Uma abordagem bi-objetiva para cobertura de pontos de interesse utilizando rede de sensores sem fio

Gabriel V. C. Rocha
Universidade Federal de Minas Gerais, Departamento de Ciências da Computação
gabreuvcr@gmail.com

Iago A. Carvalho
Universidade Federal de Alfenas, Departamento de Ciências da Computação
iago.carvalho@unifal-mg.edu.br

Thiago F. Noronha
Universidade Federal de Minas Gerais, Departamento de Ciências da Computação
tfn@dcc.ufmg.edu.br

RESUMO

Redes de sensores sem fio podem ser utilizadas para monitoramento de áreas ou pontos de interesse, com aplicações militares, na agricultura e sistemas de transporte, dentre outras. Estas redes devem ser construídas de forma a maximizar a qualidade da cobertura utilizando um pequeno número de sensores. Construir uma rede que otimize estas duas características constitui um problema NP-difícil. Este trabalho propõe uma heurística bi-objetivo para este problema baseada no algoritmo do *Differential Evolution*, uma meta-heurística evolucionária amplamente utilizada na literatura, e compara os resultados obtidos com os obtido por uma heurística *Harmony Search*, que é o algoritmo mais eficiente da literatura para este problema. Um pequeno conjunto de experimentos computacionais demonstra que o algoritmo proposto neste trabalho supera a heurística *Harmony Search* da literatura.

PALAVRAS CHAVE. Redes de sensores sem fio, Algoritmos evolucionários, Cobertura, Metaheurísticas, Otimização combinatória

ABSTRACT

Wireless sensor networks can be used to monitor areas or points of interest, with applications in military, agriculture, and transportation systems, among others. These networks need to be built in such a way that the sensing quality is maximized using a small number of sensors. Building a network that optimizes both characteristics is an NP-hard problem. This work proposes a bi-objective heuristic for this problem based on the Differential Evolution algorithm, an evolutionary meta-heuristic widely employed in the literature, and contrasts the obtained results with those obtained by a Harmony Search heuristic, which is the most efficient algorithm in the literature for this problem. A small set of computational experiments demonstrate that the algorithm proposed in this work outperforms the literature's Harmony Search heuristic.

KEYWORDS. Wireless sensor networks, Evolutionary algorithms, Coverage, Metaheuristics, Combinatorial optimization

1. Introdução

Monitoramento de áreas de interesse utilizando redes de sensores sem fio (em inglês, *Wireless Sensors Networks* - WSN) é de grande interesse no mundo atual. Este monitoramento ocorre em vários cenários, como em áreas militares, ambientais, química, biologia, agricultura e transporte, dentre outras [Farsi et al., 2019]. Com isso, o problema de determinar as melhores posições para se colocar sensores a fim de minimizar gastos e aumentar a capacidade de cobertura é de suma importância.

Um ponto de interesse (em inglês, *Point of interest* - POI) é uma abstração em que toda uma área ou conjunto de objetos de interesse é imaginado como um ponto e que devem ser monitorados. Dizemos que um sensor cobre um POI caso o raio de cobertura do sensor seja maior ou igual à distância do sensor até o POI. Todo POI coberto por um sensor é dito ser monitorado.

Sensores coletam dados de POIs e podem monitorar um ou mais ao mesmo tempo, possuem área de cobertura definida por um raio, podendo se comunicar com outros sensores a partir de uma área de comunicação definida por um segundo raio, ter restrições quanto à energia e obstáculos no ambiente. Adicionalmente, mesmo que um sensor não cubra nenhum POI, ele pode receber informações e passá-las para outro sensor, fazendo parte da conectividade dos sensores [Harizan e Kuila, 2020].

Este trabalho estuda o problema de maximização da cobertura de POIs utilizando WSN, conforme proposto por Alia e Al-Ajouri [2017]. O problema de maximizar a cobertura de uma WSN é definido com um conjunto de sensores S e um conjunto de POIs P, onde cada POI $p \in P$ e cada sensor $s \in S$ possuem uma coordenada x, y determinando sua localização. Todos os sensores e POIs estão distribuídos em um *grid* bi-dimensional com tamanho $M \times N$. Nosso problema consiste em otimizar dois objetivos. O primeiro é encontrar o menor subconjunto de sensores $S' \subseteq S$ que cubra os POIs definidos, enquanto o segundo é maximizar a razão de cobertura dos POIs, isto é, maximizar o número de POIs cobertos pelos sensores escolhidos. Este problema é um problema de otimização NP-difícil [Alia e Al-Ajouri, 2017].

