**Algoritmo 1: Pseudocódigo do Swarm-GAP**

1: token de recepção

2: Calcular recursos disponíveis ri

3: para todas as tarefas disponíveis

4: capacidade de computação kij

5: Calcular tendência T ij (st)

6: se roleta () <T ij (st) e ri cj então

7: Alocar tarefa j para o agente i

8: Diminuir o recurso ri

9: Marcar agente conforme visitado no token

10: se ainda houver tarefas disponíveis,

11: Envie o token para um agente ainda não visitado

4. Solução proposta

A proposta baseada no Swarm-GAP apresentada neste trabalho está dividida em três variações. Esta seção descreve cada um deles, assim como a motivação que explicam suas propostas.

**4.1. Loop de alocação (AL)**

Quando o EXAME-GAP é usado para resolver o problema de alocação de tarefas Seção 2, os recursos dos VANTs não foram levados em consideração: muitos as tarefas não foram executadas e os recursos foram subutilizados. Para aproveitar o código deste método. Em AL, em vez de derrubando o token depois que todos os UAVs o receberam, a lista de agentes visitados o token é limpo e uma nova rodada de envio de tokens é iniciada. Assim, cada UAV pode receber o token mais de uma vez e as tarefas não alocadas terão um novo chance de ser alocado. No entanto, um simples esquema de limpeza desta lista pode não é eficaz. Quando um token ainda tem tarefas não alocadas, mas os UAV já alocados todos os seus recursos, o token é desnecessariamente mantido em circulação, causando uma troca desnecessária de mensagens entre os UAVs.

Para evitar que o token permaneça circulando para sempre, após limpar a lista de agentes visitados, os agentes que ainda têm capacidade ou recursos disponíveis para executando tarefas, são inseridas nesta lista novamente. Então, apenas agentes com algum A probabilidade de alocação de tarefas poderá receber o token na próxima rodada. Para isso, quando um UAV recebe um token, depois de selecionar as tarefas e se marcar como visitada (linhas 1 a 3), informa o token se tem ou não disponibilidade para executar qualquer uma das tarefas restantes (linha 5). Em seguida, o token armazena uma lista de agentes que não devem mais receber o token nas próximas rodadas, chamada lista de indisponíveis, porque não podem mais selecionar nenhuma outra tarefa contida no token.

**4.1.1. Loop de Classificação e Alocação (SAL)**

No método anterior, uma vez que a escolha das tarefas é feita na ordem em que o tarefas estão no token, muitas vezes os UAVs acabam selecionando primeiro as tarefas que não são aqueles mais adequados para eles. Então, pode causar falta de recursos para outras tarefas mais apropriado para eles. Para lidar com esse problema, um mecanismo de classificação que complemente o AL é proposto. Esse algoritmo é chamado de classificação e alocação Loop (SAL).

O pseudo código da SAL é apresentado no Algoritmo 3. Na SAL, as tarefas são classificados por pendência em ordem decrescente (linha 3 a 6). Isso facilita a seleção de uma tarefa pelo UAV, ou seja, torna mais fácil para os UAVs selecionar mais

**Algoritmo 2: Pseudocódigo - Loop de alocação (AL)**

1: token de recepção

2: Selecione as tarefas que irá realizar (linhas 2 a 8 do Algoritmo 1)

3: Marcar agente conforme visitado no token

4: se ainda houver tarefas disponíveis,

5: Informar o token se tiver disponibilidade para executar qualquer uma dessas tarefas

6: se todos os agentes já receberem o token,

7: Limpe a lista de agentes visitados

8: Preencher lista de agentes visitados com os agentes indisponíveis

9: Envie o token para um agente ainda não visitado tarefas apropriadas para eles. Este tipo dá prioridade às tarefas com maior tendência, evitando a seleção de tarefas com menor pendência.

**Algoritmo 3: Pseudocódigo - SAL**

1: token de recepção

2: Calcular recursos disponíveis ri

3: para todas as tarefas disponíveis

4: capacidade de computação kij

5: Calcular tendência T i ij

6: Ordenar tarefas por tendência decrescente

7: para todas as tarefas disponíveis classificadas por tendência

8: O mesmo que as linhas 6 a 8 do Algoritmo 1

9: O mesmo que as linhas 3 a 9 do Algoritmo 2

**4.1.2. Loop de limite e alocação (LAL)**

Pelo jeito que foi concebido, o algoritmo SAL provoca um aumento na quantidade de tarefas executadas. No entanto, foi observado nos experimentos que, em algumas corridas, havia UAVs que ainda permaneciam inativos, ou seja, UAVs que não executou nenhuma tarefa. Quando os UAVs têm recursos suficientes, o primeiro deles é visitado por UAV em cada rodada. Este limite significa que um UAV não alocar mais de uma tarefa toda vez que receber o token. Assim, todos os UAVs tem a chance de escolher uma tarefa dentro de uma rodada e a carga de trabalho está em resultado mais equilibrado.

