Schemaläggning vid ett universitet med en genetisk algoritm

Daniel Fors
Institutionen för informationsteknologi
Åbo Akademi
Finland

2015

Då man skapar ett schema för kurser och tentamen vid ett universitet måste man bland annat ta i beaktande vilka lärare som finns tillgängliga, deras kompetens, olika klassrums kapacitet med mera. Man måste också möjliggöra för en studerande att ta de kurser som studieprogrammet hänvisar till. Det har visats att detta är ett NP svårt problem som därmed växer exponentiellt med schemats omfång. Utöver de hårda begränsningarna bör även preferenser tas i beaktande. På grund av problemets potentiella storlek kan det vara svårt för en människa att utföra denna schemaläggning, en heuristisk optimeringsalgoritm är därmed en god lösning. Avhandlingen behandlar en sådan heuristisk algoritm i formen av en genetisk algoritm. En genetisk algoritm är en typ av evolutionär beräkning, en typ av algoritmer inspirerad av den biologiska evolutionen. Med hjälp av denna algoritm kan man skapa ett giltigt schema som också tar olika preferenser i beaktande.

Innehåll

1 Inledning

Schemaläggning är ett vanligt problem som förekommer inom många olika områden och avser helt enkelt skapandet av ett schema över vad som skall ske när, var och hur. För en butik kan det innebära att man specificerar vilka tider personalen arbetar så att man klarar av både öppethållningstider och rusningstider. Vid en hamn å andra sidan kan det handla om när skepp anländer, vilka kajplatser skeppen skall använda samt se till att kajplatserna kan hantera det gods som transporters. Målet med schemaläggning är alltså att fördela begränsade resurser under en viss tidsperiod för att utföra en eller flera uppgifter.

Skapandet av scheman kan enkelt göras manuellt om dess omfattning är liten. Om vi ökar omfånget på schemat med fler resurser, fler begränsningar samt fler uppgifter så växer problemet snabbt. Informationsmängden blir så stor så att en människa har svårt att få överblick över hela problemet och det blir svårt att skapa ett schema som uppfyller alla begränsningar medan man på samma gång uppfyller de uppgifter som skall göras inom tidsperioden[1].

En lösning är att utöka den tillgängliga tidsperioden för att utföra uppgifterna, vilket skulle underlätta schemaläggning för en människa. Att utöka tidsperioden är dock inte alltid vare sig lämpligt eller möjligt. En bättre lösning är att använda beräkningsförmågan hos en dator för att sköta schemaläggningen. Detta kan utföras med hjälp av någon sorts optimeringsalgoritm, där valet av algoritm och dess utformning beror på vilka krav man har. Optimeringsalgoritmer bör generellt väljas noggrant beroende på problemets utformning och storlek. Vissa algoritmer, såsom genetiska algoritmer, är funktionella för en stor del problem, men de kan ofta ha nackdelar jämfört med mera specialiserade val[5].

I denna avhandling behandlas appliceringen av schemaläggning inom ett specifikt område, nämligen schemaläggning av kurser och tentamen vid universitet. Det har visats att denna typ av schemaläggning är ett så kallat NP svårt problem vilket betyder att problemet växer snabbt. Detta problem har angripits på flera olika sätt ända sedan datorer började dyka upp vid olika universitet[1]. Ett flertal olika kategorier av algoritmer har använts för att lösa denna typ av problem. Algoritmerna kan grupperas i fyra olika kategorier; sekventiella metoder, klustermetoder (eng. cluster methods), begränsningsmetoder (eng. constraint-based) samt meta-heuristiska metoder[7]. Avhandlingen kommer att koncentrera sig på endast en algoritm inom gruppen meta-heuristiska metoder, den genetiska algoritmen.

2 Genetiska algoritmer

2.1 Introduktion

En genetisk algoritm är en stokastisk optimeringsalgoritm som bygger på inspiration från den biologiska evolutionen. En stor del av terminologin som används för att beskriva den genetiska algoritmens olika element återfinns även i biologin; här används termer såsom överkorsning, population och gener.

Begreppet genetisk algoritm kommer ursprungligen från den version som Holland presenterade 1975 och tillhör gruppen evolutionära beräkningar[5], ett samlingsnamn för metoder som inspirerats av evolutionen. Gränsen för vad som är en genetisk algoritm är oklar, men generellt använder man sig av en population av kromosomer som genom överkorsning och mutationer skapar nya generationer. Det är också nödvändigt att man kan utvärdera och jämföra lösningar och ge dessa lösningar en relativ rangordning. Ett problem som endast har korrekta och felaktiga lösningar, utan någon möjlighet att värdera hur fel eller hur korrekt lösningen är, kan inte lösas med en genetisk algoritm.

Idén bakom algoritmen är en förenklad syn på den biologiska evolutionen där primärt individer som är väl anpassade till sin omgivning får avkomma och ger upphov till nya generationer. Dåliga egenskaper gallras således bort ur populationen vilket leder till att varje ny generation är bättre anpassad till sin miljö en den föregående. I den genetiska algoritmen är en individ en möjlig lösning till ett problem, medan lösningens kvalité är hur väl anpassad individen är till sin miljö. Goda lösningar kombineras med varandra eller förändras slumpmässigt för att skapa nya lösningar medan dåliga lösningar sållas bort.

