Förbättring av en självlärande schemaläggare för telekommunikation

Populärvetenskaplig sammanfattning av: Improving a Reinforcement Learning Algorithm for Resource Scheduling

Elin Wilson Andersson
& Johan Håkansson

Inom telekommunikation används basstationer för att skicka och ta emot data. På en basstation kan man tjäna på att olika sändningar delar på hårdvara för sina uträkningar, men en effektiv schemaläggning av dessa kan vara svår att uppnå eftersom olika system kan ha olika egenskaper. Vårt arbete använder förstärkningslärning för att träna en schemaläggare som minimerar latens och undviker deadline-missar.

Inom telekommunikation finns basstationer som skickar och tar emot data, exempelvis till och från en användares mobiltelefon. För att göra detta kan man använda beamforming, vilket innebär att signaler riktas mot en basstation eller en mottagare för att minska störningar. Detta kräver komplexa uträkningar, vilka kan utföras av specifika hårdvarukärnor. Dessa kan användas både när data ska skickas, och när data ska tas emot.

För att det inte ska uppstå flaskhalsar på grund av de många uträkningarna som ska köras samtidigt så måste användningen av kärnorna fördelas på ett bra sätt. Det är därför viktigt att sändningarna har låg latens, och att de inte missar sina deadlines. Olika basstationer kan dock ha olika egenskaper/konfigurationer, vilket innebär att det inte är helt lätt att komma fram till vad som är rätt schemaläggningsbeteende. Dessa olika egenskaper kan exempelvis handla om antal kärnor för beamforming-uträkningarna, eller att sändningar kommer med olika tidsintervall. Nyttan med att hitta ett effektivt schemaläggningsbeteende är att basstationen kan hantera fler sändningar, vilket ger ett mer effektivt resursanvändande.

En annan försvårande aspekt är att latensdefinitionerna skiljer sig beroende på om basstationen skickar eller tar emot data. Om den tar emot data ska arbetet vara färdigt snarast möjligt. När basstationen däremot ska skicka data görs det vid vissa bestämda tidpunkter. Då finns det ingen poäng i att arbetet är utfört långt innan datan ska skickas, eftersom det skulle ta upp onödigt minne. Målet när data ska skickas är därför att arbetet ska vara färdigt så nära denna punkten som möjligt, dess deadline. Det är lite som att ta dig hem under vintern när det är kallt ute. Har du bil kan du åka hem direkt och slippa kyla, men tar du bussen har du en fast tid att anpassa dig efter. Frågan blir då när du ska börja gå för att hinna till bussen utan att behöva stå på busshållplatsen och vänta i kylan för länge.

Ett sätt att lösa detta på är att använda förstärkningslärning för att kunna hitta det bästa schemaläggningsbeteendet för varje unik konfiguration. Förstärkninslärning är en typ av maskininlärning som går ut på att låta en så kallad agent testa olika handlingar i miljön den tränar i för att sedan få någon form av belöning baserat på hur handlingen påverkade miljön. Därigenom lär den sig vilka handlingar som är bäst i olika situationer. Därför är distinktionen mellan olika situationer väldigt viktig.

I sitt examensarbete *Dynamic Scheduling of Shared Resources using Reinforcement Learning* (se `https://lup.lub.lu.se/student-papers/search/publication/9059466`) undersökte Patrik Trulsson hur förstärkningslärning kan appliceras på detta schemaläggningsproblem. Han implementerade en simuleringsmodell och schemaläggare som använde två olika versioner av förstärkningslärning. En av versionerna, kallad Q-learning, hade flera förbättringsmöjligheter både gällande latens och anpassning till den verkliga världen. Med hjälp av samma simuleringsmodell, undersökte vi dessa möjligheter närmare.

Schemaläggaren har genom vårt arbete kunnat förbättras. I simuleringssystemet finns två olika konfigurationer, och på båda presterar nu schemaläggaren med nästan så låg latens som går att få utan att riskera deadline-missar. Agenten jämförs med en baslinje, och presterar nu med latenser som motsvarar cirka 50-58 procent av dess latens. Ändringarna som har behövts för att uppnå detta handlar främst om att ändra inställningarna i Q-learningen, samt att modifiera hur den lärande agenten skiljer på olika situationer. Nu anländer vi precis i tid till bussen.