posições

```
   Entrada: D, posição_1, posição_2
   Saída: velocidade
1  início
2      para i ← 0 até D faça
3          Acrescente a velocidade[ AntiSwap(posição_1, posição_2)];
4          Swap(posição_1, velocidade[i]);
5      fim
6      Filtre os elementos ∅ de velocidade;
7  fim
```

Algoritmo 3: AntiSwap

```
   Entrada: posição_1, posição_2
   Saída: dupla
1  início
2      para (n, i) ∈ (enumerar(posição_1), posição_1) faça
3          se i ≠ posição_2[n] então
4              dupla ← (posição_1[n],posição_2[n]);
5          fim
6      fim
7  fim
```

3.2.2 Velocidade

A velocidade v de cada partícula foi definida de forma que troque, dois a dois, os elementos do vetor posição: $v = ((i_k, j_k))$, i_k, $j_k \in N_G$, $k \in \{1, \ldots, \|v\|\}$. Em que a dupla (i_k, j_k) pode ser lida da forma "troque o elemento i_k pelo j_k", conhecido na literatura por *Swap* (WANG, 2007). O algoritmo 4 realiza essa ação e a função enumerar retorna o índice de cada elemento do vetor posição. Essa troca também pode ser realizada pelo índice dos elementos do vetor, em que o par (i, j) representa a troca do elemento de índice i, pelo elemento de índice j (WANG ; LAN HUANG, 2003). A $\|v\|$ (norma de v) é a dimensão do vetor velocidade ou número de *Swaps* a serem realizados, ou seja, o vetor velocidade é uma coleção de *Swaps*.

Algoritmo 4: Swap

```
   Entrada: posição, dupla
   Saída: posição
1  início
2      se dupla ≠ ∅ então
3          (N_A, N_B) ← dupla;
4          para (n, i) ∈ (enumerar(posição), posição) faça
5              se i = N_A então
6                  posição[n] ← N_B;
7              fim
8              se i = N_B então
9                  posição[n] ← N_A;
10             fim
11         fim
12     fim
13 fim
```

3.2.2.1 Adição de velocidades

A adição de velocidades resume-se na incorporação ordenada ou concatenação dos elementos dos vetores velocidade por outro vetor. Como exemplo temos $v_1 = ((3, 4), (5, 7), (4, 9))$ e $v_2 = ((6, 7), (2, 5))$, a velocidade resultante será $v_r = v_1 + v_2 = ((3, 4), (5, 7), (4, 9), (6, 7), (2, 5))$. Enfatiza-se que essa operação não é comutativa, uma vez que a ordem em que as trocas são realizadas está relacionada à resultante (CLERC, 2006b).

3.2.2.2 Multiplicação por escalar

A multiplicação da velocidade v pelo escalar c pode ser dividida em 3 casos:

1. $c = 0$: a multiplicação deve retornar $cv = \varnothing$;
2. $0 < c \leq 1$: devemos truncar o número de duplas do vetor velocidade no maior inteiro menor que o produto $c\|v\|$, isto é, $cv = (v_1, \ldots, v_{\lfloor c\|v\| \rfloor})$;
3. $c > 1$: podemos escrever c como um inteiro k mais um número real $j < 1$, tal que $cv = \underbrace{v + v + \cdots + v}_{k\ vezes} + jv$, em que a adição de velocidades já foi definida e a parcela jv é

correspondente à multiplicação por escalar entre 0 e 1.

A operação de multiplicação é ilustrada pelo algoritmo 5, em que é utilizado um vetor auxiliar para receber os elementos do vetor velocidade e à variável t é atribuído o valor da norma de auxiliar. Além disso, a função $\lfloor x \rfloor$ retorna o maior inteiro menor que x, ou seja, $\lfloor x \rfloor = \max\{m \in \mathbb{Z} \mid m \leq x\}$. Na linha 11, do algoritmo 5, utilizou-se a operação módulo, que retorna o resto da divisão de c por $\lfloor c \rfloor$, ou seja, a parte decimal do número real c. As linhas 13 e 15 contém o operador "$+$", que no caso refere-se à concatenação de vetores velocidade, já definida na subseção 3.2.2.1. O vetor auxiliar$[0$ **até** $(\lfloor ct \rfloor)]$, representa o vetor auxiliar com o número de elementos truncado pelo maior inteiro menor que o produto ct.

Algoritmo 5: Multiplicação

Entrada: c, velocidade
Saída: velocidade

```
1  início
2      auxiliar ← velocidade;
3      t ← ‖auxiliar‖;
4      se c = 0 então
5          velocidade ← ∅;
6      fim
7      se 0 < c ≤ 1 então
8          velocidade ← auxiliar[0 até (⌊ct⌋)];
9      fim
10     se c > 1 então
11         cl ← c  mod ⌊c⌋;
12         para i ← 0 até (⌊c⌋ − 1) faça
13             velocidade ← velocidade + auxiliar ;
14         fim
15         velocidade ← velocidade + auxiliar[0 até (⌊ct⌋)];
16     fim
17 fim
```

3.2.3 Atualizar velocidade

Para atualizar a velocidade, utiliza-se a Equação 10 com as operações redefinidas. O algoritmo 6 mostra como a velocidade é alterada, a definição dos parâmetros c_1, c_2, c_3, D, p_{best} e g_{best} constam na subseção 2.2.6.

Algoritmo 6: Atualizar velocidade

Entrada: D, c_1, c_2, c_3, g_{best}, p_{best}, posição, velocidade
Saída: velocidade

```
1  início
2      velocidade inercial ← Multiplicação(c1, velocidade);
3      velocidade cognitiva ← Multiplicação(c2, Subtração de posições(D, p_best, posição));
4      velocidade social ← Multiplicação(c3, Subtração de posições(D, g_best, posição));
5      velocidade ← velocidade inercial + velocidade cognitiva + velocidade social;
6  fim
```

3.2.4 Deslocamento

O deslocamento é feito de acordo com o algoritmo 7. A Equação 11 é mantida, porém o algoritmo foi adequado aos casos discretos. A soma do vetor posição à velocidade é feita através do laço de controle atrelado a função *Swap* para cada elemento do vetor velocidade, portanto a equação é mantida conceitualmente.

Algoritmo 7: Deslocamento

Entrada: posição, velocidade
Saída: posição

```
1 início
2     para i ← 0 até ||velocidade|| faça
3         Swap(posição, velocidade[i]);
4     fim
5 fim
```

3.2.5 Algoritmo DPSO completo

O algoritmo 8 contém o DPSO aplicado ao roteamento de uma equipe de inspeção. O vetor chaves contém 2 elementos, sendo o primeiro a chave da qual a equipe partirá para inspeção e o segundo, a chave de chegada. O vetor inspeção contém os pontos a serem inspecionados. Ao parâmetro maxit é conferido o número máximo de iterações. Se atribui à entrada chamada grafo o grafo de vias com o de rede incorporado. Os demais parâmetros foram previamente definidos.
Na linha 3, o vetor enxame recebe elementos do tipo partícula, que ao se iniciarem embaralham os elementos de inspeção e os alojam em seus respectivos vetores posição.
As linhas de 9 a 20 definem o grafo de influência, em que a função Inteiro aleatório(a, b) retorna um número inteiro aleatório em uma distribuição uniforme no intervalo $[a, b]$.
A linha 23 contém o cálculo da função de custo para posição de cada partícula do enxame já aplicada à função que retorna a rota (linha 22).
As linhas entre 24 e 27 atualizam o vetor p_{best} para cada partícula do enxame, além de manter o respectivo valor da função de custo no vetor ep_{best}. Já as linhas no intervalo de 28 a 34 atualizam o vetor g_{best}, o número eg_{best} e a menor rota encontrada. Por fim, as equações de movimento são utilizadas entre as linhas 36 e 39.