2.2 Struktur

Hollands ursprungliga genetiska algoritm använder sig av strängar av binära värden[5]. Dessa strängar är en kodad representation av en lösning till ett optimeringsproblem. Strängarnas längd är oföränderliga och de olika binära värdena i strängarna utgör från- eller närvaron av någon typ av egenskap[4]. Den terminologi som används för dessa strukturer avspeglar dess inspirationskälla. En mängd strängar är en population, strängarna kallas kromosomer, de binära värdena kallas gener, genernas möjliga värden kallas alleler och generas position i en kromosom är dess loci. En kromosom skulle exempelvis kunna representera särdragen hos ett hus. Locus tre i en kromosom, eller den tredje genen i kromosomen, antar värdet ett om huset har fönster söderut

1	0	1	1	1

Figur 1: Kromosom med binärrepresentation

3 7,8	5 5,2	3 2,2	0 10,4	7 4,3

Figur 2: Kromosom med flervärdesrepresentation

eller värdet noll om huset inte har några fönster söderut. Figur 1 visar hur en dylik kromosom kan se ut.

Genom denna representation är det möjligt att enkelt manipulera lösningarna. Domänspecifik kunskap krävs endast för att skapa kodning samt för att utvärdera kromosomerna. De metoder som manipulerar kromosomerna behöver alltså inte nödvändigtvis någon kunskap om problemets egentliga utseende, vilket gör metoderna generella och användbara för varierande problem. Dylika kunskaper kan dock utnyttjas för att justera de använda parametrarna och metoderna för att förbättra algoritmens prestanda. För att förenkla kodningen är nutida algoritmer inte alltid begränsade till en binär kodning. Istället för en binär representation används då ofta heltals- eller flyttalskodningar och gener kan bestå av fler än ett värde. I figur 2 ses en kromosom med gener som innehåller två olika värden, ett heltal och ett flyttal. Kromosomerna är inte heller alltid begränsade till en fastställd längd, gener kan både tas bort och adderas. Dessa mera invecklade implementationer är inte lika generella och domänspecifik kunskap kan därmed också bli nödvändig för manipulationen av kromosomerna.

Valet av hur ett problem kodas spelar en viktig roll i hur algoritmen fungerar[5]. En kromosom är inte alltid en representation av en giltig lösning. Kodningar som tillåter ogiltiga lösningar bör generellt undvikas, men de kan vara oundvikliga. Då en kromosom utvärderas som ogiltig kan den hanteras på två olika sätt. En metod är att bestraffa eller nedvärdera kromosomen. Om en kromosom är högt värderad även efter bestraffning så tyder det på att någon av dess egenskaper är väldigt god. Genom att endast nedvärdera kromosomen så har dessa egenskaper en möjlighet att föras vidare. Det finns dock en risk att bestraffningen inte är tillräckligt hård, vilket kan leda till att ogiltiga lösningar värderas för högt. I värsta fall kan ogiltiga lösningar värderas högre än de bästa möjliga giltiga lösningarna. Den andra möjligheten är att korrigera kromosomen eller generera en ny. Fördelen med att korrigera kromosomen direkt är att högt värderade ogiltiga lösningar inte kan dominera en population. En korrigering har möjlighet att utnyttja den information som en ogiltig kromosom innehåller, men det kan vara svårt att

skapa en god korrigeringsmetod. Om en helt ny kromosom genereras för att byta ut den ogiltiga kromosomen så förloras information[7].

Den största strukturen i algoritmen är en mängd kromosomer, eller en population. En större population utökar sökområdet men detta leder inte nödvändigtvis till en god lösning snabbare än med en liten population. En stor population ökar förutom sökområdet även tiden som krävs för att utvärdera alla kromosomer. Antalet gener som varje kromosom består av samt deras kodning är även en aspekt som påverkar vad som är en passlig populationsstorlek. Hur många gener en kromosom har bestäms till stor del av hur man beskriver problemet och den kodning som detta innebär. Hur många kromosomer man använder är lätt att ändra godtyckligt, men valet kan ha en stor inverkan på algoritmen.

2.3 Utförande

En enkel genetisk algoritm startar med att skapa en helt ny population av n-antal kromosomer. Generna i dessa kromosomer initialiseras med slumpmässigt valda värden och deras lämplighetsvärde utvärderas. Kända lösningar kan även användas för att ge algoritmen en startpunkt, men då riskerar man att algoritmen snabbt konvergerar runt denna lösning istället för att avsöka ett större område.

1. Välj n-antal föräldrar från populationen slumpmässigt men viktat enligt kromosomernas lämplighetsvärde. En kromosom kan väljas som förälder fler än en gång.

2. Skapa n-antal nya kromosomer genom överkorsning mellan de valda föräldrarna.

3. Mutera de nya kromosomerna.

4. Förkasta den gamla populationen och byt ut den med de nya kromosomerna.

5. Utvärdera varje kromosoms lämplighetsvärde.

6. Om något avslutningskriterium har nåtts, avsluta algoritmen. Annars återgår den till 1.

Detaljerna i hur den genetiska algoritmen är implementerad varierar stort. Bland annat är det möjligt att tillåta ett visst antal av den föregående generationen att överleva oförändrade till följande generation. En möjlighet är att förändra storleken på populationen, en annan är att förändra hur kromosomer

väljs för att överleva beroende på hur snabbt medeltalet på kromosomernas lämplighetsvärde varierar per generation.

