**UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA**
Autorizada pelo Decreto Federal nº 77.496 de 27/04/76
Recredenciamento pelo Decreto nº17.228 de 25/11/2016
**PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO**
*COORDENAÇÃO DE INICIAÇÃO CIENTÍFICA*

**XXVII SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS**
**SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA - 2023**

# Modelagem de cidades inteligentes considerando zonas de conectividade para detecção de emergências

**Wagner A. Ferreira Junior[1]; Thiago C. Jesus[2]**

1. Bolsista PIBIC/CNPq, Graduando em engenharia de computação, Universidade Estadual de Feira de Santana, e-mail: wagner.afjr@gmail.com

2. Orientador, Departamento de nome, Universidade Estadual de Feira de Santana, e-mail: tcjesus@uefs.br

**PALAVRAS-CHAVE:** Cidades inteligentes, Classificação de redes, detecção de emergências, sensoriamento inteligente.

## INTRODUÇÃO

Uma cidade Inteligente é caracterizada principalmente pela presença de dispositivos interconectados para a coleta de informação a fim de gerenciar recursos e ativos de maneira eficiente, a fim de melhorar a segurança e qualidade de vida nestas cidades, é necessário aumentar a velocidade de resposta a emergências, portanto, foi pensada a implementação de sensores de vigilância, dispositivos responsáveis por monitorar áreas de risco e de se comunicarem através de uma rede de conexão, estabelecendo assim, uma malha de vigilância ativa 24/7, capazes de alertar sobre possíveis desastres de maneira autônoma e quase instantânea ao momento em que correm, aumentando a agilidade e precisão com que serviços públicos de segurança e saúde sejam capazes de responder, o que por sua vez, evita maiores tragédias, deste modo, se faz necessário a criação de um algoritmo voltado para o posicionamento dos sensores responsáveis pelo monitoramento das áreas de risco com o objetivo de minimizar o desperdício de recursos enquanto maximiza a eficiência de vigilância de tais sensores, entretanto, não é viável o posicionamento destes sensores sem um devido mapeamento da área, o que envolve a caracterização das redes de conexão presentes na região, portanto o sistema a ser apresentado tem como objetivo apresentar estratégias de classificação de zonas e posicionamento de sensores ao analisar a infraestrutura de uma determinada zona.

## METODOLOGIA

Ao iniciar o desenvolvimento do algoritmo proposto, foi optado uma metodologia de pesquisa exploratória de casos aleatórios gerados por computador, estes casos foram definidos como diferentes zonas de risco em que as posições de todas as redes de conexão presentes na zona poderiam ser alteradas a cada inicialização do sistema.

O sistema utiliza como bibliotecas principais Numpy (HARRIS et al., 2020), MatplotLib (HUNTER, 2007), além da biblioteca Math (ROSSUM, 2020) e time, estas duas últimas, nativas do próprio Python. Ao definir a largura ($w$) e altura ($h$) de uma area de monitoramento $MA$, adota-se a estratégia de dividir $MA$ em blocos menores, chamados de *mb*'s, cada *mb* representando 1 ou mais metros quadrados da área a ser monitorada. Para a implantação dos *mb's*, cada vértice $V$

de cada bloco de monitoramento *B* é calculado a partir da posição do vértice inicial *A*, JESUS (2020) propõe a seguinte equação para o cálculo dos vértices da área *MA* a ser monitorada utilizando sua largura *w*, altura *h* e um ângulo de rotação β.

$$MA_{V2_x} = A_x + w\cos(\beta), \qquad MA_{V2_y} = A_y + w\sin(\beta)$$
$$MA_{V3_x} = A_x + h\sin(\beta), \qquad MA_{V3_y} = A_y - h\cos(\beta)$$
$$MA_{V4_x} = A_x + h\sin(\beta) + w\cos(\beta), \qquad MA_{V4_y} = A_y - h\cos(\beta) + w\sin(\beta)$$

Pode-se utilizar uma equação bastante similar para calcular os vértices e centroide (*Bxc* e *Byc*) de cada um dos *mb's* da zona de risco baseando-se em *Ax* , *Ay*, um ângulo de rotação β e as posições *M* e *N* do *mb* em questão na matriz *MBS*.

Após definir as coordenadas de cada *mb*, uma pontuação baseada nos atributos das redes que cobrem o centroide do bloco é atribuída ao bloco, definindo também seu nível de conexão, com o nível de conexão Aqui também são inicializadas duas matrizes vazias de tamanho *M* por *N*, *MBS* irá guardará os valores *CTSR* e *MBC*, que conterá o nível de conexão de cada bloco.

