Modellering av skördarförarens trädval vid gallring

Modelling the individual tree selection done by the harvester operator in thinning

Christian Fredriksson

Universitetet for miljø- og biovitenskap

Institutt for naturforvaltning

Masteroppgave 30 stp. 2010

Förord

Genom arbete med både skördare och skotare i gallringar har jag fått praktisk erfarenhet av mekaniserade gallringar. Denna erfarenhet i kombination med studier i skogsskötsel vid UMB har gjort att jag under åren blivit allt mer intresserad av gallringens utförande. Detta har lett fram till att jag valde att skriva min masteruppsats om maskinförarens trädval vid gallring.

Först och främst vill jag tacka min handledare, Professor Andreas Brunner vid INA, UMB. Utan all den hjälp jag fått av honom hade detta arbete inte varit möjligt. Sen vill jag, utan att nämna några namn, också tacka de personer som hjälpt till med korrekturläsning av uppsatsen så den går att förstå även för de som inte varit med under hela processen.

Sammanfattning

Målet med detta arbete var att utveckla en modell som beskriver och simulerar det arbete som utförs av föraren av gallringsskördaren vid gallring. Denna modell skapades som en algoritm. Algoritmen skulle utifrån fakta om de aktuella trädens position och diameter i brösthöjd, kombinerat med fakta om skördaren och önskemål om hur beståndet skulle se ut efter att gallringen var utförd, självständigt kunna simulera gallringsingreppet.

Den utvecklade algoritmens arbete börjar med att läsa in de fakta som fanns tillgängliga om det aktuella beståndet. Därefter lades de stickvägar där den simulerade gallringsskördaren skulle köra och de uppställningsplatser utmed stickvägen där skördaren skulle stå under arbetet ut. Vid varje sådan här uppställningsplats definierades två arbetsområden, ett på vänster och ett på höger sida om stickvägen. Därefter påbörjades själva gallringsarbetet med att först välja ut de träd innanför det aktuella arbetsområde som skulle klassificeras som framtidsträd. Efter att framtidsträden i arbetsområdet var definierade fortsatte arbetet med att hitta dessa framtidsträds största konkurrenter. För att förbättra konkurrenssituationen för framtidsträden togs sedan konkurrenter ut tills den önskade grundytan var uppnådd. Efter detta fanns ytterligare ett låggallringsalternativ där små träd togs ut för att komma närmare den önskade grundytan.

För att testa den utvecklade gallringsalgoritmen användes data från verkliga gallringar utförda i talldominerade (*Pinus sylvestris*) bestånd i Hedmark fylke, Norge. Resultatet av dessa tester visade att algoritmens simulerade uttag av träd stämde överens med det verkliga uttaget till i genomsnitt 84,3 % vid ett maximalt avvik mellan verklig och uppnådd grundyta på 5 % och 86,5 % vid ett maximalt avvik i grundyta på 10 %.

Den utvecklade algoritmen beskriver på ett bra sätt hur trädvalet vid en gallring går till och den kan användas för att simulera utförandet av gallringar.

Abstract

The aim of this study was to develop a model that describes and simulate the tree selection carried out by the operator of a harvester during a thinning operation. This model was created as an algorithm. The algorithm was based on facts about the current tree location and dbh, combined with technical details of the harvester and the aim as to appearance of the stand after the thinning has been completed.

The work of the algorithm once developed begins by loading the data available on the stand in question. After that the algorithm defines the strip roads where the simulated thinning harvester would drive, together with working locations along the strip roads where the harvester would be positioned when in operation. At each working location two work zones are defined, one on the left and one on the right side of the strip road. After this the simulated thinning will start. It begins by the algorithm defining all the trees that can be classified as crop trees. In order to improve the growth of the crop trees the algorithm identifies all competitors to the crop trees and removes the most competitive ones until the desired basal area is reached. After removing these, the algorithm starts a "thinning from below" phase where small trees are taken out to come still closer to the desired basal area.

In order to test the developed thinning algorithm, data from real thinnings in pine-dominated stands in Hedmark county, Norway were used. The results of these tests showed that the algorithm simulated the removal of trees to an average of 84.3% at a maximum difference between real and simulated basal area of 5%, and 86.5% at a maximum difference in basal area of 10%.

The test showed that the developed algorithm effectively describes the decisions taking place during a thinning operation, and that it can be used to simulate the process of a thinning on the ground.

Innehåll

1 Inledning

1.1 Bakgrund för modeller

Skogen, både det enskilda beståndet och större skogsområden, innehåller stora värden i form av trädkapital. Och oberoende av om det gäller ett litet bestånd eller ett större skogsområde är det viktigt att kunna förvalta och sköta dessa värden på bästa möjliga sätt. För att lyckas med detta krävs goda kunskaper om skogens biologi och processer, men även om hur skogen utvecklas efter och reagerar på olika typer av ingrepp. En idag vanligt förekommande metod för detta arbete är att använda sig av modeller som simulerar skogen och dess utveckling efter och reaktioner på olika ingrepp (Daume & Robertson 2000; Eliasson 1999; Söderbergh & Ledermann 2003). En modell för skogsimulering kan beskrivas som en rad små beslut (eller delmodeller) som samverkar och tillsammans beskriver skogens utveckling och resultatet av olika skötselåtgärder (Daume & Robertson 2000).

Enligt Söderberg & Ledermann (2003) behöver den enklaste formen av en skogsimulator bestå av minimum tre olika delmodeller. En för att beskriva och förutsäga diameter- och höjdväxt, en för att förutsäga trädens dödlighet och slutligen en för att förutsäga föryngringen. Däremot kan en modell som innehåller endast dessa tre delmodeller bara simulera skogar utan några som helst ingrepp och därmed är denna enkla modell oftast inte relevant i dagens skogsbruk (Söderbergh & Ledermann 2003). För att simulatorn ska kunna användas även i skogar där det sker ingrepp behöver den bestå av ytterligare delmodeller som beskriver de olika ingreppen (Söderbergh & Ledermann 2003). Exempel på ytterligare delmodeller som är viktiga för att beskriva skogar där det sker ingrepp kan vara modeller som beskriver gallring eller föryngringsavverkning (Söderbergh & Ledermann 2003).

1.2 Gallringsmodeller

Gallringen är ett mycket viktigt ingrepp i skogen eftersom det formar skogen efter uppsatta önskemål om hur denna ska utvecklas vidare fram mot föryngringsavverkningen (Albrektson et al. 2008; Daume & Robertson 2000; Hyytiainen & Tahvonen 2002; Söderbergh & Ledermann 2003). Detta har på senare tid också lett till ett ökande intresse för gallring i exempelvis Sverige (Eliasson 1999). Det ökade gallringsintresset har i sin tur lett till att gallringsvirket blivit allt viktigare för industrin. I Sverige stod gallringsvirket år 2008 för ca 44 % av industrins totala virkesvolym (*Riksskogstaxeringen* 2009) och i Norge stod gallringsvirket år 2007 för 13 % av virket till industrin (*Statistisk sentralbyrå* 2010).

På grund av gallringens betydelse för skogens utveckling är det av stor vikt att kunna optimera och förutsäga resultatet av en utförd gallring. Detta gör att en delmodell som beskriver hur en gallring går till i verkligheten är en mycket viktig del vid skogsimuleringar. Daume & Robertson (2000) menar att det är av stor vikt att kunna jämföra resultatet av olika gallringar och gallringstidspunkt redan innan de utförs i skogen och på så sätt optimera utförandet. För att kunna skapa goda modeller krävs det mycket kunskap från verkliga gallringar. Detta har också lett till att det har utförts många studier som försökt beskriva en skogs eller enskilda träds reaktioner efter olika gallringsingrepp (ex. Kantola et al. 2007; Makinen & Isomaki 2004a; Makinen & Isomaki 2004b; Makinen & Isomaki 2004c; Makinen & Isomaki 2004d; Nilsson et al. 2010; Øyen 2003).

Dessa studier har på varierande sätt visat hur många träd eller hur stor grundyta som ska tas ut eller stå kvar. Däremot finns det inte många studier som undersökt gallringsskördarförarens arbete och trädval vid en gallring. I en studie utförd av Gellerstedt (2002) undersöktes förarens användning av gallringsskördarens olika reglage och de olika arbetsmoment som utförs med hjälp av gallringsskördaren. Ovaskainen et al. (2004) studerade arbetets utförande och tidsåtgång för olika moment vid en gallring, medan Ovaskainen et al. (2006) undersökte maskinens positionering i förhållande till de kvarlämnade träden närmast stickvägen.

För att kunna simulera utförandet av en gallring på ett bra sätt används en algoritm vars uppgift är att kombinera alla tillgängliga data om de träd som växer på arealen på ett sådant sätt att algoritmen självständigt kan välja vilka träd som ska tas ut och vilka som ska stå kvar (Daume & Robertson 2000).

1.3 En gallrings utförande

Gellerstedt (2002) utförde en studie för att beskriva arbetsmomenten vid en gallring utförd med en gallringsskördare (engreppsskördare). Han kom fram till att en gallring med gallringsskördare är ett komplext arbete som innehåller många och ofta repeterade moment, många av dessa moment utförs också parallellt. På grund av denna komplexitet finns det heller inte någon dominerande arbetsteknik, utan tekniken kan skilja sig åt mellan olika maskinförare (Sirén 1998, citerat efter Ovaskainen et al. 2004). I den tidigare nämnda studien av Gellerstedt (2002) visades att arbetet kan delas upp i olika moment. Först ska maskinföraren planera stickvägarna och körningen utifrån terräng och bärighet och därefter ska maskinföraren köra maskinen på den planerade stickvägen. Allt eftersom gallringsskördaren förflyttas fram längs stickvägen är det viktigt att placera maskinen på ett bra sätt så att så många träd som möjligt kan fällas och upparbetas från varje uppställningsplats. På varje ny uppställningsplats tas först träden i stickvägen ut, därefter fortsätter arbetet med trädval och uttag bland träden på båda sidor om stickvägen. Först väljs framtidsträd ut och därefter identifieras konkurrenter och indifferenta bistammar. De indifferenta bistammarna tas enligt Albrektson et al. (2008) antingen ut eller tillåts stå kvar som utfyllnads- eller ersättningsträd. Vidare beskriver Gellerstedt (2002) att allt eftersom maskinföraren arbetar sig ut åt sidorna ”öppnas” skogen upp och det blir lättare att se och bedöma de delar av beståndet som befinner sig längre från maskinen. Ju längre från maskinen träden står och ju fler träd som står emellan desto svårare blir valet av vilka träd som ska tas ut och vilka som ska stå kvar. Efter att föraren bestämt vilket/vilka träd som ska tas ut styrs kranen så att det går att nå det aktuella trädet, gripa tag i och fälla trädet, dra det intill stickvägen och kvista det. Därefter följer aptering av stammen, kapning och uppmärkning av stockarna. Slutligen ska stockar, grenar och topp placeras på riktig plats (Gellerstedt 2002).

Begreppet konkurrens är mycket viktigt för att bestämma vilka träd som ska tas ut och vilka träd som ska tillåtas stå kvar vid en gallring. Under åren har det använts många olika mått på konkurrens (så kallade konkurrensindex) mellan träd i en skog (Daniels 1976; Miina & Pukkala 2000). En metod för att beräkna dessa konkurrensindex är att använda sig av de enskilda trädens storlek, egenskaper och inbördes avstånd vid mätningar och beräkning av konkurrensen, så kallade avståndsberoende konkurrensindex (Bella 1971; Ledermann & Stage 2001; Miina & Pukkala 2000). Vid beräkning av dessa index används information om storlek på och verkliga avstånd mellan träden. I många modeller har inte avstånd mellan träden använts direkt, utan det har i stället använts krongeometri (Ledermann & Stage 2001) eller överlapp av påverkanszoner (områden inom vilka träden påverkar sina grannar) (Bella 1971) för att beräkna konkurrensindexen.

1.4 Studiens syfte

Syftet med detta arbete var att: (1) med hjälp av den kunskap som finns tillgänglig genom både vetenskaplig litteratur och egna erfarenheter utveckla en gallringsalgoritm som så korrekt som möjligt beskriver en gallrings utförande. (2) testa den utvecklade gallringsalgoritmen mot verkliga data från gallringar utförda i olika skogtyper med olika gallringsstyrkor och av olika maskinförare. (3) med hjälp av den utvecklade gallringsalgoritmen beskriva de trädval (vilka träd som lämnas kvar och vilka som tas ut) som görs vid en gallring och (4) utifrån de utförda testerna beskriva hur den utvecklade gallringsalgoritmen ska användas för att uppnå bästa resultat.

2 Material och metod

2.1 Gallringsalgoritmen

För att utveckla en gallringsalgoritm som simulerar arbetet vid maskinell gallring krävs kunskaper om hur en gallring med maskin utförs i verkligheten, denna kunskap har så långt som möjligt hämtats från publikationer (exempelvis Gellerstedt 2002; Ovaskainen et al. 2006), medan annat kommer från egna erfarenheter och funderingar.

Gallringsalgoritmen har utvecklats i programspråket C++ (koden finns bifogad i bilaga 2) och finns tillgänglig som ett datorprogram kallat ThinningSelector. Alla parametrar och alla träddata som behövs för att kunna utföra simuleringar med detta program finns lagrade i två textfiler, en parameterfil och en trädlista. Dessa textfiler läses in av algoritmen vid start. För att ändra förutsättningarna för simuleringarna görs det ändringar i dessa två filer. Algoritmens uppgift är sedan att kombinera alla fakta i parameterfilen och trädlistan på ett sådant sätt att den självständigt kan ta beslut om vilka träd som ska tas ut och vilka som ska lämnas kvar, detta förfarande benämns vidare som att simulera eller att utföra en simulering.

2.1.1 Objektet

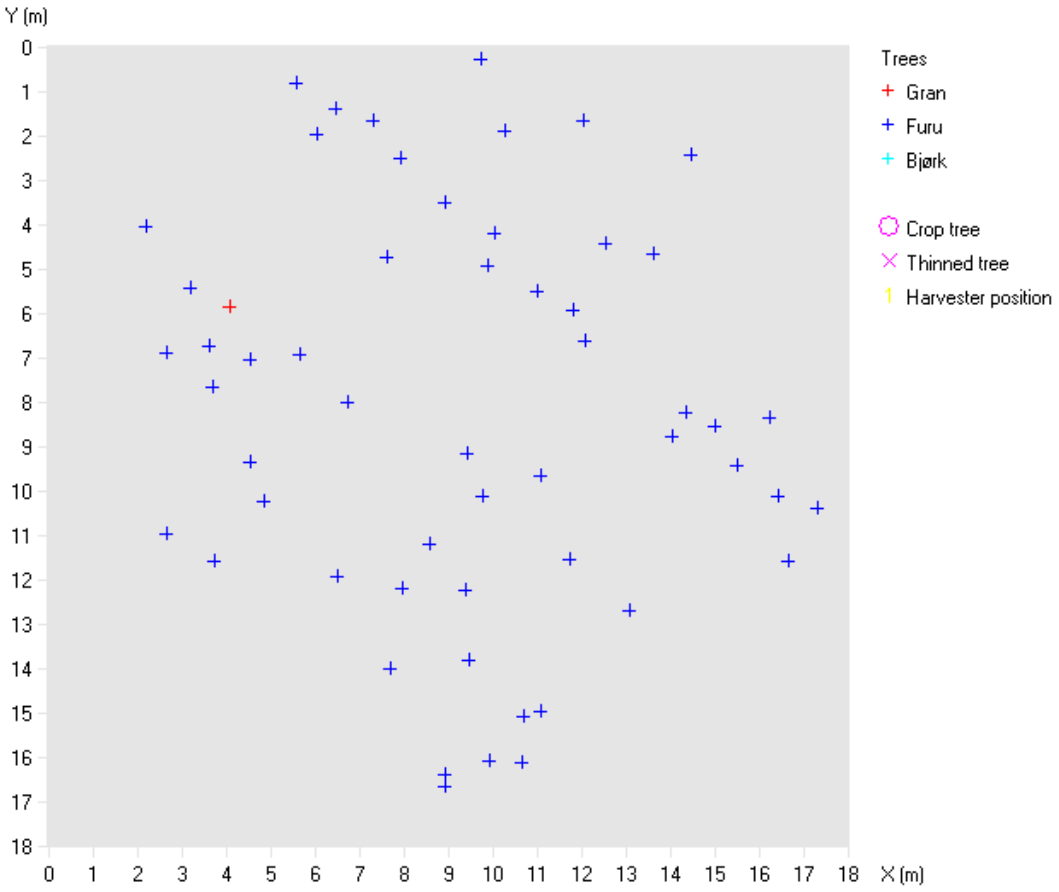
Ett avgränsat område inom vilket det utförs simuleringar benämns härefter som en plot.

Först bestäms den aktuella plotens storlek. Denna definieras som plotens avgränsning i x- och y-led (i m) och redovisas i parameterfilen (ett exempel på en parameterfil finns i bilaga 1). Här är det viktigt att notera att origo för koordinatsystemet är beläget uppe i vänstra hörnet. För ploten finns valet att använda sig av kvadratiska eller cirkulära plots, om ploten är cirkulär eller kvadratisk anges i parameterfilen. I de fall det används cirkulära plots beräknas arealer och andra arealbaserade variabler baserat på en cirkel med x- och y-koordinater som diameter (förutsätter att x och y är lika stora).

De träd som står på denna plot redovisas i trädlistan. I denna trädlista får varje träd ett eget löpnummer. I listan anges för varje träd information om trädslag (anges med ett nummer, varje trädslag ges ett eget nummer i parameterfilen), diameter i brösthöjd (dbh, i cm), trädets position i x- och y-led och trädens status (visas som ett nummer) som beskriver trädet på följande sätt: 1 = framtidsträd, 2 = uttaget vid selektiv gallring, 3 = uttaget på grund av att det stod i stickvägen, och 0 = inte definierat som något av de tidigare nämnda. För tester mot verkliga data där inte orsaken till uttaget av det enskilda trädet är känt kan trädklass 2 användas för alla uttagna träd.

I de fall simuleringarna utförs med cirkulära plots finns det möjlighet att utöka den cirkulära ploten så att den blir kvadratisk (kan beskrivas som att fylla ut hörnen med träd). Detta går till så att algoritmen kopierar ett slumpvis valt träd ur trädlistan och placerar detta träd i något av hörnen. Denna procedur upprepas tills hörnen uppnår samma grundyta per hektar som ursprungsploten.

När parameterfilen och trädlistan laddas in i programmet vid start visar programmet ThinningSelector ploten med alla träd på datorns skärm. Där går det att utläsa trädslag, vika träd som är uttagna och vilka som står kvar. Ett exempel på en sådan bild går att se i Figur 1.

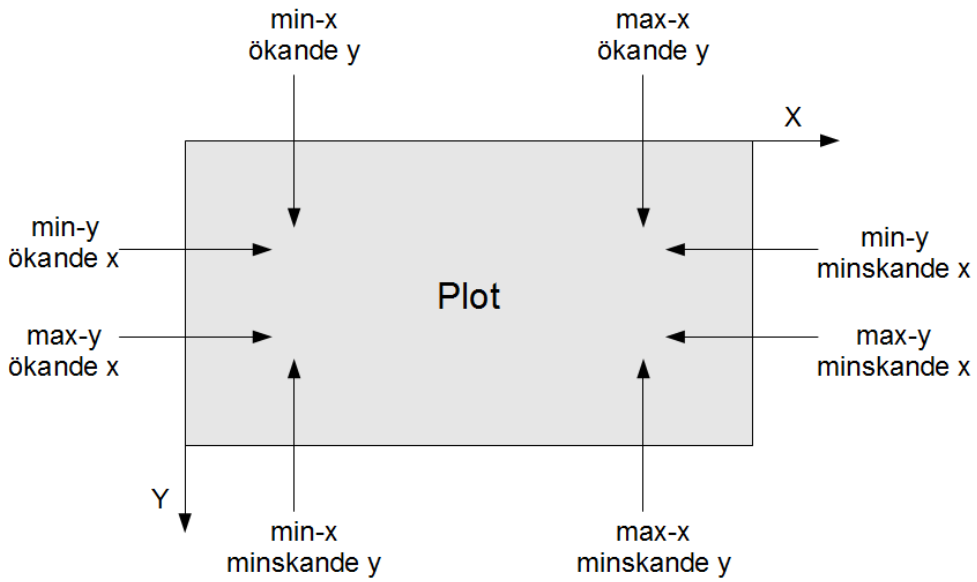


Figur 1. Exempel på den skärmbild som visas efter att en parameterfil och en trädlista från en cirkulär plot laddats in i programmet ThinningSelector. Här visas också trädslag, vilka träd som tagits ut och vilka som lämnats kvar vid en gallring.

2.1.2 Stickvägsgeometri

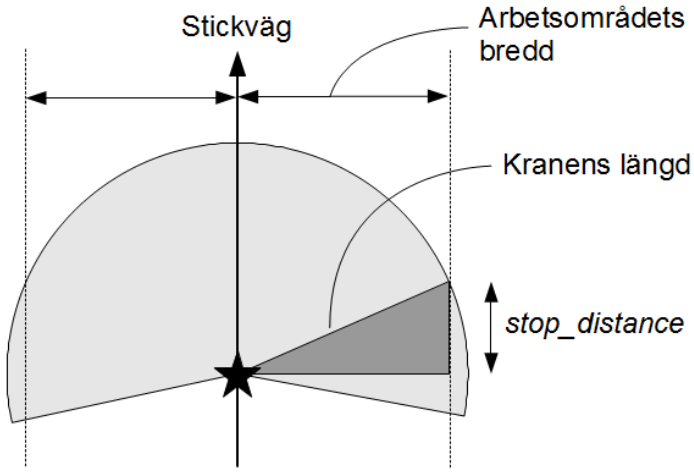
Grunddata om ploten och träden finns nu tillgänglig för ThinningSelector och det är möjligt att börja simuleringsarbetet.