Neste trabalho, nós propomos a resolução deste problema utilizando um algoritmo baseado na meta-heurística *Differential Evolution* (DE) [Storn e Price, 1997], um algoritmo evolutivo simples e genérico amplamente utilizado na literatura para a resolução de problemas de otimização. Nós comparamos os resultados obtidos pelo *Differential Evolution* com os resultados obtidos por um algoritmo *Harmony Search* (HS) [Alia e Al-Ajouri, 2017], que é o algoritmo mais recente da literatura para este problema.

O artigo está organizado desta forma. Na seção 2 são discutidos os trabalhos relacionados. A seção 3 apresenta os algoritmos implementados. A seção 4 demonstra e discute os resultados obtidos. Por fim, na última seção são apresentadas conclusões e os trabalhos futuros a serem desenvolvidos.

2. Trabalhos relacionados

Sahu e Gupta [2012] estudam o problema da cobertura e conectividade em redes de sensores sem fio, revisando três estratégias comuns usadas para resolvê-los: Baseadas em força, baseadas em *grid* e baseadas em geometria computacional. Nas baseadas em geometria computacional, a abordagem mais utilizada, Diagrama de Voronoi, foi discutida e analisada. Foi proposto um método para gerenciar a densidade de uma rede, com o objetivo de introduzir um mecanismo tolerante a falhas, aumentar a precisão e fornecer dados de várias resoluções. O método utilizado envolve o uso de Diagrama de Voronoi para determinar quais nós devem ser ligados ou desligados, permitindo que os nós sejam temporariamente retirados de serviço. Por fim, foi sugerido que esta abordagem poderia ser utilizada na arquitetura de gerenciamento de WSN.

Diversas variações do problema também foram estudadas, como em Bajaj e Manju [2014], onde o objetivo é maximizar a cobertura junto com o problema de restrição de energia, ou seja, maximizar o tempo de vida da rede. A heurística proposta é chamada *Energy-Efficient Maximum Coverage Heuristic*, MCH. Essa heurística encontra uma cobertura de sensores e após isso melhora a solução removendo sensores que a sua retirada não altere a cobertura, ajudando a reduzir o gasto de energia. O resultado mostrou que a heurística teve um desempenho melhor que as outras duas heurísticas comparadas.

Variações mais complexas, como o problema da K-Cobertura, onde cada POI precisa ser coberto por pelo menos K sensores, foram estudadas em Rebai et al. [2015]. Propuseram um algoritmo de Busca Local (em inglês, *Local Search Algorithm* - LS) e um Algoritmo Genético (em inglês, *Genetic Algorithm* - GA) para o problema K-Cobertura. No final da experimentação, o *gap* (porcentagem da distância do valor obtido para o valor ótimo) médio máximo do algoritmo GA não ultrapassou 30%, enquanto no algoritmo LS foi encontrado um gap de 42.52%. Os resultados mostraram que quanto maior o tamanho da região explorada, menor o *gap* do algoritmo GA, ao contrário do algoritmo LS, onde foi observado um aumento. Dessa forma, o algoritmo LS obteve um desempenho melhor em instâncias menores, enquanto o GA se saiu melhor nas maiores instâncias.

Alia e Al-Ajouri [2017] estudaram o problema de maximizar a cobertura de uma WSN propondo uma meta-heurística *Harmony Search* (HS). HS é uma meta-heurística evolucionária inspirada em um fenômeno de músicos, mais especificamente, imita o comportamento de músicos quanto estão improvisando. A HS possui uma memória chamada *Harmony Memory* (HM) que armazena cada solução. Essa memória inicia com soluções produzidas aleatoriamente e a cada iteração aperfeiçoa as soluções criando uma nova solução utilizando três operadores: *memory consideration* — que seleciona valores já existente na *Harmony Memory* —, *pitch adjustment* — que melhora localmente os valores obtidos pela *memory consideration* — e *random consideration* — que é usado para diversificar aleatoriamente a nova solução —. Quando uma nova solução criada é melhor que a pior existente na *Harmony Memory*, está é removida da memória e a nova solução é colocada no seu lugar. No artigo, foi comparada com um algoritmo genético e apresentou resultados superiores e com maior eficiência em todos os casos de teste.

Hanh et al. [2019] propuseram um novo algoritmo genético denominado *MIGA*, tendo como objetivo melhorar outro algoritmo genético, *IGA*, que era considerado até o momento o melhor algoritmo genético para o problema de maximizar a cobertura de uma WSN utilizando sensores com alcances distintos. O artigo propõe uma nova forma de representar os indivíduos, uma nova função de cálculo de *fitness*, uma heurística de inicialização diferente e a combinação de dois operadores genéticos. A heurística *MIGA* foi comparada com cinco outras heurísticas do estado da arte e obteve melhores resultados em 13 de 15 instâncias, demonstrando um desempenho superior às demais.