Algoritmo 8: DPSO

Entrada: chaves, inspeção, grafo, maxit, N, D, K, c_1, c_2, c_3
Saída: g_{best}, eg_{best}, rota

início
  para $n \leftarrow 0$ **até** N **faça**
    enxame[n] $\leftarrow$ partícula(inspeção);
  fim
  $ep_{best} \leftarrow 1$;
  $eg_{best} \leftarrow 1$;
  $i \leftarrow 0$;
  repita
    para $n \leftarrow 0$ **até** N **faça**
      para $m \leftarrow 0$ **até** N **faça**
        links$[m][n] \leftarrow 0$;
      fim
      links$[n][n] \leftarrow 1$;
    fim
    para $m \leftarrow 0$ **até** N **faça**
      para $k \leftarrow 0$ **até** K **faça**
        $n \leftarrow$ Inteiro aleatório$(0, N-1)$;
        links$[m][n] \leftarrow 1$;
      fim
    fim
    para $j \leftarrow 0$ **até** N **faça**
      rotas$[j] \leftarrow$ Função de rota(enxame$[j]$.posição, grafo, chaves);
      $L[j] \leftarrow$ Função de custo(rotas$[j]$, grafo);
      se $L[j] < ep_{best}$ *ou* $ep_{best} = 1$ **então**
        $p_{best}[j] \leftarrow$ enxame$[j]$.posição;
        $ep_{best}[j] \leftarrow L[j]$;
      fim
      para $g \leftarrow 0$ **até** N **faça**
        se $(L[j] < eg_{best}$ *ou* $eg_{best} = 1)$ *e* (*links*$[g][j] = 1)$ **então**
          $g_{best} \leftarrow$ enxame$[j]$.posição;
          $eg_{best} \leftarrow L[j]$;
          rota $\leftarrow$ rotas$[j]$;
        fim
      fim
    fim
    para $j \leftarrow 0$ **até** N **faça**
      Atualizar velocidade$(D, c_1, c_2, c_3, g_{best}, p_{best}$, enxame$[j]$.posição, enxame$[j]$.velocidade$)$;
      Deslocamento(enxame$[j]$.posição, enxame$[j]$.velocidade$)$;
    fim
  até *que* $i =$ *maxit*;
fim

3.2.6 Função de custo

A Função de custo no algoritmo 9 calcula a distância baseada na soma algébrica das arestas ponderadas de cada trajeto, o valor retornado é denominado custo. Esses trajetos são dados pela concatenação do primeiro elemento do vetor chaves, seguido do respectivo vetor posição e do último elemento de chaves. A Função de rota (algoritmo 10) utiliza o algoritmo de Dijkstra para encontrar o menor caminho entre dois pontos sucessivos de cada trajeto contido no vetor rotas.

Algoritmo 9: Função de custo

```
Entrada: trajeto, grafo
Saída: custo
1  início
2      custo ← 0;
3      para x ← 0 até (||trajeto|| − 1) faça
4          custo ← custo + Tamanho aresta(trajeto[x], trajeto[x + 1]);
5      fim
6  fim
```

Algoritmo 10: Função de rota

```
Entrada: posição, grafo, chaves
Saída: trajeto
1  início
2      auxiliar ← chaves[0] + posição + chaves[1];
3      i ← 0;
4      para x ← 0 até (||auxiliar|| − 1) faça
5          trajeto ← trajeto + Menor caminho Dijkstra(grafo, auxiliar[x], auxiliar[x + 1]);
6      fim
7      repita
8          se trajeto[i] = trajeto[i + 1] então
9              Deletar(trajeto[i]);
10         fim
11         senão
12             i ← i + 1;
13         fim
14     até que i < (||trajeto|| − 1);
15 fim
```

3.2.6.1 Algoritmo de Dijkstra

O algoritmo de Dijkstra foi usado para encontrar o menor caminho entre dois pontos sucessivos do vetor inspeção. A complexidade desse algoritmo é $O(n^2)$, em que n é o número de nós do grafo (ZAMBONI; PAMBOUKIAN; BARROS, 2006). Esse algoritmo é desenvolvido em uma árvore de caminhos mínimos, dado que caso haja arestas múltiplas ou laços, esses são transformados em grafos simples deixando apenas as arestas de menor peso (BARROS; PAMBOUKIAN; ZAMBONI, 2007). Além de ter um bom desempenho (ATZINGEN *et al.*, 2012), o algoritmo pode ser facilmente implementado através da biblioteca *networkx* pela função *nx.dijkstra_path* (HAGBERG; SCHULT; SWART, 2008).

3.3 Testes

Os testes foram realizados em um computador com processador de 64 bits do tipo Intel(R) Core(TM) i3-4005U com *clock* interno de 1,70 GHz e memória cache de 3Mb, 4Gb de RAM e sistema operacional *Windows* 10. A linguagem utilizada foi *Python* 2.7. O trabalho foi elaborado no ambiente de desenvolvimento científico Anaconda (ANACONDA, 2016), que facilitou a interface com o sistema operacional *Windows* 10.
Para cada ponto coletado, em cada gráfico, realizou-se 80 repetições do algoritmo com foco no tempo de execução e no custo do melhor caminho encontrado.

3.3.1 Mapa - Grafo de vias

O grafo utilizado para os testes é do tipo *lattice* não direcionado com arestas de tamanho unitário, que por simplicidade representam vias de duas mãos com o mesmo custo. Na Figura 2 temos um exemplo desse tipo de grafo de tamanho 8x8, este mesmo grafo foi utilizado para os testes de número de iterações, tamanho do enxame, quantidade de informantes por partícula e número de elementos a serem inspecionados. O grafo foi construído através do gerador de grafos grid_2d_graph da biblioteca networkx (HAGBERG; SCHULT; SWART, 2008). Essa biblioteca da linguagem *Python* proporciona a criação, manipulação e estudo da estrutura, dinâmica e função de redes complexas. A biblioteca matplotlib (HUNTER, 2007), também pertencente à linguagem *Python*, foi empregada na produção das figuras e gráficos.

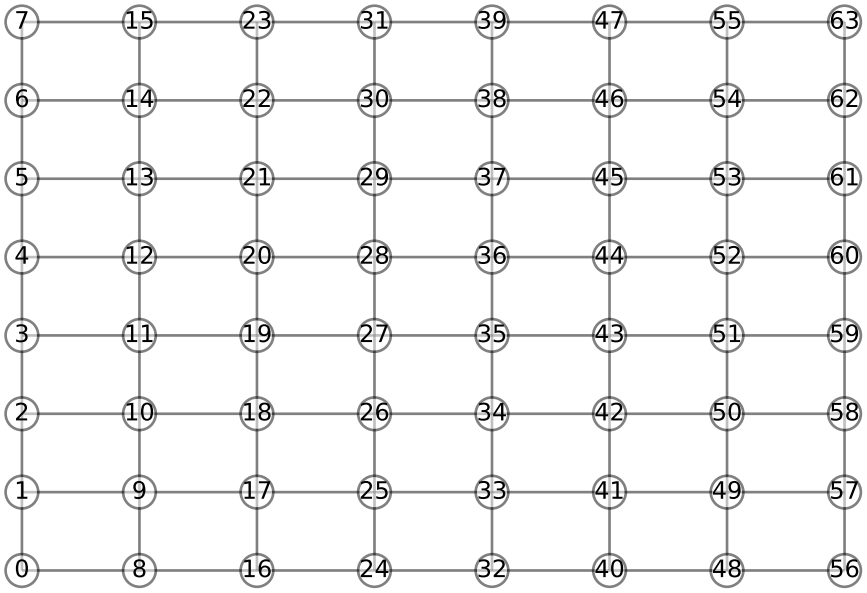

Figura 2 – Grafo do tipo *lattice* 8x8 utilizado para os testes do roteamento de uma equipe de inspeção de rede simbolizando vias de duas mãos com o mesmo custo, unitário.

