Busca Hill-Climbing Modificada aplicada à Observação de Alvos Cooperativos (CTO) movendo-se sobre um Grafo Planar

Levi Porto Figueredo
Universidade Estadual do Ceará
Fortaleza, CE, Brasil
leviportofigueredo@gmail.com

José Everardo Bessa Maia
Universidade Estadual do Ceará
Fortaleza, CE, Brasil
jose.maia@uece.br

ABSTRACT

In the Cooperative Targets Observation (CTO) problem, a group of observing agents with limited vision should be commanded in order to keep the observation of multiple target agents in motion in order to maximize the average number of targets observed during the period under consideration. Targets are cooperative in the sense that they do not run away from observers. This article describes and evaluates the application of a modified Hill-Climbing algorithm to the CTO problem when the movement of the targets is restricted to a planar graph. It is argued how this new configuration of the problem can be representative of a practical situation. The performance of the proposed algorithm surpasses that of two other algorithms widely applied to the problem that are classic Hill-Climbing and K-Means.

KEYWORDS

Cooperative Targets Observation, Multiagent Control, Modified Hill-Climbing Search, K-Means.

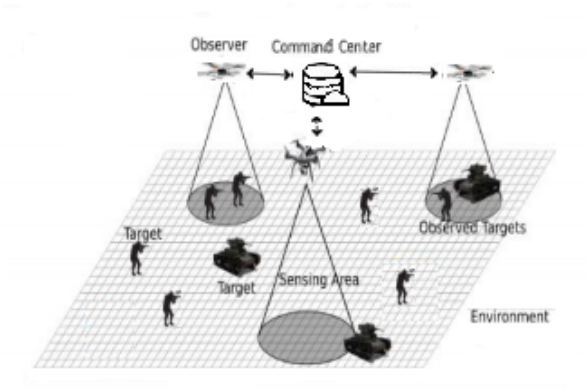


Figure 1: Ilustração do problema CTO. Os observadores são guiados por um comando central para manter a observação dos alvos. Modificado de [1].

1 INTRODUÇÃO

A Figura 1 ilustra o problema *Cooperative Targets Observation* (CTO). Nela os observadores são drones que sobrevoam o ambiente e os alvos, que são de dois tipos, soldados e veículos, se movimentam em terra. Como ilustrado, cada observador tem um raio de visibilidade limitado. Os drones são guiados por um comando central para manter a observação dos alvos em terra. Nesta configuração do problema, as posições dos alvos são informadas, por exemplo, por GPS (Sistema de Posicionamento Global), e os alvos não fogem dos observadores. Alvos e observadores podem funcionar como um time. Em [1] uma classificação geral desta classe de problemas é descrita. O leitor interessado em uma visão mais ampla deve concultar este trabalho.

Em algumas situações frequentes, como em pesquisa sobre a vida selvagem ou na detecção de movimento social de multidões [1], uma entidade, ou grupo delas, deve ser vigiado por outras entidades em movimento, de forma contínua. As entidades que observam podem ser veículos aéreos [2], terrestres ou subaquáticos [3], os quais possuem sensores e habilidades de comunicação. Esses recursos são normalmente limitados, impondo dificuldades na observação. Estes limites são, por exemplo, baixa visibilidade, dificuldades de comunicação ou grandes quantidades de alvos a serem observados por um pequeno número de observadores.

Esse problema foi amplamente discutido em [4], definido como *Cooperative Multi-Robot Observation of Multiple Moving Targets* (CMOMMT). O problema considera a separação de dois tipos de agentes, observadores e alvos, em um ambiente 2D, no qual os obervadores possuem visão parcial do ambiente. O objetivo dos observadores é maximizar o número médio de alvos monitorados no período em consideração. O problema teve uma variação descrita em [5], chamado de *Cooperative Target Observation* (CTO), no qual o ambiente é totalmente observável, além de que, a localização dos alvos são conhecidas.

Esse trabalho introduz uma variação do problema na qual o movimento dos alvos está restrito a grafo planar sobre o ambiente. Para esta configuração, uma modificação do algoritmo *Hill-Climbing*, utilizando uma heurística que favorece a separação entre os observadores, é descrita e avaliada. Os resultados demonstram que a proposta agrega melhoria de desempenho aos algoritmos desta classe existentes.

