Filtrene

Programmet giver som sagt kort så tilfældigt som det er muligt. Efter kortgivning sendes fordelingen igennem et 'filter' som undersøger om fordelingen passer med de kriterier brugeren har checket af. Nedenstående forklarer hvordan disse filtre er konstrueret. Du vil se, at jeg i vidt omfang har benyttet mig af de bøger som findes i 'bogskabet'.
Det er klart, at du ikke skal behøve at være C# - expert for at kunne læse disse sider. Rolig: Det behøver du heller ikke at være. Men jeg mener, at kildekoden suppleret med lidt forklaring nok alligevel er den mest koncise måde hvorpå jeg kan fortælle – også til en der ikke er kendt med C# - hvad der egentlig foregår. Kildekoden er stort set uden avancerede konstruktioner: Dette handler om bridge, ikke om programmering. Men en *lille* indledning forlods til de dele er dog på sin plads:

Symbol	Betydning
`Return x`	Funktionen returnerer 'x'
`&&`	'Og' i betydningen 'både og'. Returnerer sand dersom det der står til højre så vel som det der står til venstre for '&&' er sandt
`||`	'Eller' i betydningen 'og/eller'. Returnerer sand dersom det der står til venstre og/eller det der står til venstre for '||' er sandt
`!`	'Not'. Returnerer sand dersom det der står til højre for '!' *ikke* er sandt
`Count`	'Antallet af'. Fx betyder 'hearts.Count' antallet af hjertere på hånden.
`= =`	De TO lighedstegn lige efter hinanden angiver et 'udsagn' dvs en påstand om at det der står på de to sider af '= =' er ens. Denne påstand kan så være sand eller falsk. Dersom man kun sætter ét lighedstegn er der derimod tale om en 'tilegnelse': I så fald giver man venstresiden den værdi som står til højre
`!=`	Dette symbol forholder sig til '!' som '= =' forholder sig til '='. Dvs. '!=' er en påstand om, at det der står på de to sider af symbolet IKKE er ens

Kildekoden benytter sig af kald til en række funktioner som jeg har lavet. Jeg har bestræbt mig på at gøre navnene på disse så selvforklarende som muligt.
Du vil bemærke, at jeg nogle steder bruger engelske betegnelser, fx 'diamonds' og 'clubs' i stedet for ruder og klør. Dette er ikke kun for at være krukket: Æ, Ø og Å kan give problemer, så jeg undgår at bruge det i kildekode.

Åbning 1 i major

Herunder den afgørende programstump:

```
/* Nedenstående er koordineret med kravene vedr 'weak_2_opener' */
return hearts.Count > spades.Count && hearts.Count > 4 &&
    hearts.Count >= diamonds.Count && hearts.Count >= clubs.Count &&
    (hp_interval(12, 21) ||
    (hCP == 10 && hearts.Count > 5 && lTC < 7) ||
    (hCP == 11 && hearts.Count > 5 && lTC < 8));
```

For at en hånd skal kunne passere igennem dette filter skal der være flere hjertere end spar (hearts.Count > spades.Count), der skal være flere end 4 hjertere (dvs. vi åbner ikke med 4-farve i hjerter – programmet er forberedt til at dette kan ændres, men det fremgår ikke af denne kodestump) og der må ikke være en længere minorfarve.
Derudover er der nogle krav til håndens honnørstyrke: Den skal i udgangspunktet ligge mellem 12 og 21 (begge inklusive) idet dog 10 og 11 Honnørpoints kan godtages dersom der er 6-farve og lTC (Losing Trick Count, dvs antallet af 'tabere' er lavt nok. Retningslinierne for dette stammer fra en artikel af Hulgård – en artikel som jeg nu desværre har forlagt og som ikke bare lige kan findes igen; jeg kan imidlertid se, at de passer godt med det Blakset og Lars Lund Madsen skriver i 'Meld Bedre', se boglisten). Hvad angår tabskortberegningen, så henviser jeg til Ron Klinger: 'The modern losing trick count'.

Filtrene

Åbning 1 i major overfor 4+ kortstøtte

Dette er fuldstændigt som for 'åbning 1 i major' med det yderligere krav, at spillet kun slipper gennem filteret dersom svarhånden har 4 kort med i farven. Der er ingen honnørkrav til svarer.