Hur dessa detaljer skall väljas har inte ännu något definitivt svar. Som riktlinjer kan man använda sig av de erfarenheter andra har haft. Det har dock visat sig att parametrar som fungerar på en grupp av problem kan vara otillräckliga för ett annat. Söklandskapet är trots allt inte utformat på samma sätt för alla problem.

2.3.1 Val

För varje ny generation måste val göras över vilka kromosomer som används för att skapa den nya generationen[5]. Valet av kromosomerna sker slumpmässigt och de valda kromosomerna används därefter som så kallade föräldrar för att skapa nya kromosomer. För att få fram de bästa lösningarna kan populationens alla kromosomer inte ha samma sannolikhet att väljas. Valet av kromosomer måste viktas så att informationen i de goda lösningarna används till en större grad än den i de dåliga. Vid valet av de kromosomer som får avkomma använder man sig därför av kromosomernas lämplighetsvärde. En kromosom med ett högt lämplighetsvärde har en större chans att bli vald än en kromosom med låg lämplighet. Det finns inte någon begränsning på hur många gånger en enskild kromosom kan väljas, men i de flesta varianter av rekombination blir avkomman identisk till föräldrarna om föräldrarna är identiska.

Alla valda kromosomer används inte nödvändigtvis som föräldrar, ett valbart antal kopieras istället direkt till följande generation utan några förändringar. För att försäkra sig om att den bästa eller de bästa kromosomerna överlever till nästa generation används ibland elitism. Via elitism väljs alltid den bästa eller de bästa kromosomerna oförändrade till nästa generation, därmed försäkrar man att den bästa lösningen inte försvinner. Sannolikheten för att de kromosomer som överlever via elitism skall väljas via den normala valmetoden förändras vanligtvis inte.

Det finns ett stort antal valmetoder för att bestämma sannolikheten för att en kromosom skall väljas. En enkel metod är rouletthjulsviktning. Hela populationens lämplighetsvärde summeras och sannolikheten för att en enskild kromosom väljs är lika stor som dess lämplighetsvärde i proportion av den beräknade summan. Roulettеviktningen har en svaghet i och med att enskilda kromosomer med väldigt höga lämplighetsvärden i relation till andra kromosomer också har en hög chans att väljas. Populationen kan då snabbt bli homogen, då denna kromosom med största sannolikhet väljs som förälder till en stor andel av följande generation. Denna homogena population får då ett begränsat sökområde. Om detta sker tidigt under algoritmens utförande

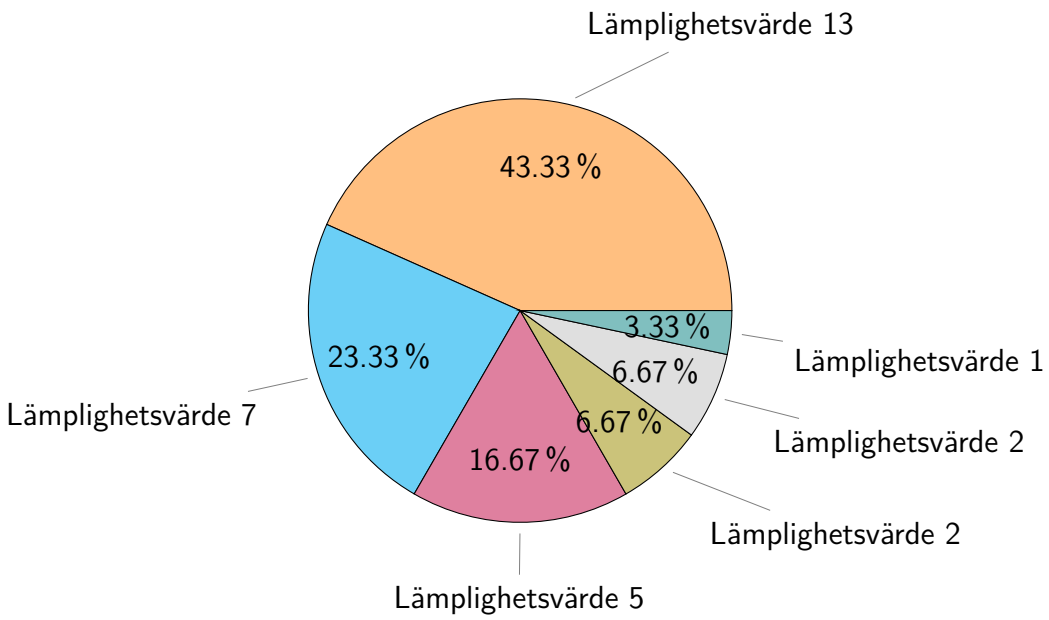
så är risken stor att den når ett lokalt optimum snarare än ett globalt.

