Hermun skautaskála hjá Ísáli

Gunnarr Baldursson & Ragnar Gísli Ólafsson

Apríl 2011

Útdráttur

Álverið í Straumsvík notar svokölluð skaut til að rafgreina súrál í súrefni og ál. Skautin, sem samanstanda af kolum og gaffli, liggja í kerum í kerskála sem hafa ákveðinn straum og með tímanum þarf að endurnýja þau. Til þess hefur álverið skautskála sem að endurnýjar skautin. Þessi skáli inniheldur tvo starfsmenn og sjö vélar sem vinna sérhæfð verk. Álverið hyggst hækka strauminn í tveimur af sínum þremur kerskálum og verður það til þess að endingartími skautana í þeim skálum styttist. Verkefnið er hermun skautskálans til þess að kasta ljósi á það hversu margar tafir skautskáli þoli til að ná lágmarksafköstum, hvort að tafir séu í biðröðum vélanna og hvernig vélarnar nýtist. Einnig hefur álverið áhuga á því hvernig straumhækkunin hefur áhrif á ferlið. Til þess að svara þessum spurningum, og fleirum, er líkan kynnt til sögunnar. Frumgerð þess er útfærð í forritunarmálinu C sem finna á í viðauka ásamt inntaksgögnum þess og úttaki.

Efnisyfirlit

1 Inngangur

Alcan á Íslandi hf, betur þekkt sem Ísál, er hluti af Rio Tinto Alcan, fjölþjóðlegu fyrirtæki sem er stærsti álframleiðandi í heimi. Ísal rekur álverið í Straumsvík sem er ellefta stærsta álverið innan samsteypunnar. Framleiðslugetan er um 185 þúsund tonn og starfsmennirnir eru um 450; vélvirkjar, verkfræðingar, stóriðjugreinar, rafvirkjar, verkafólk, tæknifræðingar, málarar, skrifstofufólk, bifvélavirkjar, viðskiptafræðingar, múrarar, matreiðslumenn, rafeindavirkjar, smiðir o.fl

Ísal rekur þrjá kerskála þar sem að svonefnd skaut eru notuð til rafgreiningar áls. Kerin eru númeruð frá 1001 til 3160 þar sem fyrsta talan stendur fyrir númer skála, og næstu þrjár númer kers í skálanum. Hvert ker hefur 24 skaut sem að slitna með tímanum. Þess vegna þarf að skipta um þau á 26 til 30 daga fresti en það er mismunandi eftir skálum. Skáli 3 hefur stærri skaut og hærri straum, 165 kA og þar endast skaut í 26 daga. Skálar 1 og 2 hafa lægri straum, 133 kA og minni skaut sem að endast í 30 daga. Daglega þarf að skipta út um það bil 404 skautum.

Í kerskála er unnið á þremur vöktum allan sólarhringinn alla daga vikunnar og fer fram skautskipting á hverri vakt. Hver vakt nemur 8 klukkustundum, næturvaktin byrjar á miðnætti, dagvaktin klukkan átta og kvöldvaktin klukkan fjögur. Hver vakt skiptir því um $404/3 = 104$ skaut. Starfsmenn kerskála taka brunnin skaut úr kerum og setja ný skaut í kerin í staðinn. Síðan kemur starfsmaður skautskálans og nær í brunnu skautin sem bíða á vögnum í kerskálanum og flytur þau á sérstakan kæligang. Þar skilur hann þau eftir og nær í staðinn í brunnin skaut sem eru orðin köld og fer með þau í skautskála til hreinsunar. Í hvert skipti sem starfsmaður skautskála sækir brunnin skaut úr kerskála kemur hann með ný skaut. Því er alltaf jafn fjöldi vagna sem fer inn í skautskálann og út úr honum.

Skautin eru flutt á tveimur tengdum vögnum með 12-14 skautum á í einu. Ferðir frá skautskála til kerskála eru aðeins farnar á dagvöktum og kvöldvöktum. Skaut sem þarf að nota á næturna eru því keyrð til kerskála á dag - og kvöldvöktum. Meðaltal fjölda ferða frá skautskala til kerskála eru um það bil 30 á sólarhring, eða 15 á vakt. Skautskáli reynir að framleiða þann fjölda skauta sem nemur skautafjölda tveggja vakta hjá kerskála á hverri vakt, og er því tveimur vöktum á undan.

