Symulacja skrzyżowania — Analiza

Tomasz Kanas

11 września 2020

1 Używanie

Aby uruchomić rozwiązanie wystarczy użyć komendy

```
java -jar TrafficController.jar
```

w katalogu z rozwiązaniem. Format wejścia i wyjścia powinien zgadzać się z podanym w treści. Można też włączyć wyświetlanie logów (do stdout) za pomocą flagi „-DLogLevel=[log level]". Możliwe poziomy logowania:

- 0: DEBUG (nie polecam używać)
- 1: TRACE — dość sporo danych na temat kolejnych stanów symulacji i przebiegu wnioskowania agentów.
- 2: INFO — informacje o numerze obecnej tury i o prędkościach i kwotach płaconych przez agentów.
- 3: ERROR — domyślny, tylko kolizje
- 4: brak logowania (nawet kolizji)

2 Opis rozwiązania

Każdy agent zaimplementowany jest jako AbleAgent, ponadto jest jeszcze jeden agent (nazwany Simulation) odpowiedzialny za powiadamianie pozostałych agentów o rozpoczęciu kolejnej tury, aktualizowaniu stanu symulacji między turami oraz sprawdzania i obsługi kolizji. Symulacja wysyła na początku każdej tury do wszystkich agentów pełen stan skrzyżowania, czyli informacje o

- pozycji
- kierunku
- pozostałym czasie oczekiwania
- maksymalnej możliwej prędkości
- pośpiechu

wszystkich agentów. Sam agent przechowuje między turami jedynie swoje argumenty startowe, nie gromadzi żadnych dodatkowych statystyk.

Większość logiki agenta jest opisana w postaci rulesetu Able. Do jej implementacji użyto głównie silnika PatternMatch. Część logiki jest też w silniku Script, oraz jeden bardzo prosty kawałek w silniku Policy. Próba użycia silnika PatternMatchRete spowodowała błędne wyniki, więc zostałem z wolniejszym PatternMatch.

Nie zaimplementowano pojazdów uprzywilejowanych.

2.1 Protokół

Wszelka komunikacja między agentami zaimplementowana za pomocą Listenerów z Able. Każda wiadomość jest rozgłaszana do wszystkich innych agentów. Z tego co mi się wydaje, agent w Able nie ma możliwości wysłania wiadomości tylko do części swoich odbiorców, więc rozgłaszanie było najprostszym wyborem.

Uzgadnianie pierwszeństwa zaimplementowane jest jako licytacja 3 etapowa. W pierwszym etapie, każdy agent którego możliwa ścieżka ruchu w tej turze krzyżuje się ze ścieżką ruchu innego agenta, wysyła ofertę ile punktów jest skłonny zapłacić za przepuszczenie.

Po tym jak wszyscy agenci, którzy mogli, wysłali ofertę, każdy agent decyduje się które oferty jest skłonny zaakceptować. Musi zaakceptować wyższą ofertę (w przypadku identycznych porównuje się pośpiech, a następnie kolejność z wejścia), ale może być miły i zaakceptować niższą ofertę.

Gdy wszystkie wiadomości o akceptacji zostaną wysłane każdy agent wylicza, czy i jak daleko może pojechać w tej turze oraz jak zmieniają się jego punkty.

Z racji tego, że istnieje agent zarządzający symulacją, to na koniec każdej rundy każdy agent musi wysłać wiadomość z wybraną prędkością. W przypadku agentów biorących udział w licytacji ta wiadomość nie jest używana we wnioskowaniu, pozostali agenci za jej pomocą sygnalizują, że nie biorą udziału w licytacji.

Protokół gwarantuje, że każdy agent wyśle maksymalnie 3 wiadomości w turze.

2.2 Liczenie punktów

Aktualizacja punktów po każdej rundzie została trochę zmieniona od wersji w treści, w celu lepszego dopasowania jej do rozwiązania. Agent płaci punkty które zalicytował jeśli ruszył się, oraz zablokował drogę jakiemuś innemu agentowi. Jeśli agent został zablokowany przez innego agenta, zarabia liczbę punktów którą tamten zalicytował, ale zarabia je tylko od jednego agenta — tego który bezpośrednio zablokował jego drogę. System ma na celu zapobieganie inflacji i deflacji punktów przy dłuższych symulacjach. Dalej możliwe jest, że liczba punktów wzrośnie, w przypadku gdy jeden agent bezpośrednio zablokuje drogę dwóm innym agentom, ale takie przypadki okazują się dość rzadkie.

W treści było napisane aby agent zarabiał połowę punktów wydanych przez innego agenta. Nie zaimplementowałem tego, gdyż obawiałem się, że będzie to powodować spadek łącznej liczby punktów i wpłynie negatywnie na symulację.

2.3 Szczególne przypadki

Zakleszczenie jest możliwe, gdy na środku skrzyżowania znajdzie się 4 agentów i żaden z nich nie będzie mógł z niego od razu zjechać. Wbrew pozorom nie jest to mało prawdopodobny przypadek — zdarzał się nawet w średnich przypadkach testowych. Aby go uniemożliwić, jeśli na koniec rundy miałoby się znaleźć 4 agentów na środku skrzyżowania, ten z najniższą ofertą nie może wjechać na środek i nie dostaje za to żadnych punktów.