Förutom rouletthjulsviktningen finns det ett flertal andra vanliga viktningsmetoder. Nämnvärda är val med hjälp av turneringar (eng. tournament selection) eller enligt rangordning. För turneringar väljs ett antal kromosomer helt slumpmässigt från populationen, varvid den bästa av de valda kromosomerna blir en förälder. Om turneringarna är små så har även kromosomer med låga lämplighetsvärden en god chans att bli en förälder, medan deras chans snabbt minskar då turneringsstorleken ökar. Genom rangordning å andra sidan är varje kromosoms chans att bli en förälder direkt proportionerlig till deras rang. I en population med n-antal kromosomer rangordnas först alla kromosomer enligt deras lämplighetsvärde, därefter beräknas deras viktning enligt denna rang. Ett sätt att använda rangordningsviktningen då vi har n-antal kromosomer är att först ge varje kromosom ett värde från 1 till N, där den sämsta kromosomen får värdet 1 och den bästa får värdet N. Sannolikheten för att den sämsta kromosomen skall väljas blir $\frac{1}{n!}$, den nästa sämsta har sannolikheten $\frac{2}{n!}$ och så vidare tills kromosomen med det högsta lämplighetsvärdet får sannolikheten $\frac{n}{n!}$. Viktningen väljs ofta så att sannolikheten för att bli vald jämnas ut över hela populationen. Till skillnad från rouletthjulsviktningen har en enskild kromosom svårt att dominera, men bieffekten blir att algoritmen endast långsamt konvergerar. En visuell representation av rouletthjulsviktning ses i figur 3 för en population med 6 kromosomer. Samma population där sannolikheten istället bestäms med hjälp av rangordningsviktning ses i figur 4.

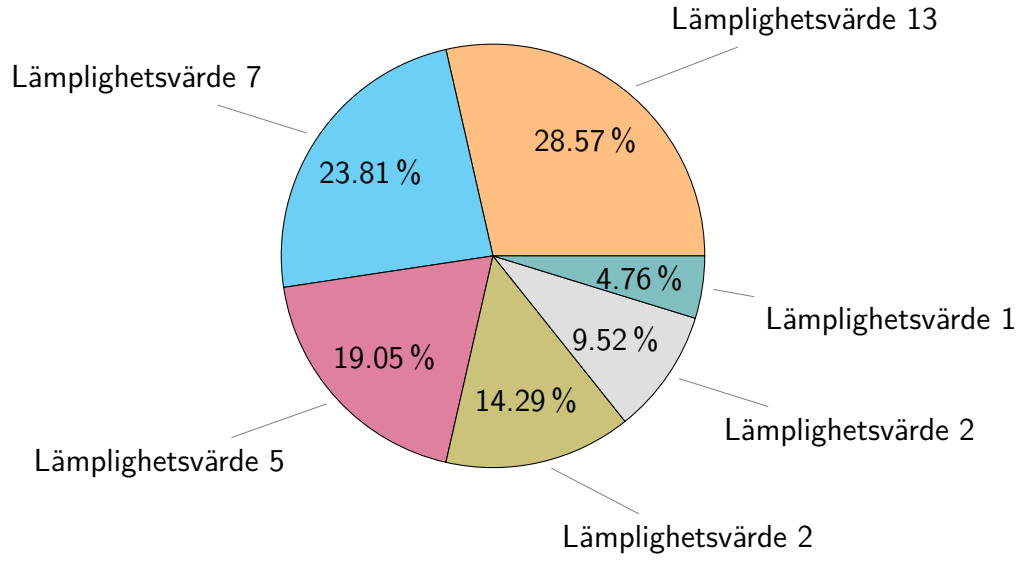
2.3.2 Överkorsning

Överkorsning är en typ av rekombination, inom genetiska algoritmer används begreppen ofta som synonymer[5]. Via överkorsningar kan gener modifieras enligt de värden som andra kromosomer har eller direkt bytas ut med gener från andra kromosomer. De ursprungliga kromosomerna kallas här föräldrar medan de resulterande kromosomerna är deras barn eller avkomma. En enkel överkorsningsvariant mellan två kromosomer med en oföränderlig längd n är 1-punkts överkorsningen. Först väljs en slumpmässig position k mellan 2 och n, sedan kopieras alla gener mellan 1 till k från den första föräldern till den första avkomman och från den andra föräldern till den andra avkomman. Slutligen kopieras alla gener från k till n från den andra föräldern till den första avkomman och från den första föräldern till den andra avkomman[4].

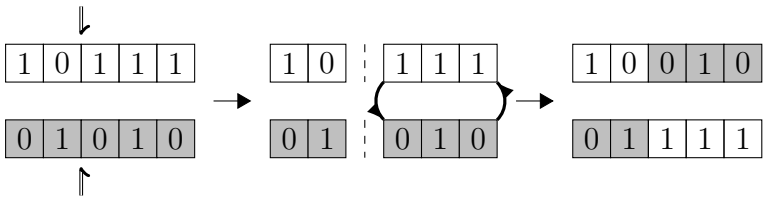
Antalet överkorsningspunkter kan varieras fritt och dessa överkorsningspunkters position kan också vara förutbestämda istället för att vara slumpmässigt valda. Om längden på en kromosom inte är bestämd kan även överkorsningspunkterna placeras på olika platser i föräldrarnas kromosomer. Fi-