Först bestäms startpositionen för gallringsskördaren. För varje plot finns det fyra olika startmöjligheter, nära x-axelns min-punkt (min-x), nära x-axelns max-punkt (max-x), nära y-axelns min-punkt (min-y) och nära y-axelns max-punkt (max-y). För var och en av dessa startpositioner finns det två möjliga körriktningar, ökande eller minskande x eller y. Tillsammans ger detta åtta olika möjligheter att köra på ploten (stickvägsalternativ) som alla behöver kunna simuleras (se Figur 2). Rent praktiskt utförs detta i algoritmen genom att startpositionen först väljs som min-x eller max-x, därefter väljs körriktningen som ökande eller minskande y. För att välja startpositioner på y-axeln vrids ploten 90° medsols så att x- och y-koordinaterna förändras och därefter väljs startposition och körriktning på samma sätt som innan vridningen.



Figur 2. Åtta möjliga stickvägsalternativ (ett för varje pil), notera att origo ligger i det övre vänstra hörnet. Det övre värdet vid varje pil anger stickvägens startposition, medan det undre anger körriktnigen.

För att ploten ska bli gallrad helt ut till kanten bestäms avståndet mellan plotens ytterkant och den första stickvägens centerlinje av skördarkranens längd (*harvester_range*, i m) och hur långt det är mellan skördarens uppställningsplatser (*stop_distance*, i m). Beräkningen är vanlig geometri med Pytagoras sats, där skördarkranens längd är hypotenusan och där avståndet mellan uppställningsplatserna är den ena katetern (Figur 3). Detta avstånd benämns i fortsättningen som arbetsområdets bredd (*harvester_x*, i m). Avståndet mellan centerlinjerna av två intill varandra liggande parallella stickvägar blir dubbelt så stort som arbetsområdets bredd.



Figur 3. Sambandet mellan arbetsområdets bredd, kranlängd och avstånd mellan uppställningsplatser. Stjärnan markerar gallringsskördarens position.

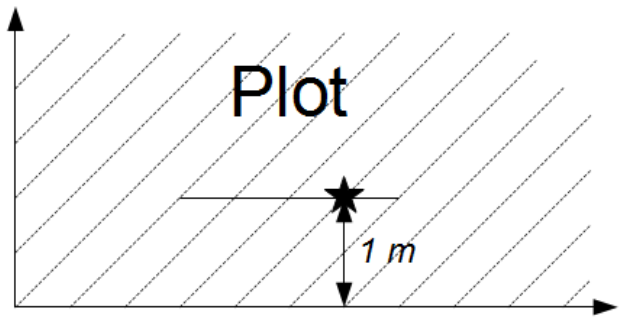
2.1.3 Placering och avverkning av stickvägarna

Efter att den första stickvägens centerlinje placerats *harvester_x* m från plotens kant placeras resterande stickvägar ut med ett avstånd av 2**harvester_x* tills hela ploten kan nås från någon stickväg. Samtidigt med utplaceringen definieras också körriktningen för varje stickväg. Körriktningen på den första stickvägen är definierad sedan tidigare, körriktningen på stickväg tre, fem, sju och så vidare är den samma som för den första stickvägen. Körriktningen för stickväg två, fyra, sex och så vidare är den motsatta av körriktningen på den första stickvägen. Här kommer det att inträffa tillfällen där det sista arbetsområdets bredd inte blir fullt utnyttjat eller där sista stickvägen behöver placeras utanför objektet för att det ska gå att nå träd i hela ploten från någon stickväg.

Bredden av stickvägarna (*striproad_width,* i m) bestäms av hur mycket plats skördaren (och skotaren) behöver för att kunna köra och arbeta utefter dessa. Halva denna bredd läggs ut på varje sida om stickvägens centerlinje. De träd som står ”på stickvägen” kommer inte att kunna stå kvar eftersom gallringsskördaren ska kunna köra där, de definieras därför som utgallrade.

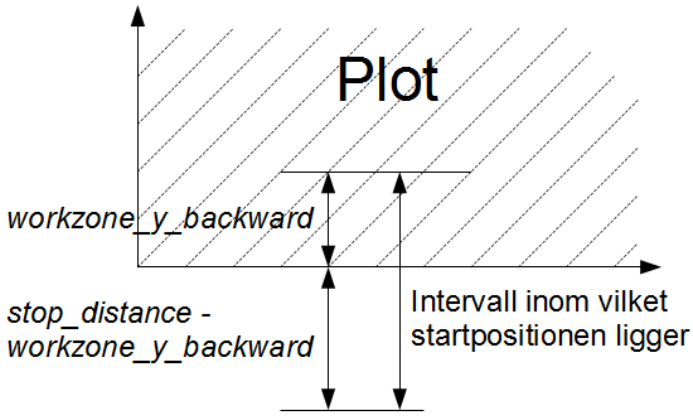
2.1.4 Placering av uppställningsplatser

För att bestämma den första uppställningsplatsen på varje stickväg finns det två olika möjligheter beroende på inställningarna i parameterfilen. Som standard används ett helt deterministiskt system där den första uppställningsplatsen för varje stickväg bestäms till 1 m in på ploten, från plotens kant längs stickvägen (Figur 4).



Figur 4. Vid deterministisk bestämning av den första uppställningsplatsen på varje stickväg bestäms den till 1 m in på ploten längs stickvägen. Den första uppställningsplatsen markeras med en stjärna.

Det andra sättet på vilket den första uppställningsplatsen bestäms är en metod där positionen slumpas fram inom ett intervall av *workzone_y_backward* m från plotens kant längs stickvägen, och *stop_distance – workzone_y_backward* m från plotens kant i motsatt riktning av stickvägen (Figur 5). Denna procedur upprepas sen för den första uppställningsplatsen för varje stickväg.



Figur 5. Intervall inom vilket den första uppställningsplatsen placeras vid simulering med slumpvis vald första position för varje stickväg.

Efter denna, den första uppställningsplatsen används alltid ett avstånd av *stop_distance* m mellan uppställningsplatserna helt tills den aktuella stickvägen är färdiggallrad.

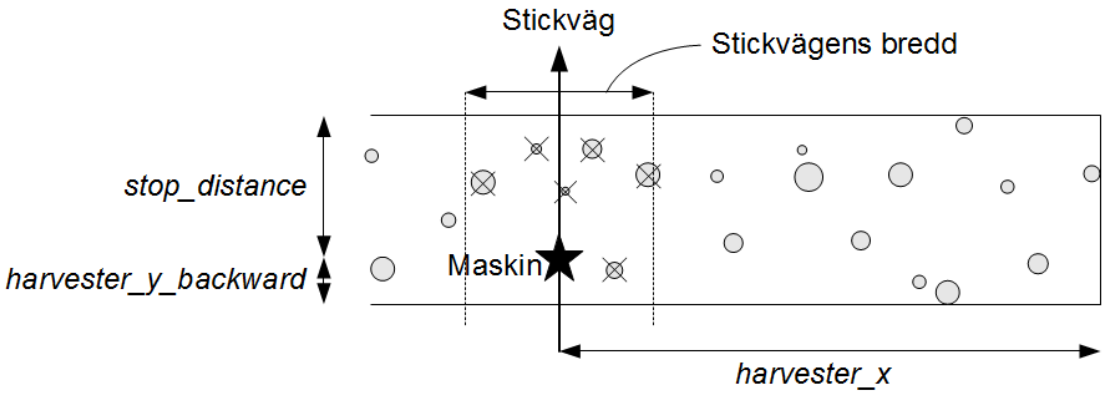
Vid start av simuleringar i programmet ThinningSelector finns två olika alternativ, ”Single run” eller ”Multiple runs”. Vid ”Single run” utförs endast en simulering, medan det vid ”Multiple runs” utförs så många repeterade simuleringar för varje stickvägsalternativ som uppges av parametern *NoReplicates* i parameterfilen. Exempelvis leder en simulering med *NoReplicates = 100* till att det utförs 8*100 = 800 simuleringar. Simulering med ”Multiple runs” benämns vidare som en simuleringsomgång.

För att inte alla simuleringar med samma stickvägsalternativ ska bli identiska vid ”Multiple runs” varieras den första uppställningsplatsen för varje stickväg mellan varje repetition. För alternativet med fast första uppställningsplats flyttas den första uppställningsplatsen fram *stop_distance/NoReplicates* m för varje repetition. Vid alternativet med slumpvis vald första uppställningsplats slumpas den första uppställningsplatsen för varje repetition fram på samma sätt som tidigare beskrivet.

2.1.5 Definiering av arbetsområden

Eftersom det inte är realistiskt att föraren av gallringsskördaren har överblick över hela ploten på en gång (Vestlund et al. 2006) definieras det två arbetsområden för varje uppställningsplats, först ett på vänster och sedan ett på höger sida av stickvägen. Ett arbetsområdes storlek i x-led definieras av *harvester_x*, medan arbetsområdets storlek i y-led definieras av *stop_distance* m i körriktningen och *workzone_y_backward* m i motsatt riktning av körriktningen (den senare parametern medför att det blir överlapp mellan arbetsområdena), allt i förhållande till kranens bas (se Figur 6). Arbetsområdets totala area blir således:

$$harvester\_x \cdot (stop\_distance + harvester\_y\_backward).$$



Figur 6. Bild av ett arbetsområde med träd uttagna för stickvägen. Gallringsskördarens position markeras med en stjärna markerad ”Maskin”, körriktningen visas av pilen.

2.1.6 Val av framtidsträd

Valet av framtidsträd styrs av flera olika parametrar. Den första parametern som påverkar detta val är den som bestämmer av vilket trädslag framtidsträden ska vara (*croptree_species*). Detta anges med hjälp av en siffra som motsvarar den trädslagssiffra som finns för varje träd i trädlistan. Genom att ange *croptree_species* = 0 väljs framtidsträd av alla arter. För att undvika att små träd väljs som framtidsträd definieras en minsta tillåtna diameter. Denna beräknas som en percentil av diameterfördelningen för alla träd på ploten (*croptree_dbh_percentile*, i %). Eftersom den rumsliga fördelningen av de grova träden kan vara ojämn kan det här uppstå tillfällen både där det finns väldigt många eller där det inte finns några framtidsträdskandidater innanför det aktuella arbetsområdet.

I parameterfilen finns en parameter för önskat antal framtidsträd per hektar (*croptree_number_*ha, i träd/ha). För att bestämma hur många framtidsträd som önskas i det aktuella arbetsområdet beräknas arean av detta (vid simulering av cirkulära plots varierar arean kraftigt från arbetsområde till arbetsområde). Därefter beräknas antalet önskade framtidsträd i det aktuella arbetsområdet på följande sätt:

$$\frac{Area\ av\ det\ aktuella\ arbetsområdet\ i\ m^2}{10\ 000\ m^2} \times croptree\_number\_ha$$

För att inte framtidsträden ska stå för nära varandra bestäms en parameter som beskriver minsta tillåtna avstånd mellan två framtidsträd (*croptree_mindist_par*, 0…1). Denna parameter utgörs av ett tal mellan 0 och 1, och definieras som en andel av medelavståndet mellan framtidsträden. Medelavståndet mellan framtidsträden definieras som det avstånd som fås om man placerar ut *croptree_number_ha* framtidsträd i ett kvadratiskt förband över ett område på ett hektar. Eftersom det i många fall är svårt att se små diameterskillnader mellan träd med blotta ögat (speciellt om träden står på olika avstånd från betraktaren) så finns det en parameter som bestämmer hur små

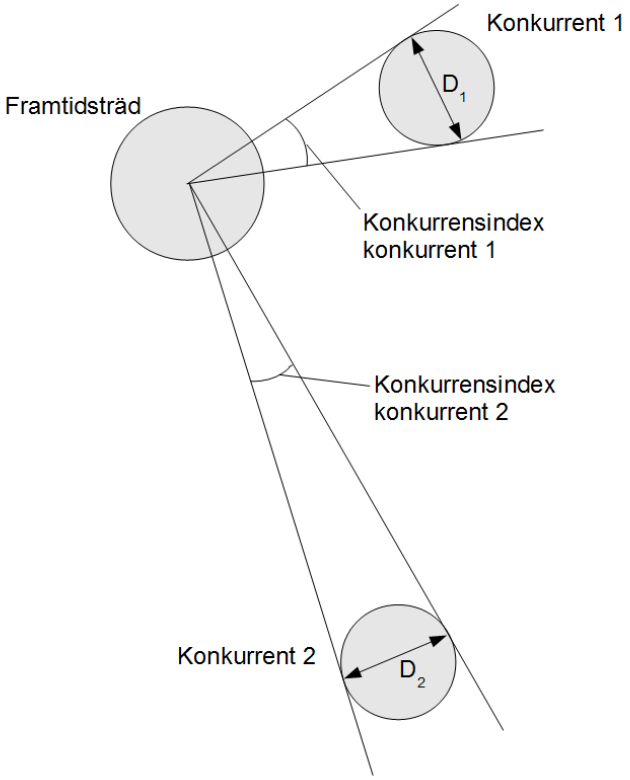
skillnader i diameter som kan avgöras från skördarförarens position (*croptree_min_dbh_diff*, i cm). Träd som inte skiljer sig mer åt i diameter i brösthöjd än denna parameter betraktas vid de vidare simuleringarna som lika grova.

Alla potentiella framtidsträd (rätt art, över minsta tillåtna diameter i brösthöjd, inte valda som framtidsträd sedan tidigare och inte utgallrade sedan tidigare) innanför arbetsområdet listas nu efter sjunkande diameter. Det grövsta trädet i denna lista grupperas sedan med de träd som inte skiljer sig mer än *croptree_min_dbh_diff* cm i brösthöjd från detta, och det träd i denna grupp som står närmast gallringsskördaren väljs som ett första potentiellt framtidsträd. För att slutligt bestämma om detta träd kan väljas som framtidsträd kontrolleras avståndet från detta träd till alla sedan tidigare valda framtidsträd. Om inget av dessa avstånd understiger det minsta tillåtna avståndet mellan två framtidsträd kan detta träd definieras som ett framtidsträd, om det däremot står för nära ett sedan tidigare valt framtidsträd kan det inte definieras som ett nytt framtidsträd. I båda fallen plockas trädet bort från listan och gruppering och urval upprepas tills riktigt antal framtidsträd inom arbetsområdet uppnåtts (det är viktigt att tänka på att det på grund av arbetsområdenas överlappning redan sedan tidigare kan finnas träd innanför arbetsområdet som kan vara klassificerade som framtidsträd, också dessa ska räknas med i antalet). Orsaken till att det träd som står närmast maskinen är det som först väljs som framtidsträd är att maskinförarens trädval börjar nära maskinen och fortsätter utöver arbetsområdet (Gellerstedt 2002).

2.1.7 Val av träd att gallra ut

Efter att det valts framtidsträd inom ett arbetsområde är det dags att hitta vilka träd som ska tas ut. Först och främst tas träd som klassificeras som konkurrenter till framtidsträden ut. För att hitta konkurrenter till framtidsträden beräknas ett konkurrensindex för alla träd inom en radie från respektive framtidsträd av 0,55 * medelavståndet mellan framtidsträden. Beräkning av konkurrensindex sker oberoende av om konkurrenterna står innanför arbetsområdet eller inte.

För att beräkna konkurrensindex användes en metod där vinkeln som skapas mellan avståndet mellan framtidsträd och konkurrent, och diametern i brösthöjd av konkurrenten beräknas (Pukkala 1989), se Figur 7.



Figur 7. Beräkningen av konkurrensindexet görs genom att beräkna vinkeln som skapas genom en tänkt triangel med sidorna ”avstånd mellan framtidsträd och konkurrent”, och ”diameter i brösthöjd (D_i) av konkurrenten”.

När konkurrensindex för alla träd innanför sökradien beräknats markeras den största konkurrenten (det träd vars vinkel i beräkningen av konkurrensindex är störst) till varje framtidsträd som uttagen (framtidsträd kan inte klassificeras som konkurrenter). Detta för att gallringen ska minska åtminstone något på konkurrensen för alla framtidsträd. Efter att dessa träd tagits ut listas alla kvarvarande konkurrenter inom arbetsområdet (oberoende av om de är konkurrenter till framtidsträd innanför aktuellt arbetsområde eller till tidigare valda framtidsträd) efter sjunkande konkurrensindex. För att ytterligare minska konkurrensen för framtidsträden tas det härefter ut ytterligare konkurrenter. Som mål för när uttaget av konkurrenter ska upphöra används parametern för önskad grundyta (*BA_target*, i m^2/ha) i parameterfilen. Hur stor den önskade grundytan är i varje arbetsområde (benämns vidare målgrundyta) beräknas på motsvarande sätt som för antalet önskade framtidsträd inom arbetsområdet. Så länge den absoluta differensen mellan målgrundytan och den aktuella grundytan reduceras, genom att ta ut den för tillfället största konkurrenten, fortsätter uttaget.

Om den aktuella grundytan fortfarande är större än målgrundytan efter att så många konkurrenter som möjligt är uttagna tas ytterligare träd ut i en låggallringsfas. Först sorteras alla träd innanför arbetsområdet efter sjunkande diameter. Med början på det klenaste trädet (sista trädet i listan) tas sedan små träd ut tills differensen mellan aktuell och önskad grundyta är så liten som möjligt (enligt samma regel som för uttaget av konkurrenter).

Här är arbetet i detta arbetsområde klart. Nu bestäms nästa arbetsområde inom vilket samma arbetsutförande upprepas.

2.1.8 Resultatfil

Resultaten av utförda simuleringar med ThinningSelector skiljer sig åt beroende på om det simulerats med ”Single run” eller ”Multiple runs”. Vid ”Single run” presenteras resultatet huvudsakligen på skärmen, men det skapas också en trädlista med alla träds status efter simuleringen och en bild av ploten efter simuleringen (liknande den i Figur 1, men med utritad stickväg). Den information som står på skärmen sparas också i en textfil. Vid ”Multiple runs” utförs mer än 10 simuleringar i sekunden och därför är det inte möjligt att hinna läsa av vad som står på skärmen mellan varje enskild simulering i simuleringsomgången. Även här skapas en textfil med den information som snabbt passerar på skärmen men eftersom det blir väldigt mycket text att läsa skapas även en summafil där alla resultat, plus ytterligare några, presenteras i en tabell. Som alternativ finns det även vid ”Multiple runs” möjlighet att skapa en trädlista och en bild av ploten för resultatet av varje simulering i simuleringsomgången (anges med parametern *MultipleRunFileOutput* = 1 i parameterfilen, om inte alla dessa filer önskas sätts parametern till 0).

Vid alternativet ”Single run” går det att få ut följande resultat från skärmen: antal valda framtidsträd vid simuleringen, grundyta före simulering, grundyta uttagen i stickvägen, grundyta uttagen vid selektiv gallring och kvarvarande grundyta efter simulering. För att ha möjlighet att testa algoritmen mot verkliga data visas förutom dessa även statistik på hur bra simuleringen sammanfaller med en verklig gallring (vilka träd som togs ut vid den verkliga gallringen presenteras för programmet genom att markera vilka träd som är uttagna i den trädlista som läses in vid start). Detta innefattar: hur många träd som sammanföll i uttaget vid den verkliga gallringen och simuleringen (i % av de som togs ut i den verkliga gallringen, benämns vidare som träff-% och är det viktigaste resultatet vid simuleringarna), hur många träd som togs ut endast i den verkliga gallringen (i % av de träd som togs ut i den verkliga gallringen) och hur många träd som togs ut endast vid simuleringen (i % av de som togs ut vid simuleringen).

Ur den summafil som skapas vid alternativet ”Multiple run” går det att förutom det som anges på skärmen vid ”Single run” även utläsa antal träd som inte stod i den simulerade stickvägen, antal träd uttagna vid den verkliga gallringen, antal träd uttagna vid simuleringen och antal träd uttagna i låggallring (de träd som togs ut i låggallringsfasen).

2.2 Test av algoritmen

2.2.1 Data som användes för att testa algoritmen

Enligt Daume och Robertson (2000) är det viktigt att använda sig av data från verkligheten för att testa tillförlitligheten av en algoritm. För att testa denna algoritm användes data insamlade sommaren 2009 efter verkliga gallringar i Hedmark, Norge. De data som användes var egentligen insamlade för ett annat projekt, men passade bra även för test av denna algoritm. Insamlingen skedde på totalt 34 plot/provytor i 10 olika bestånd som gallrats 5-10 år innan registreringen. Alla bestånd dominerades av tall, och hade valts ut för att täcka ett brett spektra av bonitet, ålder och gallringsstyrka. Fakta om de olika gallringsbestånden går att läsa i Tabell 1.

Tabell 1. Fakta om de bestånd som användes för att testa algoritmen.