Após esta breve introdução, o restante deste trabalho está organizado em 6 seções. A Seção 2 revisa um grupo de trabalhos mais diretamente relacionado com esta pesquisa e direciona o leitor para novas variações do problema. A Seção 3 descreve os algoritmos e a Seção 4, as configurações da simulação. Na Seção 5 apresenta-se os resultados obtidos e a Conclusão é mostrada na Seção 6.

2 TRABALHOS RELACIONADOS

O trabalho seminal nesta área é [4]. Ele define um problema mais geral que denominou *Cooperative Multi-Robot Observation of Multiple Moving Targets* (CMOMMT). Neste, os alvos podem reagir à presença dos observadores, por exemplo, procurando fugir para não serem observados. A métrica de performance introduzida nele, que é o número médio de alvos observados, representado por A, é usada

a partir de então em outros trabalhos, e é definida formalmente por:

$$A = \sum_{t=1}^{T} \sum_{j=1}^{n} \frac{g(B(t)_{.j})}{T}, \quad (1)$$

onde, T é o tempo total de observação discreto, n é o número de alvos, $g(B(t)_{.j}) = 1$, se o alvo j é observado por pelo menos um observador, e $g(B(t)_{.j}) = 0$, se o alvo j não é observado. $B(t)_{.j}$ nesse caso é uma função que retorna quais observadores estão vendo o alvo j nos instante t.

A função $g(B(t)_{.j})$ visa garantir que, mesmo um alvo sendo observado por mais de um observador, seja computado somente 1 no somatório.

[5] é o primeiro trabalho encontrado definindo o problema CTO, como utilizado neste trabalho, no qual os alvos executam sua rotina de movimentos e não reagem à presença dos observadores. [5] aplica os algoritmos *Hill-Climbing* e *K-Means* clássicos, isolados e combinados, para definir o movimento dos observadores tanto numa estratégia centralizada quanto em estratégias distribuídas, formando grupos. No limite, ele testa a estratégia totalmente distribuida. Os autores concluem que o algoritmo *Hill-Climbing* apresenta o melhor desempenho.

Para fins de comparação, a proposta do presente trabalho é comparada com a de [5] nas mesmas configurações de teste, além de que as soluções são testadas e comparadas com a nova configuração na qual os alvos se movimentam sobre um grafo planar. Este trabalho foi escolhido para comparação porque, além de ser uma solução *baseline* para a configuração CTO, ele é também uma pesquisa reprodutível, uma vez que os autores fornecem informações completas e precisas sobre a proposta e a implementação.

O problema CTO foi estudado em trabalhos mais recentes como em [6–8]. Entretanto estas publicações não abordam a configuração de mobilidade dos alvos introduzida no presente trabalho.

3 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta o algoritmo proposto assim como aqueles utilizados na comparação de desempenho. Inicialmente, são descritos os algoritmos *Hill-Climbing* e *K-Means* clássicos e as modificações realizadas por [5] na sua aplicação ao problema CTO em um ambiente com movimento irrestrito. Em seguida é descrita a modificação proposta para melhorar o desempenho no ambiente com movimento dos alvos restritro a um grafo planar.

3.1 O Algoritmo *K-Means*

O termo *K-Means* apareceu em [9] propondo um método de quantização vetorial para sinais multivariados. *K-Means* é um método de agrupamento em K grupos iterado em dois passos: após uma inicialização dos centros, no primeiro passo, cada ponto é atribuído a um dos K grupos, aquele de menor distância ao centro atual, utilizando alguma medida de distância ao centro; no segundo passo, os centros dos grupos são recalculados considerando as últimas atribuições. Esse processo é repetido até convergir, geralmente para um mínimo local.