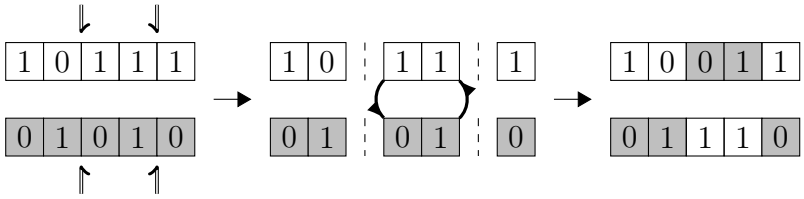
Figur 3: Rouletthjulsviktning



Figur 4: Rangordningsviktning



Figur 5: Enpunktsöverkorsning



Figur 6: Tvåpunktsöverkorsning

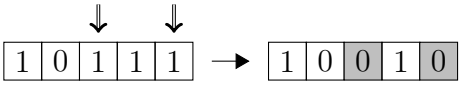
gur 5 och 6 visar överkorsning med kromosomer som använder sig av en binärrepresentation. Om till exempel flyttal används kan även aritmetiska överkorsningar beaktas. I dessa fall byts inte fullständiga gener mellan kromosomerna, istället förändras genernas värden enligt någon matematisk funktion med hjälp av båda föräldrarna.

2.3.3 Mutation

Mutationer görs genom att slumpmässigt förändra en eller flera gener i en kromosom[5]. En vanlig variant är att varje gen i kromosomen har en fastställd sannolikhet P för att en mutation ska ske i den genen. En annan möjlighet är att alltid mutera åtminstone en gen och istället välja platsen för denna mutation slumpmässigt. En binärkromosom som utsatts för två mutationer visas i figur 7. Sannolikheten för att en mutation skall ske kan även förändras under algoritmens gång.

För binära kromosomer är en mutation helt enkelt en invertering av bitvärdet. Istället för att invertera värdet vid heltals- och flyttalskromosomer så ändras istället genens värde enligt någon stokastisk funktion, till exempel enligt en normalfördelning. Då andra representationer än binära behandlas så kan även byten av värden mellan gener inom kromosomen användas. Valet av typen av mutation kan med fördel använda sig av domänspecifik kunskap.

Mutationer är delvis en metod för att undvika att vissa allel värden försvinner. Om den första genen i alla kromosomer i en population har värdet 0 skulle följande generationer aldrig kunna anta något annat värde förutom 0. Med mutationer kan detta värde introduceras på nytt. Mutationer kan också



Figur 7: Mutation

hjälpa till med att söka av närområdet i likhet med en bergsklättringsalgoritm (eng. hill climbing). Ett för stort antal mutationer i en kromosom leder likväl till att algoritmen mer liknar en helt slumpmässig sökning då den största delen av informationen från föregående generationer försvinner.

3 Schemaläggning

3.1 Introduktion

Schemaläggning vid ett universitet är ett komplicerat problem som bör ta flera varierande faktorer i beaktande. Det är också ofta nödvändigt att skapa ett nytt schema varje år och den tid det tar att manuellt skapa ett schema kan räknas i veckor[1]. Användningen av datorer för att underlätta schemaläggningen reducerar därmed arbetsbördan avsevärt och problemet har studerats sedan 60-talet på grund av detta.

Begreppet omfattar även flera olika omständigheter. På grund av de administrativa skillnaderna mellan universitet kan det vara svårt att skapa en schemaläggningsmetod som är väl anpassad för alla dessa skillnader. Schemaläggningen kan till exempel göras utifrån den studerandes perspektiv, där de studerande först bestämmer vilka kurser de är intresserade av, varefter schemaläggningen tar deras intressen i beaktande[3]. Från universitets sida finns det även skillnader i hur olika föreläsningar fördelas. Varje kurs kan ha en huvudansvarig lärare som är den som därmed också är föreläsare för denna kurs, men möjligheten finns också att föreläsare väljs enligt vem som är tillgänglig med rätt kompetens.

Att ge ett absolut svar på vad som är den bästa metoden för att lösa detta problem med alla dessa variationer är inte möjligt, istället ges en överblick över de möjligheter som studerats.

3.2 Krav

De krav och begränsningar som schemaläggningen står inför är många och varierande. Vad som är egentliga krav är inte heller alltid lätt att veta.

Både kurser och tentamen kräver lämpliga utrymmen och föreläsare med rätt kompetens måste utses[7]. För att kunna utvärdera ett schema krävs

därför väl definierade specifikationer för vilken typ av kompetens föreläsaren för en kurs behöver samt vilka krav en kurs eller tentamen har för dess utrymmen i form av kapacitet eller extra utrustning. Föreläsare kan inte heller föreläsa fler än en kurs åt gången och ett rum kan inte användas för många ändamål samtidigt. Utöver dessa direkta krav kan även mera långsiktiga begränsningar beaktas. En studerande bör rimligtvis ha möjlighet att delta i alla de kurser som är obligatoriska enligt studieprogrammet.