Ao percorrer um loop aninhado de *M.N* utilizando índices em vez de iteradores, pode-se utilizar estes mesmos índices nas matrizes *MBS* e *MBC*, facilitando para que em cada iteração do laço de repetição seja feito o cálculo do *CTSR* do bloco atual e baseando-se neste valor *CTSR*, é marcado na matriz *MBc* o nível de conexão do *mb*.

Um algoritmo de identificação cria uma matriz vazia *MBmM.N* e itera sobre cada elemento da matriz *MBS*, assim que um *mb* com nível de conexão não nulo é encontrado, é chamada uma função que realiza uma busca em profundidade, contando quais *mb's* adjacentes são do mesmo nível de conexão e os adicionando a uma lista *Zf* que será a zona achada, ao finalizar a busca em largura nesta zona, retorna uma lista de zona encontrada *Zf* que contém todos os blocos adjacentes de um mesmo nível de conexão.

O primeiro algoritmo foi chamado de "*WoProxZone*" funciona por adaptar o algoritmo de busca em largura (*breadth first search*, ou BFS) para alocar sensores enquanto percorre cada *mb* de cada zona de conexão, respeitando uma distância mínima entre as unidades de sensoreamento. O ponto de partida da alocação é computado como um "*pseudo centroide*" PC, este *pseudo centroide* é o *mb* central de uma zona, ele é achado após a ordenação de todos os blocos dentro da zona de conexão priorizando a coordenada x e fatiando esta lista de *mb*'s no meio e enquanto visita os *mb*'s adjacentes de *PC*, o algoritmo tenta posicionar sensores no *mb* atual, respeitando regras de quantidade máxima e distancia mínima. Este acaba por não ser um algoritmo justo, uma vez que tenta alocar quantos sensores quanto possível toda vez que entra no modo alocação, muitas vezes deixando várias zonas e até níveis de conexão inteiros sem sensores quando as zonas iniciais são muito grandes. Para balancear estes pontos negativos, é implementado um algoritmo de prioridade, responsável por dizer a ordem em que o *WoProxZone* deve visitar os níveis de conexão, dando prioridade para os níveis com mais sensores por zona, a seguir dos níveis com maior conectividade. Por garantir que a maior quantidade possível de *mb*'s sejam testados para o posicionamento dos sensores e que estes estejam a uma distância mínima uns dos outros, *WoProxZone* também é o algoritmo mais custoso dos 5, aumentando em um ritmo muito maior a quantidade de recursos e tempo necessários do que os outros algoritmos para concluir o posicionamento na medida que o detalhamento da área monitorada aumenta.

Os algoritmos baseados em *bps_zoneSize* ordenam as zonas para calcular um valor de blocos por sensor *bps* para cada zona ao comparar o tamanho de cada zona com a quantidade total de blocos no nível e multiplicar o resultado pela quantidade de sensores disponíveis para esse mesmo nível. O *bps* será utilizado para achar o centroide (para o algoritmo *zoneSize_TrueCentroid*) e o *pseudo centroide* (para o algoritmo

*bps_zoneSize_Pcenter*) ao fatiar um subconjunto de *mb*'s dentro das zonas, sendo assim a cada *bps* blocos, é calculado um centroide (ou pseudo centroide no caso do *bps_zoneSize_Pcenter*) entre o início e fim de cada subconjunto gerado.

Os algoritmos baseados em *bps_level* possuem o mesmo começo, o calculo do *bps* se da por comparar a quantidade de *mb*'s em todo um nível de conectividade contra a quantidades de sensores disponíveis para um nível i conectividade, sendo assim cada nível possui somente um único *bps*. Assim como *bps_zoneSize*, *bps_level*, fatia as zonas a cada *bps* blocos passados para calcular o centroide e pseudo centroide.

**RESULTADOS E/OU DISCUSSÃO (ou Análise e discussão dos resultados)**

No total foram desenvolvidos cinco algoritmos diferentes para o posicionamento dos sensores, são estes: *WoProxZone*, *bps_level_Pcenter*, *bps_level_TrueCentroid*, *bps_zoneSize_PCENTER* e *bps_zoneSize_TrueCentroid***.**