Best-ånd nr.	Område	Plats (UTM: zon, öst, nord)	Ålder (år i brösthöjd)	Bonitet (höjd vid brösthöjd-ålder 40 år)	Grundyta före gallring (medel; min-max, m^2/ha)	Grundyta efter gallring (medel; min-max, m^2/ha)	Gallrings-uttag (uttag i % av grundyta före gallring, medel; min – max, %)	Tid för gallrings-ingrepp (månad, år)	Antal plots/ prov-ytor
1	Elverum	32v, 657207, 6757643	48	15	31.0; 23.8 - 39.1	17.1; 15.5 - 18.7	43.3; 34.0 - 52.1	feb. 2003	4
2	Elverum	32v, 657699, 6757950	48	15	30.1; 26.3 - 34.2	12.8; 9.5 - 15.6	57.7; 52.4 - 64.0	apr./maj 2001	4
3	Elverum	32v, 637817, 6755953	54	12	20.0; 18.1 - 21.5	14.1; 12.9 - 16.0	29.1; 11.7 - 36.2	sept. 2003 / feb. 2004	4
5	Rendalen	32v, 608168, 6870025	55	14	26.0; 21.7 - 29.9	14.4; 12.4 - 15.8	43.7; 28.8 - 53.0	okt. 2003	4
6	Grue	33v, 340672, 6701096	25	20	27.6; 24.1 - 31.3	20.1; 18.8 - 21.8	26.8; 22.0 - 30.2	juni 2001	3
8	Kongs-vinger	33v, 343016, 6685602	35	18	21.1; 11.2 - 26.7	13.0; 8.5 - 17.8	35.8; 23.8 - 50.2	maj 2001	3
9	Kongs-vinger	33v, 339339, 6679903	23	17	16.3; 14.7 - 19.1	14.5; 12.9 - 16.8	10.7; 8.0 - 12.0	feb. – maj 1999	3
10	Våler	33v, 338871, 6747053	44	12	16.5; 13.0 - 21.4	10.3; 5.9 - 12.9	37.8; 15.3 - 54.4	maj 2004	3
12	Rendalen	32v, 619275, 6842970	72	12	28.1; 26.6 - 29.5	18.4; 15.4 - 21.8	34.7; 22.8 - 42.2	nov. 2002	3
14	Stor – Elvdal	32v, 610491, 6833159	59	11	15.2; 12.4 - 17.2	11.1; 9.9 - 12.6	26.5; 20.2 - 32.8	feb. – mar. 2001	3

Provytorna var alla på 250 m^2 och slumpmässigt utlagda längs en linje genom bestånden. För varje träd på provytorna registrerades diameter i brösthöjd och polär koordinat (avstånd och vinkeln till trädet avläst på en kompass). Det togs även borrkärnor från alla träd för att kunna beräkna ålder och tillväxt hos trädet. Denna borrkärna gjorde det även möjligt att beräkna trädens diameter vid gallringstidspunkten. För alla stubbar som härrörde från den senaste gallringen registrerades stubbdiameter och polär koordinat. Att diametern av träden vid gallringstidspunkten baserades på beräkningar skulle kunna medföra en risk att dessa inte är helt korrekta, men på grund av att spridningen i diameter i brösthöjd av träden var stor ansågs inte detta kunna påverka resultatet nämnvärt.

För att estimera diameter i brösthöjd baserat på stubbdiameter för träden som var uttagna användes data från diametermätningar från 8899 tallar som registrerats av en skördare vid föryngringsavverkning år 2007 i Aurskog-Høland kommun i sydöstra Norge (Bollandsås et al. 2010). Barktjocklek för de uttagna träden estimerades med hjälp av en modell som baserades på barktjocklek på de uppmätta provytorna.

Det var inte känt vilka som varit förare (och därmed också valt vilka träd som togs ut och vilka som lämnats kvar) av gallringsmaskinerna vid gallringarna. Men på grund av den stora geografiska spridningen av ploten ansågs det troligt att det var många olika chaufförer. Det är inte heller känt

om skogsägarna för de olika ploten har haft några speciella önskemål för hur gallringarna skulle utföras.

2.2.2 Parametervärden som användes för att testa algoritmen

För att kunna genomföra simuleringar behöver en rad parametervärden bestämmas. Plotens storlek sattes till 17,84124116 m (diameter av en cirkulär plot på 250 m^2) i både x- och y-led. Genom att jämföra tekniska data för några stora maskintillverkare bestämdes kranens längd (*harvester_range*) till 10 meter (*Eco Log* 2009; *Gremo* 2010; *John Deere* 2010; *Ponsse* 2008; *Rottne* 2010; *Valmet* 2010). Agestam (2009), Holmen skog (2003) och Ovaskainen et al. (2006) menar alla att en stickväg bör vara 4 meter bred, Fransson (2008) visade i sitt arbete att den ofta är ännu lite bredare. Men på grund av att de verkliga stickvägarnas placering i förhållande till ploten inte var kända, och det därför inte var möjligt att placera den simulerade stickvägen på samma plats som den verkliga, så sattes stickvägens bredd till 0 m vid simuleringarna. Ovaskainen et al. (2004) visade i en studie från gallringar i Finland att avståndet mellan uppställningspositionerna (*stop_distance*) varierade mellan 3,3 och 4,1 meter och med ett medelavstånd på 3,68 m, därför bestämdes *stop_distance* till 3,68 m för simuleringarna. I en studie av Ovaskainen et al. (2006) registrerades var träd som gallrades ut stod i förhållande till kranbasen. Nämnda studie visade att det var relativt vanligt att träd upp till 1 m bakom kranbasen togs ut, därför bestämdes *workzone_y_backward* till 1 m.

Målgrundytan (*BA_target*) efter simulering varierades mellan olika plots och sattes till den uppmätta grundytan efter den verkliga gallringen. Det bestämdes också att framtidsträden bara skulle väljas bland tallarna (*croptree_species = 2*). Parametern för minsta tillåtna diameter för framtidsträden (*croptree_dbh_percentile*) sattes till 50 % och minsta skillnad i diameter som kan ses av skördarföraren (*croptree_min_dbh_diff*) sattes till 2 cm. Att *croptree_dbh_percentile* sattes så lågt som till 50 % medförde att den inte hade någon stor betydelse för trädvalet utan mer begränsade extremfallen. Parametern för minsta tillåtna avstånd mellan två framtidsträd (*croptree_mindist_par*) bestämdes till 0,55 för att vara lite större än hälften då ett avstånd på under hälften hade blivit väl litet. Ett större avstånd hade däremot gjort att det kunde bli svårt att få plats med det önskade antalet framtidsträd. Eftersom det inte var känt hur många framtidsträd skördarföraren tänkt sig vid den verkliga gallringen så varierades parametern *croptree_number_ha* för att hitta det bästa alternativet.

Även om ploten för att testa algoritmen var cirkulära valdes det att inte ”fylla ut hörnen” så de blev kvadratiska eftersom de träd som placeras i hörnen är med och påverkar vilka träd som blir framtidsträd och vilka som tas ut. Ytterligare ett problem med denna utfyllnad är att träden i hörnen inte är de samma från simuleringsomgång till simuleringsomgång. Resultatet av simuleringarna kan därför variera mycket mellan olika simuleringsomgångar även om alla övriga parameterinställningar är lika. Det valdes att simulera med fast första uppställningsplats för varje stickväg (Figur 4). För att det skulle bli ett kort avstånd mellan de repeterade simuleringarna och för att i stort sett alla möjliga uppställningsplatser skulle testas simulerades det hela tiden med ”Multiple runs” och 100 repetitioner (*NoReplicates* = 100). Detta resulterade i ett avstånd på 3,68 cm mellan varje repetition.

2.2.3 Testets utförande

Testen av algoritmen utfördes genom att variera antalet önskade framtidsträd per hektar mellan varje simuleringsomgång. För varje plot utfördes simuleringar i intervallet 40 till lägst 800 önskade framtidsträd per hektar i steg om 40 träd per hektar (1 träd på 250 m^2 motsvarar 40 träd/hektar). Det

utfördes inte simuleringar för alla alternativ inom detta spann. Träff-% varierade mellan olika alternativ av *croptree_number_ha* och i de alternativ där det uppnåddes bra träff-% simulerades många alternativ, medan det i områden där träff-% var sämre endast utfördes några få simuleringar för att kontrollera att inte resultaten förändrades oförutsett (se Tabell 3 där det inte utförts några simuleringar i intervallet 160-360 önskade framtidsträd/ha eftersom träff-% var lågt här, men desto fler runt 1000 önskade framtidsträd/ha där träff-% var bättre).

2.2.4 Analys av resultaten

För att analysera resultaten av simuleringarna utöver de uppgifter som gavs i summafilen beräknades en låggallrings-% som beskrev hur stor del av de uttagna träden som tagits ut till följd av låggallringen. Det beräknades också hur stor grundytan efter simulering var jämfört med den efter den verkliga gallringen (som användes som målgrundyta).

3 Resultat

För varje plot utfördes mellan 7 och 25 simuleringsomgångar och totalt för alla 34 plot utfördes 459 simuleringsomgångar (Tabell 2). Vid varje simuleringsomgång utfördes 800 enskilda simuleringar, 100 för varje stickvägsalternativ. Efter varje simuleringsomgång sorterades den bästa enskilda simuleringen bland dessa 800 fram genom att först välja bland de med bäst träff-% och sedan välja den plot där grundytan efter simuleringen stämde bäst överens med grundytan efter den verkliga gallringen (vilken använts som *BA_target*) (benämns vidare som grundyteträff, där 100,0 % är bästa möjliga träff) av samma plot. I de fall ingen av simuleringarna med bäst träff-% uppnådde en tillfredställande grundyteträff söktes det vidare bland simuleringarna med näst bäst träff-% osv. Som gräns för vad som betraktades som tillfredställande grundyteträff sattes två olika gränser, den strängare sattes till max 5 % avvik medan den mindre stränga gränsen sattes till max 10 % avvik. Denna mindre stränga gränsen användes i de fall där en bättre träff-% kunde uppnås genom att sänka kravet på grundyteträff. I de fall där simuleringar med max 10 % avvik i grundyteträff gav bättre träff-% än max 5 % avvik i grundyteträff redovisas båda resultaten.

Det var en del skillnader mellan resultaten av simuleringarna för de olika ploten. Denna skillnad berodde på den stora variationen mellan de olika plots som användes för att testa algoritmen (Tabell 1). Även att det troligtvis var många olika förare av gallringsskördarna på de olika ploten har bidragit till att resultaten varierar.

Träff-% för simuleringarna med ett avvik i grundyteträff på maximalt 5 % varierade mellan 53,3 och 96,2 % (medel 84,3). Sänktes däremot kravet på grundyteträff till maximalt 10 % avvik varierade träff-% mellan 68,8 och 96,2 % (medel 86,5) (se Tabell 2).

Tabell 2. Resultaten av utförda simuleringar för alla testplots.

Plot nr.	Antal önskade framtidsträd /ha (*croptree_number_ha*)	Antal träd definierade som framtidsträd vid simuleringen /ha	Träff-% (i %)	Grundyte-träff (i %)	Låggallrings-% (i %)	Antal utförda simulerings-omgångar
1.1	1000	920	92,7	100,5	33	24
1.2	880	640	88,6	99,1	39	18
1.3	40	0	85,0	96,3	100	16
1.4	680	480	84,6	97,4	54	13
2.1	600	560	80,0	99,7	30	15
	480	440	80,0	99,6	38	
	400	360	80,0	100,6	28	
2.2	520	360	94,1	100,9	49	18
2.3	480	400	92,7	99,4	43	12
2.4	520	400	81,8	100,2	44	16
3.1	400	400	93,8	99,0	33	12
3.2	40	0	92,3	96,1	100	15
3.3	720	640	87,0	96,0	46	18
3.4	1160	760	91,7	97,0	39	11

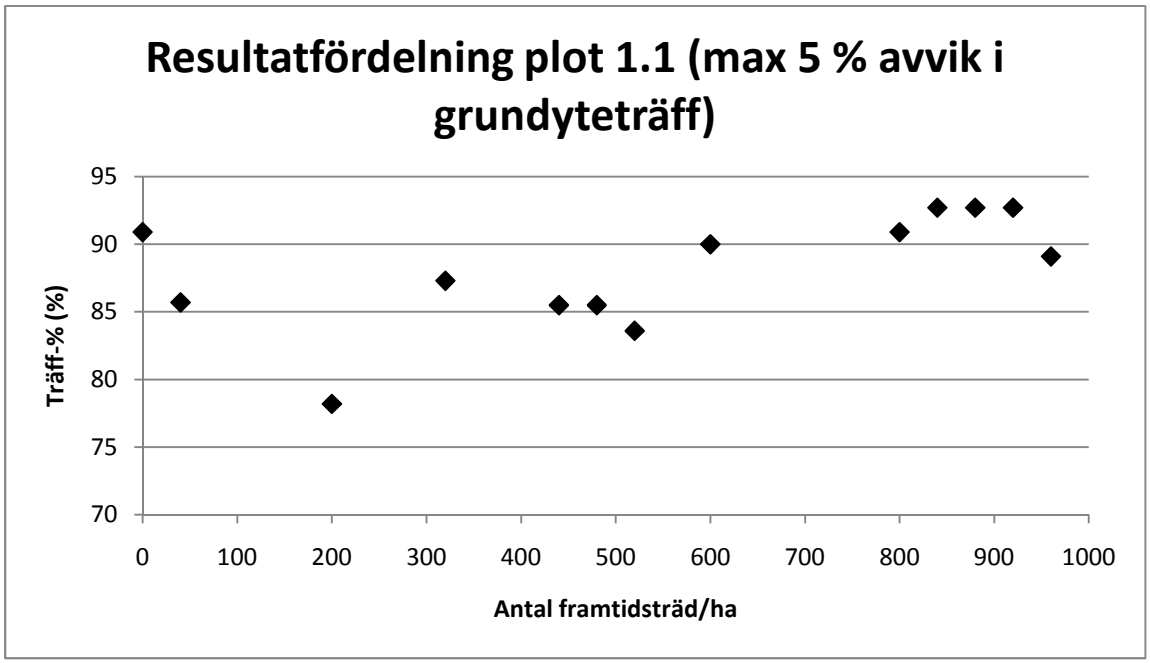
5.1	640	520	78,6	99,4	19	12
5.2	520	320	88,9	99,0	50	9
5.3	360	280	89,3	100,9	41	14
	400	360	89,3	100,9	38	
	440	400	89,3	100,9	45	
5.4	400	320	93,8	99,8	39	12
6.1	680	520	68,8	99,0	53	12
6.2	40	0	70,6	95,5	100	25
6.3	40	0	88,0	98,9	100	13
	800	720	88,0	96,7	41	
8.1	480	400	76,5	96,9	50	11
8.2	600	360	80,0	99,1	45	14
8.3	800	280	83,3	100,3	50	13
9.1	40	0	80,0	96,4	100	12
	40	0	86,7	94,2	100	
9.2	40	0	70,0	95,9	100	13
	120	80	75,0	90,6	76	
9.3	40	0	80,0	95,5	100	8
	40	0	91,4	92,2	100	
10.1	40	0	65,4	95,2	100	8
	40	0	80,0	90,3	100	
10.2	40	0	94,4	99,2	100	10
10.3	40	0	93,9	95,7	100	7
	40	0	95,5	94,8	100	
12.1	840	640	91,2	99,9	39	16
12.2	840	600	80,0	96,5	48	10
12.3	480	400	96,2	100,5	43	13
14.1	1000	520	53,3	96,3	21	11
	40	0	80,0	93,1	100	
14.2	240	200	86,7	101,7	79	14
	240	240	93,3	93,8	52	
14.3	800	640	91,7	95,6	42	14
	920	680	95,8	92,4	36	
					Summa	459
					Genomsnitt	13,5

När den bästa simuleringen bland de många tusen som utfördes för varje plot sorteras ut kan det tyckas att det inte är konstigt att den uppnår en bra träff-%. För att kontrollera att det inte bara är en slump att det hittas ett så bra resultat studerades resultaten för plot 1.1 närmare. Orsaken till att just denna plot valdes för fördjupade studier var att resultaten av simuleringsomgångarna för denna var typiska för testmaterialet. För denna plot hade det också utförts många simuleringsomgångar (24 st., vilket är näst flest) vilket gör att det är enkelt att utläsa hur resultaten av simuleringsomgångarna ofta varierar.

För plot 1.1 utfördes simuleringsomgångar i spannet *croptree_number_ha* = 40 till 1120 framtidsträd per hektar. Detta resulterade i ett resultatutfall på mellan 0 och 960 simulerade framtidsträd per hektar efter simuleringen och en träff-% på mellan 78,2 och 92,7 % (Tabell 3 och Figur 8). På samma sätt som i Tabell 2 redovisas resultaten för både 5 och 10 % avvik i grundyteträff.

Tabell 3. Resultaten av alla simuleringsomgångar för plot 1.1. Den som bedömts som bäst är markerad med grått.

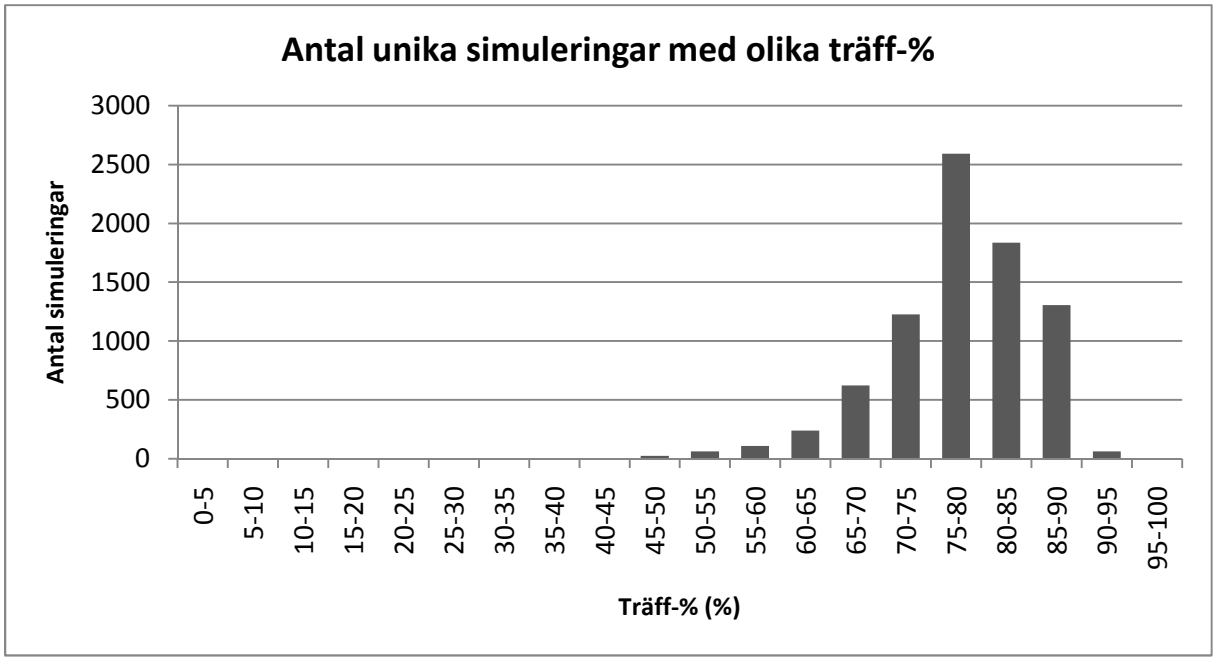
Antal önskade framtidsträd/ha (*croptree_number_ha*)	Antal hittade framtidsträd vid den bästa simuleringen/ha	Träff-% (i %)	Grundyteträff (i %)
40	0	90,9	99,3
80	0	90,9	99,3
120	40	87,3	93,6
	40	85,5	98,1
160	200	80,0	94,7
	200	78,2	101,4
360	320	87,3	97,0
400	360	87,3	91,3
	440	85,5	95,8
440	480	89,1	93,7
	480	85,5	98,4
480	440	85,5	94,5
	520	83,6	95,9
520	480	87,3	93,6
	520	83,6	96,7
560	600	85,5	96,6
600	560	92,7	94,0
	600	90,9	95,2
640	600	90,9	100,4
680	600	90,9	99,6
720	800	90,9	104,8
760	800	90,9	104,8
800	800	90,9	104,8
840	800	90,9	99,2
880	840	92,7	98,0
920	880	92,7	98,0
960	880	92,7	98,0
1000	920	92,7	100,5
1040	920	92,7	100,5
1080	920	92,7	100,5
1120	960	89,1	103,5



Figur 8. Resultatet av simuleringarna med maximalt 5 % miss i grundyta för plot 1.1, endast det bästa alternativet för simulerat antal framtidsträd redovisas.