Åbning 1 i minor

Ja, den er lidt mere indviklet end åbning 1 i major. Du får kildekoden til hele proceduren:

```
public bool one_minor_opener_(int lower_1NTlimit, int higher_1NTlimit)
{
    bool hpKrav = hp_interval(12, lower_1NTlimit - 1) || hp_interval(1 + higher_1NTlimit, 21) ||
                  (hp_interval(lower_1NTlimit, higher_1NTlimit) && !nT_fordeling);
    bool fordelingsKrav = (clubs.Count == 5 && spades.Count == 5 && hCP < 13 &&
                           hCP_(Denomination.clubs) > hCP_(Denomination.spades)) ||
                          (longest_suit == Denomination.clubs || longest_suit == Denomination.diamonds) ||
                          (spades.Count < 5 && hearts.Count < 5);
    return hpKrav && fordelingsKrav;
}
```

Oversigtsmæssigt kan man se, at en proceduren returnerer '`hpKrav && fordelingskrav`' dvs en fordeling slipper kun igennem dersom såvel et krav til honnørpoints ('hpKrav') som også et krav til fordelingen 'fordelingsKrav') er opfyldt.
hpKrav er opfyldt i 3 forskellige scenarier: Man er over kravet til en åbningshånd og under den nedre grænse for en UT – åbning, eller man er over den øvre grænse for en UT – åbning men under en kravåbning, eller man er i intervallet for en UT – åbning, men der er ikke UT – fordeling.
Fordelingskravet er ligeledes 3 – delt, og det er lidt kompliceret:
Den første mulighed: `clubs.Count == 5 && spades.Count == 5 && hCP < 13 && hCP_(Denomination.clubs) > hCP_(Denomination.spades)` dækker den situation, at der er 5 – 5 i sort og en relativt svag hånd med en svag sparfarve. Det er normalt at åbne 5 – farver 'ovenfra' men i lige netop den situation anbefaler Eddie Kantar (Take All your Chances bind 2 spil 68) at man åbner med 1♣. Han argumenterer: 'Hvis du åbner 1♠ og makker melder 2♦ eller 2♥, så bliver man nødt til at genmelde 2♠ med mindre man har aftalt at 3♣ ikke lover ekstra'. Det med '`hCP_(Denomination.clubs)`' angiver antallet af honnørpoints i klørfarven.
Næste mulighed ' `longest_suit == Denomination.clubs || longest_suit == Denomination.diamonds)`' angiver, at håndens længste farve er en minorfarve. Det skal her bemærkes, at 'longest_suit' udregnes således, at dersom 2 farver er lige lange, så returneres den højestgældende.
Tredje og sidste mulighed er, at man ikke kan åbne med 1 i major fordi dette kræver 5+farve og at man således kan blive nødt til at åbne med 1 i minor på færre end 4 kort i farven.

Åbning 1x med 18 – 21 hp

```
public bool one_x_opener_18_21()
{
    bool res = false;
    res = hp_interval(18, 21) && (one_minor_opener || one_major_opener);
    res = res && !(two_clubs_opener_(TwoClubsOpening.strong20) ||
                   two_clubs_opener_(TwoClubsOpening.strong_spillestik));
    return res;
}
```

Først kræves det at honnørstyrken skal ligge mellem 18 og 21 HP og at hånden åbnes med 1 i minor eller med 1 i major (og altså ikke i sans hvad intervallet for det så end måtte være).

Filtrene

```
public enum TwoClubsOpening
{
   undefined = 0,
   weak6card = 1,
   strong22 = 2,
   strong20 = 3,
   strong_spillestik = 4
}
```

Dernæst tilføjes et yderligere krav, der egentlig er en slags 'rettidig omhu': Det er planen at man senere skal kunne vælge flere karakteristika ved det system som ligger til grund for filtrene. Til det formål har jeg bl.a. defineret en 'Enumeration' som skal angive hvad åbning 2♣ skal betyde (se indsæt til venstre). Foreløbig har jeg sat værdien af dette til 'strong22' dvs. åbning 2♣ viser 22+hp og den sidste betingelse i ' `one_x_opener_18_21()`' er således altid opfyldt.

Åbning 1UT

```
return hp_interval(lowerlimit, higherlimit) && nT_fordeling;
```

Der kræves det rette antal honnørpoints samt desuden sansfordeling. Dette sidste er defineret ved at der ikke må være 5-farve i major, at der ikke må være en singleton og at der heller ikke må være 2 doubleton'er. Det giver så sig selv, at der ikke kan være nogen 6+farve.