3.3.2 Parâmetros constantes

Os coeficientes de confiança foram mantidos constantes durante os testes, em que $c_1 = 0.689343$, $c_2 = 1.42694$ e $c_3 = 1.42694$. Esses valores foram retirados de (CLERC, 2006a), já que o autor os indica para acelerar a convergência.

3.3.3 Número de iterações

Para os testes com o número de iterações, o tamanho do enxame utilizado foi $N = 20$ com 4 informantes por partícula ($K = 4$). O vetor chaves contém como elementos os nós 2 e 62, do grafo da Figura 2, como chave inicial e final, respectivamente. o vetor inspeção foi definido como $[17, 23, 50, 36, 12, 56]$.
O teste foi iniciado com 11 iterações e terminou com 201, com acréscimo de 10 iterações a cada ponto. Para cada um dos 20 pontos obtidos foram realizadas 80 repetições e calculou-se as médias e desvios padrão dos respectivos custos dos g_{Best} (eg_{Best}) e duração de execução.

3.3.4 Número de partículas (N)

Os parâmetros empregados nesse teste foram o número de iterações fixado em 100, número de informantes por partículas $K = 4$ e o grafo da Figura 2. Os vetores chaves $= [2, 62]$ e inspeção $= [17, 23, 50, 36, 12, 56]$ foram mantidos como no teste do número de iterações.
O número de partículas se inicia em 6, em que foram realizadas 80 repetições, então foram acrescidas 2 partículas e outras 80 repetições foram executadas. Esse processo ocorreu até o enxame atingir o tamanho de 44 e as médias e desvios padrão foram calculados para cada ponto para os custos, eg_{Best}, e duração de execução.

3.3.5 Número de informantes por partícula (K)

O número de informantes por partícula (K) foi variado de 1 a 20 com incremento de 1 informante a cada ponto coletado. Para esses pontos foram realizadas 80 repetições e calculou-se as médias e desvios padrão para os eg_{Best} e intervalos de tempo de execução.
O tamanho do enxame foi fixado em 20 partículas, foram feitas 100 iterações por repetição e foi empregado o grafo da Figura 2. Os vetores chaves $= [2, 62]$ e inspeção $= [17, 23, 50, 36, 12, 56]$ foram conservados.

3.3.6 Número de pontos de inspeção

Os parâmetros mantidos nesse teste foram: $N = 20$, $K = 4$ e número de iterações igual a 100. O grafo da Figura 2 foi utilizado como grafo de vias e o vetor chaves $= [2, 62]$.
O número de elementos do vetor de inspeção foi incrementado de 1 em 1, em que os nós foram obtidos de forma aleatória no conjunto de nós do grafo. Esse número se inicia em 7 elementos, $[17, 23, 50, 36, 12, 56, 6]$, finalizando o teste com 26. Para cada incremento de elementos do vetor inspeção foram realizados 80 repetições e foram calculadas as médias e desvios padrão dos eg_{Best} e tempos de execução.

3.3.7 Representação gráfica do menor caminho encontrado

Para essa etapa realizou-se 80 repetições com $N = 20$, $K = 4$, 100 iterações, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$. O menor caminho com maior número de repetição foi selecionado. Após a seleção do menor caminho, traçou-se o trajeto de vermelho no grafo de vias da Figura 2, em que os pontos de inspeção são sinalizados de azul e as chaves por quadrados verdes. Vale ressaltar que caso uma aresta seja percorrida mais de uma vez, sua cor terá uma tonalidade mais escura.

4 Resultados e discussão

Para cada teste da seção 3.3, traçou-se dois gráfico com os resultados, em um deles foi obtido o custo do percurso de uma equipe de inspeção e no outro o tempo de execução do código, com suas respectivas médias ($\bar{x}$) e desvios padrão ($\sqrt{s^2}$).

4.1 Número de iterações

O custo das posições g_{Best} em função do número de iterações da Tabela 1 é ilustrado pela Figura 3. O valor médio após estabilização, a partir do terceiro ponto, é de 29,56 com desvio padrão médio de 0,96. O custo se estabilizou rapidamente com o aumento do número de iterações, pois a partir do terceiro ponto, 31 iterações, o gráfico mostra um comportamento tendendo a uniformidade. Na Figura 4, o tempo de execução apresentou comportamento linear em relação ao número de iterações, cujos resultados constam na Tabela 2.
O algoritmo *DPSO* mostrou-se validado pela Figura 3 (YIN, 2004), em que esse obtém valores menores a medida que aumentamos o número de iterações, até se estabilizar num valor mínimo. Porém, o tempo de execução cresce linearmente com o número de iterações (Figura 4). Tornando-se necessário ponderar sobre esse parâmetro se é desejável uma solução mais otimizada em detrimento do tempo de execução.

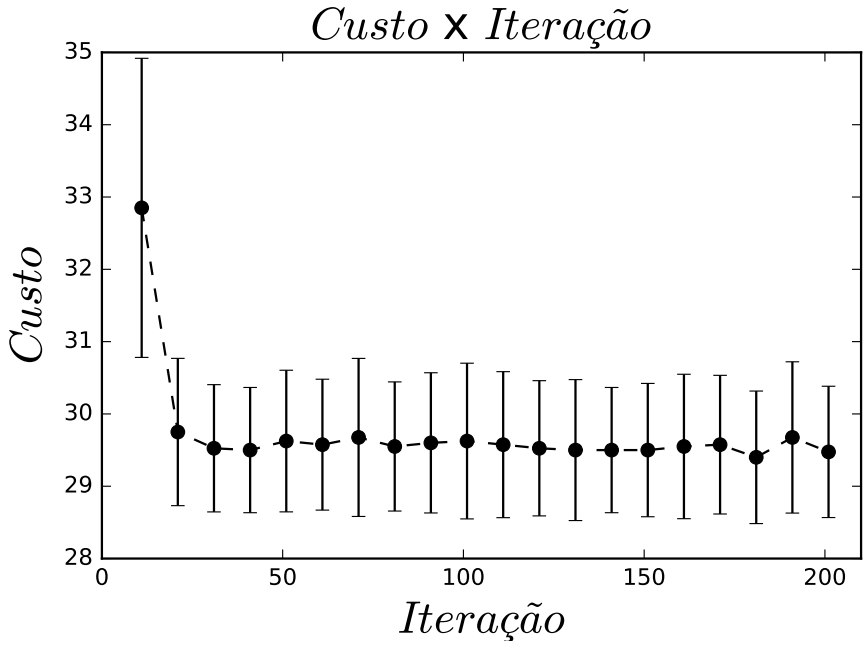

Figura 3 – Custo das posições g_{Best} em função do número de iterações. Com $N = 20$, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 10 iterações de um ponto para o próximo.