O algoritmo *K-Means* é adaptado em [5] para o problema CTO conforme mostrado no Algoritmo 1. O algoritmo é utilizado para decidir o próximo movimento dos observadores a partir da sua posição atual. Os alvos em suas posições atuais são os pontos a serem agrupados tomando as posições dos observadores como centros. Em cada execução do *K-Means*, K fica sendo igual ao número do observadores e suas posições atuais são usada como posições iniciais dos centros dos grupos. Após convergir, as posições finais são tomadas para calcular os próximos destinos dos observadores, pela relação:

$$K_i = (1 - \beta)K_i + \beta m_i, \quad (2)$$

onde K_i o centro do i-ésimo *cluster*, m_i sendo o centro final de todos os pontos associados ao i-ésimo observador calculado pelo *K-Means* e o β o tamanho do passo do observador. Note que o algoritmo posiciona os próximos destinos dos observadores em pontos intermediários entre as posições atuais e os destinos indicados pelo *K-Means*, determinadas por β.

Algoritmo 1: Algoritmo *K-Means*

Entrada: Vetor de posições atuais de cada observador
Saída: Vetor de posições desejadas de cada observador
início
  Selecione cada posição atual do observador para ser o centro de conjunto;
  enquanto *houver mudança significativa de localidade* **faça**
    Associe cada alvo em cada conjunto, baseado na menor distância entre eles e os observadores;
    Atualizar o vetor de posições desejada dos observadores baseado no local do ponto médio das posições dos alvos até este observador no *cluster*;
  fim
fim

3.2 O Algoritmo *Hill-Climbing* [5]

Hill-Climbing é um algoritmo guloso de busca local para otimização de funções que consiste basicamente em gerar novas soluções perturbando várias vezes a solução atual e escolhendo aquela com maior contribuição para melhorar a função de avaliação.

O trabalho [5] faz modificações no *Hill-Climbing* original para adaptá-lo ao problema CTO conforme mostrado no Algoritmo 2. Nessa adaptação, a cada chamada do algoritmo são simuladas 1000 pertrubações. Para cada iteração, um dos observadores é escolhido de forma aleatória. Considerando a solução pai como sendo a original e a filha, a simulada, o algoritmo adotará a escolha da próxima solução, naquela iteração do observador escolhido, baseada nos seguintes critérios de qualidade:

(1) Aceita a filho se este observar mais alvos;
(2) Caso observe a mesma quantidade de alvos, computa

$$H = \sum_{o \epsilon O} \sum_{t \epsilon T} \begin{cases} dist(o,t), & \text{se R/2 < dist(o,t) < R;} \\ 0, & \text{caso oposto,} \end{cases} \quad (3)$$

onde dist(o,t) é a distância Euclidiana entre o observador $o \epsilon O$ e o alvo $t \epsilon T$. Aceita a solução filha se $H_{filha} < H_{pai}$. Isso motiva os observadores a irem ao encontro de alvos que estão mais próximos;

Algoritmo 2: Algoritmo *Hill-Climbing*

```
Entrada: Vetor de posições atuais de cada observador
Saída: Vetor de posições desejadas de cada observador
início
    para 1000 execuções faça
        Escolha aleatório um observador;
        Escolha um ponto baseado em um quadrado, centrado no observador escolhido, com tamanho metade do ambiente;
        se posição escolhida for melhor que a atual então
            Coloque esta posição no índice referente àquele observador no vetor de posições desejadas.
        fim
        Reduza em 1% o quadrado de movimento centrado naquele observador;
    fim
fim
```

(3) Se $H_{filha} = H_{pai}$ e a solução filha e a pai observarem a mesma quantidade de alvos, então computa:

$$G = \sum_{o \epsilon O'}^{t \epsilon T'} min(dist(o, t))$$

onde O'é o conjunto de observadores que não observam nenhum alvo e T'é o conjunto de alvos que não são observados por nenhum observador. Aceita a solução filha se $G_{filha} < G_{pai}$. Isso motiva os observadores que não observam nenhum alvo a ser aproximar de alvos que não são observados por nenhum observador;

(4) Senão, rejeita a posição filha e segue para o próximo observador.

