Självlärande schemaläggare inom telekommunikation

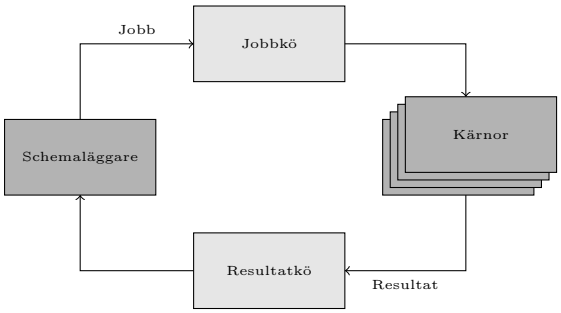
Patrik Trulsson

Populärvetenskaplig sammanfattning av:
Dynamic Scheduling of Shared Resources using Reinforcement Learning

Att dela på begränsade resurser är ofta fördelaktigt då det kan leda till att arbete kan utföras effektivare. Dock om man delar på resurser i hårdvara krävs schemaläggning vilket ibland kan vara utmanande att designa, särskilt om det finns många olika sätt som hårdvaran i sig kan vara inställd på. I detta fall skulle man kunna designa en specifik schemaläggare för varje gång man använder nya inställningar. Eller så skulle man kunna designa en generell schemaläggare som eventuellt inte är optimal för alla inställningar. Ett sätt att hantera detta vore om schemaläggaren själv kunde lära sig själv vad som är bra schemaläggning från sina tidigare handlingar. Detta är vad konceptet förstärkningsinlärning eventuellt kan användas till.

1 Modell

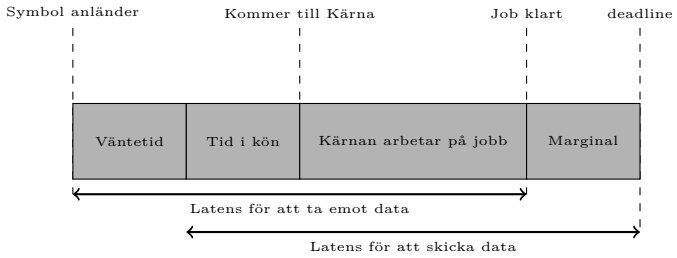
När man ska skicka eller ta emot data från användare inom telekommunikation, exempelvis från en mobil, så är det användbart att använda beamforming. Beamforming har bland annat att göra med hur signalerna från antennerna på basstationen interfererar med varandra för att öka signalstyrkan hos användaren. Dock kräver beamforming att beräkningar utförs i basstationerna. Detta arbete kan utföras snabbt men endast av ett begränsat antal hårdvarukärnor. Således så behövs arbetet delas upp på dessa kärnor för att man ska kunna skicka eller ta emot data på ett effektivt sätt, se figuren nedan. Inom detta system finns det mycket som variera, exempelvis hur många antenner finns det för att skicka information, hur många kärnor finns det?



Figur 1: Modell över systemet, schemaläggaren lägger upp jobb på hardvarukärnor och får tillbaka resultat för hur det gick.

Schemaläggaren får in så kallade symboler vid jämna tidsintervall vilket motsvarar den data som ska skickas eller tas emot. Dessa symboler ger storleken på arbetet som ska utföras och även implicit vilken deadline arbetet har, dvs. när arbetet skall vara klart. Det är schemaläggarens uppgift att dela upp arbetet i mindre jobb som skickas till jobbkön, vilken prioritet som dessa jobb har, samt om schemaläggaren bör vänta eller ej innan den skickar vidare jobben till kön. Målet när man designar en schemaläggare är att undvika att missa deadlines samt att minimera latens.

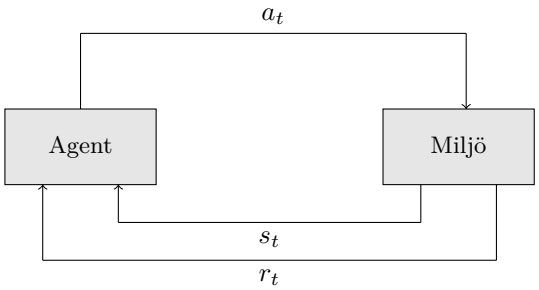
Att skicka data och att ta emot data i basstationen har olika krav. Generellt bör arbete som har med att ta emot data i basstationen utföras så snabbt som möjligt. Däremot, när basstationen ska skicka data bör arbetet utföras så snart inpå sin deadline utan att överskrida den. Detta något underliga beteende är på grund av att basstationen skickar information i intervall. Om arbete blir klart innan det ska skickas iväg så måste det lagras i minnet vilket inte är fördelaktigt. Således när basstationen skickar data motsvarar låg latens att jobbet görs klart nära inpå sin deadline. Således defineras latens på olika sätt, se i figuren nedan.



Figur 2: Latens definition.

2 Förstärkningsinlärning

Förstärkningsinlärning går ut på att lära sig av sina misstag och utnyttja erfarenhet för att utföra bättre val. Det generella systemet kan ses nedan som består av en agent och en miljö. Agenten ansvarar för att uföra en handling, a_t, på miljön som i sin tur ger status, s_t som returnerar information om miljön just nu och en belöning, r_t, som ger hur bra handlingen var.



Figur 3: Överblick för förstärkningsinlärning.

Två algoritmer för förstärkningsinlärning användes i examensarbetet för att försöka skapa en självlärande schemaläggare: tabellbaserad Q-learning samt deep Q-learning. Båda metoderna försöker uppskatta en Q-funktion som är kopplat till s_t och a_t och beskriver hur bra en handling är vid vissa omständigheter ur en långtidsperspektiv. Med en bra uppskattning av Q-funktionen kan den användas för att maximera den totala belöningen genom att ta den a_t för s_t med högst värde på Q-funktionen. Skillnaden mellan de två algoritmerna är till största delen hur de uppskattar Q-funktionen. Den tabellbaserade använder en tabell medan deep Q-learning använder en funktionsapproximerare i form av ett neuralt nätverk.

Att konvertera systemet i Figur 1 till Figur 3 är del av designprocessen och är inte alltid lätt. I designen så motsvarar s_t information som finns tillgänglig när en symbol kommer till systemet. a_t motsvarar de val schemaläggaren kan göra. r_t designas som en funktion som ger dåliga belöningar för missade deadline men bättre belöningar desto lägre latens som åstadkoms.

3 Resultat & Slutsats

De två schemaläggarna tränades och testades på simuleringar av systemet med två olika inställningar. Tyvärr gav inte schemaläggaren med deep Q-learning några bra resultat, den missade deadlines och hade hög latens. Det kan dock nämnas här att denna algoritm är svårare att implementera så det är möjligt att mänskligt fel är inblandat. Däremot, den tabellbaserade Q-learning schemaläggaren lyckades möta alla deadlines och gav bra resultat angående latens. Algoritmen applicerades på en simulering av det riktiga system och en del arbete krävs för att gå från simuleringen till det verkliga systemet. Däremot visar resultatet potentialen för att använda förstärkningsinlärning för att uppnå dynamiska självinlärande schemaläggare.