Ett välgjort schema bör utöver de hårda kraven även behandla mjuka krav eller preferenser. Preferenser är krav som inte måste tillgodoses för ett giltigt schema, men genom att de uppfylls så förbättras förhållandena för både studerande och lärare. Föreläsare som även har ett administrativt ansvar kan önska längre sammanhängande lediga perioder från undervisningen. Gränsen mellan vad som är ett hårt krav och vad som är endast en önskan kan vara oklar. Vad som anses vara ett hårt krav vid en institution kan vara ett mjukt krav vid en annan institution.

De mjuka krav som finns kan också vara direkt motsägande. Eftersom de inte måste uppfyllas är det inte ett direkt problem, men hur man prioriterar och utformar dessa krav är något som kan ses över. En ofullständig lista på möjliga krav ges här[7][1]. Dessa krav är både av den hårda och mjuka varianten.

- En föreläsning måste ha en föreläsare
- En lärare kan endast undervisa en klass
- Klassrummet måste ha kapacitet för alla deltagare
- Specialutrymmen kan krävas
- En kurs har endast en föreläsning per dag
- En föreläsningen kan delas på flera tillfällen
- Föreläsningar bör spridas ut över hela veckan
- Föreläsningar för ett specifikt studieprogram bör begränsas till endast en byggnad
- Mellanrum mellan föreläsningar bör begränsas

3.3 GA tillämpning

3.3.1 Representation

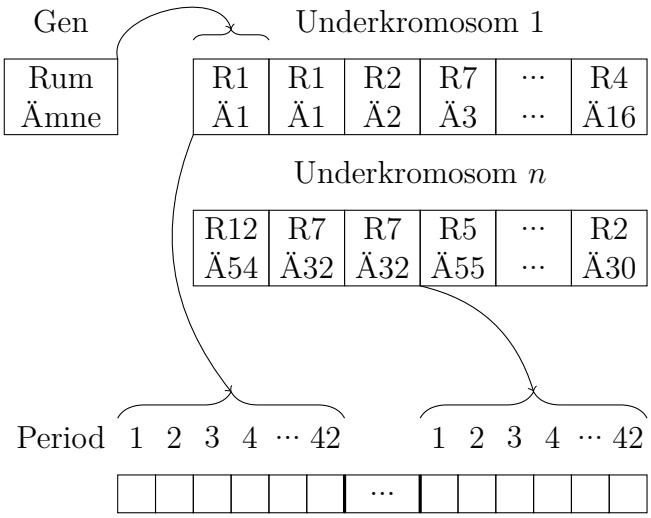
Hur schemaläggningen beskrivs är en av de viktigast aspekterna då man utformar algoritmen. En möjlig uppbyggnad är att direkt använda sig av studieprogrammet. På detta sätt är ett av de krav den ställer uppfyllt endast på grund av hur kromosomen strukturerats[7]. Varje studieprogram använder sig av en skild underkromosom (eng. sub-chromosome) i kromosomen. Generna i underkromosomen har två olika värden, ett värde för det ämne som föreläses och ett värde för det klassrum som skall användas. Genens position i underkromosomen är direkt ett schema över när kursen föreläses. Med denna struktur delas en dag in i perioder. Den första genen i en underkromosom skulle därmed representera det ämne som föreläses i den första perioden på måndag, följande gen skulle vara den andra perioden på måndag och så vidare tills den sista genen representerar den sista perioden på fredag. Hur denna struktur ser ut kan ses i figur 8

En svaghet med denna representation är att tillgängligheten till andra kurser inte tas i beaktande, endast de obligatoriska kurserna för studieprogrammet garanteras för en studerande. Denna representation förbiser även möjligheten med kurser som är gemensamma för flera studieprogram; varje kurs är endast för ett studieprogram. Representation är därmed inte nödvändigtvis den bästa möjliga för alla universitet. Ytterliga kontroller kan emellertid implementeras för att också tillgodose dylika situationer, men de skulle ändå inte få någon fördel av kromosomernas struktur för att automatiskt uppfylla dessa krav. Strukturen ignorerar även tillgängligheten till lärare, hur de allokeras till olika kurser bestäms utanför algoritmens ramar. Tillgängligheten till lärare kan likväl kontrolleras av algoritmen och kromosomer som inte klarar denna kontroll korrigeras.

Då man använder sig av kromosomerna som direkta scheman så kan man också använda sig av scheman med flera värden istället för att använda underkromosomer[6]. Generna består av räckor av heltalsvärden där värdena står för olika händelser. Kromosomens struktur begränsar i detta fall inte problemet till någon större grad och hjälper därmed inte till att lösa problemet.

Istället för den ovannämnda representationen kan även all information sparas direkt i kromosomens gener. Olikt den tidigare nämnda representationen är denna kromosom inte direkt ett schema, då genernas position inte direkt hänvisar till en viss period. Endast efter att informationen i en gen har kontrollerats så kan man veta var i ett schema den hör hemma.

En alternativ representation är där genernas position representerar vilken



Figur 8: Gen, underkromosom och kromosomstruktur för n studieprogram, 42 perioder och 16 ämnen per studieprogram av den typ använd av Pongcharoen at al.[7]

händelse det handlar om. Den tidsperiod som en händelse är schemalagd till sätt därmed in dess gen. Ett exempel är den som Donlon använder, detta dock med ett enklare problem[3]. Algoritmen används för att schemalägga valbara kurser för vilka de studerande redan visat sitt intresse. Kromosomstrukturen som används har en gen för varje kurs, där genens värde representerar i vilken period kursen skall undervisas. Den reducerade omfattningen på problemet betyder dock att den inte direkt kan jämföras med ett försök att schemalägga alla kurser vid ett universitet.