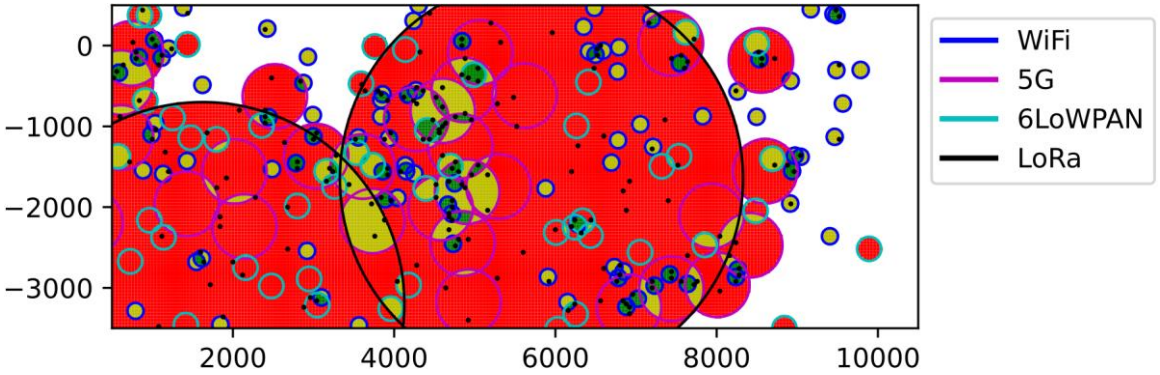


Figura 1 - Bps por nível de conexão - Pseudo Centroide (ALEXANDRE, W. F. J; JESUS, T. C. 2023).

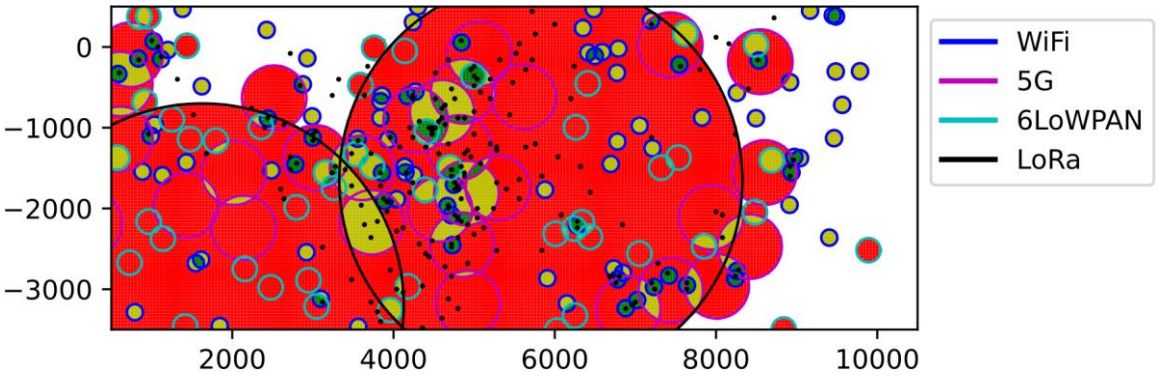


Figura 2 - Bps por nível de conexão - Centroide (ALEXANDRE, W. F. J; JESUS, T. C. 2023).

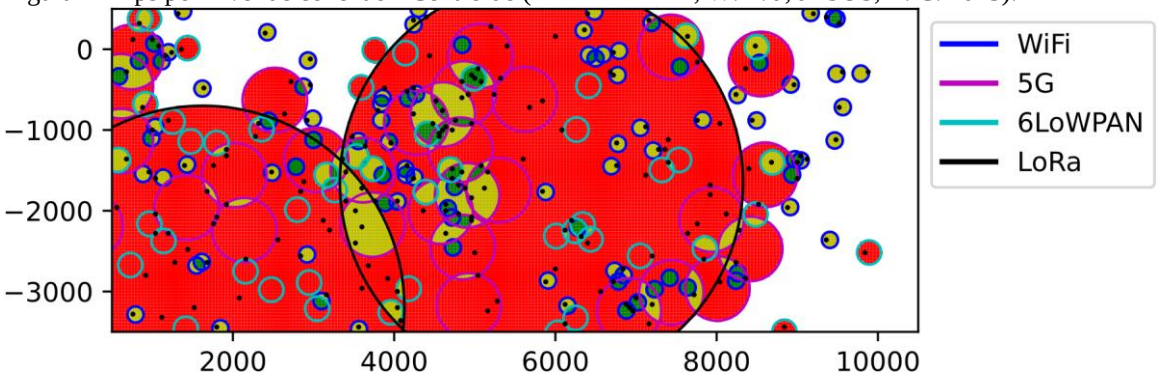


Figura 3 - Bps por razão de zona/nível - Pseudo Centroide (ALEXANDRE, W. F. J; JESUS, T. C. 2023).

