Modelagem em multithread da interação entre colônias de formiga

Germano Andrade, João Alcindo e Tiago da Silva

14 de Abril de 2022

Conteúdo

1 Introdução

Neste documento, vamos descrever os aspectos que enformaram nossas implementações e caracterizaram as nossas decisões de modelagem; contemplaremos, além disso, situações tremendamente atribuladas que caracterizaram tanto a implementação serial – em uma thread – quanto a multithread deste sistema. Na seção seguinte, portanto, introduzimos os atributos gerais de nosso programa, enfatizando como os agentes – as formigas, os formigueiros, as comidas e os objetos subjacentes, como os mapas e as coordenadas – foram desenhados, e apontando, também, para a contemplação dos mecanismos que ensejam sua interação.

Na seção subsequente, vislumbramos os alicereces que culminaram na versão mulithread do programa, explicitando a utilização de variáveis de exclusão mútua, de variáveis de condição e de semáforos com o objetivo de lograr as idiossincrasias inconvenientes das programação paralela, como as condições de corrida e a inanição (*starvation*).

2 Desenho dos agentes

A simulação da interaçao entre formigueiros contempla, neste cenário, múltiplos agentes; em um momento inicial, portanto, é importante que os caracterizemos, garantindo que eles

possam interagir consistentemente durante a simulação. Em nossa implementação, em particular, identificamos cada agente com uma classe; alguns gozam de múltiplas instâncias (como o agente `Ant`, formiga), e outros, não (como o agente `Map`, mapa). Explicitamente, introduzimos as classes

1. `Anthill`, formigueiro,
2. `Ant`, formiga,
3. `Food`, comida,
4. `Map`, mapa e
5. `Tile`, azulejo (ou, equivalentemente, coordenada);

elas estão, assim, descritas na Figura 1. Perceba, logo, que as formigas estão amarradas ao formigueiro e, além disso, interagem com ele, incrementando o seu armazém de alimento; em contraste, a interação entre as formigas e a comida consiste no decremento de um atributo – o volume da instância de comida; por outro lado, as formigas não modificam, objetivamente, os atributos das coordenadas do mapa – eles que, na verdade, rastreiam, em uma tabela hash (em que as chaves correspondem aos nomes dos formigueiros) de pilhas, as formigas depositadas neles em cada iteração. Esta é, aliás, a interação mais crucial da simulação: as formigas se movem pelo mapa, identificam a comida, a capturam e, em um deslocamento para o formigueiro, incrementam o seu armazenamento de alimento. Nestas condições, o movimento das formigas é delicado; ele é, desta maneira, o tópico da seção seguinte.

2.1 Movimento das formigas

Em cada iteração, há incisivamente um tripleto de ações que as formigas podem executar: ou elas se movem aleatoriamente (método `moveRandomly`), ou elas se direcionam ao formigueiro (método `moveToColony`), ou elas procuram o alimento (método `moveToFood`; elas podem, também, entrar em combate; descreveremos, mais tarde, nossa aproximação para isso). Vislumbramos, nos itens seguintes, cada um desses movimentos.

1. `moveRandomly`. No movimento aleatório, identificamos as coordenadas em que a formiga está e, em seguida, capturamos seus vizinhos[1], verificando, importantemente, a sua existência (com esse objetivo, o método `getTile`, da classe `Map`, levanta uma exceção `BorderError`, sinalizando que o programa está acessando uma coordenada que transcende as bordas do mapa; o tratamento desta exceção, assim, garante a consistência da identificação das coordenadas vizinhas). Escolhemos, então, uma coordenada aleatoriamente; a probabilidade de uma instância ser escolhida é diretamente

[1]Que não contêm alimento; as formigas não interagem com estes azulejos – elas modificam as instâncias da classe `Food` depositadas neles.

