**Teoriuppgiften**

**1.** Heuristik är en inexakt lösning till ett optimeringsproblem som genererar en suboptimal lösning till problemet. Det finns ingen garanti på varken prestanda eller algoritms korrekthet i alla fall även om det finns vissa underliggande teorier om hur lokalsökning kan specificeras på så sätt att man snabbt och sannolikt genererar grannar som leder till en förbättring.

Approximation är en familj av algoritmer som approximera den exakta lösningen genom försäkra att den givna lösningen har en kvot (mellan kvoten av tid det ta för den optimal lösning och den approximativa lösningen alltid mindre är ett avsatt värde). Mindre än lika med p(N) det N är indata storlek

**2.** Vår närmaste målfunktion är att minimera antalet skådespelare in den givna lösningen. Målfunktion är att söka lösning på så sätt att den optimala lösningen S\*  alltid gäller för alla lösningar S i grannskapet d.v.s f(S\*) < f(S). Funktion är antal använda skådespelare i problems lösning. Samt minimera konfliter beroende på hur lösningen är konstruererad

3. Vi behöver superskådisar för att kunna hitta en heuristisk lösning varje gång vi fastna i ett schema som inte går att uppfylla givet de nuvarande begränsningar på vårt optimeringsproblem.

**4.** Vi kan börja med en viss lösning från till exempel en girig algoritm som går på polynomisk tid. Given den lösning som är suboptimal kan vi ändra den lösning till en lösning som liknar den suboptimal där en variabel är ändrad. Givet vårt problem vet att vi har superskådisar vi kan ändra den givna lösning genom att byta ut en superskådis som har givits en roll, swap(Pj,Pk) om ”swappen” fortfarande behåller problemets struktur och Pj kan ersättas av Pk, tar vi Pk ur vår lösning och minska antalet skådespelarna #actors = #actors – 1.

**5.** Om vi fastna i lokala maximum under vår ändring så kan det inte leda till globala maximum. Under en lokalsökning kan vi ta bort en superskådis och få en lokal granne till S1 av en lösning S0 sedan kan vi under nästa lokalsökning få en lokal granne S2 sådan att S1 = S2. Då kommer vår lösning ingenstans och vi fastnar i lokalt maximum i alla iterationer.

**6.** I simulated annealing använder vi en Temperatur T som vid varje iteration minska med kvoten . Vi accepterar en lösning med en viss probabilitet som bestäms av slumpen vilket gör att sannolikhet att fastna i lokala maximum minskas. Vår lokalsökning ändras med en viss temperatur och för varje iteration ändras temperatur så att vår sökning går långsam till en bättre maximum än den nuvarande, Vilket inte gjort tydligt i upprepade lokal sökningar där allt är deterministiskt.

**7.** Det finns flera sätt att stoppa sökningen. Antingen genom att deklarera ett maximalt antal iterationer som upprepade lokala sökningar får göras eller så kan man utifrån sökshistoriken bestämma när en viss lösning har fastnat i lokala maximum. Det vill säga att det inte finns några vidare förbättringar.

**8.** För att minska risken, (dock inte tar bort) att fastna i lokala maximum under sökningen. Vilket gör att vi får en mer spridning och vi kan börja med olika sökningar vid varje tidpunkter vilket ökar chansen att hitta ett maximum.