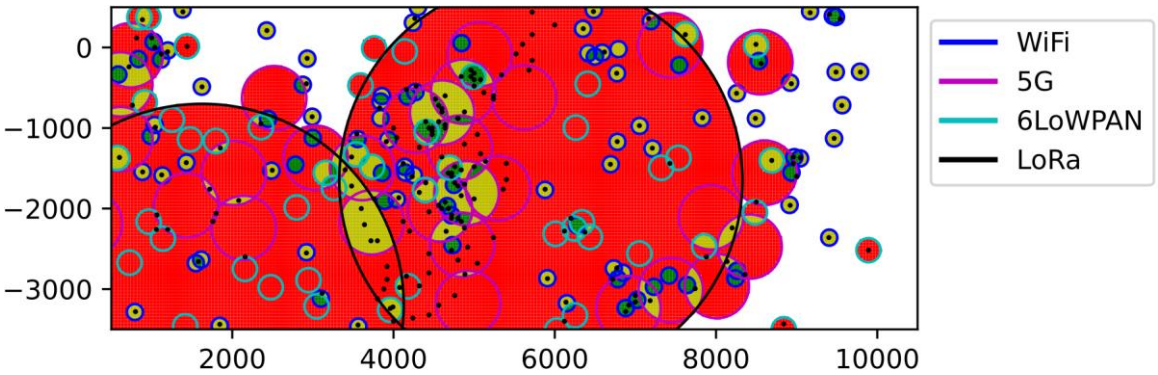

Figura 4 - Bps por razão de zona/nível - Centroide (ALEXANDRE, W. F. J; JESUS, T. C. 2023).

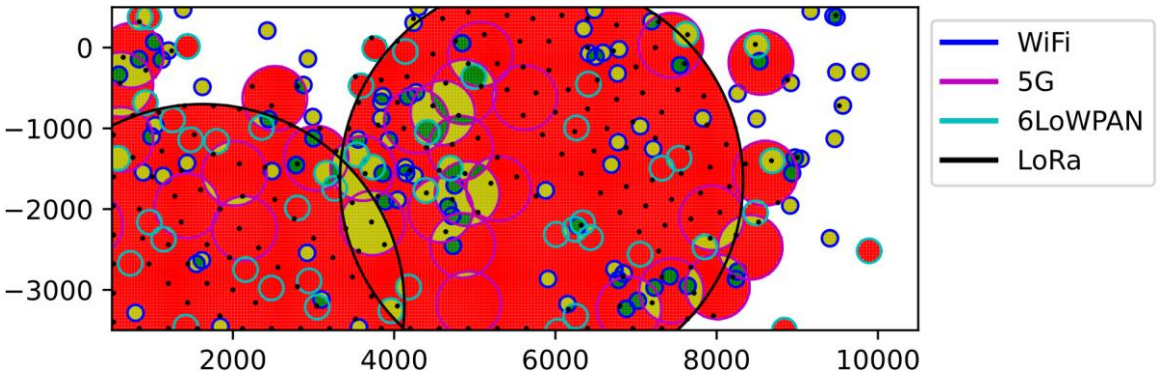

Figura 5 – WoProxMzone - Sensores com 320 metros de distância (ALEXANDRE, W. F. J; JESUS, T. C. 2023).

## CONSIDERAÇÕES FINAIS

Apesar de ser o algoritmo mais custoso para executar, o *WoProxZone* se mostrou o melhor algoritmo para posicionar de forma adequada os sensores, isto é, quando a quantidade de sensores disponíveis é suficientemente grande para suprir todas as zonas de conectividade ou quando os sensores podem estar a uma distância mínima longa o suficiente para que uma baixa quantidade de unidades de sensoreamento não seja um obstáculo logo sendo seguido pelos algoritmos *zoneSize_PCENTER* e *bps_level_PCENTER,* enquanto que o *bps_level_ TrueCentroid e zoneSize_TrueCentroid* tiveram os piores posicionamentos.

## REFERÊNCIAS

HARRIS, C. R. et al. Array programming with NumPy. Nature, Springer Scienceand Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em:<https://doi.org/10.1038/s41586-020-2649-2>.

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

JESUS, T. C. et al. A comprehensive dependability model for qom-aware industrial wsn when performing visual area coverage in occluded scenarios. Sensors, v. 20, n. 22, 2020. ISSN1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/22/6542>.

ROSSUM, G. V. The Python Library Reference, release 3.8.2. [S.l.]: Python Software Foundation,2020.