Protokół umożliwia utworzenie się cyklu przepuszczeń. Aby uniemożliwić powstanie takiej sytuacji, jeśli agent przepuścił innego agenta, choć nie musiał, to nie może pojechać dalej, nawet jeśli przepuszczany agent będzie zablokowany przez innego agenta, w tym przypadku nie dostaje też punktów za czekanie. Jest to nieoptymalnie rozwiązanie (i kara za bycie miłym), ale nie udało mi się wymyślić żadnego lepszego prostego rozwiązania.

3 Eksperymenty

3.1 Zbiór testowy

Aby zbadać wpływ profili na symulację przeprowadzimy szereg testów. Będziemy testowali skrzyżowanie dla 5 lub 12 agentów, o długości 5 jednostek przed skrzyżowaniem. Każdy test będzie zawierał 10000 tur. Przetestujemy przypadki gdy wszyscy agenci na skrzyżowaniu mają ten sam profil, oraz przypadek gdy znajdują się reprezentanci wszystkich profili. Dla odniesienia zbadamy też przypadek, gdy wszyscy agenci mają profil nijaki (oznaczamy — w tabelce), czyli akceptują tylko kiedy muszą, a do wyceny oferty biorą pod uwagę tylko natężenie ruchu przed skrzyżowaniem. Ponadto zbadamy przypadki, gdy wszyscy agenci mają losowy pośpiech, oraz takie w których mają przez całą symulację stały (w miarę równo rozłożony) pośpiech. Zbadamy jeszcze jeden przypadek dla maksymalnej liczby 100 agentów, powiększymy wtedy długość skrzyżowania do 50 jednostek, zwiększymy maksymalną prędkość agentów do 3 oraz ograniczymy liczbę tur do 1000. Wyniki przedstawiono na Tabeli 1.

Wszystkie wykorzystane dane wejściowe są dostępne w katalogu „tests". Duży test był jedynym testem generowanym losowo.

3.2 Wyniki eksperymentów

#agents	profile	haste	avg. time	haste 0	haste 1	haste 2	haste 3	haste 4	haste 5
5	all	const	6,5824	6,5463	6,6130	6,7057	6,4579	6,5904	0
5	all	rand	6,5589	6,5910	6,6333	6,5403	6,5353	6,4905	6,5560
5	R	rand	6,5699	6,5907	6,6306	6,5813	6,5762	6,5152	6,5214
5	H	rand	6,5571	6,6092	6,5380	6,5828	6,5125	6,5422	6,5551
5	N	rand	6,5921	6,6465	6,6222	6,5814	6,6035	6,5183	6,5784
5	F	rand	6,5691	6,6643	6,6157	6,5487	6,5186	6,4974	6,5722
5	A	rand	6,5773	6,6222	6,5864	6,6326	6,5466	6,5476	6,5297
5	—	rand	6,5747	6,6654	6,6230	6,6236	6,5327	6,4876	6,5088
12	all	const	6,5187	6,5161	6,5162	6,5194	6,5334	6,5011	6,5259
12	all	rand	6,5235	6,5332	6,5382	6,5460	6,5569	6,5071	6,4581
12	R	rand	6,5141	6,5575	6,4899	6,5164	6,5005	6,4839	6,5381
12	H	rand	6,5473	6,6140	6,5401	6,5632	6,5243	6,5331	6,5100
12	N	rand	6,5090	6,4983	6,5033	6,5112	6,5239	6,4903	6,5259
12	F	rand	6,5318	6,5563	6,5154	6,5570	6,5132	6,4989	6,5461
12	A	rand	6,5377	6,5370	6,5374	6,5742	6,5241	6,5482	6,5079
12	—	rand	6,5583	6,5983	6,5869	6,5686	6,5626	6,5342	6,4967
100	all	rand	99,2050	105,1544	77,0314	83,0462	111,8621	94,4318	128,9076

Tabela 1: Wyniki eksperymentów — czas

Po pierwsze możemy zauważyć, że wyniki różnych testów są dość podobne. Przy 5 jednostkach przed skrzyżowaniem, każdy agent jadąc tam i z powrotem musi pokonać 24 jednostki, a przekracza skrzyżowanie 2 razy. Przy maksymalnej prędkości 2, minimalny średni czas przekroczenia skrzyżowania to 6, a wyniki na tabeli oscylują między 6.4 a 6.7, w tym różnice w kolumnie „czas" (czyli

#agentów	profile	haste	max msg	min msg	avg msg
5	all	const	15	6	6,8568
5	all	rand	16	6	6,8268
5	R	rand	17	6	6,8584
5	H	rand	16	6	6,8400
5	N	rand	16	6	6,8516
5	F	rand	17	6	6,8162
5	A	rand	14	6	6,7838
5	—	rand	21	6	6,8464
12	all	const	26	6	6,5506
12	all	rand	24	6	6,5616
12	R	rand	30	6	6,5284
12	H	rand	27	6	6,6008
12	N	rand	26	6	6,5197
12	F	rand	29	6	6,5558
12	A	rand	25	6	6,7025
12	—	rand	20	6	6,7682

Tabela 2: Wyniki eksperymentów — liczba wiadomości

średni czas dla wszystkich agentów) są rzędu 10^{-2}.