Além da K-Cobertura, considerando também o problema da M-Conectividade, onde os sensores precisam se comunicar e estarem conectados, tolerando até $M - 1$ falhas, surge o problema K-Cobertura e M-Conectividade. Barkhoda e Sheikhi [2020] propuseram um método meta-heurístico baseado no *Imperialist Competitive Algorithm* (ICA), o *Immigrant Imperialist Competitive Algorithm* (IICA), no qual possui uma alta capacidade exploratória, além de tratar do problema com multi-restrições, com posições potenciais já pré-definidas e lidar tanto com regiões 2D quanto 3D. Uma característica interessante da meta-heurística ICA é a capacidade de fornecer um conjunto de soluções para escolha, o que é bastante útil no contexto de abordagem bi-objetiva. Em comparação com outras cinco meta-heurísticas, o IICA obteve resultados superiores e com maior eficiência.

3. Definição formal do problema

Separando o que é da natureza do problema do que é particularidade dos algoritmos, temos que uma rede de sensores sem fio é uma rede contendo um conjunto de sensores S e um conjunto de pontos de interesse (POIs) distribuídos em um *grid* $M \times N$. Os pontos de interesse são definidos a partir de um valor de tamanho de célula, gerando quadrados idênticos por todo o *grid*, com cada um localizado no centro de um quadrado. Cada sensor possui um raio R de alcance de cobertura e um valor de incerteza Re, definido aqui como a metade do raio de alcance. Além disso, existem algumas constantes ligadas às características dos sensores, sendo estas: $\alpha_1 = 1$, $\alpha_2 = 0$, $\beta_1 = 1$, $\beta_2 = 0.5$ e $Coverage\ Threshold = 0.9$. Elas são utilizadas para verificar se um POI está sendo coberto e foram definidas por [Alia e Al-Ajouri, 2017]. Então, para cada sensor s_i, existe uma probabilidade $P_{cov}(s_i, p_j)$ de cobrir um POI p_j, definida como

$$P_{cov}(s_i, p_j) = \begin{cases} 1 & \text{se } dist(s_i, p_j) \leq R - Re \\ exp(-\frac{\alpha_1 \lambda_1^{\beta_1}}{\lambda_2^{\beta_2}} + \alpha_2) & \text{se } R - Re < dist(s_i, p_j) < R + Re \\ 0 & \text{se } dist(s_i, p_j) \geq R + Re \end{cases}$$

onde os valores α_1, α_2, β_1 e β_2 são os valores pré-definidos citados anteriormente. A distância entre um sensor e um POI é calculada pela fórmula da distância euclidiana. Já os valores λ_1 e λ_2 são definidos, respectivamente, pelas Equações (1) e (2).

$$\lambda_1 = Re + R - dist(s_i, p_j) \tag{1}$$

$$\lambda_2 = Re - R + dist(s_i, p_j) \tag{2}$$

Com a fórmula de cobertura definida acima, calculamos utilizando a Equação (3) o valor da cobertura conjunta, onde a sobreposição da cobertura de vários sensores podem aumentar a chance de um POI ser coberto. O conjunto Sov contém todos os sensores da solução. O resultado dessa cobertura conjunta é comparado com a constante $Coverage\ Threshold$, definindo um limite do valor da cobertura conjunta para definir se um POI é coberto, ou seja, um POI p_j é dito coberto se $P_{Sov}(p_j) > Coverage\ Threshold$.

$$P_{Sov}(p_j) = 1 - \prod_{s_i \in Sov} (1 - P_{cov}(s_i, p_j)) \tag{3}$$

Por fim, a partir das dimensões $M \times N$ (Largura x Altura) e dos raios de alcance e de incerteza, definimos o mínimo e o máximo de sensores factíveis para uma solução, calculados da seguinte forma:

$$MinS = \left\lceil \frac{Largura}{2 \times (R + Re)} \times \frac{Altura}{2 \times (R + Re)} \right\rceil \tag{4}$$

$$MaxS = \left\lceil \frac{Largura}{2 \times (R - Re)} \times \frac{Altura}{2 \times (R - Re)} \right\rceil \tag{5}$$

4. Algoritmos

A meta-heurística utilizada como *baseline* foi a *Harmony Search*, sendo implementada conforme proposto em Alia e Al-Ajouri [2017]. Além disso, este trabalho propõe uma meta-heurística bi-objetivo baseado no algoritmo *Differential Evolution*, outro algoritmo populacional baseado em computação evolutiva. A seguir, ambos os algoritmos serão explicados em detalhes.

4.1. Harmony Search

Além das entradas intrínsecas ao problema, o algoritmo *Harmony Search* recebe como entradas o tamanho da *Harmony Memory*, o número de iterações e algumas taxas que definem a forma na qual acontecerão as melhorias das soluções: *Harmony Memory Consideration Rate* ($HMCR$), *Pitch Adjustment Rate* (PAR) e *Bandwidth* (BW).