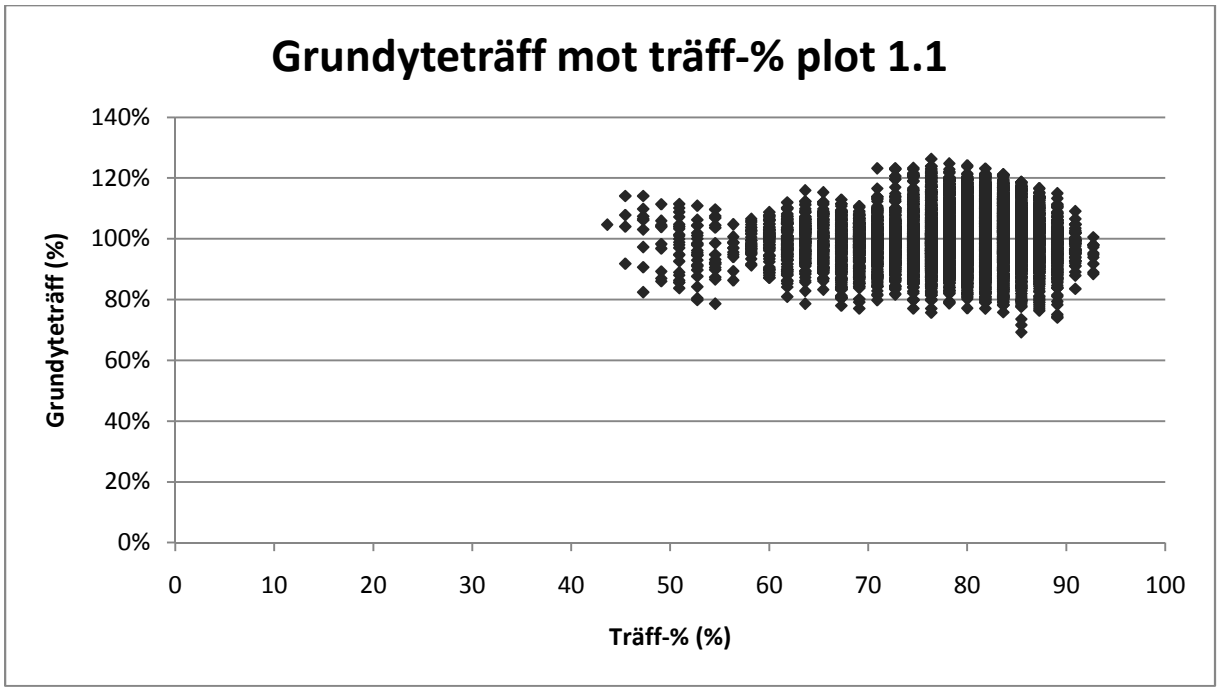
Som bästa simuleringsomgång för plot 1.1 utsågs alternativet med *croptree_number_ha* = 1000 framtidsträd/ha. Att just denna simuleringsomgång valdes bland de tre som är helt identiska berodde på att antalet simulerade framtidsträd/ha skilde sig minst från det önskade antalet framtidsträd/ha. Den bästa simuleringen vid denna simuleringsomgång hade en träff-% på 92,7 % och en grundyteträff på 100,5 % (Tabell 2 och gråmarkerad rad i Tabell 3). Innan något ingrepp skett stod det 84 träd på plot 1.1 och i den verkliga gallringen hade 55 av dessa tagits ut. Vid den bästa simuleringen för plot 1.1 togs 61 träd ut, av vilka 51 sammanföll med de som tagits ut vid den verkliga gallringen. Av simuleringens uttag på 61 träd skedde 41 på grund av att de var konkurrenter, medan de resterande 20 togs ut vid låggallring. Att grundytan efter simuleringen var högre än efter den verkliga gallringen tyder på att diametern på träden uttagna vid simuleringen var mindre än vid den verkliga gallringen.

På grund av att avståndet mellan uppställningsplatserna inte var längre än 3,68 cm mellan de olika simuleringarna blev många av simuleringarna identiska. Av de 19 200 simuleringar (24 simuleringsomgångarna * 800 simuleringar för varje simuleringsomgång) som utfördes för plot 1.1 fanns det 8 072 unika lösningar. Fördelningen av de 8 072 unika resultaten kan ses i Figur 9.

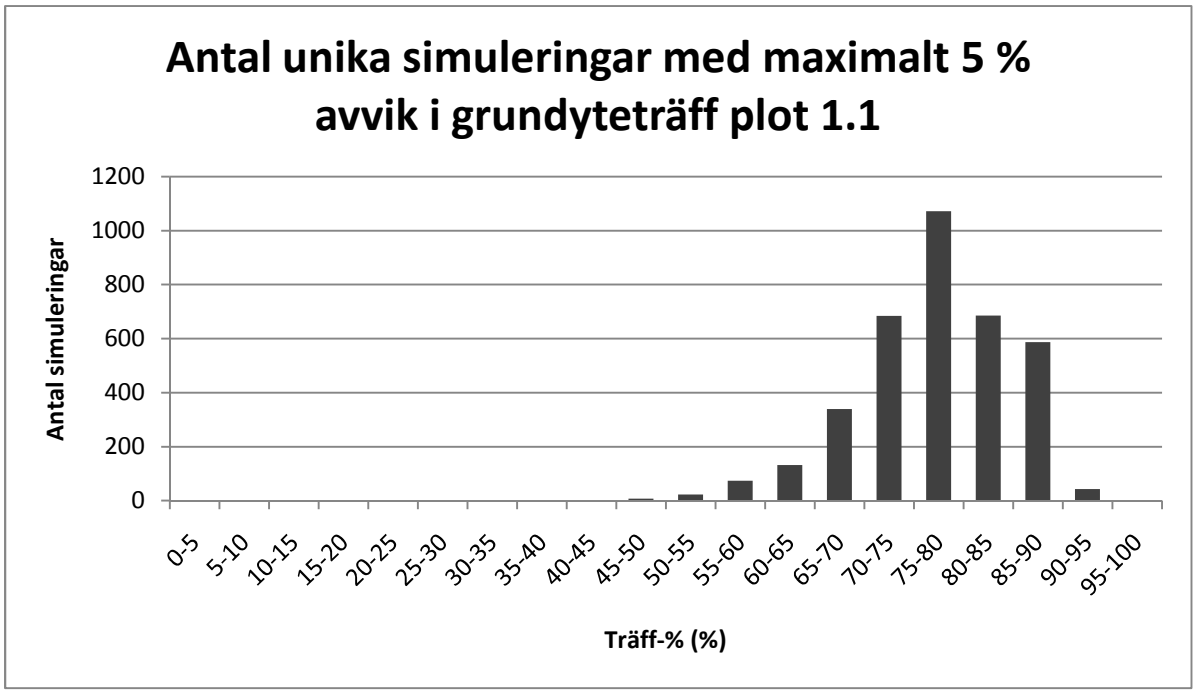


Figur 9. Fördelningen av träff-% för alla 8 072 unika simuleringar för plot 1.1.

Bland dessa 8 072 unika resultat var det inte alla som nådde upp till de fastsatta maximala avviken i grundyteträff (se Figur 10), 6 080 simuleringar låg innanför maximalt 10 % avvik och 3 649 låg innanför maximalt 5 % avvik i grundyteträff. I samma figur syns också tydligt att de simuleringar som inte klarade av gränserna för grundyteträff ligger utspridda över i stort sett hela spektret med träff-%. Detta medför att fördelningen av resultaten när alla som inte klarade av gränsen för avvik i grundyteträff tagits bort fortfarande liknande den i Figur 9. Fördelningen av alla unika resultat innanför ett maximalt avvik av 5 % i grundyteträff går att se i Figur 11.

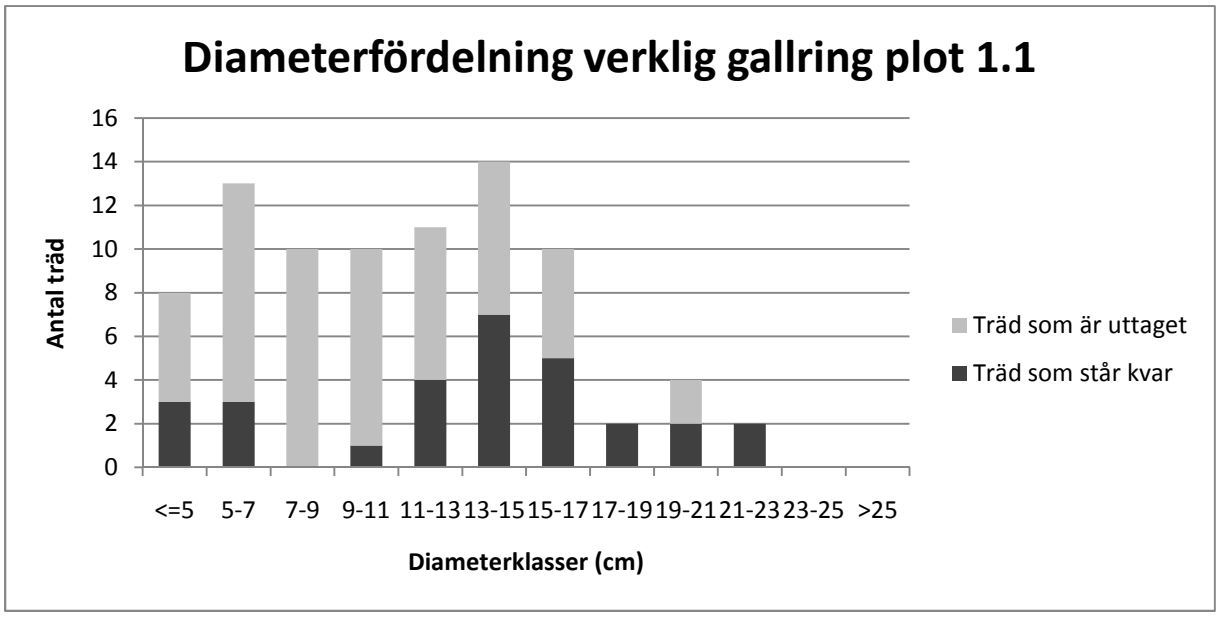


Figur 10. Fördelning i grundyteträff och träff-% för alla 8072 unika resultat för plot 1.1

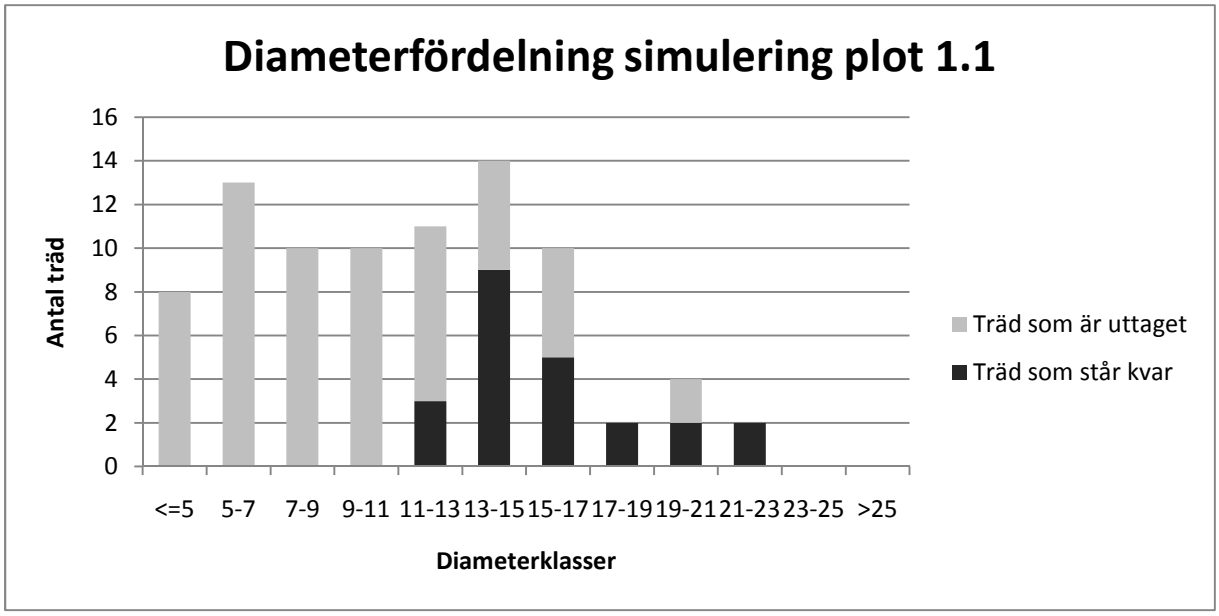


Figur 11. Fördelningen av träff-% för de 3 649 unika resultat för plot 1.1 som hade ett maximalt avvik i grundyteträff på 5 %.

För att uppnå en så hög träff-% som de flesta av simuleringarna gjorde så borde uttaget i de olika diameterklasserna vid den enskilda simuleringen också vara bra. För att kontrollera detta jämfördes uttaget i de olika diameterklasserna från den verkliga gallringen (Figur 12) med uttaget från den bästa simuleringen (Figur 13). En jämförande blick på dessa båda figurer visade att så också är fallet.

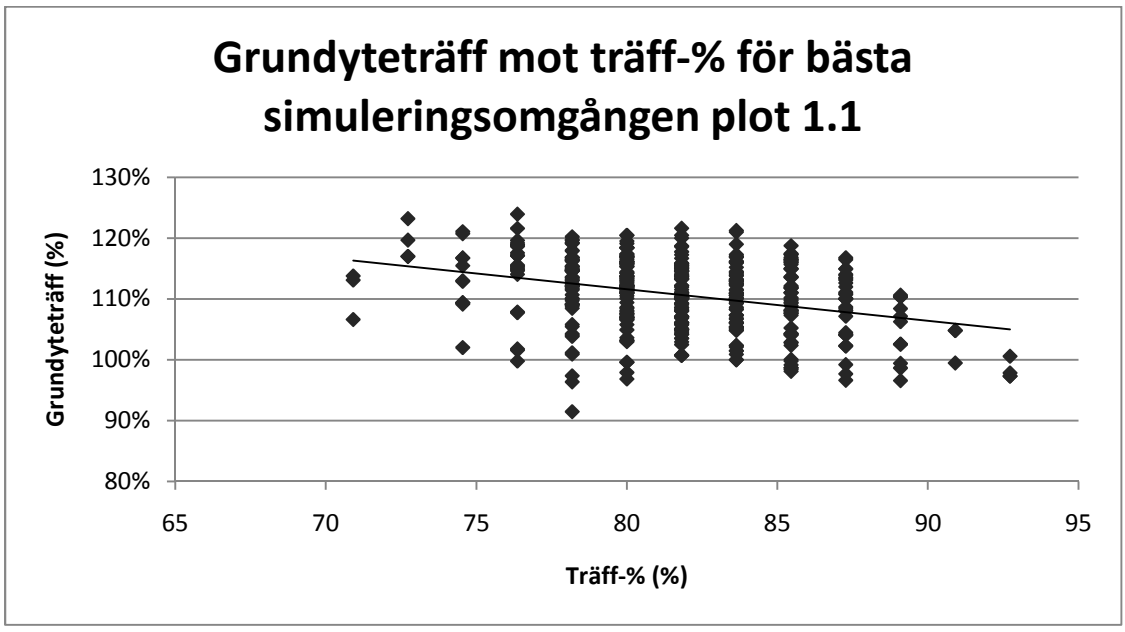


Figur 12. Uttagna och kvarlämnade träd i olika diameterklasser efter den verkliga gallringen av plot 1.1.



Figur 13. Uttagna och kvarlämnade träd i olika diameterklasser efter den bästa simuleringen av plot 1.1.

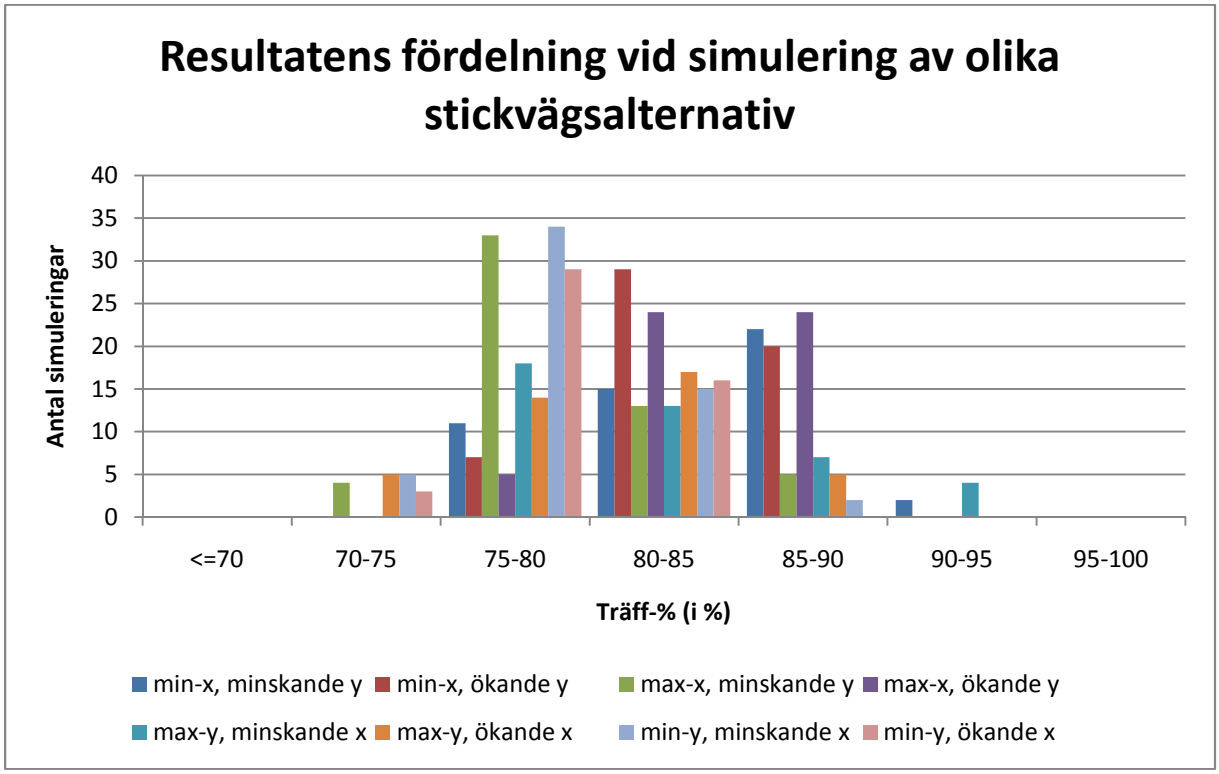
Ju fler träd som tas ut vid simuleringen desto större är sannolikheten att fler av de som togs ut vid den verkliga gallringen tas ut även vid simuleringen. Och ju fler träd som tas ut desto lägre blir den kvarvarande grundytan efter simuleringen. Detta är orsaken till att den största delen av simuleringarna har en grundyteträff som ligger under 100 % (Tabell 2). För plot 1.1 kan man också se att det är ett samband mellan ökande träff-% och sjunkande grundyteträff (Figur 14).



Figur 14. Grundyteträff mot träff-% för alla unika simuleringar för den ”bästa” simuleringsomgången för plot 1.1. Den inritade svarta linjen är trendlinjen.

Vilket stickvägsalternativ som simulerades hade även det betydelse för resultatens fördelning (se Figur 15). Skillnaden var inte stor mellan de olika stickvägsalternativen och det var inget alternativ som klart skilde ut sig som bättre eller sämre. I samma figur går det också att se att resultaten för

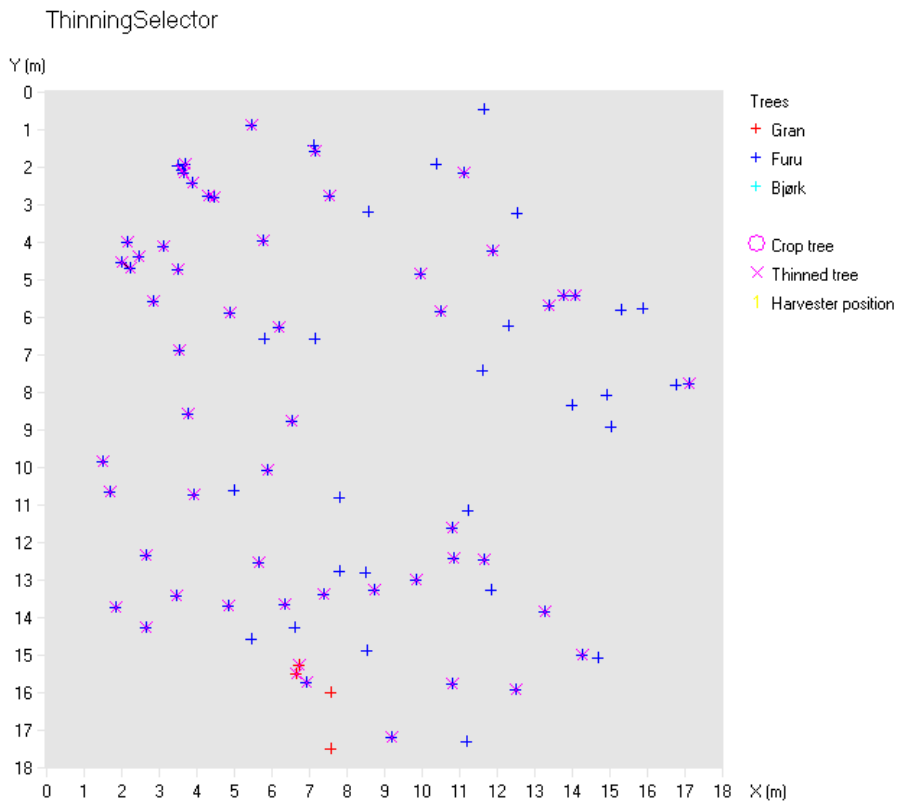
varje stickvägsalternativ fördelade sig över i stort sett hela spektrumet av resultat. Eftersom det enda som förändrades mellan de olika simuleringarna vid samma stickvägsalternativ var positionen för den första uppställningsplatsen så tyder det på att denna har stor betydelse för resultatet. Avståndet mellan två olika resultat var endast i genomsnitt 7,5 cm för alla åtta olika stickvägsalternativen (7,4; 6,6; 6,7; 7,0; 8,9; 9,0;6,6 respektive 7,5 cm för de åtta olika).