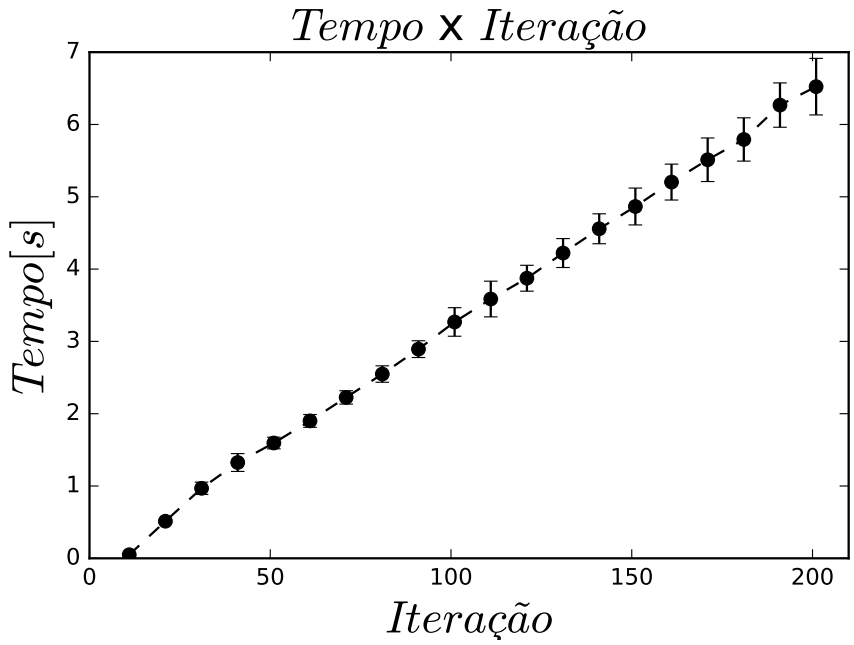

Figura 4 – Tempo de execução em função do número de iterações. Com $N = 20$, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 10 iterações de um ponto para o próximo.

Tabela 1 – Médias e desvios padrão dos custos das posições g_{Best} em função do número de iterações. Utilizando-se $N = 20$, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 10 iterações de um ponto para o próximo.

Ponto	Número de iterações	$\bar{x}$	$\sqrt{s^2}$
1	11	32,85	2,07
2	21	29,75	1,02
3	31	29,52	0,88
4	41	29,50	0,87
5	51	29,62	0,98
6	61	29,58	0,91
7	71	29,68	1,09
8	81	29,55	0,89
9	91	29,60	0,97
10	101	29,62	1,08
11	111	29,58	1,01
12	121	29,52	0,94
13	131	29,50	0,97
14	141	29,50	0,87
15	151	29,50	0,92
16	161	29,55	1,00
17	171	29,58	0,96
18	181	29,40	0,92
19	191	29,68	1,05
20	201	29,48	0,91

Tabela 2 – Médias e desvios padrão do tempo de execução em função do número de iterações. Utilizando-se $N = 20$, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 10 iterações de um ponto para o próximo.

Ponto	Número de iterações	$\bar{x}$	$\sqrt{s^2}$
1	11	0,05	0,01
2	21	0,51	0,03
3	31	0,97	0,09
4	41	1,32	0,12
5	51	1,59	0,08
6	61	1,90	0,09
7	71	2,23	0,09
8	81	2,55	0,11
9	91	2,89	0,12
10	101	3,27	0,20
11	111	3,59	0,25
12	121	3,87	0,18
13	131	4,22	0,20
14	141	4,56	0,21
15	151	4,87	0,25
16	161	5,20	0,25
17	171	5,51	0,30
18	181	5,79	0,30
19	191	6,27	0,31
20	201	6,52	0,39

4.2 Número de partículas (N)

Na Figura 5 temos o custo das posições g_{Best} em relação ao número de partículas do enxame. Notam-se elevados desvio padrão e média para o primeiro ponto, cujos valores foram de aproximadamente 5,41 e 39,20, respectivamente. A partir do décimo segundo ponto ($N = 28$) o resultado se estabiliza de acordo com a Tabela 3. A Tabela 4 contém os resultados relacionados ao tempo de execução. Nota-se pela Figura 6 que o tempo de execução apresenta crescimento linear com o número de partículas empregado.

O tamanho do enxame influenciou de forma semelhante ao número de iterações na obtenção dos mínimos locais e globais. De forma que quanto maior o enxame, menores os custos obtidos até estabilizar-se no valor mínimo. Entretanto, o tempo de execução também cresce de forma linear com o número de partículas, sendo necessário encontrar um ponto de equilíbrio.

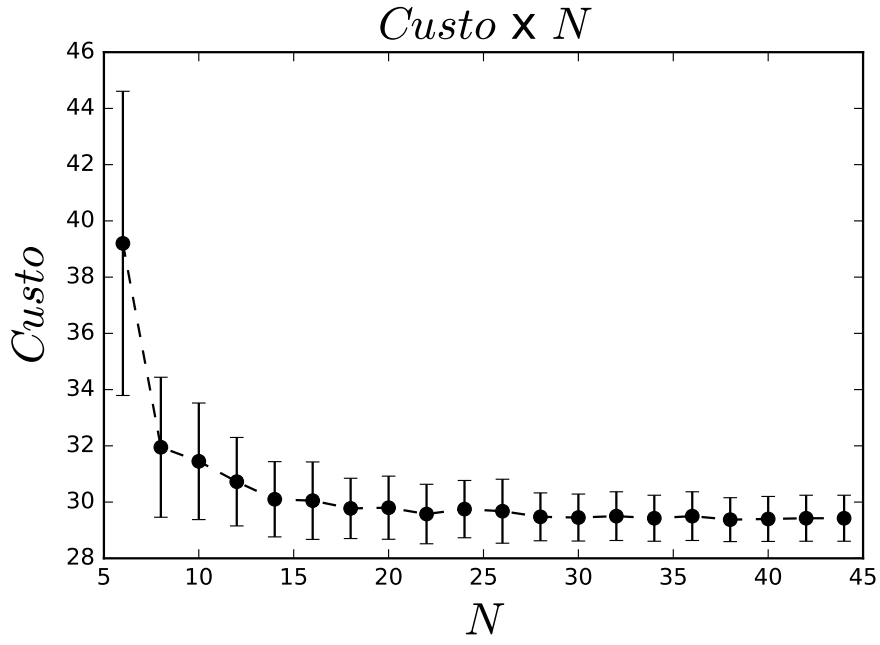

Figura 5 – Custo das posições g_{Best} em função do número de partículas do enxame. Com número de iterações igual a 100, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 2 partículas de um ponto para o próximo.