Det finns inte någon representation som kan sägas vara mer korrekt än den andra. De olika representationerna används på något varierande problem och det är därmed inte möjligt att jämföra dem direkt. Eftersom alla universitet inte använder sig av samma administrativa upplägg så är det inte heller självklart att det finns en representation som är objektivt bättre än de andra för alla universitet.

3.3.2 Initialisering

En normal genetisk algoritm initialiseras helt slumpmässigt. Om kodningen tillåter ogiltiga kromosomer kan det leda till att en stor del av populationen är ogiltig. Användning av domänspecifik kunskap och enkla heuristiska metoder under initialisering hjälper därmed algoritmen att snabbare hitta goda lösningar och dessa lösningar är också potentiellt bättre[6]. Som ett exempel

kan de olika kurser eller tentamina som skall schemaläggas först klassificeras i hur svårplacerade de är. En kurs som kräver specifika utrymmen, specialutrustning och endast kan ske under en viss tid på dygnet bedöms som väldigt svårplacerad, medan en kurs som endast kräver ett litet klassrum betecknas som lättplacerad. Under initialiseringen försöker därefter algoritmen att sätta ut kurser på ett sådan sätt att alla kursens krav uppfylls och ordningen kurserna placeras i bestäms av hur svårplacerade de är.

Relevansen av hur initialisering sker beror på hur problemet representeras. Det stora antalet krav som ett kursschema ställs inför leder till att de flesta representationer som tillåter ogiltiga lösningar bör använda sig av en initialisering som inte är slumpmässig. Orsaken är att majoriteten av de möjliga slumpmässigt genererade kromosomerna kommer att vara ogiltiga, såvida problemet inte är trivialt[8].

3.3.3 Lämplighetsvärde

En god orsak till att undvika möjliga ogiltiga scheman är lämplighetsvärdet. Om ogiltiga lösningar tillåts så måste dessa bestraffas genom att reducera deras lämplighetsvärde. Med många mjuka krav är det svårt för en kromosom att uppfylla dem alla, resultatet blir att bestraffningen från de mjuka kraven kan hålla algoritmen borta från områden med giltiga lösningar. Om å andra sidan möjligheten för de mjuka kraven att bestraffa lämplighetsvärdet begränsas kan bestraffningen från de mjuka kraven bli så låga att de knappt påverkar algoritmen[8].

Istället för att bestraffa ogiltiga lösningar kan man korrigera dem eller helt förkasta kromosomen och generera en ny kromosom. Eftersom de flesta kromosomer troligtvis är ogiltiga så är det inte någon större fördel med att generera en ny kromosom. Genom att se till att alla ogiltiga kromosomer korrigeras kan man lättare välja bestraffningsvärden för de preferenser och mjuka krav som finns. Algoritmen behöver inte optimera två olika mål samtidigt. Reparationer av kromosomer kan dock vara svårt att göra om kromosomernas struktur är alltför fri. Därmed återkommer man till hur viktig kodning av problemet är.

Om endast giltiga kromosomer ges ett lämplighetsvärde kommer algoritmen att söka ett globalt minimum. Varje krav som inte uppfylls bestraffar kromosomen genom att öka lämplighetsvärdet. Det blir därmed även relativt lätt att prioritera olika mjuka krav genom att öka bestraffningen för de högre prioriterade kraven. Då en kromosom uppfyller alla mjuka krav får lösningen ett lämplighetsvärde på 0.

3.3.4 Problem delning och hybrider

Den återkommande poängen om möjligheten av ogiltiga kromosomer visar att i princip två olika saker sökes. Giltiga lösningar, sådan som endast behöver klara de hårda kraven, och goda scheman, som beaktar preferenser[8][10] Genom att antingen använda en representation som endast tillåter giltiga lösningar eller genom att bruka korrigeringar används algoritmen i princip endast för optimeringen av de mjuka kraven som ställs. De metoder som används för att initialisera och korrigera ogiltiga lösningar kan till exempel anses ligga utanför vad som definieras som en genetisk algoritm.

Denna delning av problemet i två delar är en orsak till användningen av hybrider av den genetiska algoritmen. Dessa kombinationer med andra metoder förbättrar helhetens prestanda genom att låta den genetiska algoritmen arbeta på den del som den är bäst på. I initialiseringsstycket ovan nämns till exempel användningen av heuristiska metoder under initialiseringen. Redan denna använding av andra heuristiska metoder kan anses vara en typ av hybrid. Den genetiska algoritmen kan till och med kombineras med sig själv, där en implementation först arbetar på att hitta ett giltigt schema, medan den andra använder denna information för att optimera schemat enligt de mjuka krav som finns.

Vanligtvis används den genetiska algoritmen för att optimera de mjuka kraven snarare än de hårda. Det finns dock undantag, där till exempel luddig logik (eng. fuzzy logic) används med de mjuka kraven tillsammans med den genetiska algoritmen[2].