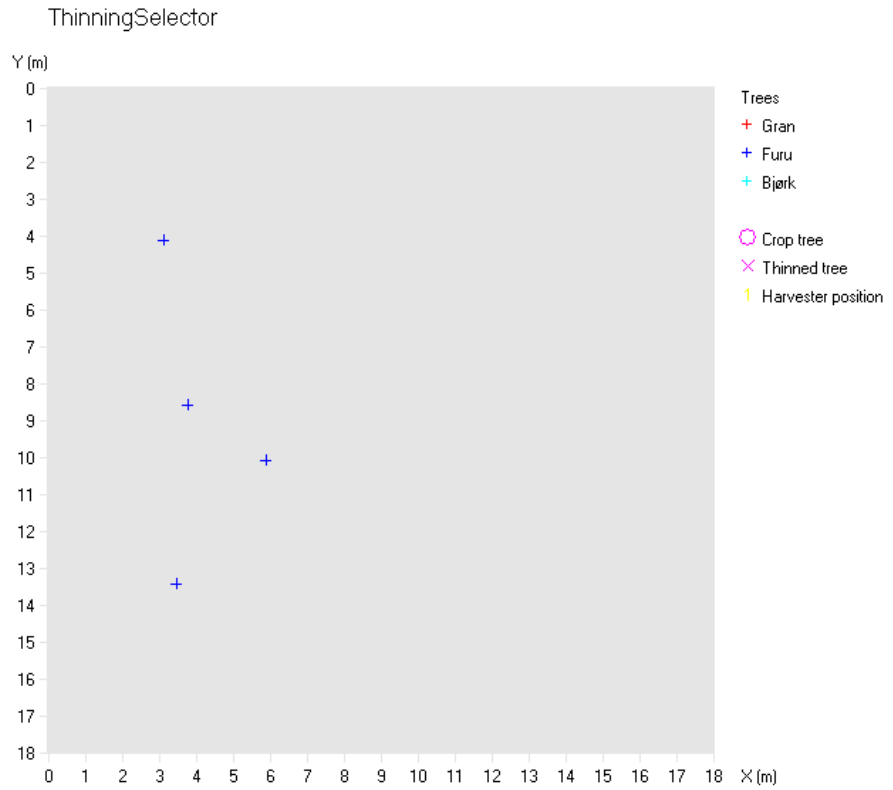
Figur 15. Resultatens fördelning för de åtta olika stickvägsalternativen (Figur 2) för den bästa simuleringsomgången av plot 1.1 (endast unika resultat).

Genom att se på programmets uppritning av ploten efter den verkliga gallringen var det till en viss grad möjligt att utläsa om det kunde gå någon stickväg genom ploten. På sex av ploten (plot 1.1, 2.4, 3.2, 10.2, 12.1 och 14.1) såg det ut som det gick en stickväg genom ploten, medan det för ytterligare fyra plot (plot 3.1, 3.3, 5.2 och 12.3) var möjligt att det kunde gå en stickväg genom, även om de var mer osäkra. På plot 2.2 och 2.3 var så många träd uttagna att det kunde gå en stickväg i stort sett var som helst.

För plot 1.1 såg det ut som det gick en stickväg genom vänsterkanten (se Figur 16), av de fyra träd som tagits ut vid den verkliga gallringen men inte vid den bästa simuleringen av plot 1.1 verkade tre stå i stickvägen (Figur 17). Om dessa träd stått kvar efter den verkliga gallringen om de inte stått i stickvägen är ovisst, men möjligheten finns. På detta sätt kan stickvägen ha varit med och påverkat träff-% på även de andra plot där det möjligtvis går en stickväg igenom.

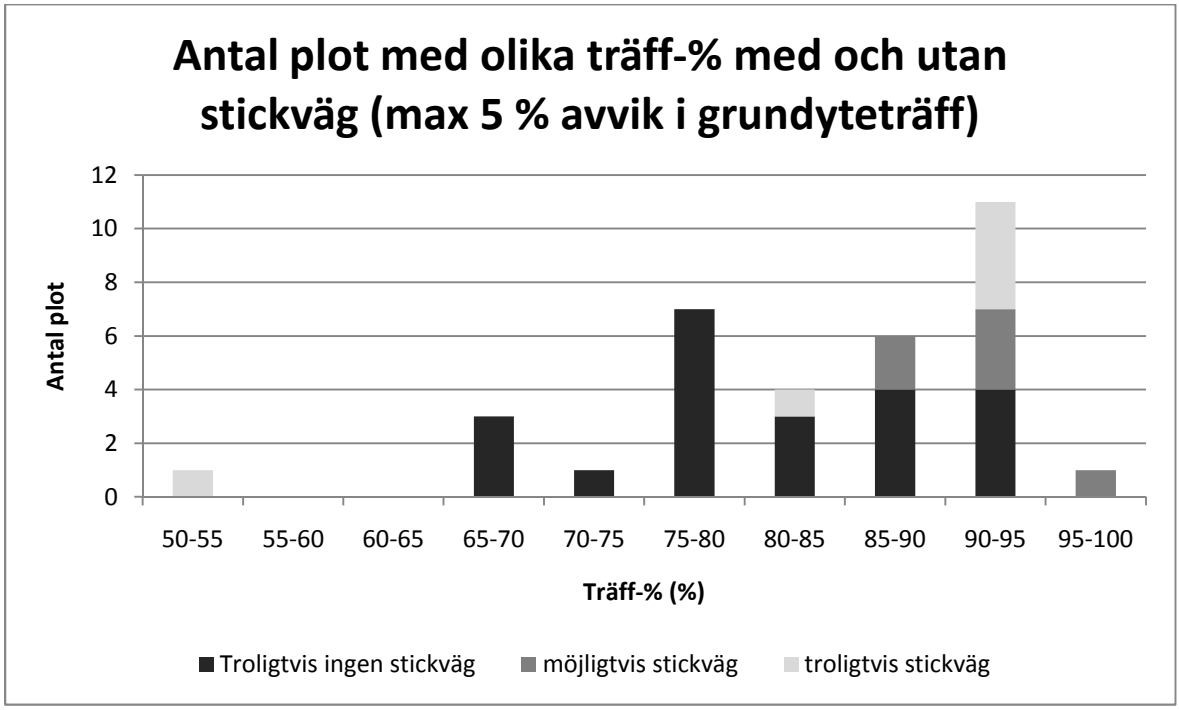


Figur 16. Plot 1.1, där det verkar finnas en stickväg från den verkliga gallringen i vänsterkanten (alla träd här är uttagna).



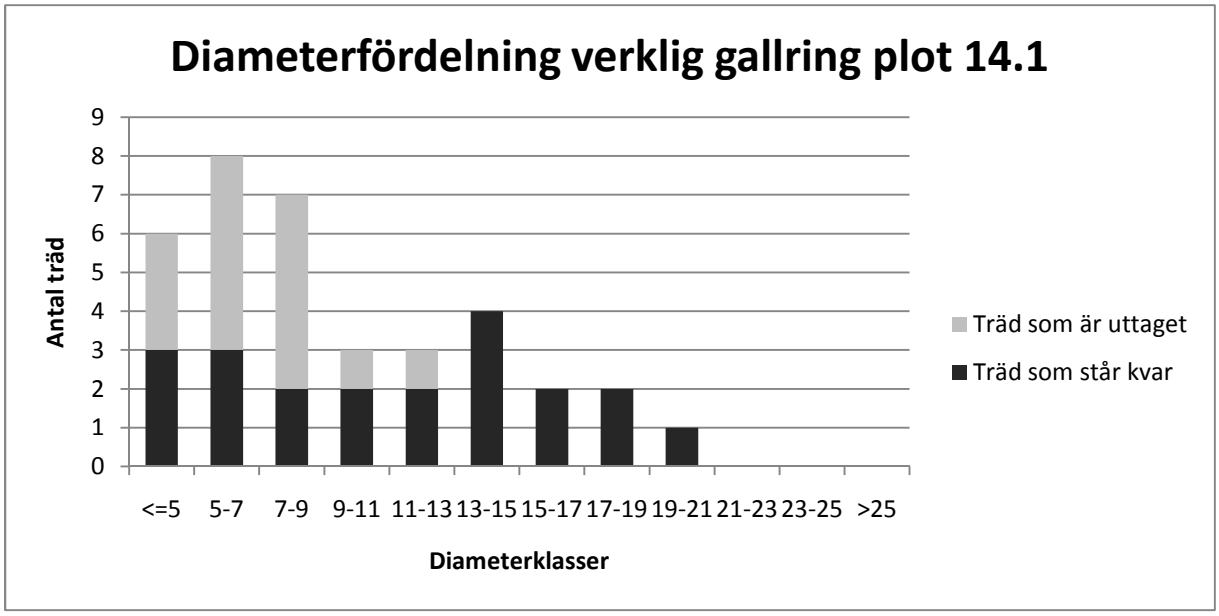
Figur 17. Av de träd som inte togs ut vid simuleringen verkar tre stå i den verkliga stickvägen (de tre till vänster), jämför med Figur 16.

Även om ovissheten med stickvägarna troligtvis påverkar träff-% av simuleringarna negativt så verkar det inte vara den enda orsaken till att träff-% är sämre på vissa plot än på andra. I Figur 18 syns det att de flesta plots där det möjligtvis kan finnas en stickväg finns i den övre delen av träff- % -skalan. Undantaget är ploten med den dåligaste träff-% i hela undersökningen.

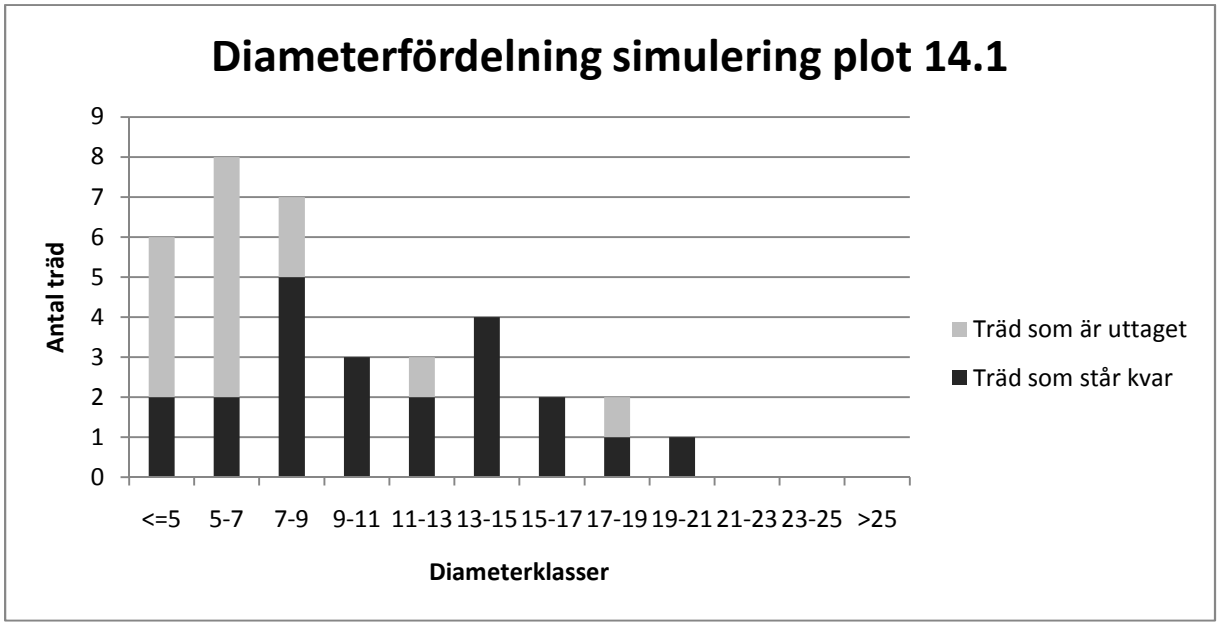


Figur 18. Antal plot med olika träff-% och om de innehåller någon stickväg vid den verkliga gallringen. Ploten där det kunde gå en stickväg i stort sett var som helst finns med i gruppen "möjligtvis stickväg".

Den plot som i Figur 18 skiljer ut sig med en klart sämre träff-% än de andra är plot 14.1. Detta är en plot med förhållandevis många små träd och där endast träd ur de klenare diameterklasserna tagits ut vid den verkliga gallringen (se Figur 19). Denna kan jämföras med det simulerade uttaget som syns i Figur 20.

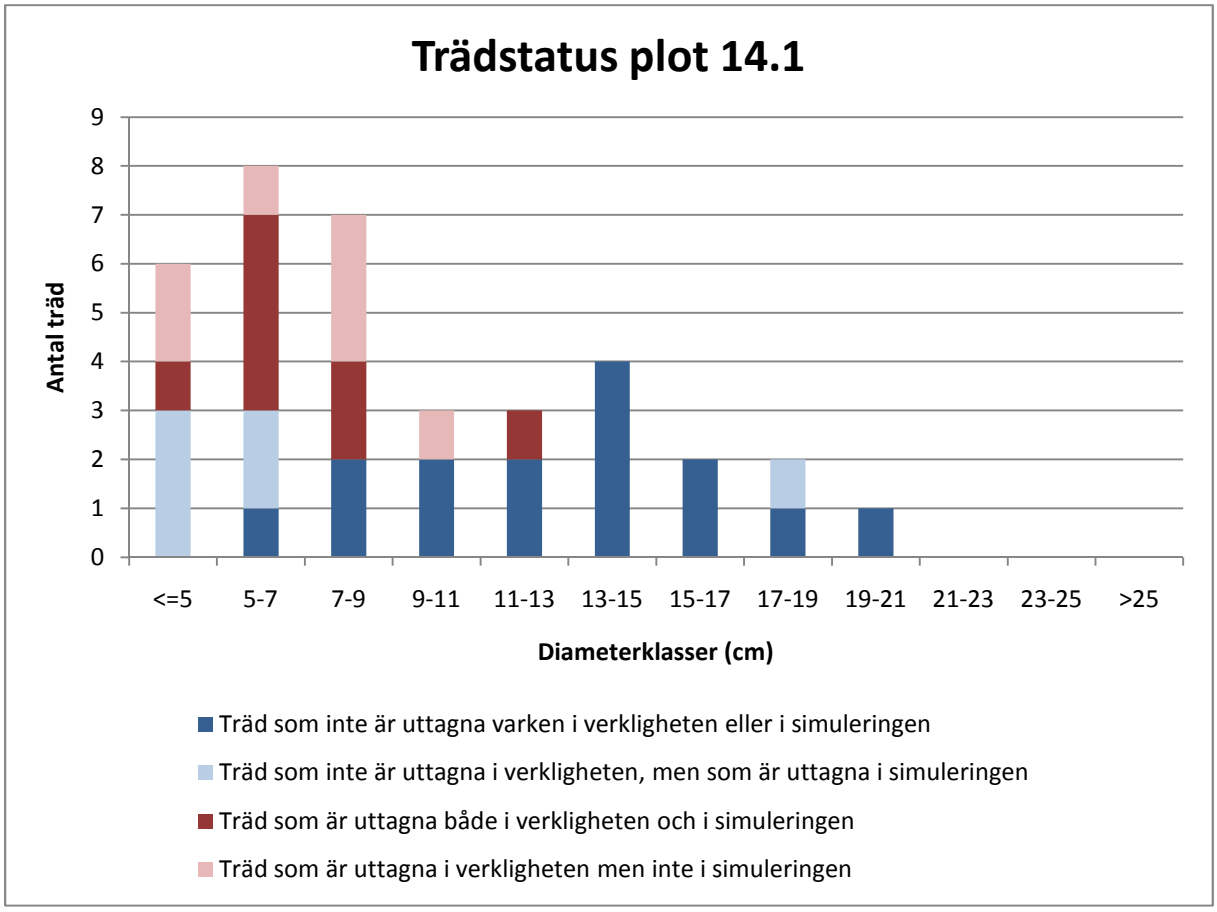


Figur 19. Diameterfördelning och gallringsuttag efter verklig gallring plot 14.1.



Figur 20. Diameterfördelning och gallringsuttag efter simulering plot 14.1 (simuleringsalternativet med 560 sparade framtidsträd).

Trots att uttaget i de olika diameterklasserna ser bra ut vid den bästa simuleringen, går det att med en mer noggrann kontroll av vilka träd som tas ut i respektive ingrepp se att även om antalet uttagna träd i de olika diameterklasserna är bra så har ”fel” träd tagits ut vid flera tillfällen, det vill säga att många av de träd som tagits ut i den verkliga gallringen inte togs ut i simuleringen, istället var det andra träd som togs ut vid simuleringen (Figur 21).



Figur 21. Plot 14.1 är den plot som har sämst träff i uttaget, och det syns här att det beror på att ”fel” träd tagits ut, främst i de klenaste diameterklasserna.

Hur stor andel av de uttagna träden vid simuleringarna som berodde på att träden var konkurrenter eller på låggallringen varierade mellan de olika ploten (se Tabell 2). Däremot var det ingen plot där det inte utfördes någon låggallring alls. Detta visar att det i olika omfattning alltid utförts låggallring vid de verkliga gallringarna, oftast i kombination med att konkurrenter tagits ut. Men det förekommer också plot där det endast utförts låggallring. Det senare inträffar i de fall där det inte simuleras några framtidsträd, och därmed inga konkurrenter, utan alla träd som tas ut tas ut i låggallring.

Resultaten av de utförda testerna visar att det är tillräckligt att variera antalet önskade framtidsträd mellan olika simuleringar för att uppnå ett tillfredställande resultat av simuleringarna.

4 Diskussion

4.1 Utveckling av algoritmen

Det konkurrensindex som användes bygger på principen att ju större en konkurrent är, eller ju närmare framtidsträdet den står, desto större konkurrenspåverkan har den. Detta är ett samband som verkar logiskt (Pukkala 1989) och som är enkelt att beräkna. Om detta mått på konkurrens är det biologiskt optimala är tveksamt, men det är inte det intressanta. Det använda konkurrensindexet verkar i alla fall beskriva hur skördarföraren tänker vid gallringen.

Ett problem som inte har kunnat påvisas men som skulle kunna vara med och påverka resultaten vid simuleringar med en algoritm som denna där det definieras arbetsområden är att det inte tas hänsyn till de träd som står strax utanför dessa. I verkligheten finns det inga så skarpa gränser som de som skapas i algoritmen (enligt Gellerstedt (2002) kan föraren av gallringsskördaren observera trädens positioner inom ett avstånd av upp till ca 1,5 gånger kranens räckvidd), och ett träd som står strax utanför arbetsområdet kan på så vis tas hänsyn till i verkligheten medan det inte ingår i det aktuella arbetsområdet för algoritmen och därför ignoreras helt i söket efter framtidsträd. Problemet med detta uppstår först om det, i det aktuella arbetsområdet, väljs ett halvstort framtidsträd som står nära gränsen och som gör att ett träd som står strax utanför inte kan väljas som framtidsträd i nästa arbetsområde, även om det egentligen är ett mer lämpat framtidsträd. Felet förvärras ytterligare om det träd som står utanför det aktuella arbetsområde tas ut som en konkurrent till trädet innanför gränsen, även om det egentligen borde varit motsatt. Det kan också uppstå fel på grund av att arbetet alltid sker på vänster sida av stickvägen före höger. Detta gör att bland grova träd nära stickvägen så favoriseras de på vänster sida som framtidsträd framför de på höger. Problemet blir större när simuleringarna utförs (som i de utförda testerna) utan eller med endast en smal stickväg. Vid simuleringar med bred stickväg tas träden i en så pass bred gata (stickvägen) ut att det inte är många träd som kan påverka träden på andra sidan stickvägen. Ingen av dessa har kunnat påvisas, men det är viktigt att ha dessa båda felkällor i åtanke vid denna typ av simuleringar.

Vid valet av framtidsträd och konkurrenter är det flera faktorer som är svåra att definiera. Alla parametrar kring maskinen och stickvägen, och därmed arbetsområdet, är relativt enkla att bestämma. Däremot är det värre att bestämma minsta tillåtna avstånd mellan framtidsträden och inom vilken radie det går att räkna med att ett träd kan anses som en konkurrent (Bella 1971). Men även om det är bra att veta inom vilket avstånd ett träd kan klassas som konkurrent så har det inte så stor betydelse för denna studie eftersom den främst eftersträvar att modellera skördarförarens arbete och inte biologin bakom trädens växt.

4.2 Resultaten

Daume & Robertson (2000) beskriver svårigheten med att välja vilken metod som ska användas för att beskriva en gallringsalgoritms prestation i jämförelse med verkliga gallringar. Liksom hos Daume & Robertson (2000) användes för testerna av denna gallringsalgoritm en metod där uttaget av det enskilda trädet jämfördes mellan simulering och verklig gallring. Denna metod gör det möjligt att i detalj studera om de träd som tagits ut i den verkliga gallringen också tagits ut vid simuleringarna, vilket i sin tur är en bra mätsticka på om algoritmen klarar av att simulera den verkliga gallringen.