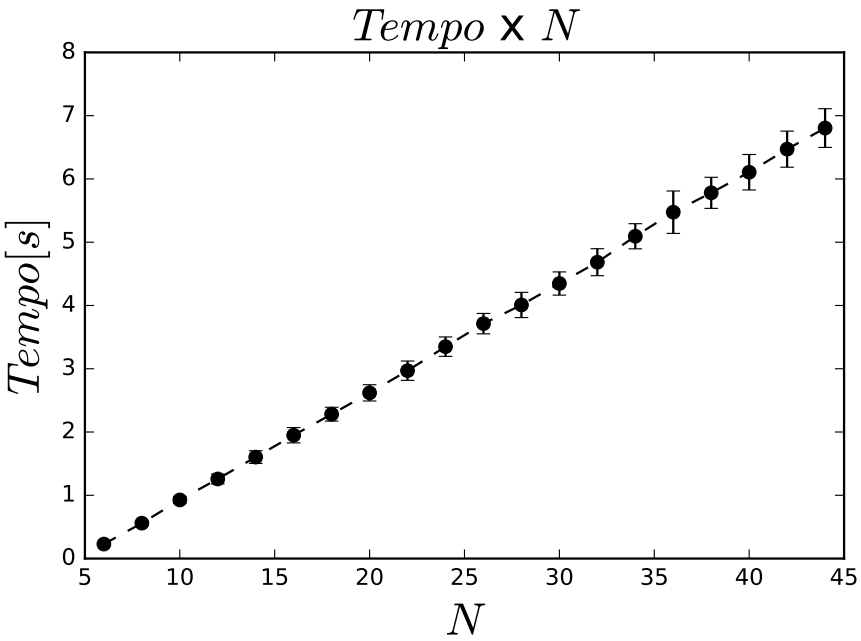

Figura 6 – Tempo de execução em função do número de partículas do enxame. Com número de iterações igual a 100, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 2 partículas de um ponto para o próximo.

Tabela 3 – Médias e desvios padrão dos custos das posições g_{Best} em função do número de partículas do enxame. Com número de iterações igual a 100, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 2 partículas de um ponto para o próximo.

Ponto	N	$\bar{x}$	$\sqrt{s^2}$
1	6	39,20	5,41
2	8	31,95	2,49
3	10	31,45	2,07
4	12	30,72	1,57
5	14	30,10	1,34
6	16	30,05	1,38
7	18	29,78	1,07
8	20	29,80	1,12
9	22	29,58	1,06
10	24	29,75	1,02
11	26	29,68	1,14
12	28	29,48	0,85
13	30	29,45	0,84
14	32	29,50	0,87
15	34	29,42	0,82
16	36	29,50	0,87
17	38	29,38	0,78
18	40	29,40	0,80
19	42	29,42	0,82
20	44	29,42	0,82

Tabela 4 – Médias e desvios padrão do tempo de execução em função do número de partículas do enxame. Com número de iterações igual a 100, $K = 4$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 2 partículas de um ponto para o próximo.

Ponto	N	$\bar{x}$	$\sqrt{s^2}$
1	6	0,23	0,03
2	8	0,56	0,04
3	10	0,93	0,06
4	12	1,26	0,08
5	14	1,60	0,10
6	16	1,95	0,12
7	18	2,28	0,11
8	20	2,62	0,13
9	22	2,97	0,15
10	24	3,35	0,15
11	26	3,71	0,16
12	28	4,01	0,20
13	30	4,35	0,18
14	32	4,68	0,21
15	34	5,09	0,20
16	36	5,47	0,34
17	38	5,78	0,25
18	40	6,11	0,28
19	42	6,47	0,29
20	44	6,80	0,31

4.3 Número de informantes por partícula (K)

A Tabela 5 contém os resultados referentes aos custos em função de K, ilustrados pela Figura 7. Na Tabela 6 temos o tempo de execução em função de K e seu comportamento pode ser melhor observado pela Figura 8. As médias e desvios padrões se mantiveram na mesma faixa para todos os pontos, tanto no custo quanto no tempo de execução.

O número de informantes por partícula não teve influência expressiva nos testes em virtude do número de iterações ser elevado e a propagação da informação ocorrer rapidamente como discutido na subseção 2.2.5.

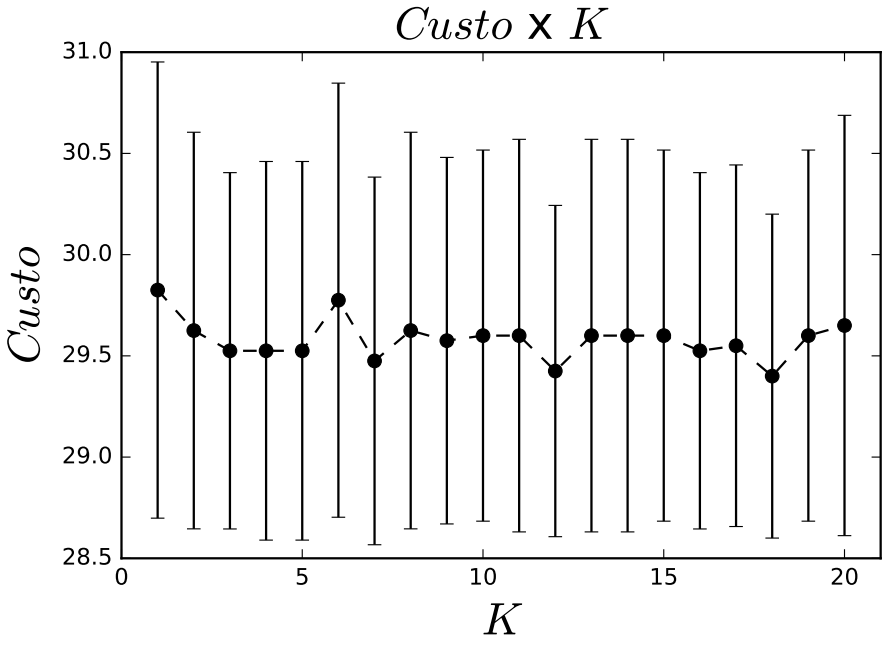

Figura 7 – Custo das posições g_{Best} em função do número de informantes por partículas do enxame. Com número de iterações igual a 100, $N = 20$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 1 partícula de um ponto para o próximo.

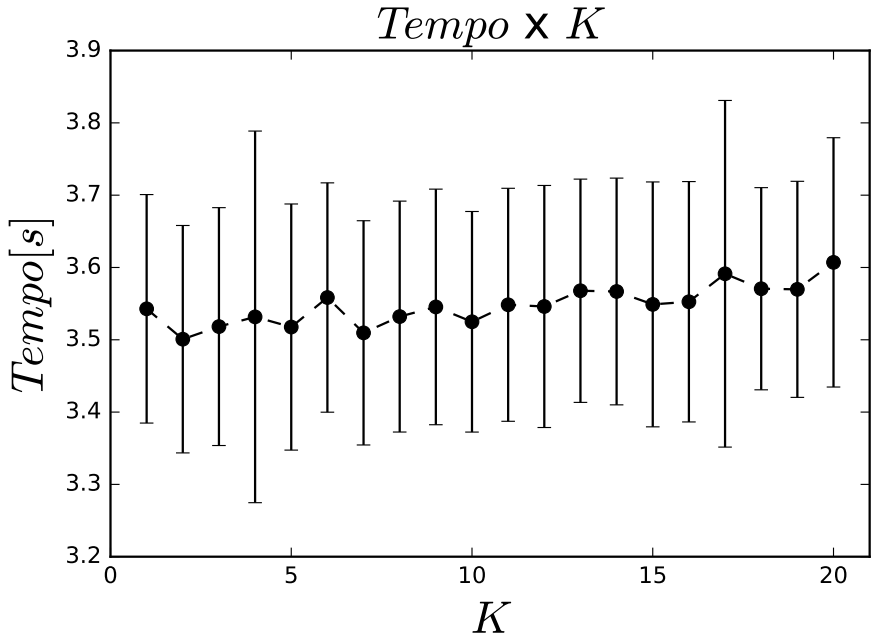

Figura 8 – Tempo de execução em função do número de informantes por partículas do enxame. Com número de iterações igual a 100, $N = 20$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 1 partícula de um ponto para o próximo.