3.3 O algoritmo *Hill-Climbing* modificado proposto (MHC)

O algoritmo *Hill-Climbing* como descrito e modificado por [5] foi testado no ambiente com movimentos restritos sobre um grafo planar, havendo um ganho de desempenho em relação ao algoritmo clássico pelo fato de que o grafo força a um maior espaçamento entre alvos, em relação ao caso de movimento irrestrito. Entretanto, notou-se que frequentemente existe interseção nas coberturas dos observadores nas soluções, o que é um desperdício pois isso leva a que mais de um observador cubra um mesmo alvo. Com isto, houve a motivação de criar uma heurística que visa aumentar o espaçamento entre os observadores.

O algoritmo MHC é mostrado no Algoritmo 3. A modificação em *Hill-Climbing* feita para MHC foi que além de o procedimento aceitar uma solução derivada (filho) caso esta observe mais alvos que a solução atual (pai), caso ocorra uma situação de o filho e o pai observarem a mesma quantidade de alvos, MHC irá aceitar o filho que, além disso, observar uma menor quantidade de outros observadores. A intuição por trás desta heurística é que ela incentiva o distanciamento entre observadores, desde que não sacrifique o índice A, o que é desejado. O algoritmo MHC foi avaliado por simulação.

Algoritmo 3: Algoritmo modificado MHC

```
Entrada: Vetor de posições atuais de cada observador
Saída: Vetor de posições desejadas de cada observador
início
    para 1000 execuções faça
        Escolha aleatório um observador;
        Clone sua posição atual e some um valor entre -10 a 10 no X ou Y desta posição clonada;
        se posição clonada observar mais alvos então
            Coloque esta posição no índice referente àquele observador no vetor de posições desejadas.
        fim
        senão se posição clonada observar mesma quantidade de alvos então
            se posição clonada observar menos observadores então
                Coloque esta posição no índice referente àquele observador no vetor de posições desejadas;
            fim
        fim
        Reduza em 1% o quadrado de movimento centrado naquele observador;
    fim
fim
```

4 SIMULAÇÃO

Todas as simulações foram realizadas no ambiente de simulação Multi-Agente MASON [10]. O ambiente simulado é um retângulo 2D no qual alvos e observadores realizam seus movimentos. Os algoritmos são comparados em duas configurações de ambientes: o ambiente irrestrito, no qual os alvos realizam um passeio livre aleatorio e o ambiente restrito, no qual os movimentos dos alvos está restrito a acontecer sobre um grafo planar. Os parâmetros comuns das simulações foram os seguintes:

(1) Comprimento e largura do ambiente: 150 × 150 unidades;
(2) Passos executados por simulação: 1500;
(3) Número de simulações executadas por dados: 30;
(4) Número de alvos: 24;
(5) Número de observadores: 12;
(6) Velocidade dos observadores: 1 unidade por passo;
(7) Para o ambiente com grafo planar, número de vértices: 40;
(8) α de execução dos algoritmos: 10.
(9) β do *K-Means*: 0,25.

Os algoritmos, proposto e já existentes na literatura [5], são rodados em paralelo, a cada α passos, junto com a movimentação dos observadores. Ao término de uma execução do ciclo de qualquer um desses algoritmos, o retorno será a posição desejada do observador, que irá caminhar buscando atingir aquela posição. A estrutura geral da simulação está apresentada no algoritmo 4.

Uma imagem do primeiro ambiente gerada pelo MASON está mostrada na Figura 2. O movimento dos alvos nesse ambiente ocorre assim: sendo $p_i = [x_i, y_i]$ a posição atual do alvo i, ele calcula as coordenadas do próximo destino somando ou subtraindo à posição atual um valor aleatório de amplitude 40, ou seja $p_i =$

Algoritmo 4: Algoritmo da simulação

Entrada: Vetor das posições iniciais de alvos e observadores e parâmetros da simulação.
Saída: Visualização da simulação e cálculo do índice A.
início
  para *t = 1:T* **faça**
    Movimentar alvos e observadores na direção dos destinos atuais;
    Calcular e armazenar $g(B(t)_{.j})$ para $j = 1...n$;
    Atualizar os destinos dos alvos;
    se $t \% \alpha == 0$ **então**
      Executar o algoritmo de comando do destino dos observadores (KM, HC ou MHC);
      Atualizar os destinos dos observadores;
    fim
  fim
  Calcular o índice *A* pela Equação 1.
fim

$p_i \pm random(40, 40)$, que é aproximadamente um quarto das dimensões do ambiente. Isso é justificado em [5] pelo fato de que assim evita que muitos alvos se cruzem no centro do ambiente. Se o ponto gerado cair fora das fronteiras do ambiente ele é descartado e um novo ponto é gerado.