5. Configuração experimental

A fim de avaliar as soluções propostas para o problema de alocação de tarefas entre os UAV, foi implementada uma simulação no NetLogo [13] (versão 5.3.1).

As tarefas e os UAVs são representados, cada um, por um símbolo específico. Figura

1 ilustra a interface gráfica de simulação, onde o símbolo para as tarefas é representado por uma forma \ X "e o símbolo para um UAV tem a forma de um avião.

De acordo com o problema formulado (Seção 2), cada tarefa tem sua localização (x, y), um tipo de destino associado e a quantidade de recursos necessários para sua execução. A diferença nas cores das tarefas representa os diferentes tipos de alvos. Cada UAV, além de sua localização (x, y) e sua lista de sensores, tem "fazer" lista de tarefas. Esta lista começa vazia, e quando o UAV seleciona uma tarefa, adiciona a tarefa a esta lista. A lista de tarefas é uma FIFO: a primeira tarefa selecionada é o primeiro a ser realizado.

A simulação foi implementada para que, a cada tick, todos os UAVs movam um pixel. Os ticks são usados ​​como makespan (tempo total que decorre desde o início até o final da execução). Nunca excederá o prazo da missão. O a execução termina quando os UAV concluem todas as tarefas que selecionaram para fazer. Isto é Vale ressaltar que o makespan pode ser facilmente traduzido em distância percorrida por cada UAV, já que todos os UAVs movem um pixel por tick.

Apenas um token é liberado para os UAVs. O token contém a missão tarefas e uma lista de agentes visitados. Esta lista começa vazia e cada UAV que receber o token adicionar seu ID nele. A cada tick, o token é enviado para um aleatório UAV que ainda não recebeu o token na rodada atual, ou seja, um UAV que não está na lista de agentes visitados. O token também possui uma lista de agentes indisponíveis.

Esta lista é usada pelos algoritmos AL, SAL e LAL (consulte a Seção 4.1). Quando um UAV recebe o token, ele executa o algoritmo para escolher as tarefas que ele irá executar.

Vários cenários experimentais foram criados para analisar diferentes situações. Os cenários variam em quantidade de tarefas de missão, número de agentes e tamanho da área (pixels) em que eles estão situados. Conforme descrito a seguir.

i) 3 UAVs; 4 tarefas; 300 carrapatos como prazo final; Tamanho da área de 100 x 80 px.

ii) 3 UAVs; 8 tarefas; 300 carrapatos como prazo final; Tamanho da área de 100 x 80 px.

iii) 3 UAVs; 16 tarefas; 300 carrapatos como prazo final; Tamanho da área de 100 x 80 px.

iv) 3 UAVs; 32 tarefas; 300 carrapatos como prazo final; Tamanho da área de 100 x 80 px.

v) 6 UAVs; 64 tarefas; 300 carrapatos como prazo final; Tamanho da área de 200 x 160 px.

vi) 9 UAVs; 96 tarefas; 300 carrapatos como prazo final; Tamanho da área de 300 x 240 px.

Cada conjunto de UAVs {com 3, 6 e 9 UAVs {foram criados aleatoriamente uma vez.

Por exemplo, o conjunto de 3 UAVs foi criado uma vez e usado nos cenários i, ii, iii, iv. A localização dos VANTs (x, y) foram definidos por valores aleatórios, respeitando o limite da área do cenário. O número de sensores que cada UAV foi equipado como bem como os tipos de sensores também foram atribuídos com valores aleatórios. Cada UAV foi equipado com um ou dois sensores dos tipos s0, s1, s2 ou s3.

O mesmo foi aplicado para as tarefas. A localização das tarefas (x, y) e o tipo de alvo foram definidos por valores aleatórios. Cada tarefa foi atribuída a um único tipo de alvo, que pode ser dos tipos a0, a1, a2 ou a3. Apenas a quantidade de recursos necessário para executar uma tarefa (cj) foi xed em dez unidades de tempo (ticks) para todos.