Tabela 5 – Médias e desvios padrão dos custos das posições g_{Best} em função do número de informantes por partículas do enxame. Com número de iterações igual a 100, $N = 20$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 1 partícula de um ponto para o próximo.

Ponto	K	$\bar{x}$	$\sqrt{s^2}$
1	1	29,82	1,13
2	2	29,62	0,98
3	3	29,52	0,88
4	4	29,52	0,94
5	5	29,52	0,94
6	6	29,78	1,07
7	7	29,48	0,91
8	8	29,62	0,98
9	9	29,58	0,91
10	10	29,60	0,92
11	11	29,60	0,97
12	12	29,42	0,82
13	13	29,60	0,97
14	14	29,60	0,97
15	15	29,60	0,92
16	16	29,52	0,88
17	17	29,55	0,89
18	18	29,40	0,80
19	19	29,60	0,92
20	20	29,65	1,04

Tabela 6 – Médias e desvios padrão dos tempos de execução em função do número de informantes por partículas do enxame. Com número de iterações igual a 100, $N = 20$, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$, 80 repetições por ponto, incremento de 1 partícula de um ponto para o próximo.

Ponto	K	$\bar{x}$	$\sqrt{s^2}$
1	1	3,54	0,16
2	2	3,50	0,16
3	3	3,52	0,16
4	4	3,53	0,26
5	5	3,52	0,17
6	6	3,56	0,16
7	7	3,51	0,16
8	8	3,53	0,16
9	9	3,55	0,16
10	10	3,52	0,15
11	11	3,55	0,16
12	12	3,55	0,17
13	13	3,57	0,15
14	14	3,57	0,16
15	15	3,55	0,17
16	16	3,55	0,17
17	17	3,59	0,24
18	18	3,57	0,14
19	19	3,57	0,15
20	20	3,61	0,17

4.4 Número de pontos de inspeção

Na Tabela 7 temos os resultados relativos ao custo e sua representação na Figura 9. A média e o desvio padrão crescem a medida em que são inseridos novos elementos. Na Tabela 8 temos os resultados relativos ao tempo de execução e o gráfico na Figura 10. Este apresenta um crescimento aproximadamente linear com o número de elementos do vetor inspeção.

O número de elementos a serem inspecionados relaciona-se com tempo de execução do código. A medida que a norma do vetor inspeção aumenta, o tempo de execução cresce de forma aproximadamente linear. Isso pode ser um problema devido ao número de elementos a serem inspecionados ser elevado em um determinado setor.

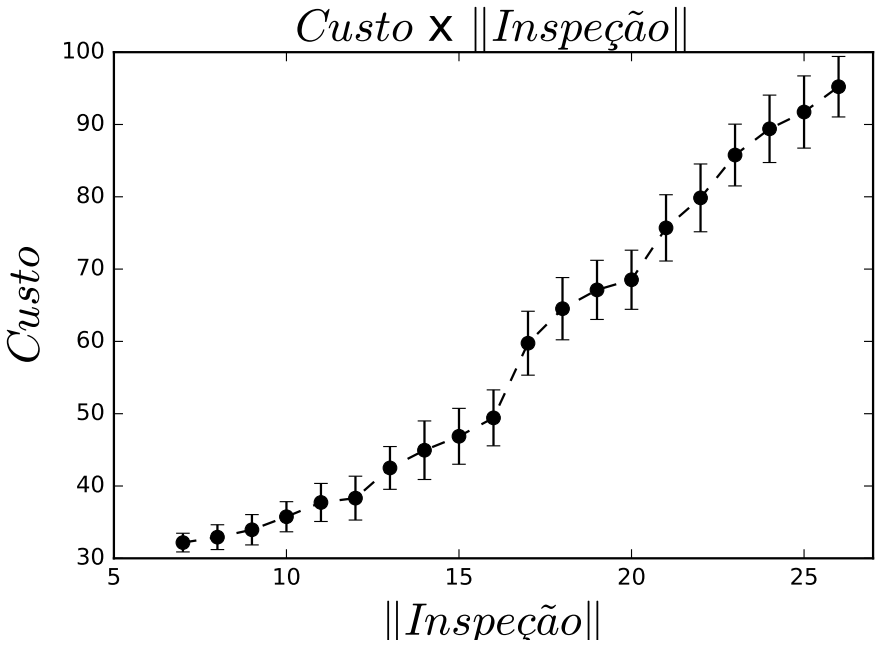

Figura 9 – Custo das posições g_{Best} em função do número de pontos de inspeção. Com número de iterações igual a 100, $N = 20$, $K = 4$, chaves $= [2, 62]$, 80 repetições por ponto, incremento de 1 elemento a ser inspecionado de um ponto para outro.

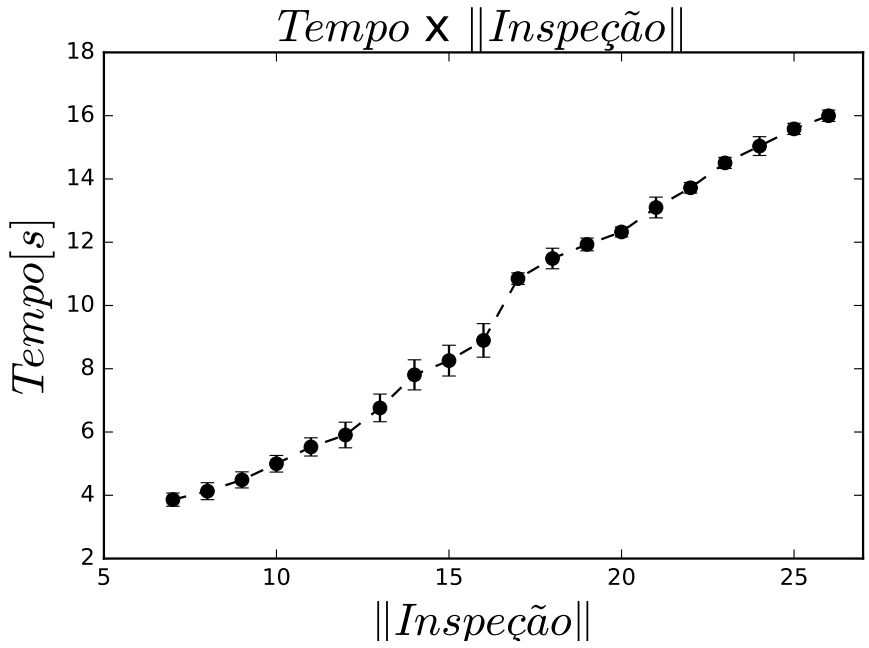

Figura 10 – Tempo de execução em função do número de pontos de inspeção. Com número de iterações igual a 100, $N = 20$, $K = 4$, chaves $= [2, 62]$, 80 repetições por ponto, incremento de 1 elemento a ser inspecionado de um ponto para outro.

Tabela 7 – Médias e desvios padrão dos custos das posições g_{Best} em função do número de pontos de inspeção. Com número de iterações igual a 100, $N = 20$, $K = 4$, chaves $= [2, 62]$, 80 repetições por ponto, incremento de 1 elemento a ser inspecionado de um ponto para outro.