Fræðileg hámarks afkastageta skautskála eru 52 skaut á klukkustund en vegna bilanna er afkastageta á hverri vakt í besta falli um það bil 40 skaut á klukkustund. Skálinn er framleiðslulína sem að samtímis tekur skautleifar af vögnum og hreinsar ásamt því að taka á móti nýjum skautum og setja á vagnanna. Lestun nýrra skauta og losun brunninna skauta er samtengt ferli, ef ekki er hægt að taka brunnin skaut af vagni þá er heldur ekki hægt að setja ný skaut á vagn.

Framleiðsluferli skautskála hefst þegar skautleifar koma á vögnum að vél A, sem að hífir þær af vögnum. Eftir það fara þær í gegnum vélar B til E þar sem að leifarnar eru hreinsaðar þannig að gaffallinn stendur einn eftir. Gaffallinn heldur síðan áfram inn í aðra byggingu sem að nefnist skautsmiðja, þar sem hann er skoðaður, réttur af og sandblásinn áður en hann fer í steypun þar sem að ný kol eru steypt við hann. Þá er hann tilbúinn sem nýtt skaut. Þegar þessu ferli er lokið kemur skautið að vél F þar sem það er merkt og sent til vélar G. Vél G lestar skautið á vagn, sem er síðar keyrður til kerskála. Þessu ferli er lýst á Mynd 1.

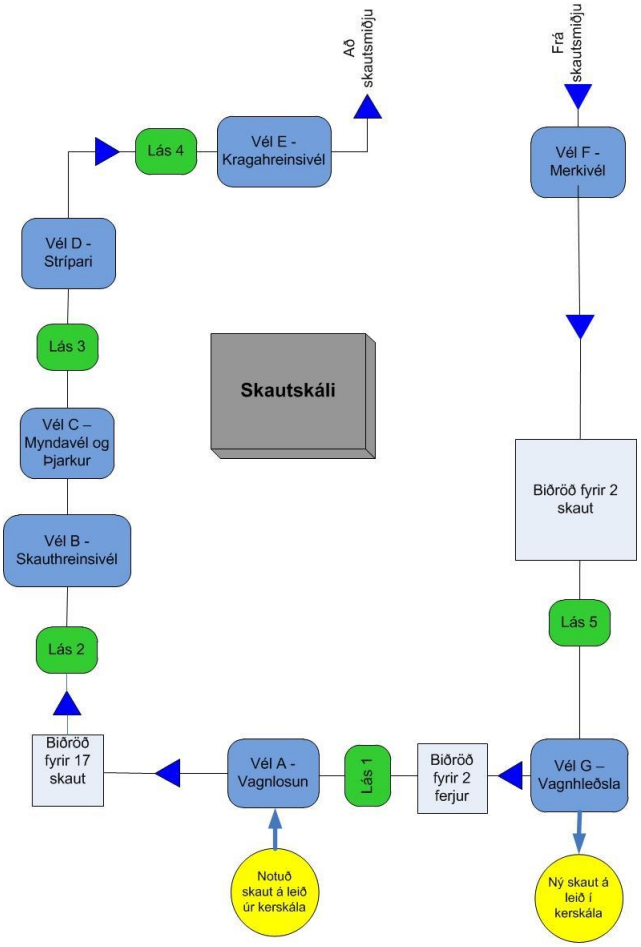
Skautið er tekið inn í ferlið þannig að það er hengt á ferju sem að er dregin áfram af keðju, sem að fer í gegnum allan skautskálann og inn í skautsmiðjuna og til baka. Ferlið er raðgengt svo ef vél er að afgreiða skaut þarf skautið á eftir að bíða þangað til að vélin hefur lokið sér af. Til að stýra þessu flæði eru svokallaðir lásar staðsettir með regulegu millibili á keðjunni og kúpla þeir ferjum út til að stöðva þær. Þannig geta sum skaut verið á hreyfingu á meðan önnur eru kyrrstæð því að keðjan sjálf stöðvar ekki nema slökkt sé á henni handvirkt. Á bak við sumar vélar eru biðraðir en þar bíða skaut eftir afgreiðslu ef að vélin er upptekin. Lása og biðraðir má sjá á Mynd 1. Lásar á undan fullum biðröðum mega ekki sleppa sínum skautum þangað til að það rúmast til í röðinni. Skaut geta ekki farið framhjá vélum þannig að ef að vél bilar lengi og röð hennar fyllist heldur sá lás sem kemur þar á undan sýnu skauti föstu og þannig koll af kolli. Þannig getur löng bilun stöðvað skautahreinsiferlið í einhvern tíma þó að keðjan sem ber ferjurnar haldi áfram keyrslu. Hún er þá eins og bílvél með einhvern snúningshraða sem er í hlutlausum gír.