Para quantificar um valor para cada solução encontrada, é definida pelo artigo uma função objetivo. Porém, apesar de utilizar o número de sensores e a taxa de cobertura, que são os valores que queremos otimizar, tentar encontrar um bom valor para esta função não representa uma otimização bi-objetiva, como será mostrado nos resultados obtidos. A fórmula pode ser vista abaixo:

$$Obj = \frac{1}{S} \times \text{Taxa cobertura} \times MinDist \times K \tag{6}$$

Onde S é a quantidade de sensores utilizados na solução, a taxa de cobertura é dada pela razão entre a quantidade de POIs considerados cobertos e o número total de POIs, K é uma constante para se evitar valores muito próximos de zero e $MinDist$ é a razão entre a menor distância entre sensores da solução e a maior distância possível entre dois sensores, servindo para evitar convergências prematuras e também que sensores fiquem muito próximos, cobrindo uma área muito similar.

Com isso, inicia-se a *Harmony Search*. É criada a *Harmony Memory* (HM) de forma aleatória, onde cada solução é representada por um vetor de sensores, possuindo $A \in [MinS \leq A \leq MaxS]$ sensores aleatórios ativos espalhados pelo *grid*, os outros $MaxS - A$ sensores serão considerados desativados. Cada solução possui também um valor objetivo calculado como descrito acima.

Para cada iteração do algoritmo, é criado um novo vetor, que irá adicionar sensores em sua solução com três operadores:

1. Com probabilidade $HMCR$, irá realizar o que chamamos de *Harmony Memorization*, buscando aleatoriamente na *Harmony Memory* um sensor já existente em alguma solução.

2. Considerando que houve uma *Harmony Memorization* e que o sensor selecionado está ativo, existe uma probabilidade PAR de se realizar um *Pitch Adjustment*, adicionando ou removendo das coordenadas do sensor selecionado um valor aleatório $\in [0, 1]$ multiplicado pela constante BW — que define o quanto será diversificado —. Isso gera uma busca local na solução já existente, podendo melhorar seu valor objetivo e evitar ótimos locais.

3. Com probabilidade $1 - HMCR$, será realizada a operação *Random Consideration*, onde será criado e adicionado na solução um sensor de forma aleatória, aumentando a exploração do algoritmo.

Depois de construir o vetor da solução realizando as três operações, comparamos essa solução com a pior existente na *Harmony Memory*. Caso o novo vetor possua um valor objetivo melhor, a pior solução é excluída e colocamos a nova solução na HM. Isso ocorre até que o critério de parada seja alcançado, definido como a quantidade de iterações. Ao final, teremos a melhor solução encontrada.

Vale ressaltar que a função objetivo mostrada na Equação 6 não pode ser considerada bi-objetiva, fazendo com que a meta-heurística, quando executada, apenas retorne como solução um único ponto. Para obter todos os pontos, é necessário fixar e executar para cada número de sensores. Dessa forma, a nossa meta-heurística tem como propósito avançar e encontrar os pontos da fronteira de Pareto de forma realmente bi-objetiva, como será explicado a seguir.

4.2. Differential Evolution

Da mesma forma, além das entradas do problema, a meta-heurística *Differential Evolution* recebe como parâmetro a quantidade de indivíduos na população, o número de gerações e algumas taxas, como o fator F que controla o ritmo da evolução e a taxa CR que controla a taxa de *crossover*. Entretanto, como o problema foi modelado para possuir dois tipos de vetores, foram introduzidos operadores tradicionais de algoritmos genéticos, recebendo também como entrada a taxa de mutação MR.

Como mencionado acima, cada indivíduo é representado com dois vetores: o primeiro vetor possui as posições de sensores, ou seja, as posições x e y no *grid*. Já o segundo é um vetor binário que apenas determina se o sensor está ativado ou desativado. Ambos os vetores possuem o mesmo tamanho, definido pelo cálculo do número máximo de sensores possíveis.

De forma similar ao algoritmo HS, iniciamos a execução criando a primeira população. É gerado cada agente de forma aleatória, ou seja, gerando posições dos sensores e seu estado (ligado ou desligado) de forma randômica. Esse primeiro passo é finalizado ao atingir a quantidade da população determinada na entrada do programa.

Em seguida, tem início a aplicação dos operadores evolutivos. Em toda geração, será percorrido cada indivíduo e criado um novo indivíduo experimental. Podemos aplicar dois tipos de operadores evolutivos. O primeiro são os operadores clássicos do *Differential Evolution* e são aplicados no vetor que contém as posições x, y dos sensores. Já o segundo conjunto são operadores clássicos de algoritmos genéticos e são aplicados no vetor binário. Cada indivíduo experimental é criado através da aplicação de somente um destes conjunto de operadores. A aplicação deles é realizada de forma uniforme e aleatória, sendo que existe 50% de chance da aplicação dos operadores do *Differential Evolution* e 50% de chance de aplicação do segundo conjunto de operadores.