Åbning 1UT og majorfit

Dette er fuldstændigt som for 'åbning 1UT med det yderligere krav, at spillet kun slipper gennem filteret dersom svarer har 4+kort i en major hvori UT-åbneren også har 4. Der er ingen honnørkrav til svarer.

Åbning 2UT

Stort set som åbning 1UT men naturligvis med andre pointskrav. Bemærkelsesværdigt er dog, at kravene til fordeling er lidt anderledes, idet der her tillades en 5-farve i major. Dette skyldes en anbefaling af Eddie Kantar ('Take All your Chances' bind 1 spil 53) hvor han argumenterer, at man ellers let kommer til at spille i sans fra den forkerte side

Revershænder

Her er koden noget langstrakt, og jeg vedlægger den derfor ikke. Men kriterierne bygger på det der engang blev beskreveet som 'moderne reversprincipper' i en forløber for 'meld bedre' (som har lidt anderledes men dog meget lignende principper, det kan være at teksten her opdateres på et senere tidspunkt). Programmet angiver at det er en revershånd når:
Der åbnes i en 5+ farve og genmeldes i en *højeregældende* 4+ farve
Hvis svarer har meldt 1 – o – 1 (altså fx 1♠ efter vores åbning 1♣) så lover revers (genmelding 2♦ eller 2♥) at man har 17+ ('Meld bedre' skriver 18) honnørpoints. Det er rundekrav.
Dersom svarer har meldt 2 – o – 1 (altså fx 2♣ efter vores åbning 1♦) så lover revers (2 i major) 15+ honnørpoints ('Meld bedre' skriver 16) og er til gengæld udgangskrav.
Programmet medtager kun hænder hvor defensiven ikke melder andet end pas.

Kravåbning

I afsnittet 'åbning 1x med 18 – 21 hp' har jeg omtalt min 'enumeration' TwoClubsOpening og det er klart, at værdien af denne har betydning for definitionen af hvad en kravåbning kan indeholde. Men foreløbig har jeg sat den til at love 22+hp og dette er så indtil videre definitionen af hvad en kravåbning er. I det system jeg selv spiller med i turneringer har vi 'byttet' så åbning 2UT direkte lover 22-24 hp og 2♣ efterfulgt af genmelding 2UT lover 20-21 HP (for så kan en meget svag svarhånd stå af med en 5+farve i major overfor 20-21 hp); hvis det skal implementeres i programmet er det således ikke alle 22+ hænder der er 'krav'.

Spærreåbninger

Herunder koden vedr. spærreåbninger:

```
public bool spaerreaabning(int niveau, Zone zonestilling, Corner me)
{
  bool taberKrav = false;
  bool distKrav = false;
  bool hpKrav = false;
  Denomination farve = longest_suit_();
  if (suit_length(farve) > 3 + niveau)
  {
    switch (zonestilling)
    {
      case Zone.None:
      /* nødvendigt antal stik = 6 + niveau - 2
       * antal vindere = 12 - LTC
       * 12 - LTC >= 6 + niveau - 2
       * LTC <= 8 - niveau */
       taberKrav = lTC <= 8 - niveau;
       break;
      case Zone.All:
       taberKrav = lTC <= 8 - niveau;
       break;
      case Zone.NS:
       if (me == Corner.E || me == Corner.W)
       {
         /* alene udenfor zonen */
         taberKrav = lTC <= 9 - niveau;
       }
       else
       {
         /* alene i zonene */
         taberKrav = lTC <= 7 - niveau;
       }
       break;
      case Zone.EW:
       if (me == Corner.E || me == Corner.W)
       {
        /* alene i zonen */
        taberKrav = lTC <= 7 - niveau;
       }
       else
       {
        /* alene udenfor zonene */
        taberKrav = lTC <= 9 - niveau;
       }
       break;
      }
      if(farve == Denomination.spades)
      {
       distKrav = hearts.Count < 4 && diamonds.Count < 5 && clubs.Count < 5;
      }
      else if(farve == Denomination.hearts)
      {
       distKrav = spades.Count < 4 && diamonds.Count < 5 && clubs.Count < 5;
      }
      else if(farve == Denomination.diamonds)
      {
       distKrav = spades.Count < 4 && hearts.Count < 4 && clubs.Count < 5;
      }
      else if (farve == Denomination.clubs)
      {
       distKrav = spades.Count < 4 && hearts.Count < 4 && diamonds.Count < 5;
     }
     if(niveau == 3)
     {
      hpKrav = hp_interval(5, 11);
     }
     else
     {
      hpKrav = hp_interval(5, 14);
     }
 }
 return taberKrav && distKrav && hpKrav;
}
```

Filtrene

Her er 'niveau' det trin der spærres til (ved meldingen '3♦' er niveau = 3).