Testerna av denna algoritm utfördes på många olika plot som gallrats av många olika maskinförare. Att resultaten ändå blir så lika med i stort sett samma parameterinställningar (endast målgrundyta

och antal önskade framtidsträd varierades) är ett bra betyg för gallringsalgoritmen eftersom den då inte bara är anpassad till en enda maskinförare och tillvägagångssätt. Att resultaten inte skiljer sig mer åt visar också att det finns stora likheter mellan olika förare av gallringsmaskiner i denna region i Norge. Att en så stor del av gallringsuttaget kunde beskrivas som låggallring var väl egentligen inte helt väntat eftersom en gallring till största delen har till uppgift att förbättra förutsättningarna för de kvarvarande träden. Dessa små träd som togs ut vid låggallringen är troligtvis så små att de i de flesta fall inte påverkar de större framtidsträden som främst ska gynnas. Av egna erfarenheter vet jag däremot att det kan finnas flera orsaker till att de små träden tas ut. För det första så underlättar det arbetet med kranen om många av de små obetydliga träden tas bort, detta i sin tur minskar också risken för att skada framtidsträden eftersom det blir bättre plats att manövrera kranen. En annan orsak är att många skogsägare vill ha ett ”rent och snyggt” bestånd där allt ”småskräp” är borta. Detta leder till att träd som egentligen lika gärna kunde stå kvar ändå tas ut.

De cirkelformade ploten som användes för testerna av algoritmen hade en area på 250 m^2, och en diameter på ca 17,84 m, detta medförde att det endast krävdes ett enda gallringsstråk, och därmed också endast en stickväg, för att simulera gallring på hela ploten (2*harvester_*x* =19,08 m). Att diametern av ploten och gallringsstråkets bredd var nästan lika stora medförde också att stickvägen hamnade på i stort sett samma plats oberoende av om startpositionen var min- eller max-x. På detta sätt blev troligtvis inte heller skillnaden i trädval speciellt stor.

Att denna gallringsalgoritm uppnår en så hög träff-% som den gör utan att ta någon som helst hänsyn till annat än trädens diameter och position i förhållande till varandra är mycket bra. Daume & Robertson (2000) uppnådde i test av sin gallringsalgoritm en träff-% på 52 %. De menade också att orsaken till att den inte uppnår en bättre träff-% beror på att skogsarbetaren (i detta fall föraren av gallringsskördaren) ser många fel och andra egenskaper hos träden som inte algoritmen tar hänsyn till. Denna gallringsalgoritm visar att för gallring i norska tallbestånd av den typen som användes för att testa gallringsalgoritmen så går det att utan kännedom av kvalitetsegenskaper hos träden uppnå en betydligt bättre träff-% än den Daume & Robertson (2000) gjorde i sin studie.

Det finns troligtvis flera orsaker till att den utvecklade gallringsalgoritmen inte klarar av att simulera den verkliga gallringen på ett ännu bättre sätt. En viktig orsak är troligtvis att stickvägarnas placering i den verkliga gallringen inte var kända. Detta ledde (som kan ses i Figur 16 och Figur 17) till att det fanns vissa stickvägsuttag i den verkliga gallringen som nästan är omöjliga att simulera med den utvecklade algoritmen. I Figur 18 syns det att de plot där det kan misstänkas att det går en stickväg igenom oftast ändå uppnår en bra träff-%. Detta visar att stickvägarna inte är det enda som påverkar resultatet av simuleringarna. Vad som förutom stickvägarna troligtvis haft betydelse är, som nämndes som huvudorsaken av Daume & Robertson (2000), trädens egenskaper. Gellerstedt (2002) och Lageson (1997) menar att av de träd som tas ut vid gallring (speciellt vid förstagallring) så beror en stor del på att träden har skador som inte är önskade i framtidsbeståndet. Här har detta bevisligen inte haft så stor betydelse som Gellerstedt (2002) och Lageson (1997) beskriver, men det kan vara en av orsakerna till att inte ett bättre resultat uppnåddes. Orsaken till de skilda tolkningarna av varför inte algoritmerna nådde upp till ett bättre resultat kan kanske också ha att göra med skillnader i tillvägagångssätt när det gäller gallring i talldominerade bestånd (som i detta fall) och blandbestånd av bok (*Fagus sylvatica*) och gran (*Picea abies*) (Daume & Robertson 2000).

Det visade sig att låggallring av en plot ofta kan utföras på två olika sätt. Det första och naturligaste sättet är att inte definiera några framtidsträd vid simuleringen utan endast ta ut träd i låggallringsfasen. Det andra sättet är att definiera väldigt många framtidsträd. På detta sätt blir i stort sett alla, eller i alla fall de flesta, stora träd definierade som framtidsträd så de enda träd som finns kvar att ta ut som konkurrenter är träd av klenare dimensioner.

Att positionen för den första uppställningsplatsen för varje stickväg hade betydelse var väntat. Detta beror på systemet med avgränsade arbetsområden som gör att ett val i det första arbetsområdet på en stickväg kan påverka trädvalen utöver hela ploten. Att startpositionen skulle ha så stor betydelse som det visade sig att den hade (Figur 15 och att det endast var 7,5 cm i genomsnitt mellan olika resultat) var däremot inte lika väntat. Detta resultat visar hur mycket de tidiga trädvalen påverkar resten av simuleringen. Vid en verklig gallring hade denna första position troligtvis inte haft en lika stor betydelse eftersom föraren ser lite mer än arbetsområdet och kan förhoppningsvis ta lite mer hänsyn till de val som kommer senare. I denna algoritm är det däremot inte möjligt. Detta visar att det var riktigt att använda sig av många olika positioner vid simuleringarna för att simulera många olika möjligheter.

Även om det utfördes många tusen simuleringar för varje plot så visar fördelningen av resultaten i Figur 9 att inte alla (eller inte ens de flesta) simuleringsmöjligheter har simulerats eftersom det inte finns några direkt dåliga resultat. Om detta varit fallet hade det varit helt naturligt att det bland dessa hittades goda resultat. I stället visar Figur 9 att för plot 1.1 var det en klar förskjutning till höger i figuren, vilket visar att de flesta simuleringarna var ”goda”. Ett ytterligare bevis för att inte alla möjligheter är testade är att det inte i något fall inom den accepterade träffen i grundyta fanns resultat där det uppnåddes 100 % träff. I det fall där alla möjligheter varit testade hade det uppnåtts 100 % träff i någon simulering.

Det faktum att det är 3,68 cm mellan varje simulering och i genomsnitt 7,5 cm mellan olika resultat vid simuleringarna visar att 100 repetitioner var tillräckligt. Även om det hade använts fler repetitioner så hade det troligtvis inte uppnåtts fler, och möjligtvis bättre, resultat.

Av de resultat som uppnåtts vid simuleringarna går det att dra slutsatsen att algoritmen, med de parametervärden som användes vid denna studie, på ett mycket bra sätt beskriver hur skördarföraren tänker vid en gallring.

5 Slutsats

Genom att försöka efterlikna det arbete som utförs vid en gallring har det utvecklats en gallringsalgoritm som på ett bra sätt beskriver hur gallringar av talldominerade bestånd i Hedmark, Norge utförs. Även om det använts enkla samband och metoder för att beskriva komplicerade processer som konkurrens, så visar testerna av algoritmen att den på ett mycket bra sätt lyckas simulera det uttag av träd som sker vid en gallring. Att resultaten uppnåddes vid tester på bestånd av så skiftande karaktär visar också att den utvecklade algoritmen klarar av att simulera gallringar vid olika förutsättningar. Testerna visar att det med hjälp av den utvecklade gallringsalgoritmen även går att beskriva de val som utförs vid en gallring.

Testerna visade att det är relativt enkelt att använda sig av den utvecklade gallringsalgoritmen. Genom att anpassa alla maskinparametrar efter den aktuella skördaren och de övriga parametrarna på samma sätt som gjordes i studien så räcker det att variera antalet önskade framtidsträd för att simulera olika lösningar.

6 Referenser

Agestam, E. (2009). Skogsskötselserien nr 7, Gallring. *Skogsskötselserien*. 83 s.

Albrektson, A., Elfving, B., Lundqvist, L. & Valinger, E. (2008). Skogsskötselserien nr 1, Skogsskötselns grunder och samband. *Skogsskötselserien*. 84 s.

Bella, I. E. (1971). New competition model for individual trees. *Forest Science*, 17 (3): 364-372.

Bollandsås, O. M., Maltamo, M., Gobakken, T. & Næsset, E. (2010). Comparing parametric and non-parametric modelling of diameter distributions om independent data using airborne laser scanning. *(Manuskript)*.

Daniels, R. F. (1976). Simple competition indexes and their correlation with annual Loblolly pine tree growth. *Forest Science*, 22 (4): 454-456.

Daume, S. & Robertson, D. (2000). A heuristic approach to modelling thinnings. *Silva Fennica*, 34 (3): 237-249.

Eco Log, Tekniska specifikationer. (2009). Tilgjengelig fra: http://www.eco-log.se/datasheets/harvesters/Eco_Log_Harvesters_sv.pdf (lest 14.01.2010).

Eliasson, L. (1999). Simulation of thinning with a single-grip harvester. *Forest Science*, 45 (1): 26-34.

Fransson, A. (2008). *Vindskador vid stickväg i 1:a och 2:a gallring i Boxholm, Östergötland - i stormen Pers fotspår*. Examensarbete. Alnarp, Sveriges lantbruksuniversitet, Institutionen för sydsvensk skogsvetenskap. 32 s.

Gellerstedt, S. (2002). Operation of the single-grip harvester: motor-sensory and cognitive work. *International Journal of Forest Engineering*, 13 (2): 35-47.

Gremo 1050H - Teknisk data. (2010). Tilgjengelig fra: http://www.gremo.se/?ID=SKORDARE1050H_DATA&sLang=sv-se (lest 14.01.2010).

Holmen Skog. (2003). Gallringshandledning. Örnsköldsvik, Holmen Skog. 24 s.

Hyytiainen, K. & Tahvonen, O. (2002). Economics of forest thinnings and rotation periods for Finnish conifer cultures. *Scandinavian Journal of Forest Research*, 17 (3): 274-288.

John Deere 1070E, Mått. (2010). Tilgjengelig fra: http://www.deere.com/sv_SE/forestry/forestry_equipment/harvesters/machines/index.html (lest 14.01.2010).

Kantola, A., Makinen, H. & Makela, A. (2007). Stem form and branchiness of Norway spruce as a sawn timber - Predicted by a process based model. *Forest Ecology and Management*, 241 (1-3): 209-222.

Lageson, H. (1997). Effect of Thinning Type on the Harvester Productivity and on the Residual Stand. *Journal of Forest Engineering*, 8 (2): 7-14.

Ledermann, T. & Stage, A. R. (2001). Effects of competitor spacing in individual-tree indices of competition. *Canadian Journal of Forest Research-Revue Canadienne De Recherche Forestiere*, 31 (12): 2143-2150.

Makinen, H. & Isomaki, A. (2004a). Thinning intensity and growth of Norway spruce stands in Finland. *Forestry*, 77 (4): 349-364.

Makinen, H. & Isomaki, A. (2004b). Thinning intensity and growth of Scots pine stands in Finland. *Forest Ecology and Management*, 201 (2-3): 311-325.

Makinen, H. & Isomaki, A. (2004c). Thinning intensity and long-term changes in increment and stem form of Norway spruce trees. *Forest Ecology and Management*, 201 (2-3): 295-309.

Makinen, H. & Isomaki, A. (2004d). Thinning intensity and long-term changes in increment and stem form of Scots pine trees. *Forest Ecology and Management*, 203 (1-3): 21-34.

Miina, J. & Pukkala, T. (2000). Using numerical optimization for specifying individual-tree competition models. *Forest Science*, 46 (2): 277-283.

Nilsson, U., Agestam, E., Ekö, P. M., Elfving, B., Fahlvik, N., Johansson, U., Karlsson, K., Lundmark, T. & Wallentin, C. (2010). Thinning of Scots pine and Norway spruce monocultures in Sweden – Effects of different thinning programmes on stand level gross- and net stem volume production. *Studia Forestalia Suecia*, 219: 1-46.

Ovaskainen, H., Uusitalo, J. & Väätäinen, K. (2004). Characteristics and Significance of a Harvester Operators' Working Technique in Thinning. *International Journal of Forest Engineering*, 15 (2): 67-77.

Ovaskainen, H., Uusitalo, J. & Sassi, T. (2006). Effect of edge trees on harvester positioning in thinning. *Forest Science*, 52 (6): 659-669.

Ponsse Beaver, Tekninka data. (2008). Tilgjengelig fra: http://www.ponsse.com/svenska/produkter/skordare/beaver/tekniska_data.php (lest 14.01.2010).

Pukkala, T. (1989). Methods to Describe the Competition Process in a Tree Stand. *Scandinavian Journal of Forest Research*, 4 (2): 187-202.

Riksskogstaxeringen. (2009). Tilgjengelig fra: http://www-riksskogstaxeringen.slu.se/ (lest 19.02.2010).

Rottne H14, Tekniska fakta. (2010). Tilgjengelig fra: http://www.rottne.com/se/ (lest 14.01.2010).

Sirén, M. (1998). *Hakkuukonetyö, sen korjuujälki ja puustovaurioiden ennustaminen. (One-grip harvester operation, it's silvicultural result and possibilities to predict tree damage)*. Doktorsavhandling, Finnish Forest Research Institute. 179 s.

Statistisk sentralbyrå. (2010). Tilgjengelig fra: http://www.ssb.no/skog/ (lest 19.03.2010).

Söderbergh, I. & Ledermann, T. (2003). Algorithms for simulating thinning and harvesting in five European individual-tree growth simulators: a review. *Computers and Electronics in Agriculture*, 39 (2): 115-140.

Valmet 911.4, Specifications. (2010). Tilgjengelig fra: http://www.komatsuforest.com/default.aspx?id=2737&mode=specs (lest 14.01.2010).

Vestlund, K., Nordfjell, T., Eliasson, L. & Karlsson, A. (2006). A decision support system for selective cleaning. *Silva Fennica*, 40 (2): 271-289.

Øyen, B. H. (2003). Tynning i granskog på Sørlandet - effekter på tilvekst, dimensjoner og økonomi. *Rapport fra skogforskningen*, 2/03: 1-16.

7 Bilagor

7.1 Bilaga 1. Exempel på en parameterfil

```
[* ThinningSelector Settings *]
[*** Files ***]
Parameter_input_file          E:\Skola\Masteroppgave\Data\Hedmark\1.1\Hedmark_furu.par
Tree_list_input_file          E:\Skola\Masteroppgave\Data\Hedmark\1.1\Hedmark_furu_1.1.tls
[*** Settings ***]
[- Tree list settings -]
NoSpecies      3
SpeciesNames   Gran          Furu          Bjørk
[- Plot settings -]
plotX_(m)      17.84124116
plotY_(m)      17.84124116
[*** Parameters ***]
[- Strip road parameters -]
striproad_start_(0/1)         0
harvester_direction_(0/1)     0
striproad_width_(m)           0
harvester_range_(m)           10
stop_distance_(m)             3.68
workzone_y_backward_(m)       1
RandomHarvStart_(0/1)         0
[- Crop tree parameters -]
croptree_number_ha_(/ha)      800
croptree_mindist_par_(0..1)   0.55
croptree_species_(listno.)    2
croptree_min_dbh_diff_(cm)    2.0
croptree_dbh_percentile_(%)                 55.0
[- Basal area target -]
BA_target_(m2/ha)             12.5
[*** Options ***]
Circular_plot_(0/1)           1
Circular_plot_expansion_(0/1)               0
NoReplicates   100
MultipleRunFileOutput         0
```