4.2.1. Operadores do *Differential Evolution*

A mutação produz novos indivíduos, denominados indivíduos experimentais, a partir da adição de um indivíduo aleatório γ à diferença entre dois outros indivíduos aleatórios α e β da mesma população. Este processo utiliza um fator de perturbação $F \in [0; 2]$. Seja X_α, X_β, e X_γ três indivíduos distintos que pertencem à população e selecionados aleatoriamente. Um indivíduo experimental V é gerado como como

$$V = X_\gamma + F(X_\alpha - X_\beta)$$

Em seguida, é aplicado o operador de *crossover*. Seja X_i^j a j-ésima posição do vetor de coordenadas do indivíduo X_i. O operador de *crossover* sorteia um indivíduo X_i da população e gera um novo indivíduo experimental W como

$$W^j = \begin{cases} V^j, & \text{se } rand(0,1) \leq CR \\ X_i^j, & \text{caso contrário} \end{cases}$$

onde $CR \in [0, 1]$ é um parâmetro do algoritmo.

4.2.2. Operador aplicado ao vetor binário

O operador de *crossover* escolhido é o *uniform crossover*. Este operador realiza o cruzamento entre o indivíduo X_i escolhido e um novo indivíduo X_j selecionado de forma aleatória e define um novo vetor binário. Cada posição do vetor binário pode manter seu valor ou herdar o valor do indivíduo X_i com probabilidade equiprovável. Além disso, com probabilidade MR, ocorre uma mutação, ou seja, a alteração do valor de duas posições do vetor, sem que a solução torne-se inviável.

4.2.3. Avaliação do novo indivíduo

Com o agente experimental criado, é verificado se possui uma "dominância fraca" sobre o original. A dominância fraca define se o novo agente é melhor ou pelo menos igual ao original, apesar de modificado [Kukkonen e Lampinen, 2005]. Caso domine, será adicionado à nova população. Entretanto, caso falhe nesta condição, o agente original será adicionado para a próxima geração, além disso, existe mais uma condição: se o original não dominar o agente experimental — note que não é a dominância fraca, mas sim verificar se além da dominância fraca, há pelo menos uma função objetivo melhor —, ele será incluído também na próxima geração, porém como adicionamos dois novos indivíduos, precisamos definir uma variável m para controlar quantas vezes este caso ocorreu, para após finalizar uma geração, remover os indivíduos excedentes.

Após criar a nova população, é calculada a sua $crowding\ distance$, e como último passo, enquanto houverem mais agentes do que o permitido, ou seja, enquanto $m > 0$, iremos encontrar algum agente que não domine nenhum outro e que além disso, possua o menor valor de $crowding\ distance$, removendo-o da população [Kukkonen e Lampinen, 2005].

Isso ocorre enquanto não se atinge o critério de parada, sendo este a quantidade de gerações passadas como entrada. No final da execução, teremos uma fronteira de Pareto constituída com as melhores soluções encontradas para determinados números de sensores. Uma observação importante é que ao rodar o algoritmo apenas uma vez, não é garantido que se obtenha um valor de cobertura para cada número de sensores.

5. Experimentos computacionais

Ambos os algoritmos foram desenvolvidos utilizando C++ 14 e compilados com o GNU G++ 9.4.0. Para gerar os resultados da *Harmony Search*, foi utilizado os mesmos parâmetros expostos no artigo [Alia e Al-Ajouri, 2017], sendo estes: $tamanho\_harmony\_memory = 30$, $HMCR = 0.90$, $PAR = 0.3$, $BW = 0.2$ e $num\_iteracoes = 60000$. Já para a *Differential Evolution*, foram utilizados os parâmetros $F = 0.15$, $CR = 0.2$, $MR = 0.15$, $tamanho\_populacao = 250$ e $num\_geracoes = 1000$.

Como já foi mencionado anteriormente, a *Harmony Search* utilizando a Equação 6 não é de fato um problema bi-objetivo (já que apenas tenta otimizar uma única função), gerando apenas uma única saída contendo o número de sensores e a sua cobertura, exigindo que para obter todos os resultados seja necessário executar o algoritmos para cada número de sensores.

5.1. Instâncias

As instâncias utilizadas, apresentadas em Alia e Al-Ajouri [2017], estão apresentadas na Tabela 1. Podemos observar que todas as instâncias possuem a mesma dimensão, mas variam quanto ao raio de sensoriamento do sensor e quanto ao tamanho de célula utilizada.