Princippet er i øvrigt som det fremgår af kommentaren i kodeteksten for det tilfælde, at ingen er i zonen ('`case Zone.None`', at '123 – reglen' følges idet antallet af vindere beregnes som 12 – antallet af tabere ifølge 'losing trick count' (der er højst 12 'tabere' på en hånd!).

'me' er den hånd for hvilken filteret bruges. Verdenshjørnet (Corner) skifter med spilnummeret men i min seneste udgave af programmet er det altid kortgiveren.

I øvrigt følges anvisningerne i 'Meld bedre' på side 381 ff. Så nogenlunde idet jeg må vedgå, at det er svært at definere 'ikke for gode defensive værdier' i éntydigt C#. Kravet om at hånden skal være 'uegnet til at spille med andre farvere som trumf' forsøges opfyldt ved ikke at tillade 5+ farve i noget som helst andet end spærrefarven (og koden sikrer naturligvis at der ikke kan være 4+ farve i major ved siden af). Jeg har heller ikke taget hensyn til håndens 'placering', idet jeg jo nu har lavet det sådan, at det altid er 1. hånd der har spærrehånden.

Når man vælger 'Spærreåbning' i programmet, så sættes 'niveau' altid til 3. Jeg har således ikke (endnu, men det er jo let at gøre) implementeret et filter som finder spærreåbninger på 4 – trinet. Jeg har bemærket, at frekvensen af spærreåbninger som følger disse regler er lav – faktisk kan man komme ud for at skulle prøve flere gange før der overhovedet dukker nogen op.

Multi (svag variant)

Når du husker, at '&&' betyder 'både og' og at '||' betyder 'eller/og' så tror jeg at nedenstående klip fra koden ret klart fortæller hvad der slipper igennem dette filter.

```
public bool multi()
{
    /* Her følges anvisningerne fra 'Meldebogen'
        * 6 farve med 2 af de 4 tophonnører - eller en lidt dårligere 7-farve
        * 6 - 10 HP
        * Der må ikke være 4-farve i major ved siden af
        * Der må ikke være 2 esser; helst ikke et es og to konger
        * Højst 3 kontroller
        * 7- farve kan accepteres, men så er den tynd
        *  */
    bool kravHearts = ((hearts.Count == 6 && topHonorsOfFour(hearts) >= 2) ||
                    (hearts.Count == 7 && hCP_(Denomination.hearts) < 3)) &&
                    spades.Count < 4;
    bool kravSpades = ((spades.Count == 6 && topHonorsOfFour(spades) >= 2) ||
                    (spades.Count == 7 && hCP_(Denomination.spades) < 3)) &&
                    hearts.Count < 4;
    bool kravHP = hp_interval(6, 10);
    bool kravControls = controls <= 3;
    return kravHP && kravControls && (kravHearts || kravSpades);
}
```

Jeg overvejede at øge kravene – for der slipper nogle hænder igennem som jeg ikke synes er egnede til en 'multi', fx nogle med 5-farve i minor og/eller renonce. Men jeg prøver at lave filtrene så de er begrundet i andet end min intuition.

Styrke efter fjendens svage 2 – åbning

Her bruges filteret for en svag 2 – åbning kombineret med et krav om, at næste hånd har mere end 12 honnørpoints.

Filtrene

Oplysningsdobling efter åbning 1UT eller 1 i farve

Filteret følger nedenstående retningslinier:

- Almindeligvis kræves mindst 3 kort i uanset hvilken farve makker måtte vælge
- Med major 5+ farve dobles ikke – vi melder i stedet farven ind, også med en stærk hånd
- Men over en vis hp – grænse dobles uanset fordeling

Indmelding 1UT

Filteret checker, at der er åbnet 1 i farve og at der er den fornødne honnørstyrke og fordeling samt hold i åbningsfarven (jeg regner A, Kx, Qxx, Jxxx og Txxx samt 5+ kort i farven for hold).