7.2 Bilaga 2. C++ koden för programmet ThinningSelector

```
// Program ThinningSelector
/*
Andreas Brunner, andreas.brunner@umb.no
*/
//---------------------------------------------------------------------------
// ABR-header
#include <stdio.h>
#include <dir.h>
#include <stdlib.h>
#include <time.h>

#include "ThinningSelectorFunc.h"
//---------------------------------------------------------------------------
// Global variables

// File names
char currdir[MAXPATH];
char parinfile[100];
char tlsinfile[100];
char paroutfile[100];
char tlsoutfile[100];
char sumfile[100];

// Constants

// Calculation variables
bool ShowTrees = true;
bool ShowCropTrees = true;
char ShowTreeVar = '0';     // Tree parameter to be displayed in result graph

double Zoom = 100;     // Zoom factor (%)
double CenterX, CenterY;       // Result graph center

bool TreelistCreated = false;
long NTreeList;           // no. of trees in tree list

double h_x[1000], h_y[1000];  // harvester position list
int h_dir[1000];
int h_positionN;  // No of harvester positions

double BA_striproads, BA_selected, BA_remaining, BA_total; // Basal area sums (m2/ha)
double CT_selected;     // Number of selected crop trees
double matchpct1, matchpct2, matchpct3;   // Match between real and simulated thinning trees
int plotorient; // plot orientation for multiple runs
int Nrealthin;    // Number of real thinning trees
int Nsimthin;    // Number of simulated thinning trees
int Nnotstrip;  // Total number of trees not in strip roads
int N_thin_below;      // Number of trees thinned from below (not as competitor for crop trees)
```

```
// Plot, tree list
int nspec;
char spec[11][21];                              // species labels
double plotx;                   // plot dimensions
double ploty;         // plot dimensions

// Parameters
int striproad_start; // strip road system's starting point on the x-axis: 0=minx, 1=maxx from the edge of the stand
int harvester_direction; // 1= increasing y / 0 = decreasing y;

double striproad_width; // Width of strip road (m)
double harvester_range; // Crane maximum range (m)
double stop_distance; // Distance between harvester stops (m)
double workzone_y_backward; // workzone extent opposite to driving direction (m)
int RandomHarvStart; // Option for random start of harvester positions in each strip road

int croptree_number_ha; // number of crop trees / ha
double croptree_mindist_par; // Minimum allowed distance between two crop trees (0 - 1)
int croptree_species; // species for crop trees: 0 = all, 1 ... numbers related to species list in parameter file
double croptree_min_dbh_diff;  // (cm) min. dbh-difference visually detectable by harvester driver
double croptree_dbh_percentile; // (%) Pecentile in dbh distribution that will be used as minimum dbh for crop trees

double BA_target; // Target basal area (m2/ha)

// Options
int Circular_plot;      // Circular plots as input
int Circular_plot_expansion; // Expansion of circular plots into corners
int NoReplicates;       // Number of replicates per strip road system lay out
int MultipleRunFileOutput;      // Option for output of individual run results output during multiple run

// Dynamic data arrays
// tree list
//int numlist[100000];          // list of internal tree number by real tree number
int *treenum;          // external tree number
int *TreeSpecies;
int *TreeClass;        // 0 = default, 1 = crop tree, 2 = selected for thinning, 3 = in strip road
int *TreeClassOld;
int *TreeMarked;
double *treex, *treey;
double *treedbh;

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ThinningSelectorMain.h"
```

```
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
      : TForm(Owner)
{

 // Initialize random number generator
 randomize();

 // Display result image
 whiteimage();

 // Initialize filenames
 char copybuffer[MAXPATH];
 getcwd(currdir, MAXPATH);

 strcpy(parinfile, "\\Data\\Example.par");
 strcpy(copybuffer, currdir);
 strcat(copybuffer, parinfile);
 strcpy(parinfile, copybuffer);
 strcpy(tlsinfile, "\\Data\\Example.tls");
 strcpy(copybuffer, currdir);
 strcat(copybuffer, tlsinfile);
 strcpy(tlsinfile, copybuffer);
 strcpy(paroutfile, "\\Data\\Example-Out.par");
 strcpy(copybuffer, currdir);
 strcat(copybuffer, paroutfile);
 strcpy(paroutfile, copybuffer);
 strcpy(tlsoutfile, "\\Data\\Example-Out.tls");
 strcpy(copybuffer, currdir);
 strcat(copybuffer, tlsoutfile);
 strcpy(tlsoutfile, copybuffer);

}
//---------------------------------------------------------------------------
void __fastcall TForm2::Exit1Click(TObject *Sender)
{
 exit(0);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Opentreelist1Click(TObject *Sender)
{
 if(OpenDialog1->Execute())
 {
 StrCopy(parinfile, OpenDialog1->FileName.c_str());

 // Reset result image
 whiteimage();
 Form2->Image1->Repaint();
```

```
// Reset harvester position list (to delete graph from previous runs)
for (int n=0; n<1000; n++)
                    {
                     h_x[n] = 0;
                     h_y[n] = 0;
      h_dir[n] = 0;
                    }
h_positionN = 0;

// Read parameter file
parfileread(parinfile);

// Delete old data arrays
if (TreelistCreated)
     {
     DeleteTreelist();
     TreelistCreated = false;
     }

// Data array allocation
NTreeList = tlsfilelength(tlsinfile);
if (Circular_plot_expansion==0)
     NewTreelist(NTreeList);
     else
     NewTreelist(round(NTreeList*1.5));
TreelistCreated = true;
if (Circular_plot_expansion==0)
     ResetTreelist(NTreeList);
     else
     ResetTreelist(round(NTreeList*1.5));

// Read tree list file
tlsfileread(tlsinfile);

// Expand circular plots
if (Circular_plot_expansion==1) Expand_circular_plot();

// Graphs
ResultGraph();

}
}
//---------------------------------------------------------------------------

void __fastcall TForm2::SingleRun1Click(TObject *Sender)
{
ThinningSelectorRun();
}
//---------------------------------------------------------------------------

void __fastcall TForm2::MultipleRuns1Click(TObject *Sender)
```

```
{
ThinningSelectorMultipleRuns();
}
//---------------------------------------------------------------------------
```

```
// Program ThinningSelector
/*
Andreas Brunner, andreas.brunner@umb.no
*/
//---------------------------------------------------------------------------
#include <string.h>
#include <math.h>

#include "ThinningSelectorMain.h"
#include "ThinningSelectorFunc.h"
//---------------------------------------------------------------------------
class BGRunError {};    // Exception for error in Run
BGRunError BGRE;
//---------------------------------------------------------------------------
void ThinningSelectorRun()
{
 extern char parinfile[100];
 extern char tlsinfile[100];
 extern char paroutfile[100];
 extern char tlsoutfile[100];

 extern double BA_striproads, BA_selected, BA_remaining, BA_total;
 extern int NTreeList;
 extern double plotx, ploty;
 extern double *treedbh;
 extern int *TreeClass;

 char copybuffer[50];
 char value[10];
 char RunName[100];
 char OutFile[100];

 try
 {
 // Screen log
 Form2->Memo1->Lines->Add("--------------------");
 Form2->Memo1->Lines->Add("ThinningSelector single run started.");
 Form2->Memo1->Lines->Add(DateTimeToStr(Now()));
 Form2->StatusBar1->SimpleText = "Run started.";
 Form2->Memo1->Lines->Add("---");

 // Create run file name
 memset(RunName,'\0',sizeof(RunName));
 if (strstr(parinfile,".par")!=NULL + strstr(parinfile,".PAR")!=NULL)
      strncpy(RunName,parinfile,strlen(parinfile)-4);
      else
                  strncpy(RunName,parinfile,strlen(parinfile));

 // Parameter plausibility check
//if (ParameterCheck()==1) EndRun("Parameter error",1);

 //Read files, check data, establish data structures
```

```
//newplot(ReadGrid);

 // Create sumfile
// memset(sumfile,'\0',sizeof(sumfile));
// strncpy(sumfile,RunName,strlen(RunName));
// strcat(sumfile,".sum");
// sumfileheader(sumfile);
// sumfilewrite(sumfile); // summary statistics for SimYear 0

 // Reset TreeClass
 for (int n=1; n<=NTreeList; n++)
      {
      TreeClass[n] = 0;
      }

 // Reset basal area sums
 BA_striproads = 0;
 BA_selected = 0;
 BA_remaining = 0;
 BA_total = 0;

 // Calculate total basal area
 BA_total = BasalArea(4);

 // Write total basal area to log window
 Form2->Memo1->Lines->Add("Total basal area:");
 gcvt(BA_total,4,value);
 StrCopy(copybuffer, value);
 StrCat(copybuffer, " m2/ha");
 Form2->Memo1->Lines->Add(copybuffer);

 StripRoads(0);
 ThinningSelection();

  // Summary statistics
//  sumfilewrite(sumfile);
 //---------------------------------------------------------------------------
 //Output

 // Write files
      Form2->Memo1->Lines->Add("---");
      Form2->StatusBar1->SimpleText = "Writing result files.";
      Form2->Memo1->Lines->Add("Writing result files.");

      // create par file name
      memset(OutFile,'\0',sizeof(OutFile));
      strncpy(OutFile,RunName,strlen(RunName));
      strcat(OutFile,"-Out.par");
      parfilewrite(OutFile);

      // create tls file name
      memset(OutFile,'\0',sizeof(OutFile));
```

```
      strncpy(OutFile,RunName,strlen(RunName));
      strcat(OutFile,"-Out.tls");
      tlsfilewrite(OutFile);

      // create bmp file name
      memset(OutFile,'\0',sizeof(OutFile));
      strncpy(OutFile,RunName,strlen(RunName));
      strcat(OutFile,".bmp");
      Form2->Image1->Picture->SaveToFile(OutFile);
//---------------------------------------------------------------------------
//Screen log
Form2->StatusBar1->SimpleText = "Run ended.";
Form2->Memo1->Lines->Add("---");
Form2->Memo1->Lines->Add("ThinningSelector single run ended.");
Form2->Memo1->Lines->Add(DateTimeToStr(Now()));
Form2->Memo1->Lines->Add("--------------------");
//---------------------------------------------------------------------------
}     // End of try loop
catch(BGRunError)
{
 // Log
 Form2->StatusBar1->SimpleText = "Run ended.";
 Form2->Memo1->Lines->Add("---");
 Form2->Memo1->Lines->Add("Ended due to error.");
 Form2->Memo1->Lines->Add(DateTimeToStr(Now()));
 Form2->Memo1->Lines->Add("--------------------");
}
//---------------------------------------------------------------------------
// End of run block (for normal and abnormal end of run)
//Delete data structures

// Create logfile
memset(OutFile,'\0',sizeof(OutFile));
strncpy(OutFile,RunName,strlen(RunName));
strcat(OutFile,".log");
Form2->Memo1->Lines->SaveToFile(OutFile);
//---------------------------------------------------------------------------

return;
}
//---------------------------------------------------------------------------
void ThinningSelectorMultipleRuns()
{
 extern char parinfile[100];
 extern char tlsinfile[100];
 extern char paroutfile[100];
 extern char tlsoutfile[100];
 extern char sumfile[100];

 extern double BA_striproads, BA_selected, BA_remaining, BA_total;
 extern int NTreeList;
 extern double plotx, ploty;
```

```
extern double *treedbh;
extern double *treex, *treey;
extern int *TreeClass;
extern int striproad_start;
extern int harvester_direction;
extern int NoReplicates;
extern int plotorient;
extern int MultipleRunFileOutput;

char copybuffer[50];
char value[10];
char RunName[100];
char OutFile[100];
char Repstring[10];
int RunNo;
double newploty;
double newtreey;

try
{
// Screen log
Form2->Memo1->Lines->Add("----------");
Form2->Memo1->Lines->Add("ThinningSelector multiple runs started.");
Form2->Memo1->Lines->Add(DateTimeToStr(Now()));
Form2->StatusBar1->SimpleText = "Run started.";
Form2->Memo1->Lines->Add("---");

// Create run file name
memset(RunName,'\0',sizeof(RunName));
if (strstr(parinfile,".par")!=NULL + strstr(parinfile,".PAR")!=NULL)
     strncpy(RunName,parinfile,strlen(parinfile)-4);
     else
               strncpy(RunName,parinfile,strlen(parinfile));

// Parameter plausibility check
//if (ParameterCheck()==1) EndRun("Parameter error",1);

//Read files, check data, establish data structures
//newplot(ReadGrid);

// Create sumfile
memset(sumfile,'\0',sizeof(sumfile));
strncpy(sumfile,RunName,strlen(RunName));
strcat(sumfile,".sum");
sumfileheader(sumfile);

//---------------------------------------------------------------------------
RunNo = 0;

// Plot orientation loop
for (plotorient=0; plotorient<=1; plotorient++)
{
```

```
if (plotorient==1)
     {
     // Switch tree positions
     for (int tln=1; tln<=NTreeList; tln++)
               {
          newtreey = treex[tln];
          treex[tln] = ploty - treey[tln];
          treey[tln] = newtreey;
               }
     // Switch plot dimensions
     newploty = plotx;
     plotx = ploty;
     ploty = newploty;
     }

// Strip road system loops
for (striproad_start=0; striproad_start<=1; striproad_start++)
{
for (harvester_direction=0; harvester_direction<=1; harvester_direction++)
{
// Replicates loop
for (int n=1; n<=NoReplicates; n++)
{
RunNo++;

// Screen log
Form2->Memo1->Lines->Add("Run no. ");
Form2->Memo1->Lines->Add(IntToStr(RunNo));
Form2->StatusBar1->SimpleText = "Run started.";
Form2->Memo1->Lines->Add("---");

// Reset TreeClass
for (int n=1; n<=NTreeList; n++)
     {
     TreeClass[n] = 0;
     }

// Reset basal area sums
BA_striproads = 0;
BA_selected = 0;
BA_remaining = 0;
BA_total = 0;

// Calculate total basal area
BA_total = BasalArea(4);

// Write total basal area to log window
Form2->Memo1->Lines->Add("Total basal area:");
gcvt(BA_total,4,value);
StrCopy(copybuffer, value);
StrCat(copybuffer, " m2/ha");
Form2->Memo1->Lines->Add(copybuffer);
```

```
if (RunNo==542)
     RunNo=542;
StripRoads(n);
ThinningSelection();

 // Summary statistics
//  sumfilewrite(sumfile);
//---------------------------------------------------------------------------
//Output

// Summary statistics
sumfilewrite(sumfile, RunNo, n);

// Write files
if (MultipleRunFileOutput==1)
     {
     Form2->Memo1->Lines->Add("---");
     Form2->StatusBar1->SimpleText = "Writing result files.";
     Form2->Memo1->Lines->Add("Writing result files.");

     itoa(RunNo,Repstring,10);    // for output file names

     // create par file name
     memset(OutFile,'\0',sizeof(OutFile));
     strncpy(OutFile,RunName,strlen(RunName));
     strcat(OutFile,"-Out-");
     strcat(OutFile,Repstring);
     strcat(OutFile,".par");
     parfilewrite(OutFile);

     // create tls file name
     memset(OutFile,'\0',sizeof(OutFile));
     strncpy(OutFile,RunName,strlen(RunName));
     strcat(OutFile,"-Out-");
     strcat(OutFile,Repstring);
     strcat(OutFile,".tls");
     tlsfilewrite(OutFile);

     // create bmp file name
     memset(OutFile,'\0',sizeof(OutFile));
     strncpy(OutFile,RunName,strlen(RunName));
     strcat(OutFile,"-");
     strcat(OutFile,Repstring);
     strcat(OutFile,".bmp");
     Form2->Image1->Picture->SaveToFile(OutFile);
     }
//---------------------------------------------------------------------------
}    // End of Replicates loop
}    // End of strip road system loops
}    // End of strip road system loops
}    // End of plot orientation loop
```

```
}     // End of try loop
catch(BGRunError)
{
 // Log
 Form2->StatusBar1->SimpleText = "Run ended.";
 Form2->Memo1->Lines->Add("---");
 Form2->Memo1->Lines->Add("Ended due to error.");
 Form2->Memo1->Lines->Add(DateTimeToStr(Now()));
 Form2->Memo1->Lines->Add("--------------------");
}
//---------------------------------------------------------------------------
// End of run block (for normal and abnormal end of run)
//Delete data structures

//Screen log
Form2->StatusBar1->SimpleText = "Run ended.";
Form2->Memo1->Lines->Add("---");
Form2->Memo1->Lines->Add("ThinningSelector multiple runs ended.");
Form2->Memo1->Lines->Add(DateTimeToStr(Now()));
Form2->Memo1->Lines->Add("--------------------");
//---------------------------------------------------------------------------
// Create logfile
memset(OutFile,'\0',sizeof(OutFile));
strncpy(OutFile,RunName,strlen(RunName));
strcat(OutFile,".log");
Form2->Memo1->Lines->SaveToFile(OutFile);
//---------------------------------------------------------------------------

return;
}
//---------------------------------------------------------------------------
void EndRun(char *message, bool endloop)
{
Form2->Memo1->Lines->Add("---");
Form2->Memo1->Lines->Add(message);

Application->MessageBox(message,"Run error", MB_OK);

if (endloop)
     throw BGRE;
}
//---------------------------------------------------------------------------
double BasalArea(int treeclass)
{
extern int NTreeList;
extern double plotx, ploty;
extern double *treedbh;
extern double *treex, *treey;
extern int *TreeClass;
extern int Circular_plot;

double plotarea;
```

```
double circleradius;
int treeinplot;
double basum = 0;

plotarea =  plotx * ploty / 10000;
if (Circular_plot==1)
     {
     circleradius = plotx / 2;
     plotarea =  pow(circleradius,2) * 3.141592654 / 10000;
     }

for (int n=1; n<=NTreeList; n++)
 if(TreeClass[n]==treeclass || treeclass==4)
     {
     treeinplot = 1;
     if (Circular_plot==1)
          if (sqrt(pow(circleradius-treex[n],2) + pow(circleradius-treey[n],2)) > circleradius)
               treeinplot = 0;
     if (treeinplot==1)
          basum += pow(treedbh[n]/200,2) * 3.141592654 / plotarea;
     }
 return basum;
}
//---------------------------------------------------------------------------
double NoCroptrees()
{
 extern int NTreeList;
 extern double plotx, ploty;
 extern double *treex, *treey;
 extern int *TreeClass;
 extern int Circular_plot;

 double plotarea;
 double circleradius;
 int treeinplot;
 double nocts = 0;

 plotarea =  plotx * ploty / 10000;
 if (Circular_plot==1)
     {
     circleradius = plotx / 2;
     plotarea =  pow(circleradius,2) * 3.141592654 / 10000;
     }

 for (int n=1; n<=NTreeList; n++)
  if(TreeClass[n]==1)
     {
     treeinplot = 1;
     if (Circular_plot==1)
          if (sqrt(pow(circleradius-treex[n],2) + pow(circleradius-treey[n],2)) > circleradius)
               treeinplot = 0;
     if (treeinplot==1)
```

```
            nocts += 1 / plotarea;
        }
 return nocts;
}
//---------------------------------------------------------------------------
```

```
// Program ThinningSelector
/*
Andreas Brunner, andreas.brunner@umb.no
*/
//---------------------------------------------------------------------------
#include <stdlib.h>
#include <math.h>

#include "ThinningSelectorMain.h"
#include "ThinningSelectorFunc.h"
//---------------------------------------------------------------------------
 extern double plotx, ploty;
 // Tree list
 extern int NTreeList;
 extern int *treenum;
 extern int *TreeSpecies;
 extern int *TreeClass;
 extern int *TreeClassOld;
 extern int *TreeMarked;
 extern double *treex, *treey;
 extern double *treedbh;

 extern double h_x[1000], h_y[1000];
 extern int h_dir[1000];
 extern int h_positionN;  // No of harvester positions

 extern double BA_striproads, BA_selected, BA_remaining, BA_total;
 extern double matchpct1, matchpct2, matchpct3;
 extern double CT_selected;
 extern int Nrealthin;
 extern int Nsimthin;
 extern int Nnotstrip;
 extern int N_thin_below;

 // Parameters
 extern int striproad_start;
 extern int harvester_direction;

 extern double striproad_width;
 extern double harvester_range;
 extern double stop_distance;
 extern double workzone_y_backward;
 extern int RandomHarvStart;

 extern int Circular_plot;
 extern int Circular_plot_expansion;

 // Variables
 double workzone_x; // width of working area
 double ct_meandist;      // target
 double ct_min_dbh;     // minimum dbh for crop trees
 int CT_list[1000];      // list of selected crop trees
```

```
int CT_comp_no[1000][31];  //  competitor list per crop tree
double CT_comp_CI[1000][31];  // competition index per competitor

//---------------------------------------------------------------------------
void StripRoads(int h_start_rep)
{
 //Lay out strip road system and harvest trees on strips roads

 extern int NoReplicates;

 double harvester_x, harvester_y; // harvester positions
 double harv_y;
 int harvester_dir;
 double striproad_distance;
 int striproad_number;

 char copybuffer[50];
 char value[10];

 // Screen log
 Form2->Memo1->Lines->Add("---");
 Form2->StatusBar1->SimpleText = "Calculating strip roads.";
 Form2->Memo1->Lines->Add("Calculating strip roads.");

 workzone_x = sqrt(pow(harvester_range,2) - pow(stop_distance,2)); // Width of the working area

 striproad_distance = 2 * workzone_x; // distance between the centre of two strip roads

 striproad_number = abceil(plotx / striproad_distance); // total number of strip roads on this plot

 if (harvester_direction == 1)
      harvester_dir = 0;
      else
      harvester_dir = 1;

 // Reset harvester position list
 for (int n=0; n<1000; n++)
                    {
                     h_x[n] = 0;
                     h_y[n] = 0;
       h_dir[n] = 0;
                    }
 h_positionN = 0;

 // Loop over harvester positions
 for (int striproad_no=1; striproad_no<=striproad_number; striproad_no++)
  {
       harvester_x = (striproad_no - 1) * striproad_distance + striproad_distance / 2;
       if (striproad_start==1)
            harvester_x = plotx - harvester_x;

       // Start position of harvester in strip road
```

```
      harvester_y = 1;      // default: single runs without random start position
      if (h_start_rep==0 && RandomHarvStart==1) // single runs with random start position
           harvester_y = workzone_y_backward - stop_distance * random(101) / 100.0;
      if (h_start_rep > 0)   // multiple runs: systematic variation of start positions
           harvester_y = workzone_y_backward - stop_distance + (h_start_rep * stop_distance /
NoReplicates);

      if (harvester_dir == 0)
           harvester_dir = 1;
           else
           harvester_dir = 0;

      // Remove all trees on strip road
       for (int n=1; n<=NTreeList; n++)
                  {
                        if (treex[n]>= harvester_x - striproad_width/2 &&
             treex[n]<= harvester_x + striproad_width/2)
                TreeClass[n] = 3;
                  }
      for (; harvester_y<=ploty; harvester_y+=stop_distance)
       {
       if (harvester_dir == 0)
           harv_y = ploty - harvester_y;
           else
           harv_y = harvester_y;

        // Create harvester position list
        h_positionN++;
        h_x[h_positionN] = harvester_x;
        h_y[h_positionN] = harv_y;
        h_dir[h_positionN] = harvester_dir;

        // Graph update
        ResultGraph();

      } // end of harvester_y loop
 } // end of striproad_no loop

 // Calculate strip road basal area
 BA_striproads = BasalArea(3);

 // Write strip road basal area to log window
 Form2->Memo1->Lines->Add("Basal area removed in strip roads:");
 gcvt(BA_striproads,4,value);
 StrCopy(copybuffer, value);
 StrCat(copybuffer, " m2/ha");
 Form2->Memo1->Lines->Add(copybuffer);

 Form2->StatusBar1->SimpleText = "Done.";
 return;
}
//---------------------------------------------------------------------------
```

```
void ThinningSelection()
{
 // Parameters
 extern int croptree_number_ha; // number of crop trees / ha
 extern double croptree_mindist_par; // Minimum allowed distance between two crop trees (0 - 1)
 extern int croptree_species; // species for crop trees: 0 = all, 1 ... numbers related to species list in
parameter file
 extern double croptree_min_dbh_diff;  // (cm) min. dbh-difference visually detectable by harvester
driver
 extern double croptree_dbh_percentile; // (%) Pecentile in dbh distribution that will be used as
minimum dbh for crop trees
 extern double BA_target; // Target basal area (m2/ha)

