Semestrální práce - Bludišťáci

Ondřej Karlíček

16. května 2023

1 Představení problému

Jako semestrální práci jsem si zvolil Bludišťáky. Kde máme mřížkovou mapu, nejlépe nějaké bludiště a v něm jsou rozmístěni bludišťáci. Dále máme skupinu robotů, kteří mají tyto bludišťáky posbírat.

V mém problému je struktura bludiště neznámá, takže prvotním krokem je bludiště prozkoumat a najít pozice všech bludišťáků. Také je potřeba lokalizovat sběrné místo pro bludišťáky. Roboti dopředu nevědí, kolik je bludišťáků ve mřížce, takže musí prozkoumat celou mřížku. Druhým krokem je nalezené bludišťáky odvozit na sběrné místo. Každý robot uveze maximálně jednoho bludišťáka.

Roboti vidí vždy jenom jedno políčko před sebe, zasebe a vedle sebe. Stejné je to s jejich pohybem, vždy můžou jet jenom o jedno políčko dopředu, dozadu, vlevo nebo vpravo.

Roboti celou dobu spolu komunikují, respektive mají společnou pamět ohledně nalezených a odvezených bludišťáků a také spolu sdílejí jakou část mřížky již prozkoumali.

Pro prozkoumávání a odvážení beru jako jednu iteraci jednu akci všech robotů. Ale v rámci jedné iterace se roboti rozhodují sekvenčně. Nejdřív si zvolí akci první robot, na základě toho udělá akci druhý robot, atd.

Pro zjednodušení neberu v potaz srážky robotů, takže v jedné buňce se může nacházet najednou více robotů.

2 Prozkoumávání bludišťě

K prozkoumávání využívám prohledávání do šířky (BFS). Kde ale ještě volím akci, která nejvíc přispěje k prozkoumání bludiště.

K tomu využívám takzvanou úroveň explorace, která se pro danou buňku v mřížce počítá jako počet neprozkoumaných sousedů. Každá buňka má 4 sousedy, jako sousedy neberu buňky diagonálně, protože tam robot nevidí a ani se tam nemůže pohybovat.

BFS posléze probíhá tak, že ze současné pozice robota se postupně prochází všechny možné směry a prohledávání běží, tak dlouho dokud se nenarazí na buňku jejíž úroveň explorace je větší než nula, respektive na buňku, která má neprozkoumané sousedy. V tento moment algoritmus narozdíl od klasického BFS ještě dokončí expandaci buněk ve stejné vzdálenosti, protože ve stejné vzdálenosti jako první buňka s nenulovou úrovní explorace se

může nacházet buňka, která má tuto úroveň vyšší. Až když narazíme na buňky jejíž expandací by vznikla delší cesta, algoritmus ukončíme a vybereme buňku s nejvyšší hodnotou explorace.

Tato buňka je cílové místo kam se robot snaží dostat. Takže robot si uloží trasu do této buňky a každou iteraci se posune o jedno políčko blíže k ní. Takže BFS se spouští jenom v případě pokud robot nemá uloženou žádnou cestu, po které by měl jít.

Zezačátku, hlavně pro jednodušší mřížky, je BFS algoritmus jenom takové hladové koukaní kolem sebe a rozhodování jestli se robotovi vyplatí jít nahoru, dolů nebo do strany. Jeho důležitost je pak hlavně v momentě, kdy se robot nalezne v oblasti, která je již prozkoumaná a je potřeba, aby se přemístil do neprozkoumané oblasti.

Tady by se dala ještě vylepšit komunikace mezi roboty, protože pokud se jich více nachází v prozkoumané oblasti, může se stát, že BFS je navede na stejné místo. Toto se snažím aspoň lehce řešit tím, že pokud cílové destinaci robota klesne úroveň explorace na nulu, tak cestu vymažu a dojde k výpočtu nové trasy.

3 Odvážení bludišťáků

V momentě, kdy je prozkoumána celá dostupná mřížka (část mřížky může být ohraničená zdí a nedá s k ní dostat), se spustí odvážení bludišťáků na sběrné místo. Jelikož v tento moment máme rozházené roboty po mřížce, musíme jim přiřadit jednotlivé bludišťáky. Vypočítat si jak daleko je každý bludišťák pro každého robota, by bylo až příliš výpočetně náročné, takže si tuto vzdálenost odhadneme pomocí manhattanské vzdálenosti. Takže máme matici všech vzdáleností mezi roboty a bludišťáky. Následně využijeme scipy.optimize.linear_sum_assignment, která přiřadí každému robotu jednoho bludišťáka, taky aby součet vzdáleností byl co nejmenší. Posléze si každý robot najde pomocí A* algoritmu cestu k jeho bludišťáku, kterého sebere, a pak si zase pomocí A* algoritmu najde cestu do sběrného místa. Jako heuristiku používá A* manhattanskou vzdálenost. Po odvezení bludišťáka na sběrné místo, se robotovi přiřadí nejbližší bludišťák a proces s vyhledáním cesty tam a zpět se opakuje. V případě, že už žádní volní bludišťáci na mapě nejsou, tak robot mizí z mřížky. Program končí odvezením všech bludišťáků.

4 Popis jednotlivých souborů

- robot.py - implementace třídy robota, která se stará o jeho pohyb a exploraci okolí.
- grid.py - implementace třídy bludiště, která se chová jako společná pamět pro roboty ohledně prozkoumávání mřížky.
- bfs.py - implementace prohledávání do šířky pro prohledávání bludiště.
- astar.py - implementace A* algoritmu, který se využívá během odvážení bludišťáků.
- maincontroler.py - implementace třídy, která spojuje všechny soubory popsané výše. Dochází, zde ke kontrole jednotlivých cyklů obou částí programu.
- main.py - nachází se zde jednoduchá vizualizace pomocí pygame balíčku a spouští se zde celý program.

5 Výsledky

Kvalitu výsledku je těžké prezentovat, když nemám nic na porovnání, ale věřím, že jsem postupným vylepšováním došel k obstojnému výsledku. Největší problém byla explorační část, kde bylo potřeba vyřešit, aby roboti se chovali efektivně a zároveň tu byl potřeba vyřešit problém, co s robotem, který se nacházel v části bludiště, která už byla prozkoumána. Myslím, že tato část nabízí největší prostor pro zlepšení, např. vylepšením komunikace mezi roboty. Druhá část věřím, že se chová efektivně.