Fordeling til multiforsvar efter åbning 1UT

```
public bool interference_after_1NT(int lowerNTlimit, int higherNTlimit)
{
    // Der er - som det vil bemærkes - INGEN pointskrav, vi ser udelukkende på fordelingen
    bool res = false;
    /* Multiforsvar:    2 kl viser begge majorfarver 5 - 4, der skal være 5 hjerter
        *                  2 ruder viser i princippet 6 - farve i ukendt major, men en god 5-farve er OK
        *                  2 i major viser den 5 kort i den viste major (kan være mindre god) + side 4+ farve i minor
        *                  Nedenstående 2 bruger vi ikke
        *                  2UT viser mindst 5-5 i minor eller en vilkårlig meget skæv udgangskrævende hånd
        *                  3x er en spærremelding efter zonestillingen. 3 i minor kan være ret honnørstærk
        */
    if(hearts.Count > 4 && spades.Count > 3)//der meldes 2 klør
    {
        res = true;
    }
    else if (OK_for_multi(hearts) || OK_for_multi(spades))//der meldes 2 ruder
    {
        res = true;
    }
    else if((spades.Count == 5 || hearts.Count == 5) && (diamonds.Count > 3 || clubs.Count > 3))// 2 i major
    {
        res = true;
    }
    return res;
}
```

Det bemærkes, at indmeldingen ikke nødvendigvis kommer fra den hånd som følger direkte efter 1UT – åbningen; dersom UT – åbnerens svarhånd melder pas kan fjerdehånden balancere.
Der er som understreget INGEN pointskrav vedr. denne indmelding. Det kan tænkes, at jeg senere vil implementere noget sådant, men det er ikke så let som det måske lyder: Dels skal der være forskel på pointskravet i den direkte position og i fjerde hånd, dels mener jeg også at der skal være forskel alt efter hvad 1UT lover; her er min 'gut feeling' at det kræver *bedre* kort at melde ind overfor en svag 1UT. Mod en stærk 1UT er chancerne for at den anden led har udgang reelt ret små så her handler det bare om at finde en farve. Mod en svag 1UT kan der derimod lettere være udgang på den defensive led, så i den situation bør man have bedre styr på honnørstyrken. Men det er ikke noget jeg har set beskrevet nogen steder og jeg har så valgt helt at undlade pointskravet fra dette filter. Den særlige funktion 'OK_for_multi' skal også ses i dette lys.
Det er naturligvis oplagt at implementere andet end multiforsvaret, fx 'D.O.N.T'.

Filtrene

Indmelding straks efter åbning 1 i farve

Det er en meget lang kodesekvens der styrer dette så jeg vil her – stort set – nøjes med at beskrive principperne:

- Det sikres, at hånden ikke kan oplysningsdoble eller indmelde i UT
- Hvis der er en 7+ farve indmeldes i denne
- Hvis der er mere end en 6-farve vælges den mest honnørstærke af disse (man kunne have valgt anderledes)
- Med 6-farve vælges niveauet efter honnørstyrke:

```
// Hvad niveauet angår, så er det billigst muligt med en god hånd og spring
// dersom hånden er honnørsvag og fordelingspræget - hvor PAS i øvrigt er en mulighed
if (hCP >= 10 && lTC < 8 && hCP_(res.Couleur) > 4)
{
    //simpel indmelding på gode værdier og mindst 5 HP i 6-farven
    res.Level = lowestOver(openingBid, res.Couleur);
}
else if (hCP >= 8)
{
    res.Level = niveau;
}
else
{
    res = new Bid(0, Denomination.pass);
}
```

Her er 'lTC' losing trick count', 'res' er den indmelding som programmet overvejer her. 'lowestOver(openingBid, res.Couleur)' finder det niveau man mindst er nødt til at melde på efter åbningsmeldingen 'openingBid' i farven 'res.Couleur'

- Hvis der er mere end en 5-farve vælges den mest honnørstærke af disse. Der springes ikke på en 5-farve.

- Hvad angår indmelding på en 4-farve, så tillader programmet dette (Lawrence er en varm tilhænger af indmelding af 4-farver, men kun på gode farver, kun på 1-trinet og naturligvis kun dersom der ikke er et bedre alternativ – se hans 'The Complete Book on Overcalls' jf. bogskabet). Det forekommer sjældent men jeg har, lidt arbitrært, sat kravet til at der skal være mindst 7 honnørpoint i den valgte farve.