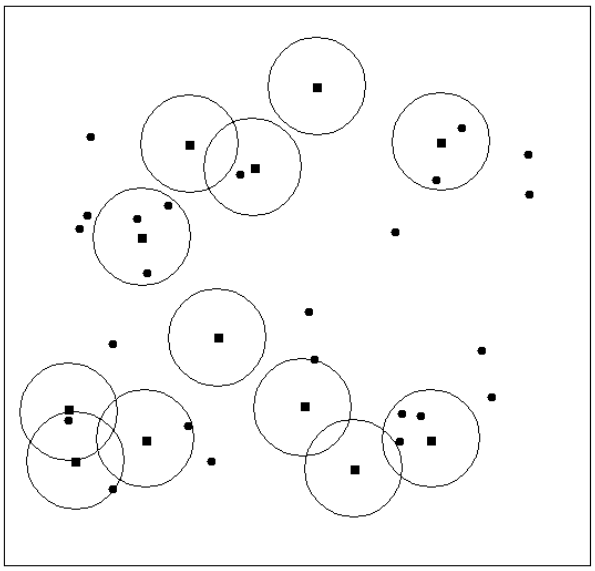


Figure 2: Instantâneo da simulação no ambiente com movimento irrestrito.

A Figura 3 mostra um instatâneo do ambiente grafo planar gerado no MASON. Neste ambiente considera-se que os destinos dos alvos são sempre vértices. Para determinar o próximo destino, cada alvo seleciona um vértice aleatório na lista de vértices do grafo e segue pelo caminho mais curto sobre o grafo. Para determinar o caminho mais curto é utilizado o algoritmo de Dijkstra [11]. Note entretanto, que apenas os alvos tem movimento restrito, os observadores podendo mover-se livremente.

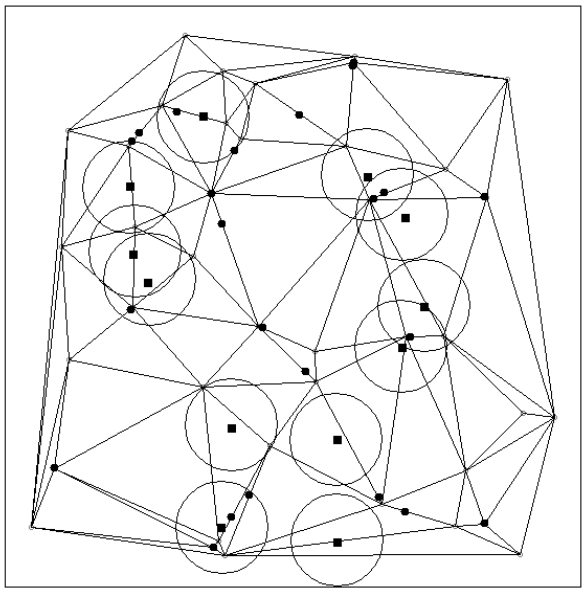


Figure 3: Instantâneo da simulação no ambiente onde o movimento dos alvos está restrito a um grafo planar. Em aplicações, esse grafo pode representar uma malha de ruas ou estradas.

5 RESULTADOS

Nesta seção mostra-se os resultados dos experimentos de simulação. Cada simulação correu por 1500 passos e foram executadas 30 execuções com posições iniciais aleatórias de alvos e de observadores para permitir que testes estatísticos sejam realizados. Como em [5], considerou-se que os observadores são mais rápidos que os alvos e isso é especificado pelo parâmetro V_r que é a velocidade relativa dos alvos em relação à velocidade dos observadores, que é constante.