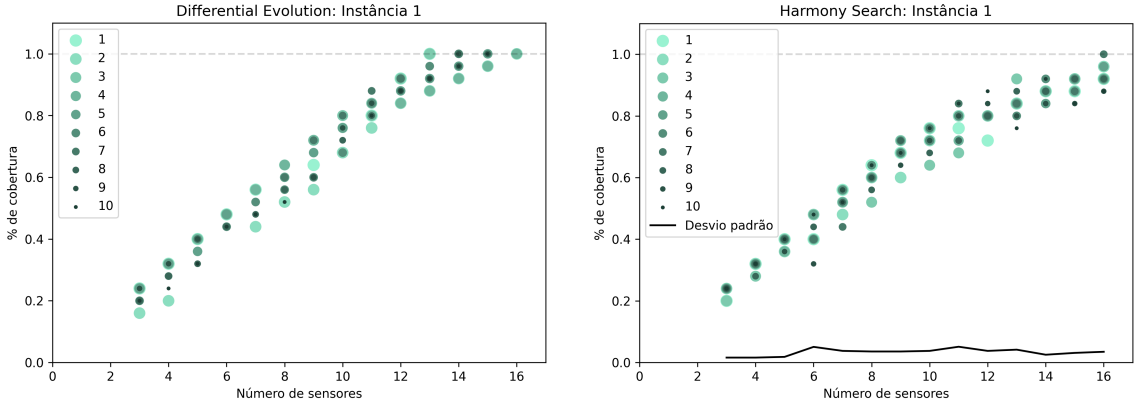
Instância	Dimensão	Raio do sensor	Tamanho da célula
1	50×50	10	10
2	50×50	5	10
3	50×50	10	5

Tabela 1: Instâncias utilizadas nos experimentos computacionais

5.2. Distribuição das soluções

Para comparar a dispersão das meta-heurísticas, foram executadas 10 vezes cada instância e obtendo a fronteira de Pareto para cada uma das execuções. No final foram plotadas as 10 fronteiras de Pareto para cada instância a fim de ser possível observar a distribuição das soluções.

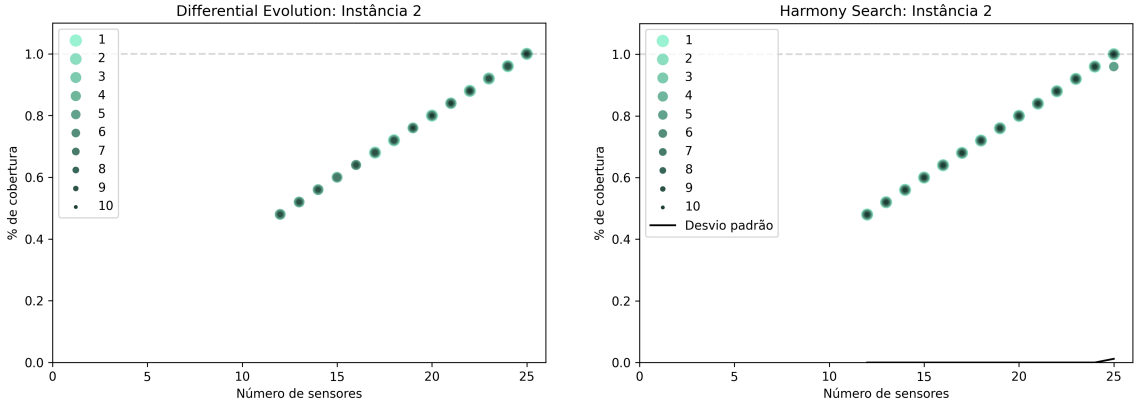
Observando as Figuras 1a e 1b, podemos notar que ambas heurísticas em 10 execuções possuem uma distribuição bastante dispersa, variando bastante em cada execução. Para a Instância 1 é possível observar uma tendência da *Differential Evolution* possuir melhores coberturas em relação à *Harmony Search*.



(a) Differential Evolution (b) Harmony Search

Figura 1: Distribuição das soluções em 10 execuções para a Instância 1

Já para as Figuras 2a e 2b, podemos notar que as soluções para as 10 execuções são praticamente as mesmas. Dessa forma, para a Instância 2, há uma tendência das soluções convergirem e possuírem um comportamento linear. Aprofundando o motivo da linearidade da fronteira de Pareto formada, podemos verificar pela Tabela 1 que ocorre no caso onde há POIs no centro de células 10×10 e com sensores com 5 de raio, ou seja, para cada sensor colocado, necessariamente irá cobrir no máximo um POI. Então, ao executar o algoritmo e otimizar as soluções, chegamos no caso onde para cada quantidade de sensores, cada sensor irá cobrir apenas um POI, gerando o comportamento linear observado.