De her anførte principper checkes 'ovenfra og ned': Først sikres det, at der ikke kan oplysningsdobles. Dernæst – hvis det kriterium er opfyldt – undersøges det om der er en 7+ farve. Hvis der så ikke er det gås videre til 6 – farver etc.

Københavnerkonventionen

Filteret undersøger om der er den rigtige fordeling og det rigtige antal 'tabere' (igen jf Losing Trick Count' til at hånden kan bruge Købnehavnerkonventionen efter den givne åbningsmelding.

Balancering i 4. hånd

Det som slipper igennem her er hænder hvor en åbningsmelding på 1- trinet er efterfulgt at 2 passer (efter de principper som dette program anvender for indmelding/svar). Fjerdehånden kan være hvad som helst, men det kan man jo så netop bruge til at øve sig på konsekvenserne af at passe eller melde.

Filtrene

Gamezonen

```
public bool game_considerations(bool alwaysDealer)
{
    /* vedrører 2 hænder og ligger derfor her, ikke i hand.cs */
    Side ds = dealer == Corner.N || dealer == Corner.S ? Side.NS : Side.EW;
    Side os = ds == Side.NS ? Side.EW : Side.NS;
    bool type1 = (majorfit(ds) && (hp(ds) > 22 || losers(ds) < 15)) ||
                    (!alwaysDealer && (majorfit(os) && (hp(os) > 22 || losers(os) < 15)));
    bool type2 = (minorfit(ds) && (hp(ds) > 25 || losers(ds) < 14)) ||
                    (!alwaysDealer && (minorfit(os) && (hp(os) > 25 || losers(os) < 14)));
    /* Herefter sanstypen - og uden krav om sansfordeling på den enkelte hånd */
    bool type3 = hp(ds) > 24 || (!alwaysDealer && hp(os) > 24);
    return type1 || type2 || type3;
}
```

Indrømmet: Det kan måske se lidt kryptisk ud. Men idéen er den, at jeg opererer med 3 typer af hænder hvorpå game er en mulighed (ikke en 'vished'): Majorfit, minorfit og type3 med råstyrken. ' `Side`' er den 'led' programmet overvejer (NS eller EW). Der er så 'ds' (denne side) og 'os' som er 'other side':

```
Side ds = dealer == Corner.N || dealer == Corner.S ? Side.NS : Side.EW;
```

Heer sættes `ds` til ' `Side.NS`' dersom giver (dealer) er nord eller syd. I modsat fald sættes ds til '`Side.EW`'. Og tilsvarende (modsat) for `os`. ' `losers(ds)`' angiver summen af tabere (Losing Trick Count) for den angivne led.
Hvad angår parameteren '`alwaysDealer`' så kan du ignorere den. Jeg lavede oprindeligt programmet således at kriterierne ikke nødvendigvis altid skulle opfyldes straks på kortgivers hånd: Åbningshånden kunne passe og filteret kunne så slippe en af de følgende hænder igennem. Det er for så vidt OK og for visse filtres vedkommende ligefrem hensigtsmæssigt, men jeg er gået bort fra det for hvis programmet skal bruges til øvelsesformål er bedre sådan. Som det er nu sætter jeg bare ' alwaysDealer' til 'true' ved starten.

Slemzonen

Dette ligner programmæssigt filteret med 'Gamezonen'.
Filteret accepterer fordelingen blot der er slemmuligheder på én af lederne. Der er 2 typer af hænder som giver mulighed for slem. Den ene er de meget honnørstærke, den anden de mere fordelingsprægede.

Hvad angår de honnørstærke er kravet her sat til mindst 32 hp. Det giver plads til 2 manglende esser, men det må parret jo så undersøge.

Den anden type er mere kompliceret: Det kræves, at der er sekundær kontrol i alle farver og at der højst mangler primær kontrol i én af farverne. Jeg har sat det maksimale LTC (Losing Trick Count) til 11 selv om det formelt sel burde være 12. Dette er en pragmatisk overvejelse: Med LTC = 12 producerer programmet alt for mange hænder hvor man i praksis ikke kan vinde slem.
Efter en tilsvarende overvejelse har jeg sat et HP krav til mindst 26. På denne måde findes mange tynde slemmer selv om programmet også producerer nogle som er komplet urealistiske.