Os experimentos se referem aos aos algoritmos descritos nas seções anteriores: *K-Means*, *Hill-Climbing* e MHC. Foram tirados resultados nos dois tipos de ambientes: os alvos com movimento livre em um passeio aleatório e os alvos movimentando-se sobre um grafo planar. Além disso, para cada ambiente foram produzidos dois gráficos. Em um deles a velocidade relativa dos alvos é fixa em V_r = 0,5 e varia-se o raio de visão dos observadores no conjunto X = {15, 20, 25}. No outro, o raio de visão é fixado em X = 20 e varia-se a velocidade relativa dos alvos no conjunto V_r = {0,1; 0,25; 0,5; 0,75; 0,9}.

A Tabela 1 mostra as médias e desvios padrão de cada cenário descrito no texto, considerando os três algoritmos, inseridos nos dois tipos de ambiente. Os dados desta tabela são plotados e analisados nas próximas duas subseções.

5.1 Ambiente sem Grafo

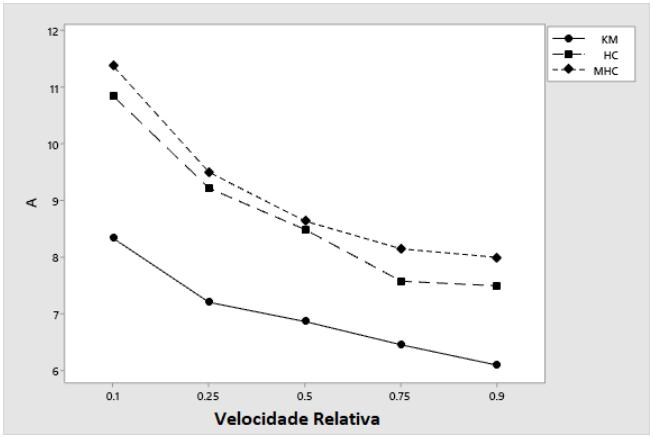
A Figura 4 apresenta os resultados para o ambiente com movimento irrestrito no qual os alvos realizam um passeio aleatório livre.

Table 1: Média (m) e desvio padrão (dp) de A em 30 corridas nos ambientes irrestrito e grafo planar quando se varia o raio de visão R ou a velocidade relativa V_r.

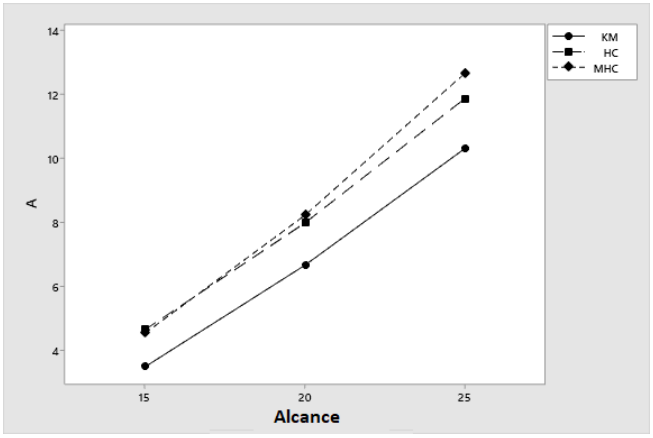
Irrestrito	R/Vr	A	15	20	25	0.10	0.25	0.50	0.75	0.90
	KM	m	3.49	6.67	10.32	8.34	7.21	6.87	6.46	6.10
		dp	0.16	0.38	0.47	0.42	0.42	0.30	0.32	0.30
	HC	m	4.65	7.99	11.88	10.85	9.22	8.49	7.58	7.50
		dp	0.10	0.12	0.16	0.42	0.42	0.30	0.32	0.30
	MHC	m	4.54	8.23	12.67	11.38	9.50	8.64	8.15	8.00
		dp	0.10	0.17	0.24	0.27	0.26	0.13	0.18	0.15
G. Planar	R/Vr	A	15	20	25	0.10	0.25	0.50	0.75	0.90
	KM	m	4.89	8.49	10.00	10.92	9.04	8.40	8.05	6.86
		dp	1.11	1.97	2.10	1.40	2.02	1.49	1.70	0.89
	HC	m	8.70	11.10	13.54	13.47	13.11	12.32	12.26	12.12
		dp	1.18	0.84	1.11	0.83	0.96	0.82	0.76	0.94
	MHC	m	8.19	12.78	15.10	15.00	13.77	13.10	12.80	12.70
		dp	1.47	1.16	1.19	1.04	0.85	1.08	1.18	1.17