3.4 Alternativ

Många alternativ för schemaläggning har studerats[7][1][10]. Jämförelser mellan genetiska algoritmer visar inte att det finns någon klar vinnare i dessa jämförelser. Problemet är att många implementationer av schemaläggningsalgoritmerna är byggda för ett specifikt problem, det blir därmed svårt att säga något definitivt om hur stora skillnaderna är. Även inom den genetiska algoritmen finns det ett stort antal variationer i hur de byggs och för vilken del av schemaläggningsproblemet de används till inom ett universitet. Metaheuristiska algoritmer, såsom den genetiska algoritmen, anses emellertid vara väl passade för lösningar av denna storlek.

4 Sammanfattning

Avhandlingen har behandlat användning av genetisk algoritmer för att lösa kursschemaläggningsproblem. Algoritmen har visat sig vara fullt duglig för

skapande av giltiga scheman. Schemaläggning har också applicerats på verkliga problem och inte endast teoretiska med goda resultat[9]. De jämförelser som gjorts med andra algoritmer visar att en genetisk algoritm inom kurs- och tentamensschemaläggning är konkurrenskraftig med andra metoder[7][6]. Men bristen på mera utförliga jämförelser med andra schemaläggninsmetoder gör det svårt att direkt jämföra deras kvalitéer. En genetisk algoritm som inte har anpassats för schemaläggning ger emellertid inte tillräckligt goda resultat[11]. Förutom genetiska algoritmer har ett antal snarlika metoder applicerats på schemaläggningsproblem, såsom simulerad glödgning (eng. simulated annealing).

En av de stora problemställningarna för den genetiska algoritmen när den appliceras till dylika problem är hur kromosomerna skall kodas. Problemens storlek och många begränsningar gör att om ogiltiga lösningar tillåts så är sannolikheten stor att en stor majoritet av de möjliga kromosomerna är ogiltiga[8]. Genom att välja en lämplig kodning av kromosomerna kan en del del av de hårda kraven uppfyllas redan genom kromosomstrukturen. Genom att utnyttja specialdesignade överkorsnings- och mutationsoperatorer som inte skapar ogiltiga scheman eller reparationer av ogiltiga scheman samt kodningar som automatiskt uppfyller vissa hårda krav kan den algoritmen användas för att optimera de mjuka krav som ställs. Heuristiska metoder kan vara något mer lämpad för de hårda kraven.

Problemet kan också reduceras så att inte alla delar tas i beaktande. Istället för att både beakta alla mjuka och hårda krav så är det bättre att minska problemet till endast en av de två[8]. Genom att till exempel förutbestämma vilken lärare som ansvarar för vilken kurs kan man även minska problemets omfång. Dessa strukturer reducerar emellertid även flexibiliteten i algoritmen, vilket gör det svårare att använda samma implementation för olika universitet. Då målet är att skapa goda scheman för specifika användningar är det trots allt inte rimligt att begränsa utnyttjandet av dylik kunskap om så inte krävs.

Utnyttjandet av hybridformer av den genetiska algoritmen tyder på att algoritmen inte är lämpad för alla de krav som kursschemaläggningen ställs inför. Eftersom algoritm likväl i dessa hybridformer ger goda resultat så är framtida utveckling i detta område en möjlighet. Mer forskning inom ämnet samt jämförelser mellan de olika möjliga schemaläggningsmetoder krävs i framtiden.

5 Referenser

[1] Victor A. Bardadym. Computer-aided school and university timetabling: The new wave. In *Practice and Theory of Automated Timetabling*, sidor 22–45. Springer Berlin Heidelberg, 1996.

[2] Arindam Chaudhuri och Kajal De. Fuzzy genetic heuristic for university course timetable problem. *International journal of Advance Soft Computing Application*, 2(1), 2010.

[3] James J. Donlon. An application of genetic algorithms to course scheduling at the united states army war college. In *Engineering of Intelligent Systems*, sidor 412–427. Springer Berlin Heidelberg, 2001.

[4] John H. Holland. Genetic algorithms. *Scientific American*, 267(1), juli 1992.

[5] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1988.

[6] Nelishia Pillay och Wolfgang Banzhaf. An informed genetic algorithm for the examination timetabling problem. *Scientific American*, 267(1), juli 1992.

[7] Pupong Pongcharoen, Weena Promtet, Pisal Yenradee, och Christian Hicks. Stochastic optimisation timetabling tool for university course scheduling. *International Journal of Production Economics*, 112(2): 903–918, 2008.

[8] Lewis Rhydian och Ben Paechter. Application of the grouping genetic algorithm to university course timetabling. In *Evolutionary Computation in Combinatorial Optimization*, sidor 144–153. Springer Berlin Heidelberg, 2005.

[9] Ricardo Santiago-Mozos, Sancho Salcedo-Sanz, Mario DePrado-Cumplido, och Carlos Bousoño-Calzón. A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a spanish university. *Computer & operations research*, 32(7):1761–1776, 2005.

[10] Andrea Schaerf. A survey of automated timetabling. *Artifical intelligence review*, 13(2):87–127, 1999.

[11] Shengxiang Yang och Sadaf Naseem Jat. Genetic algorithms with guided and local search strategis for university course timetabling. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(1):93–106, 2011.