Ponto	$\|\|$Inspeção$\|\|$	$\bar{x}$	$\sqrt{s^2}$
1	7	32,17	1,29
2	8	32,92	1,72
3	9	33,95	2,10
4	10	35,75	2,08
5	11	37,72	2,63
6	12	38,33	3,03
7	13	42,50	2,96
8	14	44,95	4,05
9	15	46,88	3,86
10	16	49,42	3,87
11	17	59,75	4,42
12	18	64,53	4,30
13	19	67,12	4,09
14	20	68,53	4,09
15	21	75,70	4,58
16	22	79,85	4,69
17	23	85,78	4,27
18	24	89,40	4,66
19	25	91,72	4,99
20	26	95,22	4,18

Tabela 8 – Médias e desvios padrão dos tempos de execução em função do número de pontos de inspeção. Com número de iterações igual a 100, $N = 20$, $K = 4$, chaves $= [2, 62]$, 80 repetições por ponto, incremento de 1 elemento a ser inspecionado de um ponto para outro.

Ponto	$\|$Inspeção$\|$	$\bar{x}$	$\sqrt{s^2}$
1	7	3,86	0,21
2	8	4,13	0,27
3	9	4,49	0,25
4	10	5,00	0,26
5	11	5,53	0,29
6	12	5,91	0,41
7	13	6,76	0,44
8	14	7,81	0,48
9	15	8,26	0,49
10	16	8,90	0,53
11	17	10,85	0,18
12	18	11,48	0,33
13	19	11,93	0,20
14	20	12,32	0,16
15	21	13,10	0,33
16	22	13,72	0,17
17	23	14,51	0,17
18	24	15,04	0,30
19	25	15,58	0,18
20	26	16,00	0,18

4.5 Representação gráfica do menor caminho encontrado

Na Figura 11 temos a representação gráfica do menor caminho encontrado. Devido ao grafo de vias utilizado ser não direcionado e com arestas de tamanho unitário, existem trajetos diferentes que levam ao mesmo valor. Nesse teste, 70% (56 trajetos) dos resultados obtidos em 80 repetições foram iguais ou de custo igual a esse trajeto. Excetuando-se esses 56 trajetos com redundância de custo, que são ótimos globais, os demais se tratam de ótimos locais.

Os pontos azuis são os locais a serem inspecionados e os quadrados em verde as chaves de manobra. Em vermelho temos a rota, com uma tonalidade mais escura para arestas percorridas mais de uma vez. O custo desse trajeto resultou em 29 e a duração de execução do algoritmo 3,54$[s]$. Portanto, mesmo não encontrando o menor percurso possível, o algoritmo ainda responde de forma satisfatória encontrando ótimos locais.

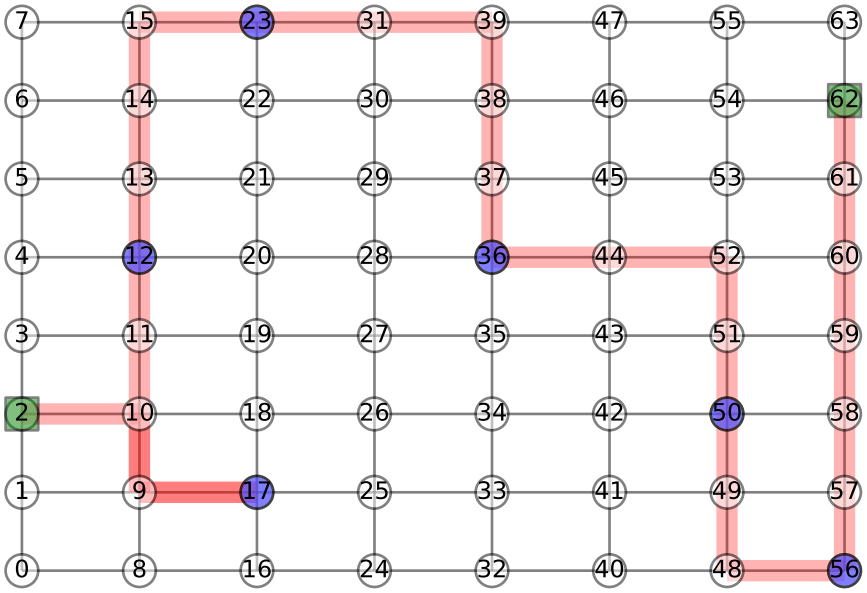

Figura 11 – Representação gráfica para a solução do roteamento com custo de 29, obtido em 70 % dos casos, e 3,625$[s]$ de duração da execução. Utilizando $N = 20$, $K = 4$, 100 iterações, chaves $= [2, 62]$, inspeção $= [17, 23, 50, 36, 12, 56]$.

5 Conclusão

Os parâmetros do DPSO devem ser selecionados de forma que se deseje um resultado mais rápido ou mais otimizado. O número de pontos de inspeção é proporcional ao tempo de execução, a medida que esse número aumenta, o tempo de execução também aumenta. Isso se torna um fator limitante, pois um setor pode ter um elevado número de pontos a inspecionar.

A meta-heurística se mostrou eficiente para roteirizar a inspeção de uma equipe visando um custo reduzido do trajeto. Como a função de custo depende dos pesos das arestas do grafo, pode-se atribuir a esses pesos uma ponderação de diferentes características, dentre elas tempo e deslocamento. Isso pode significar redução no tempo ou deslocamento da equipe em encontrar a falha e repará-la e consequentemente reduzir o tempo de restabelecimento de energia.

A redução de tempo na recomposição de UCs pode diminuir o DEC, além de aumentar a confiabilidade da rede elétrica. Isso reduz os transtornos de ordem social e econômica aos clientes.

Como proposta para trabalhos futuros sugere-se analisar a resposta do algoritmo a diversas configurações e tamanhos de grafos, incluindo grafos de vias reais. Pois ao utilizar o algoritmo de Dijkstra, o custo computacional passa a depender do tamanho do grafo e quando se trata de um grafo de vias reais, o número de nós e arestas é elevado.

Além disso, propõe-se estudar topologias de conectividade entre as partículas do enxame. Estruturas fixas e regulares possuem desempenho melhor do que as aleatoriamente selecionadas, como foi abordado nesse trabalho.

A Ponderação dos pesos das arestas também fornece um objeto de estudo relevante, pois ela pode estar relacionada a diversos fatores como: número de clientes, gastos, clientes especiais e distância.

A paralelização da execução do programa também pode ser um campo estudado, em virtude da redução do tempo de execução.

Referências

ANACONDA. 2016. Disponível em: <https://continuum.io>.

ATZINGEN, J. von; CUNHA, C. B. da; NAKAMOTO, F. Y.; RIBEIRO, F. R.; SCHARDONG, A. Análise comparativa de algoritmos eficientes para o problema de caminho mínimo. 2012.

BARROS, E. A.; PAMBOUKIAN, S. V.; ZAMBONI, L. C. Algoritmo de dijkstra: apoio didático e multidisciplinar na implementação, simulação e utilização computacional. In: *INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION, São Paulo*. [S.l.: s.n.], 2007.

BECCENERI, J. C. Meta-heurísticas e otimização combinatória: Aplicações em problemas ambientais. *INPE, Sao José dos Campos*, 2008.