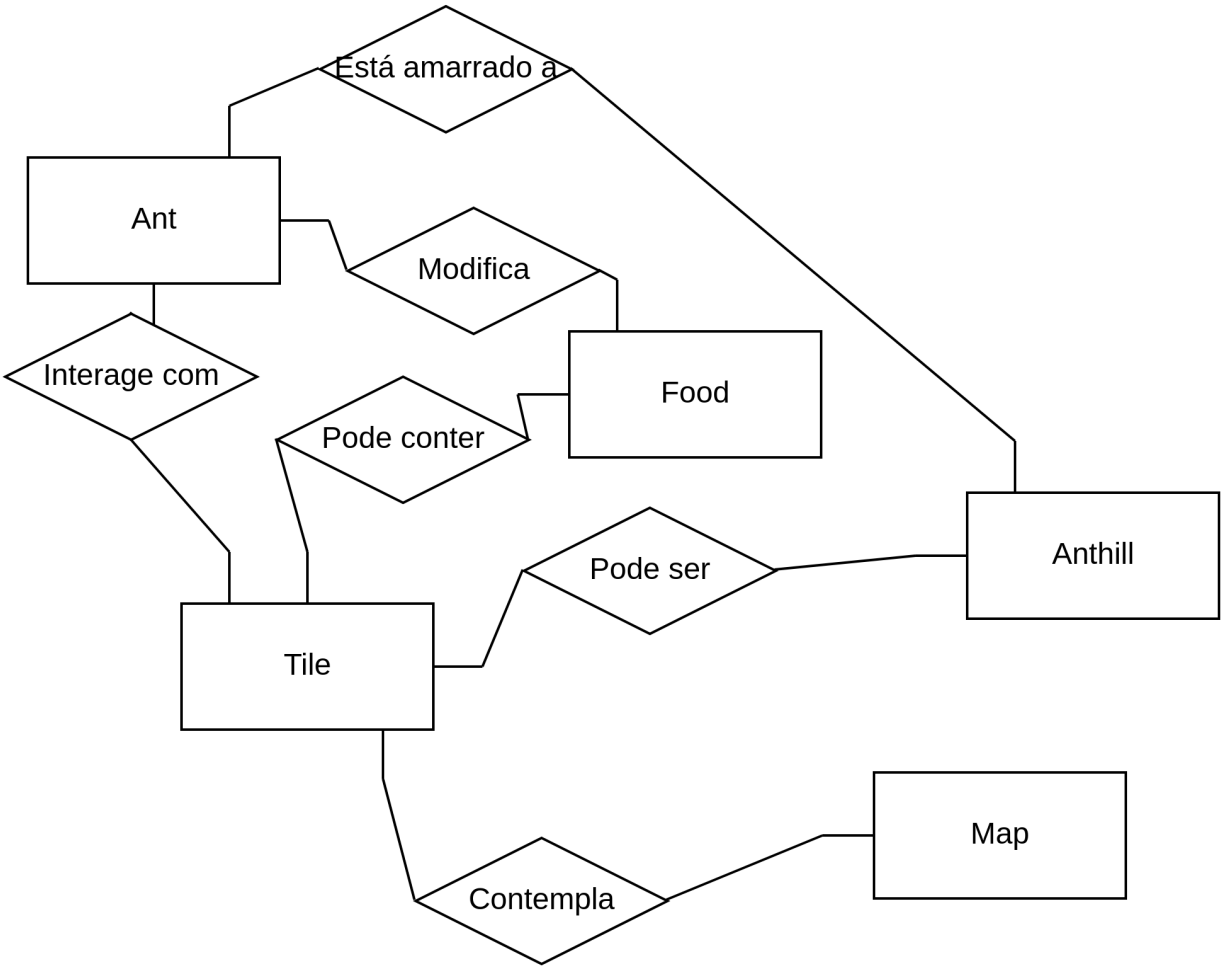

Figura 1: Modelagem e interação dos agentes na simulação.

proporcional à quantidade de feromônio que ela contém[2] (a liberação de feromônios é contemplada na seção seguinte).

2. `moveToColony`. Equipadas com alimento, as formigas se direcionam ao formigueiro – elas têm a informação de sua localização. Elas executam, para isso, um movimento retilíneo, implementado como uma adaptação do algoritmo de Breseham: inicialmente, traçamos, no plano cartersiano, o segmento que amarra a coordenada atual da formiga à de seu formigueiro; na etapa subsequente, computamos a coordenada vizinha da formiga mais próxima deste segmento, e a movemos nesta direção.

3. `moveToFood`. As formigas envisionam, em seu movimento, coordenadas a uma distância caracterizada pelo atributo – do mapa; todas as formigas, em todos os formigueiros, gozam de campos de visão idênticos (contudo, poderíamos modificar a função que identifica instâncias de `Food` nas vizinhanças e distinguir a busca por comida para cada formiga – escolhemos, neste sentido, o desenho mais parcimonioso) –; portanto, elas podem verificar se há alimentos e executar um movimento informado. Neste caso, com as restrições de acesso simultâneo aos alimentos, este "movimento informado"pode, com efeito, consistir em aguardar e, possivelmente, lutar.