A Tabela 1 apresenta a qualidade dos sensores para detectar cada tipo de alvo. Este valores foram utilizados em todos os cenários. De acordo com essa configuração, o alvo a0, por exemplo, pode ser detectado pelos sensores s0 e s2, mas com diferentes qualidades.

Tabela 1: Qualidade dos sensores para detectar os tipos de alvo

Sensor / alvo a0 a1 a2 a3

s0 1: 0 0 0: 3 0: 5

s1 0 0 1: 0 0

s2 0: 2 0 0 1: 0

s3 0 1: 0 0 0: 3

Os cenários foram executados 30 vezes para cada algoritmo. Em todos os experimentos,

A capacidade do UAV foi calculada usando a Equação 2 com? = 0: 6, dando maior importância para o fator distância. Valores de estímulo diferentes também foram testados, mas apenas resultados com o valor 0,6 são mostrados, pois este valor apresentou melhor resulta na maioria dos casos, e também concorda com o que é conhecido pelo Swarm-GAP.

Os experimentos foram conduzidos em um PC com 1.70GHz, 4GB de RAM e SO Windows 8.1 Pro 64 bits. Os resultados obtidos utilizando as soluções propostas bem como Swarm-GAP são apresentados e discutidos na próxima seção.

6. Resultados e análises experimentais

Para avaliar os métodos propostos, as seguintes medidas foram sidered: (a) Recompensa total {o objetivo principal é maximizá-lo; (b) Makespan, que é o tempo total que decorre do começo ao fim da execução.

ção; c) Quantidade de tarefas concluídas; d) Qualidade das tarefas concluídas;

(e) Número de mensagens trocadas {relacionadas com a passagem do token; e

(f) tempo de execução do algoritmo.

Estas medidas são apresentadas para os métodos AL, SAL e LAL e comparado aos obtidos usando o Swarm-GAP. Os resultados completos com todas as medidas sures (média e desvio padrão) dos cenários i a vi são apresentados no

Tabelas 2 e 3. No entanto, apenas um conjunto específico destes cenários é utilizado para acenda os resultados para cada medida. Após a apresentação, os resultados são discutidos.

No final desta seção, uma análise da escalabilidade dos algoritmos é também feito.

A seguir, os cenários i, ii, iii e iv, que são compostos por 4, 8, 16 e 32 tarefas, respectivamente, são usadas para demonstrar os resultados da recompensa total, makespan, quantidade e qualidade das tarefas concluídas e número de trocou mensagens. Estes quatro cenários foram escolhidos porque têm o mesmo número de UAVs ea quantidade de tarefas é variada, mas com a mesma missão de data limite. Permite avaliar situações nas quais os UAVs possuem tempo mais que suficiente e o ambiente tem poucas tarefas (cenários i e

ii), e os UAVs têm pouco tempo e há muitas tarefas a serem executadas

(cenários iii e iv). No primeiro, os UAV podem realizar todas as tarefas, mas eles

deve fazê-lo no menor tempo possível. Por outro lado, na segunda situação

Os UAVs não têm tempo suficiente para executar todas as tarefas. Então, eles devem executar

tantas tarefas quanto possível.

6.1. Recompensa total

A recompensa total de cada método para diferentes cenários é mostrada na Figura

2. Nos cenários com 4 e 8 tarefas, o algoritmo AL superou o Swarm-

8. Conclusão

Este artigo propôs uma solução para o problema de alocação de tarefas em uma equipe de UAVs de forma descentralizada. O problema enfrentado pressupõe que as tarefas são criados por uma entidade central, como em várias operações militares, e para a atividade de monitoramento de uma área com o objetivo de detectar tipo de alvo.

A solução proposta foi desenvolvida a partir do algoritmo Swarm-GAP, que use uma abordagem de inteligência enxame com base no modelo de limite de resposta. Através observações de experimentos, foram identificadas características que poderiam promover alocação de tarefas e, consequentemente, aumenta a recompensa total. As características foram usados para evoluir a solução, resultando em três variações de algoritmo:

Loop de Alocação (AL), Loop de Ordenação e Alocação (SAL) e Limite e Alocação

Loop (LAL). Esses algoritmos foram avaliados, apresentando resultados positivos comparado ao Swarm-GAP.

Nos experimentos, assumiu-se que a comunicação entre os UAVs é completo e nunca falha. No entanto, em um ambiente real, esta situação raramente acontece. Assim, um importante trabalho futuro é considerar a tolerância a falhas. Outro A questão a ser abordada é a dependência entre tarefas e trabalho em equipe (por exemplo, dois ou mais agentes são necessários para executar a mesma tarefa).