 // Variables
 double wz_miny, wz_maxy, wz_minx, wz_maxx;     // work zone borders
 double wz_area;       // work zone area (m2)
 int wztrees[1000];     // list of trees inside work zone
 int Nwztrees;  // number of trees in work zone

 int ct_cand_num_wz;      // target
 double ct_num_wz;      // target
 double ct_mindist;      // target
 int ct_sel_num_wz;  // counter for number crop trees selected in work zone
 int SortedTreeList[1000];
 int sortedlistlength;
 double MaxDbh;
 int MaxDbhNo;
 int percentile_n;

 double maxwzdbh;       // max. dbh of crop tree candidates in work zone
 int ct_cand_group_size; // Number of trees crop tree candidates in work zone with visually identical
dbh
 int closest_no; // internal number of closest tree
 double closest_dist;   // distance (m) of closest tree
 double dist;   // distance (m) of crop tree candidate to harvester
 int treelistpos; // posistion in tree list
 int leftinlist; // no. of trees left in sorted list

 double BA_wz;  // remaining basal area in work zone (m2/ha)
 double tree_ba;       // individual tree basal area (m2/ha)

 int ncomp;
 double maxci;
 int maxcino;
 int complist[100];
 double  compci[100];
 int sortedcomplist[100];
 int targetmet;
 double nctwz;
 double dx, dx2, dy, midypos, hx;    // variables for calculation of work zone area for circular plots
 int wzdir;
```

```
char copybuffer[50];
char value[10];

// Screen log
Form2->Memo1->Lines->Add("---");
Form2->StatusBar1->SimpleText = "Selecting thinning trees.";
Form2->Memo1->Lines->Add("Selecting thinning trees.");

N_thin_below = 0;

// Find percentile dbh per plot as minimum dbh for crop trees
// Sort plot tree list by decreasing dbh
sortedlistlength = 0;
MaxDbh = 9999;
percentile_n = abceil(NTreeList * (1 - croptree_dbh_percentile/100));
for (int l=1; l<=NTreeList; l++)
    TreeMarked[l] = 0;

while (MaxDbh>0 && sortedlistlength < percentile_n)
    {
    MaxDbh = 0;
    for (int l=1; l<=NTreeList; l++)
        {
        if (treedbh[l]>MaxDbh && TreeMarked[l]==0)
            {
            MaxDbh = treedbh[l];
            MaxDbhNo = l;
            }
        }
        if (MaxDbh>0)
            {
            TreeMarked[MaxDbhNo] = 1;
            sortedlistlength++;
            if (sortedlistlength == percentile_n)
                ct_min_dbh = treedbh[MaxDbhNo];
            }
    }

// Reset crop tree competitor list
for (int l=1; l<=999; l++)
    {
    CT_list[l] = 0;
    for (int l1=1; l1<=30; l1++)
        {
        CT_comp_no[l][l1] = 0;
        CT_comp_CI[l][l1] = 0;
        }
    }

// Loop over all harvester positions / work zones
for (int n=1; n<=h_positionN; n++)
    {
```

```
// Left (0) & Right (1) work zones
for (int workzone_dir=0; workzone_dir<=1; workzone_dir++)
      {
      // Define working area
      if (h_dir[n]==1)
            {
            wz_miny = h_y[n] - workzone_y_backward;
            wz_maxy = h_y[n] + stop_distance;
            }
            else
            {
            wz_miny = h_y[n] - stop_distance;
            wz_maxy = h_y[n] + workzone_y_backward;
            }

      if (h_dir[n]==1 && workzone_dir ==1)
            {
            wz_minx = h_x[n] - workzone_x;
            wz_maxx = h_x[n];
            }
      if (h_dir[n]==1 && workzone_dir ==0)
            {
            wz_minx = h_x[n];
            wz_maxx = h_x[n] + workzone_x;
            }
      if (h_dir[n]==0 && workzone_dir ==0)
            {
            wz_minx = h_x[n] - workzone_x;
            wz_maxx = h_x[n];
            }
      if (h_dir[n]==0 && workzone_dir ==1)
            {
            wz_minx = h_x[n];
            wz_maxx = h_x[n] + workzone_x;
            }

      if (wz_miny < 0) wz_miny = 0;
      if (wz_minx < 0) wz_minx = 0;
      if (wz_maxy > ploty) wz_maxy = ploty;
      if (wz_maxx > plotx) wz_maxx = plotx;
      if (wz_minx > plotx) wz_minx = plotx;  // only for last strip road outside plot
      if (wz_maxx < 0) wz_maxx = 0;  // only for last strip road outside plot

      wz_area = (wz_maxx - wz_minx) * (wz_maxy - wz_miny);

      // Work zones as part of circle for circular plots without expansion of plot into corners
      if (Circular_plot==1 && Circular_plot_expansion==0)
            {
            wzdir = workzone_dir;
            hx = h_x[n];
            // For strip roads outside plot
            if (hx > plotx)
```

```
                    {
                    hx = hx - workzone_x;
                    if (workzone_dir==1)    wzdir = 0;
                    if (workzone_dir==0)    wzdir = 1;
                    }
               if (hx < 0)
                    {
                    hx = hx + workzone_x;
                    if (workzone_dir==1)    wzdir = 0;
                    if (workzone_dir==0)    wzdir = 1;
                    }

               // Find center y of complete work zone
               if (h_dir[n]==1)
                    {
                    midypos = h_y[n] - workzone_y_backward + (stop_distance +
workzone_y_backward)/2;
                    if (h_y[n] - workzone_y_backward < 0)
                         midypos = ((stop_distance + workzone_y_backward) + (h_y[n] -
workzone_y_backward))/2;
                    if (h_y[n] + stop_distance > ploty)
                         midypos = ploty - (ploty - h_y[n] + workzone_y_backward)/2;
                    }
                    else
                    {
                    midypos = h_y[n] + workzone_y_backward - (stop_distance +
workzone_y_backward)/2;
                    if (h_y[n] - stop_distance < 0)
                         midypos = ((stop_distance + workzone_y_backward) + (h_y[n] -
stop_distance))/2;
                    if (h_y[n] + workzone_y_backward > ploty)
                         midypos = ploty - (ploty - h_y[n] + stop_distance)/2;
                    }

               // Find x distance to circle for given y
               dy = fabs(midypos - ploty/2);
               dx = pow((pow((plotx/2),2) - pow(dy,2)),0.5);

               // Work zone size in x-direction depending on position of strip road and work zone
direction
               if (hx < plotx/2)
                    {
                    if ((h_dir[n]==1 && wzdir==1) || (h_dir[n]==0 && wzdir==0))
                         dx2 = dx - (plotx/2 - hx);
                         else
                         dx2 = dx + (plotx/2 - hx);
                    }
                    else
                    {
                    if ((h_dir[n]==1 && wzdir==1) || (h_dir[n]==0 && wzdir==0))
                         dx2 = dx + (hx - plotx/2);
                         else
```

```
                    dx2 = dx - (hx - plotx/2);
                }
            if (dx2 < 0)    dx2 = 0;

            wz_area = dx2 * (wz_maxy - wz_miny);
            }

        // Create list of trees inside work zone
        for (int k=0; k<1000; k++)
            wztrees[k] = 0;
        Nwztrees = 0;
        for (int l=1; l<=NTreeList; l++)
         if (   (treex[l]>=wz_minx || fabs(treex[l]-wz_minx)<0.00001) &&
            (treex[l]<=wz_maxx || fabs(treex[l]-wz_maxx)<0.00001) &&
            (treey[l]>=wz_miny || fabs(treey[l]-wz_miny)<0.00001) &&
            (treey[l]<=wz_maxy || fabs(treey[l]-wz_maxy)<0.00001)
            )
            {
            Nwztrees++;
            wztrees[Nwztrees] = l;
            }

        /** Find crop trees **/

        // Define target variables
        if (croptree_number_ha > 0)
            ct_meandist = sqrt(10000.0 / (croptree_number_ha)); // Average distance between two
crop trees
            else
            ct_meandist = 100;
        ct_mindist = croptree_mindist_par * ct_meandist;       // Minimum allowed distance
between two crop trees
        nctwz = wz_area / 10000 * croptree_number_ha;     // Number of crop trees candidates in
the work zone
        if (nctwz > 0.5)                        // 0.5 is just a first guess. With 0.5 this can be all
condensed to 1 line.
//                ct_cand_num_wz = abceil(nctwz);       // earlier version
            ct_cand_num_wz = round(nctwz);
            else
            ct_cand_num_wz = 0;     // no crop trees in small work areas (sections of circular plots,
works zones for strip roads outside plot, work zones at start/end of strip road)

        // Reset counter for number crop trees selected in work zone
        ct_sel_num_wz = 0;

        // Count number of existing crop trees in work zone
        for (int l=1; l<=Nwztrees; l++)
         if (TreeClass[wztrees[l]] == 1)
                ct_sel_num_wz++;

        // Sort work zone tree list by decreasing dbh
        // Reset sorted list
```

```
        for (int no=1; no<=999; no++)
              SortedTreeList[no] = 0;
        sortedlistlength = 0;
        MaxDbh = 9999;
        for (int l=1; l<=NTreeList; l++)
              TreeMarked[l] = 0;

        while (MaxDbh>ct_min_dbh && MaxDbh>0)
            {
            MaxDbh = 0;
            for (int l=1; l<=Nwztrees; l++)
            {
            if (
              (TreeSpecies[wztrees[l]]==croptree_species || croptree_species==0) &&   // correct
species
              TreeClass[wztrees[l]]==0 &&                    // not a crop tree or thinned
              treedbh[wztrees[l]]>ct_min_dbh &&                  // larger than minimum dbh for
crop trees
              TreeMarked[wztrees[l]]==0 &&                   // not already listed
              treedbh[wztrees[l]]>MaxDbh                    // the largest
              )
                 {
                 MaxDbh = treedbh[wztrees[l]];
                 MaxDbhNo = wztrees[l];
                 }
            }
            if (MaxDbh>0)
             {
             TreeMarked[MaxDbhNo] = 1;
             sortedlistlength++;
             SortedTreeList[sortedlistlength] = MaxDbhNo;
             }
            }

        // Select crop trees
        if (sortedlistlength > 0)
              {
              leftinlist = sortedlistlength;
              while (ct_sel_num_wz < ct_cand_num_wz && leftinlist > 0)
              {
              // Find group of largest trees
              maxwzdbh = 0;
              for (int l=1; l<=sortedlistlength; l++)
               if (SortedTreeList[l] != 0)
                    {
                    if (maxwzdbh < 0.00001)
                         {
                         maxwzdbh = treedbh[SortedTreeList[l]];
                         ct_cand_group_size = l;
                         }
                         else
                         {
```

```
                    if (treedbh[SortedTreeList[l]] > maxwzdbh - croptree_min_dbh_diff)
                        ct_cand_group_size = l;
                    }
                }
            // Find tree closest to harvester
            closest_no = 0;
            closest_dist = 9999;
            for (int l=1; l<=ct_cand_group_size; l++)
             if (SortedTreeList[l] != 0)
                {
                dist = sqrt(pow(h_x[n]-treex[SortedTreeList[l]],2) + pow(h_y[n]-
treey[SortedTreeList[l]],2));
                if (dist < closest_dist)
                    {
                    closest_dist = dist;
                    closest_no = l;
                    }
                }
            // Check for distance to already existing crop trees and accept/discard crop tree
            if (closest_no != 0 && ct_sel_num_wz < ct_cand_num_wz)
                {
                treelistpos = SortedTreeList[closest_no];
                SortedTreeList[closest_no] = 0;
                leftinlist--;
                TreeClass[treelistpos] = 1;
                ct_sel_num_wz++;

                for (int l=1; l<=NTreeList; l++)
                 if (TreeClass[l]==1 &&
                    l != treelistpos &&
                    sqrt(pow(treex[l]-treex[treelistpos],2) + pow(treey[l]-treey[treelistpos],2)) <
ct_mindist
                    )
                    {
                    TreeClass[treelistpos] = 0;
                    ct_sel_num_wz--;
                    }

                 if (TreeClass[treelistpos]==1)
                    CompetitorList(treelistpos);    // sorting, listing competitors and removing
strongest competitor
                 }
            } // end of while loop
            } // end of sortedlistlength condition

        /** Select thinning trees **/
        // Calculate remaining basal area in work zone (after strip road trees and strongest
competitor per crop tree are removed)
        BA_wz = 0;
        for (int l=1; l<=Nwztrees; l++)
            {
            if (TreeClass[wztrees[l]] < 2 && wz_area > 0)
```

```
                BA_wz += pow(treedbh[wztrees[l]]/200,2) * 3.141592654 / (wz_area/10000);
            }

        if (BA_wz > BA_target)
        {
        // Find all reamining competitors in this work zone for selected crop trees
        ncomp = 0;
        for (int l=1; l<=Nwztrees; l++)
         if(TreeClass[wztrees[l]]==0)
            {
            for (int l1=1; l1<=999; l1++)
             if (CT_list[l1]>0)
                {
                for (int l2=1; l2<=30; l2++)
                 if (CT_comp_no[l1][l2]==wztrees[l])
                    {
                    ncomp++;
                    complist[ncomp] = CT_comp_no[l1][l2];
                    compci[ncomp] = CT_comp_CI[l1][l2];
                    }
                }
            }

        //  Sort competitors after decreasing CI
        for (int l=1; l<=NTreeList; l++)
            TreeMarked[l] = 0;

        for (int l0=1; l0<=ncomp; l0++)
            {
            maxci = 0;
            for (int l=1; l<=ncomp; l++)
                {
                if (TreeMarked[complist[l]]==0 &&
                    compci[l] > maxci)
                    {
                    maxci = compci[l];
                    maxcino = complist[l];
                    }
                }
             sortedcomplist[l0] = maxcino;
             TreeMarked[maxcino] = 1;
             }

        // Remove competitors until target basal area is met for this work zonne
        targetmet = 0;
        for (int l0=1; l0<=ncomp; l0++)
         if (targetmet==0)
            {
            if (wz_area > 0)
                tree_ba = pow(treedbh[sortedcomplist[l0]]/200,2) * 3.141592654 /
(wz_area/10000);
                else
```

```
                tree_ba = 0;
            if (fabs(BA_target - (BA_wz - tree_ba)) < fabs(BA_target - BA_wz))
                {
                TreeClass[sortedcomplist[l0]] = 2;
                BA_wz -= tree_ba;
                }
                else
                targetmet = 1;
            }

        // Remove additional trees as thinning from below after all competitors are removed and
basal area still is above target
        if (BA_target < BA_wz && targetmet == 0)
            {
            // Find remaining trees in work zone
             ncomp = 0;
             for (int l=1; l<=Nwztrees; l++)
              if(TreeClass[wztrees[l]]==0)
                {
                ncomp++;
                complist[ncomp] = wztrees[l];
                }

            // Sort list of remaining trees in work zone by decreasing dbh
            for (int l=1; l<=NTreeList; l++)
                TreeMarked[l] = 0;

            for (int l0=1; l0<=ncomp; l0++)
                {
                MaxDbh = 999;
                for (int l=1; l<=ncomp; l++)
                    {
                    if (TreeMarked[complist[l]]==0 &&
                        treedbh[complist[l]] < MaxDbh)
                        {
                        MaxDbh = treedbh[complist[l]];
                        MaxDbhNo = complist[l];
                        }
                    }
                 sortedcomplist[l0] = MaxDbhNo;
                 TreeMarked[MaxDbhNo] = 1;
                 }

            // Remove trees starting from lowest dbh until target basal area is met
             targetmet = 0;
             for (int l0=1; l0<=ncomp; l0++)
              if (targetmet==0)
                {
                if (wz_area > 0)
                    tree_ba = pow(treedbh[sortedcomplist[l0]]/200,2) * 3.141592654 /
(wz_area/10000);
                    else
```

```
                        tree_ba = 0;
                    if (fabs(BA_target - (BA_wz - tree_ba)) < fabs(BA_target - BA_wz))
                        {
                        TreeClass[sortedcomplist[l0]] = 2;
                        BA_wz -= tree_ba;
                        // Count trees thinned from below, but only in circle, if circular plot
                        if (Circular_plot==0 || (Circular_plot==1 &&
sqrt(pow(treex[sortedcomplist[l0]]-plotx/2,2) + pow(treey[sortedcomplist[l0]]-ploty/2,2)) <=
plotx/2))
                            N_thin_below++;
                        }
                        else
                        targetmet = 1;
                    }

                }   // end of thinning from below loop

            }   // end of if for ba_wz > ba_target


        }   // end of left right work zone loop

        // Graph update
        ResultGraph();

    }   // End of harvester position loop

// Calculate basal area for trees selected for thinning outside strip roads and count number of
selected crop trees
BA_selected = BasalArea(2);
CT_selected = NoCroptrees();

// Write number of selected crop trees to log window
Form2->Memo1->Lines->Add("Number of selected crop trees:");
gcvt(CT_selected,4,value);
StrCopy(copybuffer, value);
StrCat(copybuffer, " /ha");
Form2->Memo1->Lines->Add(copybuffer);

// Write selected basal area to log window
Form2->Memo1->Lines->Add("Basal area selected for thinning:");
gcvt(BA_selected,4,value);
StrCopy(copybuffer, value);
StrCat(copybuffer, " m2/ha");
Form2->Memo1->Lines->Add(copybuffer);

// Write remaining basal area to log window
BA_remaining = BA_total - BA_striproads - BA_selected;
Form2->Memo1->Lines->Add("---");
Form2->Memo1->Lines->Add("Basal area remaining:");
gcvt(BA_remaining,4,value);
StrCopy(copybuffer, value);
```

```
 StrCat(copybuffer, " m2/ha");
 Form2->Memo1->Lines->Add(copybuffer);

 // Calculate statistics for thinnnig tree selection vs. old selections
 int match1 = 0;
 int match2 = 0;
 int match3 = 0;
 Nnotstrip = 0;
 for (int n=1; n<=NTreeList; n++)
  if (Circular_plot==0 || (Circular_plot==1 && sqrt(pow(treex[n]-plotx/2,2) + pow(treey[n]-ploty/2,2))
<= plotx/2))
      {
      if (TreeClass[n] == 2 && TreeClassOld[n] == 2)  match1++;
      if (TreeClass[n] == 2 && (TreeClassOld[n] == 0 || TreeClassOld[n] == 1))  match2++;
      if ((TreeClass[n] == 0 || TreeClass[n] == 1) && TreeClassOld[n] == 2)  match3++;
      if (TreeClass[n] != 3) Nnotstrip++;
      }
 Nrealthin = match1 + match3;
 matchpct1 = double(match1) / Nrealthin * 100;
 matchpct3 = double(match3) / Nrealthin * 100;
 Nsimthin = match1 + match2;
 matchpct2 = double(match2) / Nsimthin * 100;

 // Write remaining basal area to log window
 Form2->Memo1->Lines->Add("---");
 Form2->Memo1->Lines->Add("Match of selected thinning trees:");
 StrCopy(copybuffer, "Thinned real only: ");
 gcvt(matchpct3,3,value);
 StrCat(copybuffer, value);
 StrCat(copybuffer, " % of real");
 Form2->Memo1->Lines->Add(copybuffer);
 StrCopy(copybuffer, "Thinned real and simulated: ");
 gcvt(matchpct1,3,value);
 StrCat(copybuffer, value);
 StrCat(copybuffer, " % of real");
 Form2->Memo1->Lines->Add(copybuffer);
 StrCopy(copybuffer, "Thinned simulated only: ");
 gcvt(matchpct2,3,value);
 StrCat(copybuffer, value);
 StrCat(copybuffer, " % of simulated");
 Form2->Memo1->Lines->Add(copybuffer);

 Form2->StatusBar1->SimpleText = "Done.";
 return;
}
//---------------------------------------------------------------------------
void CompetitorList(int ct_no)
{
double dist;
double ci;
int ncomp = 0;
double maxci;
```

```
int maxcino;
int emptyfound;

int complist[100];
double  compci[100];
int sortedcomplist[100];

 /* Find all competitors */
 for (int l=1; l<=NTreeList; l++)
  if (  TreeClass[l]==0 &&
       sqrt(pow(treex[l]-treex[ct_no],2) + pow(treey[l]-treey[ct_no],2)) < ct_meandist*0.55
//      &&  treedbh[l] > ct_min_dbh / 2.0            // accept only competitors that have a minimum dbh
       )
       {
       dist = sqrt(pow(treex[l]-treex[ct_no],2) + pow(treey[l]-treey[ct_no],2));
       ci = 2 * atan((treedbh[l] / 200) / dist) / 3.141592654 * 180; // Angle CI

       ncomp++;
       complist[ncomp] = l;
       compci[ncomp] = ci;
       }

 /* Sort competitors by decreasing strength of competition */
 for (int l=1; l<=NTreeList; l++)
       TreeMarked[l] = 0;

 for (int l0=1; l0<=ncomp; l0++)
       {
       maxci = 0;
       for (int l=1; l<=ncomp; l++)
             {
             if (TreeMarked[complist[l]]==0 &&
                   compci[l] > maxci)
                   {
                   maxci = compci[l];
                   maxcino = complist[l];
                   }
             }
        sortedcomplist[l0] = maxcino;
        TreeMarked[maxcino] = 1;
        }

 /* Remove strongest competitor  */
 if (ncomp>0)
       TreeClass[sortedcomplist[1]] = 2;

 /* Create entry in permanent list of remaining competitors */
 emptyfound = 0;
 for (int l=1; l<=999; l++)
  if (CT_list[l]==0 && emptyfound==0)
       {
       emptyfound = 1;
```

```
      CT_list[l] = ct_no;
      for (int l0=1; l0<=ncomp-1; l0++)
            {
            CT_comp_no[l][l0] = sortedcomplist[l0+1];
            for (int l1=1; l1<=ncomp; l1++)
             if (complist[l1]==sortedcomplist[l0+1])
                  CT_comp_CI[l][l0] = compci[l1];
            }
      }

 return;
}
//---------------------------------------------------------------------------
/* Expand circular plots by filling corner with trees */
void Expand_circular_plot()
{
double circlearea, plotarea;
double ba_circle = 0;
double ba_total = 0;
int treeno = 0;
int listcounter;
double x, y;
int posfound;
int tooclose;
double mindist;

// Calculate plot level N and G, current and target
circlearea =  pow(plotx / 2,2) * 3.141592654 / 10000;
plotarea =  pow(plotx,2) / 10000;

for (int n=1; n<=NTreeList; n++)
      ba_circle += pow(treedbh[n]/200,2) * 3.141592654 / circlearea;
ba_total = ba_circle / plotarea * circlearea;

mindist = sqrt((circlearea / NTreeList) * 10000) * 0.5;

// Randomly draw trees from tree list until N and G reached for entire plot
// This random selection is proportional to tree number and not size, would be better PPS
listcounter = NTreeList;
while (ba_total < ba_circle && listcounter <= round(NTreeList*1.5))
      {
      treeno = random(NTreeList)+1;
      listcounter++;

      treenum[listcounter] = treenum[treeno] + 1000;
      TreeSpecies[listcounter] = TreeSpecies[treeno];
      treedbh[listcounter] = treedbh[treeno];
      TreeClass[listcounter] = 0;

      ba_total += pow(treedbh[listcounter]/200,2) * 3.141592654 / plotarea;

      // Find random position, check if outside circle, check for min. distance to exsiting trees
```

```
    posfound = 0;
    while (posfound==0)
        {
        x = random(1001) / 1000.0 * plotx;
        y = random(1001) / 1000.0 * ploty;

        if (sqrt(pow(x-plotx/2,2) + pow(y-ploty/2,2)) > plotx/2)
            {
            tooclose = 0;
            for (int n=1; n<=listcounter-1; n++)
                if (sqrt(pow(x-treex[n],2) + pow(y-treey[n],2)) < mindist)
                    tooclose = 1;
            if (tooclose == 0)
                {
                posfound = 1;
                treex[listcounter] = x;
                treey[listcounter] = y;
                }
            }
        }
    }

NTreeList = listcounter;

return;
}
//---------------------------------------------------------------------------
```