Na Figura 4(a), com range fixo em R = 20, nota-se que há uma clara ordenação nos desempenhos sendo o algoritmo proposto superior aos demais. Embora a proposta seja o movimento sobre um grafo, ele também apresentou melhor desempenho no ambiente irrestrito. Nota-se também desse gráfico que a variação do desempenho quando a velocidade relativa dos alvos varia 0,1 à 0,9, ou seja, de 10% até 90% da velocidade do observador, os valores de A variam pouco. Isso se deve ao fato desta curva ter sido traçada para o valor relativamente grande R = 20. É intuitivo que para raios de observação grandes o efeito de V_r seja menor.

A Figura 4(b) mostra que todos os algoritmos são mais sensíveis ao raio de visão que à velocidade relativa, mas ainda assim o algoritmo proposto é vitorioso. Quando o raio visão varia entre 15 e 25 a performance A varia de 4 para 13 para o algoritmo MHC, e de 3 para 10 para o *K-Means*.



(a) Variando a velocidade relativa com range fixo igual a 20.



(b) Variando o range com a velocidade relativa fixa igual a 0,5

Figure 4: Desempenho dos algoritmos para o ambiente onde o movimento dos alvos é um passeio aleatório irrestrito: MHC (Losango), KM (Círculo), HC (Quadrado).

5.2 Ambiente com Grafo

A Figura 5 apresenta os resultados para o ambiente restrito no qual os alvos realizam um passeio aleatório sobre um grafo planar. O desempenho relativo geral dos algoritmos foi semelhante com MHC superando *K-Means* e *Hill-Climbing*.

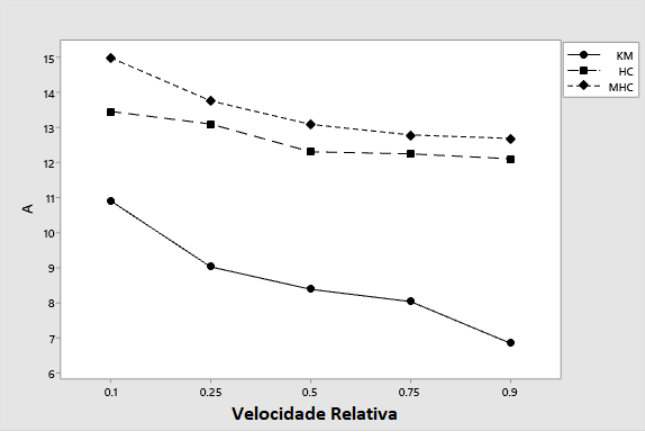
Na Figura 5(a) está mostrada a variação de A com o range, para V_r=0,5. Para MHC, o de melhor desempenho está entre os ranges 14 e 15 enquanto que no ambiente irrestrito essa faixa fica entre 11 e 12. Para o *K-Means*, o de menor desempenho, o valor cai da faixa 7 à 11 para a faixa 6 à 8.

A Figura 5(b) apresenta o desempenho A com a variação da velocidade relativa, para R=20. O desempenho de *Hill-Climbing* fica na faixa entre 9 e 12 enquanto que no ambiente irrestrito era entre 5 e 12. Para o MHC a variação foi de 8 a 15, enquanto que no ambiente irrestrito, foi de 5 a 13.