CLERC, M. *[Studies in Fuzziness and Soft Computing] New Optimization Techniques in Engineering Volume 141 || Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem*. [S.l.: s.n.], 2004. ISBN 978-3-642-05767-0,978-3-540-39930-8.

CLERC, M. *Particle Swarm Optimization*. [S.l.]: ISTE, 2006. ISBN 9781905209040,1905209045.

CLERC, M. *Particle Swarm Optimization || Combinatorial Problems*. [S.l.: s.n.], 2006. ISBN 9780470612163,9781905209040.

COELHO A.; RODRIGUES, A. D. S. M. [ieee 2004 international conference on power system technology - powercon - singapore (21-24 nov. 2004)] 2004 international conference on power system technology, 2004. powercon 2004. - distribution network reconfiguration with reliability constraints. In: . [S.l.: s.n.], 2004. v. 2. ISBN 0-7803-8610-8.

CURSO, T. d. C. de. Multi-ring: Uma nova topologia para otimização por enxame de partículas (pso).

DISTRIBUIÇÃO, P. de. Módulo 8–qualidade da energia elétrica. *Agência Nacional de Energia Elétrica–ANEEL.*, 2010.

DU, H.; NI, Q.; PAN, Q.; YAO, Y.; LV, Q. An improved particle swarm optimization-based coverage control method for wireless sensor network. In: SPRINGER. *International Conference in Swarm Intelligence*. [S.l.], 2014. p. 114–124.

FANUCCHI, R. Z.; CAMILLO, M. Inspeção de alimentadores utilizando equipes de campo com recomposição parcial de trechos entre chaves operáveis com carga. In: *VI Simpósio Brasileiro de Sistemas Elétricos*. [S.l.: s.n.], 2016. ISSN 2177-6164.

GERACE IVAN; GRECO, F. The travelling salesman problem in symmetric circulant matrices with two stripes. *Mathematical Structures in Computer Science*, Cambridge University Press, v. 18, 2 2008.

GOLBERG, D. E. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, v. 1989, p. 102, 1989.

GOMES, L. d. C. T. *et al.* Inteligencia computacional na sintese de meta-heuristicas para otimização combinatoria e multimodal. Campinas, SP, 2006.

GRIMMETT, D. W. G. *Probability: an introduction*. [S.l.]: Oxford University Press, USA, 1986. (Oxford Science Publications). ISBN 9780198532729,0198532725.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA: [s.n.], 2008. p. 11–15.

HUANG, S.; YANG, J. B.; DING, H. J. Research on ga-dpso virtual machine scheduling algorithm based on comprehensive utilization of host resource. In: TRANS TECH PUBL. *Advanced Materials Research*. [S.l.], 2013. v. 791, p. 1373–1376.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

KAO, Y.; CHEN, M. Solving the cvrp problem using a hybrid pso approach. In: *Computational Intelligence*. [S.l.]: Springer, 2013. p. 59–67.

KAVOUSI-FARD ABDOLLAH; NIKNAM, T. Optimal distribution feeder reconfiguration for reliability improvement considering uncertainty. *IEEE Transactions on Power Delivery*, IEEE, v. 29, 2014.

KAZEMI SHAHRAM; MILLAR, R. J. L. M. Criticality analysis of failure to communicate in automated fault-management schemes. *IEEE Transactions on Power Delivery*, IEEE, v. 29, 2014.

KENNEDY J.; EBERHART, R. [ieee icnn'95 - international conference on neural networks - perth, wa, australia (27 nov.-1 dec. 1995)] proceedings of icnn'95 - international conference on neural networks - particle swarm optimization. In: . [S.l.: s.n.], 1995. v. 4. ISBN 0-7803-2768-3.

KEZUNOVIC, M. Smart fault location for smart grids. *IEEE Transactions on Smart Grid*, v. 2, 2011.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. *et al.* Optimization by simmulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983.

LI JUAN; MA, X.-Y. L. C.-C. S. K. P. Distribution system restoration with microgrids using spanning tree search. *IEEE Transactions on Power Systems*, IEEE, v. 29, 11 2014.

LINARES PEDRO; REY, L. The costs of electricity interruptions in spain. are we sending the right signals? *Energy Policy*, Elsevier Science, v. 61, 10 2013.

LUCCHESI, C. L. *Introduçao à Teoría dos Grafos: 12 Coloquio Brasileiro de Matemática*. [S.l.]: Instituto de Matemática Pura e Aplicada, 1979.

MILIOUDIS APOSTOLOS N.; ANDREOU, G. T. L.-D. P. Enhanced protection scheme for smart grids using power line communications techniques—part i: Detection of high impedance fault occurrence. *IEEE Transactions on Smart Grid*, v. 3, 12 2012.

MOMOH J.A.; DIAS, L. L. D. An implementation of a hybrid intelligent tool for distribution system fault diagnosis. *IEEE Transactions on Power Delivery*, IEEE, v. 12, 4 1997.

REEVES, C. R. *Modern heuristic techniques for combinatorial problems*. [S.l.]: John Wiley & Sons, Inc., 1993.

SALES CLAUDIO; MONTEIRO, E. H. R. Qualidade do fornecimento de energia elétrica: confiabilidade, conformidade e presteza. 2014.

SOUZA J.C.S.; RODRIGUES, M. S. M. D. C. F. M. Fault location in electrical power systems using intelligent systems techniques. *IEEE Transactions on Power Delivery*, IEEE, v. 16, Jan. 2001.

TEO C.Y.; GOOI, H. Artificial intelligence in diagnosis and supply restoration for a distribution network. *IEE Proceedings - Generation Transmission and Distribution*, The Institution of Electrical Engineers, v. 145, 1998.

VASCO JOHN; RAMLACHAN, R. W. J. L. W. [ieee 2008 61st annual conference for protective relay engineers - college station, tx, usa (2008.04.1-2008.04.3)] 2008 61st annual conference for protective relay engineers - an automated fault location system as a decision support tool for system operators. In: . [S.l.: s.n.], 2008. ISBN 978-1-4244-1949-4.

WANG ; LAN HUANG, . C.-G. Z. . W. P. K.-P. [ieee 2003 international conference on machine learning and cybernetics - xi'an, china (2-5 nov. 2003)] proceedings of the 2003 international conference on machine learning and cybernetics (ieee cat. no.03ex693) - particle swarm optimization for traveling salesman problem. In: . [S.l.: s.n.], 2003. ISBN 0-7803-7865-2.

WANG, X. S. Y. L. H. L. C. L. Q. Particle swarm optimization-based algorithms for tsp and generalized tsp. *Information Processing Letters*, Elsevier Science, v. 103, 2007.

YANG, M.-S. C. S.-J. L. S.-I. L. D.-S. L. X. A direct three-phase circuit analysis-based fault location for line-to-line fault. *IEEE Transactions on Power Delivery*, IEEE, v. 22, 2007.

YIN, P.-Y. A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. *Journal of Visual Communication and Image Representation*, Elsevier Science, v. 15, 2004.

ZAMBONI, L. C.; PAMBOUKIAN, S. V. D.; BARROS, E. d. A. R. *C++ para Universitários*. [S.l.]: Páginas & Letras, 2006.

ZIDAN ABOELSOOD; KHAIRALLA, M. A.-A. M. K.-T. S. K. A. A. E. S. R. G. A. M. Fault detection, isolation, and service restoration in distribution systems: State-of-the-art and future trends. *IEEE Transactions on Smart Grid*, 2016.