W małym teście najlepsze ogólne rezultaty są osiągane dla Hoarderów, oraz dla testu z wszystkimi typami i losowym pośpiechem. Stały pośpiech w tym teście dał najgorsze ogólne rezultaty, oraz spowodował, że agenci nie spieszący się osiągali lepsze średnie wyniki, od tych którzy się mocno spieszyli.

Wśród eksperymentów z tylko jednym rodzajem agentów w małym teście wygrali Hoarderzy, trochę gorzej wypadł test Frustratów oraz Sprawiedliwych (Righteous). Najgorzej wypadli nerwowi i altruiści. Jak popatrzymy jednak na wyniki dla najbardziej spieszących się agentów, widzimy, że tu najlepsze wyniki osiągają Sprawiedliwi (R) i Altruiści, a Nerwowi najgorzej. Warto jeszcze zauważyć, że najbardziej spieszący się agenci wcale nie uzyskują najlepszych średnich czasów. Zwykle najlepsze wyniki osiągane są dla pośpiechu 3 lub 4. Widzimy jednak pewną (choć lekką) tendencję spadania średnich czasów wraz z wzrostem pośpiechu.

W średnim teście najlepiej wypadł test z Nerwowymi agentami. Co ciekawe test ze stałym pośpiechem wypadł lepiej niż test z pośpiechem losowym dla wszystkich profili, choć statystyki dla agentów najbardziej spieszących się wypadają odwrotnie. Poza Nerwowymi, dobrze w tym teście wypadli Sprawiedliwi (R), natomiast najgorzej wypadli Nijacy i Hoarderzy. Dla agentów z pośpiechem 5 najlepsze wyniki zostały osiągnięte w teście z wszystkimi rodzajami. Dobre wyniki osiągneli też Nijacy, Altruiści i Hoarderzy, natomiast najgorzej wypadli Frustraci.

3.3 Dyskusja wyników

Główną różnicą między testem małym (5 agentów) i średnim (12 agentów) było „zatłoczenie skrzyżowania". Dla małego testu na środku skrzyżowania krzyżowały się trasy zwykle nie więcej niż 2 agentów, natomiast dla testu średniego zdarzały się skrzyżowania nawet 4 agentów. Jak mogliśmy

się przekonać, spowodowało to znaczną zmianę w optymalnych strategiach.

Po pierwsze pamiętajmy, że rozważane przypadki są dość specyficzne, oraz licytacji w tych testach odbywało się stosunkowo niewiele, szczególnie w małym teście, więc wyniki mogą nie być zbyt miarodajne.

Zauważmy, że w eksperymentach w których pośpiech był stały, zależność między pośpiechem a średnim czasem była wręcz odwrotna od oczekiwanej. Może być to powodowane tym, że gdy agent nieustannie mocno się spieszy, to szybko kończą mu się punkty i musi puszczać przed sobą nawet stosunkowo niewiele płacących rywali.

Eksperymenty w których były wszystkie rodzaje agentów miały nie najgorsze (ale też nie najlepsze) wyniki. Eksperymenty zawierające tylko jeden typ agentów nie pozwalają wyłonić żadnego lidera, ponadto wyniki tych wykresów nie muszą odzwierciedlać tego, jak dobrze radzą sobie agenci tego typu w przypadku mieszanym.

Zauważmy znacznie gorsze wyniki dla dużego (100 agentów) testu. Przy tak dużej liczbie agentów zapewne powstał na skrzyżowaniu korek, co spowodowało tak znaczne pogorszenie się wyników. Eksperyment ten ujawnił też problem z efektywnością mojej implementacji. Nie skupiałem się na optymalności obliczeń, licząc na to, że dość niewielka maksymalna liczba agentów nie spowoduje poważnych problemów, ale eksperyment ten ujawnił, że już dla 100 agentów moja implementacja znacząco zwalnia. Częściowo ten problem może być powodowany dość znaczną liczbą 100 procesów potrzebnych w tym eksperymencie, co może powodować znaczne narzuty przy synchronizacji. Niska wydajność mojego rozwiązania zmusiła mnie do ograniczenia się do 1000 tur w tym eksperymencie.

Zastosowany protokół umożliwia wysyłanie stosunkowo niewielu wiadomości. Maksymalnie w jednej turze zostaną wysłane 3 wiadomości, minimalnie 1. Jak widzimy na Tabeli 2, średnia liczba wysłanych wiadomości w trakcie przejazdu przez skrzyżowanie jest niewiele większa od średniej liczby tur potrzebnych do tego, choć dla podenyńczego przejazdu potrawi być dość wysoka.