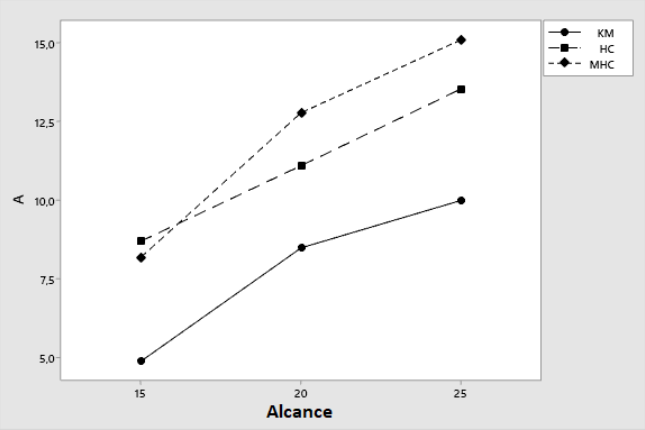
6 CONCLUSÃO

Foi apresentada uma nova configuração do problema CTO na qual os alvos movimentam-se sobre um grafo planar. Grafos planares podem representar ruas, estradas e outras limitações de movimento. Foi proposto e avaliado um algoritmo *Hill-Climbing* modificado (MHC) para guiar os observadores numa abordagem centralizada. O desempenho, utilizando a métrica A que é o número médio de alvos observados no período, foi comparado com o de dois algoritmos publicados em [5], os quais são seminais na abordagem desta classe de problemas por métodos de agrupamento. Para controle da comparação, os mesmo parâmetros da simulação foram adotados e os testes foram realizados também na configuração original do problema. Os resultados mostram que o algoritmo aqui proposto supera os concorrentes nesta nova configuração do problema.

Uma limitação importante deste trabalho é a configuração de comando centralizado. O comando centralizado não é de todo irrealista para muitas aplicações mas ele sofre do problema de escalabilidade. A evolução deste trabalho está em andamento em duas direções. A primeira é o desenvolvimento de uma configuração com comando descentralizado. Na segunda pretende-se considerar uma configuração na qual o movimento dos alvos apresente alguma regularidade desconhecida, como em [4], além de aplicar técnicas de aprendizagem de máquina para melhorar o desempenho dos observadores. Essa configuração também apresenta interesse prático.



(a) Variando a velocidade relativa com range fixo igual a 20.



(b) Variando o range com a velocidade relativa fixa igual a 0,5

Figure 5: Desempenho dos algoritmos para o ambiente onde o movimento dos alvos é restrito a está sobre um grafo: MHC (Losango), KM (Círculo), HC (Quadrado).

REFERENCES

[1] Asif Khan, Bernhard Rinner, and Andrea Cavallaro. Cooperative robots to observe moving targets. *IEEE Transactions on Cybernetics*, 48(1):187–198, 2018.

[2] J. Banfi, J. Guzzi, A. Giusti, L. Gambardella, and G. A. Di Caro. Cooperative control of uavs for localization of intermittently emitting mobile targets. page 5411–5418, 2015.

[3] Juliane Kuhn, Christian Reinl, and Oskar Von Stryk. Predictive control for multi-robot observation of multiple moving targets based on discrete-continuous linear models. *IFAC Proceedings Volumes*, 44(1):257–262, 2011.

[4] Lynne E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation & Soft Computing*, 5(1):5–19, 1999.

[5] Sean Luke, Keith Sullivan, Liviu Panait, and Gabriel Balan. Tunably decentralized algorithms for cooperative target observation. pages 911–917, 2005.

[6] Rashi Aswani, Sai Krishna Munnangi, and Praveen Paruchuri. Improving surveillance using cooperative target observation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[7] Sai Krishna Munnangi and Praveen Paruchuri. Improving wildlife monitoring using a multi-criteria cooperative target observation approach. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[8] Joao PB Andrade, Robson Oliveira, Thayanne F da Silva, José E Bessa Maia, and Gustavo AL de Campos. Organization/fuzzy approach to the cto problem. In *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 444–449. IEEE, 2018.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. pages 281–297, 1967.

[10] Sean Luke, Claudio Cioffi-Revilla, Keith Sullivan Liviu Panait, and Gabriel Balan. Mason: A multi-agent simulation environment. *Simulation: Transactions of the society for Modeling and Simulation International*, 82(7):517–527, 2005.

[11] E.W. Dijkstra. A note on two problems in connexion with graphs. *J. Numeris. Math*, pages 269–271, 1959.