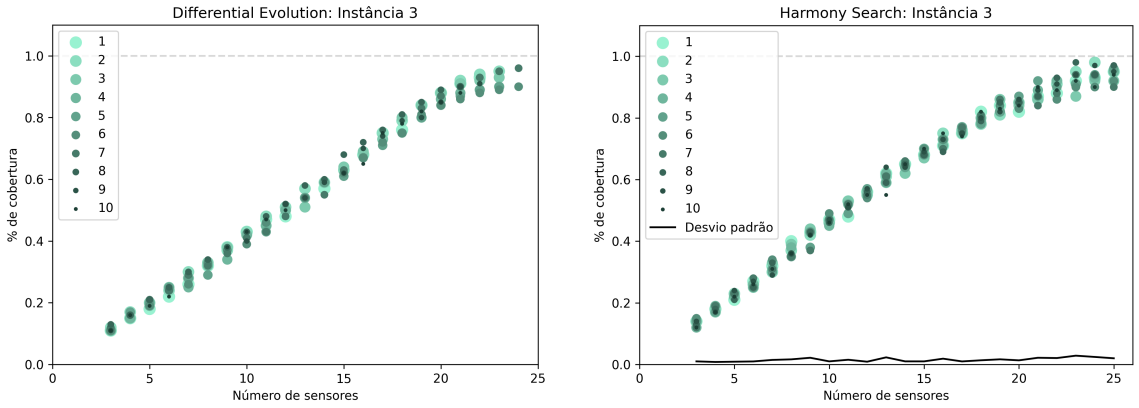


(a) Differential Evolution (b) Harmony Search

Figura 2: Distribuição das soluções em 10 execuções para a Instância 2

E agora para as Figuras 3a e 3b, a distribuição apresenta um certa dispersão, porém variando pouco os valores das coberturas. Ao contrário do observado para a Instância 1, na Instância

3 a *Harmony Search* apresenta um comportamento melhor do que da meta-heurística *Differential Evolution*, onde o centro da curva está mais elevado e atinge valores maiores em algumas execuções.



(a) Differential Evolution

(b) Harmony Search

Figura 3: Distribuição das soluções em 10 execuções para a Instância 3

5.3. Comparação

Com intuito de comparar a fronteira gerada por cada meta-heurística de maneira justa, já que a *Differential Evolution* possui caráter bi-objetivo, enquanto que a *Harmony Search* precisa executar para cada número de sensores separadamente, executamos a heurística de *baseline* 10 vezes, obtendo a média para cada número de sensores e a comparamos com uma única execução da nossa meta-heurística.

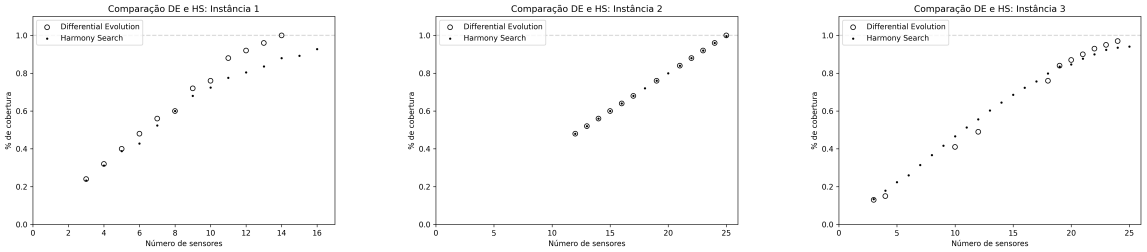
Além de comparar o comportamento por meio dos gráficos, foi calculado o *hypervolume* das soluções encontradas. O *hypervolume* fornece um parâmetro de comparação, já que a heurística DE, ao contrário da HS, nem sempre tem como solução todos os pontos.

Para a primeira instância, a Figura 4a nos mostra um comportamento superior da *Differential Evolution*, onde sua fronteira é nitidamente melhor do que a produzida pela *Harmony Search*. Na Tabela 2 comprovamos isso observando que o *hypervolume* da DE atinge 18.22, enquanto que o da HS atinge 16.428.

Como já foi mostrado nas figuras de distribuição da Instância 2, na Figura 4b não é diferente, o comportamento de ambas se mantém linear, com pouquíssima diferença no último valor encontrado na DE, atingindo um valor mais alto. Na Tabela 2 podemos notar que o *hypervolume* obtidos pela DE e pela HS foram, respectivamente, 84.62 e 84.06, uma diferença de apenas 0.56

E por fim, analisando a Figura 4c, é possível constatar que a *Differential Evolution* não obteve os valores intermediários da curva, e os obtidos, foram inferiores aos da HS. Entretanto, seus valores finais foram ligeiramente melhores do que o da HS, mas apesar disso, seu *hypervolume* é inferior, obtendo 12.725 contra os 12.96 obtidos pela *Harmony Search*.

Um detalhe importante e que deve ser mencionado, é o tempo de execução dos algoritmos aqui expostos. Como a *Harmony Search* precisa ser executada algumas vezes para tirar a média dos valores, o tempo da tabela é o tempo médio de uma execução para encontrar uma fronteira. Analisando os valores, nas duas primeiras instâncias, a *Differential Evolution* possui um tempo bastante acima do apresentado na média da HS. Já na última instância, o contrário ocorre, demonstrando um possível gargalo para o algoritmo da *baseline*.