Em cada iteração da simulação, assim, cada formiga executa um movimento condicional ao seu estado atual: se ela estiver equipada com comida, ela, incondicionalmente ao seu ambiente, ao formigueiro; se não, ela escaneia o ambiente que a entorna e verifica se há alimentos – se houver, ela se direciona a eles e, se não, ela se movimenta aleatoriamente. Em algumas circunstâncias, ela escolhe lutar; postergamos a descrição desta ação, contudo, para as seções seguintes. Na seção seguinte, caracterizamos o movimento de formigas com comida e a introdução de feromônios.

2.2 Liberação de feromônios

Quando capturam alimentos, as formigas se direcionam impetuosamente ao seu formigueiro; em cada movimento, contudo, elas liberam feromônios (`releasePheromone`; este é um dos métodos da classe `Ant`) com o objetivo de sinalizar a outras formigas (possivelmente, de outros formigueiros) que existe um caminho, nas proximidades, para uma instância de alimento. Estes feromônios, `struct Pheromone` com o atributo `lifetime`, devem ser extraídos do mapa a cada intervalo de iterações; assim, cada instância de `Tile` contém uma lista, `pheromones`, de feromônios liberados pelas formigas, que é, em cada iteração, percorrida, ensejando a captura de feromônios mortos – a intensidade de feromônios nesta coordenada, a propósito, é igual ao tamanho desta lista, que, logo, também é modificada em cada iteração.

[2]Encetamos, para isso, gerando uma variável aleatória uniformemente distribuída no intervalo unitário da reta real; instanciamos, também, uma `array` com as somas acumuladas das quantidades de feromônios (mais um; por exemplo, um objeto com um feromônio teria peso igual a dois, e um com dois, igual a três) em cada coordenada, `cumSum`. Aplicamos, então, uma divisão de cada elemento de `cumSum` pela quantidade total de fermônios nas vizinhanças, e verificamos, assim, o intervalo correspondente à variável aleatória uniforme, que equivale à nossa escolha aleatória.

2.3 Combate entre formigas

Em cada iteração, as formigas que não estão direcionando alimentos ao formigueiro podem escolher combater as adversárias; há, neste sentido, um par de cenários em que a luta é executada. Em um deles, a formiga vislumbra, a uma distância (na métrica L_1) igual a uma unidade, um alimento com volume positivo e, neste caso, seu objetivo é o defender – neste caso, ela luta deterministicamente, se houver adversários em sua coordenada. Em outro, ela não contempla alimentos em suas vizinhanças e, portanto, ela joga uma moeda justa (utilizamos, neste caso, o algoritmo de amostragem utilizado no movimento aleatório com pesos de feromônios) e executa uma escolha entre um movimento aleatório e uma luta – se, outra vez, houver adversários. Mais precisamente, ela escolhe guerrear, não lutar, conforme descrevemos nas sentenças seguintes.

Quando, desta maneira, uma formiga decide lutar, ela executa um ataque coordenado a todas as formigas adversárias em sua coordenada; a probabilidade de que ela vença é, neste caso, proporcional à quantidade de aliadas em sua coordenada. Apesar do ponto culminante desta luta, uma das formigas é *gravemente ferida* e rotulada com o atributo `isDead`; na iteração seguinte, quando percorrermos a lista de feromônios, também capturaremos as formigas mortas e as extrairíremos da simulação. Enfatizamos, aliás, que, mesmo que a rotulação das formigas seja executada em threads distintas, a sua extração é executada na thread principal; isso porque, como elas estão contempladas em um atributo da classe `Map`, `allAnts`, precisaríamos garantir que cada thread modificasse este atributo consitentemente, o que, no entanto, é essencialmente equivalente à execução por uma thread (é bastante difícil particionar o tratamento de uma lista entre threads).

2.4 Sumário dos agentes

A movimentação das formigas, a liberação de feromônios e os aspectos bélicos foram, logo, particularmente inconvenientes, na medida em que eles culminavam na caracterização de ações bastante entrelaçadas – as formigas modificam as comidas e, também, modificam a si mesmas; a dinamicidade introduzida pelo surgimento da morte foi, também, aflitivo, porque precisávamos, neste caso, ser mais cautelosos na escolha da formiga seguinte a executar o movimento e interagir com o mapa. A propósito, a introdução de aspectos estocásticos na simulação injetou oscilações no sistema que, em alguns cenários, foram inadequadas.

Tivemos, aliás, confrontos tétricos com a própria linguagem de programação, `C++`; procedimentos elementares, como o cômputo da distância entre uma coordenada e um segmento, exigiram tremenda introspecção.