(a) Instância 1 (b) Instância 2 (c) Instância 3

Figura 4: Comparação entre Harmony Search e Differential Evolution

	Harmony Search		Differential Evolution	
Instância	Hypervolume	Tempo Médio	Hypervolume	Tempo
1	16.428	$18.9068s$	18.22	$48.6789s$
2	84.06	$27.8724s$	84.62	$57.8723s$
3	12.96	$131.508s$	12.725	$110.427s$

Tabela 2: Hypervolume e tempo de execução das meta-heurísticas

6. Conclusões

Desse modo, o objetivo do trabalho de conseguir superar a meta-heurística *baseline* não foi totalmente alcançado. Das três entradas apresentadas, duas obtiveram valores satisfatórios, apesar de que na segunda a diferença seja menos significativa. Na última instância, a *Differential Evolution* conseguiu atingir valores um pouco melhores em quantidades maiores de sensores, entretanto, para valores menores de sensores, os resultados foram inferiores aos da *Harmony Search*.

Além disso, a questão do tempo é de suma importância. A meta-heurística DE na maioria dos casos levou duas vezes mais tempo para atingir o resultado. Em um caso especial, na Instância 3, o tempo de execução foi menor do que a da *baseline*, podendo significar que encontramos uma possível limitação do algoritmo, sendo necessário testar outras instâncias maiores para poder concluir essa suspeita, o que faria a DE se destacar em instâncias grandes.

Compreensivelmente, a implementação da heurística utilizada com *baseline* não será completamente fiel ao proposto pelos autores, pois é baseada em um artigo e em suas ideias apresentadas, não possuindo assim todos os detalhes e algoritmos utilizados. Porém, a ideia principal foi extraída e implementada o mais próximo possível dada as informações fornecidas.

A implementação da *Differential Evolution* proporcionou grande aprendizado e apesar de não superar totalmente a HS, apresentou bons resultados, e diferentemente da *baseline*, realiza uma busca de fato bi-objetiva, embora não encontre um valor de cobertura para todos os números de sensores possíveis.

Como trabalhos futuros, pode-se pensar em melhorias e otimizações possíveis para a implementação aqui apresentada, como encontrar maneiras de contornar os problemas expostos, além de construções de novas meta-heurísticas mais apropriadas para o problema a fim de obter-se resultados melhores e de maneiras mais eficientes.

Referências

Alia, O. M. e Al-Ajouri, A. (2017). Maximizing wireless sensor network coverage with minimum cost using harmony search algorithm. *IEEE Sensors Journal*, 17(3):882–896.

Bajaj, D. e Manju (2014). Maximum coverage heuristics (mch) for target coverage problem in wireless sensor network. In *2014 IEEE International Advance Computing Conference (IACC)*, p. 300–305, Gurgaon, India.

Barkhoda, W. e Sheikhi, H. (2020). Immigrant imperialist competitive algorithm to solve the multi-constraint node placement problem in target-based wireless sensor networks. *Ad Hoc Networks*, 106:102183. ISSN 1570-8705.

Farsi, M., Elhosseini, M. A., Badawy, M., Arafat Ali, H., e Zain Eldin, H. (2019). Deployment techniques in wireless sensor networks, coverage and connectivity: A survey. *IEEE Access*, 7: 28940–28954.

Hanh, N. T., Binh, H. T. T., Hoai, N. X., e Palaniswami, M. S. (2019). An efficient genetic algorithm for maximizing area coverage in wireless sensor networks. *Information Sciences*, 488:58–75. ISSN 0020-0255.

Harizan, S. e Kuila, P. (2020). Evolutionary algorithms for coverage and connectivity problems in wireless sensor networks: A study. In Das, S. K., Samanta, S., Dey, N., e Kumar, R., editors, *Design Frameworks for Wireless Networks*, p. 257–280. Springer Singapore, Singapore.

Kukkonen, S. e Lampinen, J. (2005). Gde3: the third evolution step of generalized differential evolution. In *2005 IEEE Congress on Evolutionary Computation*, p. 443–450, Edinburgh, UK.

Rebai, M., Le berre, M., Snoussi, H., Hnaien, F., e Khoukhi, L. (2015). Sensor deployment optimization methods to achieve both coverage and connectivity in wireless sensor networks. *Computers & Operations Research*, 59:11–21. ISSN 0305-0548.

Sahu, P. e Gupta, S. R. (2012). Deployment techniques in wireless sensor networks. *International Journal of Soft Computing and Engineering*, 2:525–526.

Storn, R. e Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359. ISSN 1573-2916.