Það er nokkuð slembið hvaða vélar stoppa nema vél F sem að bilar nánast aldrei. Þegar stærri bilanir eiga sér stað þarf að kalla út viðgerðarmenn en í flestum tilfellum tekur það 5 til 30 mínútur að koma bilaðri vél aftur af stað. Skakkt skaut í vél flokkast sem bilun og þá þarf starfsmaður að bakka því út úr vélinni, leiðrétta það og senda inn aftur. Svoleiðis atvik eiga sér stað nokkrum sinnum á sólarhring og eru helsta ástæða þess að afköst skautskála nema um það bil 40 skautum á klukkustund. Ef viðgerðartímar eru þeim mun lengri á einhverri vakt þá þarf vaktin sem kemur á eftir að vinna upp framleiðslutapið. Skautskáli keyrir aðeins á dagvöktum og kvöldvöktum.

Framkvæmdir eru hafnar við að auka strauminn í kerum 1 og 2 sem veldur dræmri endingartíma skauta, og koma þau þá til með að endast í 26 til 28 daga eftir straumhækkun. Gerð verður sú nálgun að alltaf sé nóg til af nýjum skautum í skautsmiðju sem koma að vél F. Verkefnið er að herma ferli skautskála með eftirfarandi vangaveltur í huga:

1. Hversu mikið af töfum (í mínútum talið) þolir skautskálinn til að ná lágmarksafköstum?
2. Er það ráðlegt að stækka biðraðir eða bæta við biðröðum?
3. Hve miklu munar það fyrir ferlið ef að starfsmenn koma vélum af stað eins fljótt og þeir geta?
4. Ef tafir eru litlar, hvenær hefur vakt náð lágmarksafköstum?
5. Hversu fljótur er skálinn að vinna upp langar viðgerðatafir?
6. Hvaða áhrif hefur hækkun straums á ferlið?

Þar sem að meðaltími, besti og versti tími skauta í gegnum kerfið, meðalhámarkslengd biðraða og nýtni véla verða til hliðsjónar.



Mynd 1: Ferli skautskála

2 Niðurstöður

Í grein 3.3 kemur fram að talan *nokkrar bilanir* séu 8 bilanir, svo að niðurstöður skýrslu miðast við þann bilanafjölda á sólarhring. Í heimsókn til Ísal [3] kom fram að vegna bilana nemi raunframleiðsla skautskála um það bil 44 skautum á klukkustund þrátt fyrir að skálinn geti fræðilega afkastað meiru. Inntaksgögn líkans (sjá grein 5.2) er þannig að við átta bilanir á sólarhring nemur framleiðsla um það bil 44 skautum á klukkustund. Í inntaki eru skilgreindar tvær heiltölubreytur, önnur er fyrir lágmarksfjölda bilana á sólarhring og hin fyrir hámarksfjölda bilana á sólarhring. Þessar breytur mynda því bil, og fyrir hverja heiltölu á þessu bili er hermunin framkvæmd. Í inntaki eru þessar breytur 0 og 10. Í grein 3.1 er talað um að lágmarksframleiðsla skautskála á sólarhring fyrir straumhækkun eru 404 skaut. Eftir hækkun er talan 444 skaut. Tímabilið sem hermt var yfir eru þrír mánuðir þar sem að 2 vaktir eru unnar á sólarhring og miðast niðurstöðurnar við átta bilanir á sólarhring.

Vél	**Nýtni**	**Röð**	**Meðalbið í sekúndum**
A	0.657697	*A*	591.513442
B	0.284283	*B*	0.003355
C	0.231497	*C*	*-engin röð-*
D	0.615817	*D*	1.618098
E	0.607614	*E*	1.957143

Eins og sjá má liggja engin gögn fyrir um vélar F og G en hægri hlið skautskála er undanskilin hermum að mestu leiti (sjá grein 3.4)

3 Forsendur og Líkan

Til að komast að niðurstöðum smíðuðum við líkan sem að hermir eftir ferli skautskála. Í næstu undirgreinum er forsendum líkansins lýst.

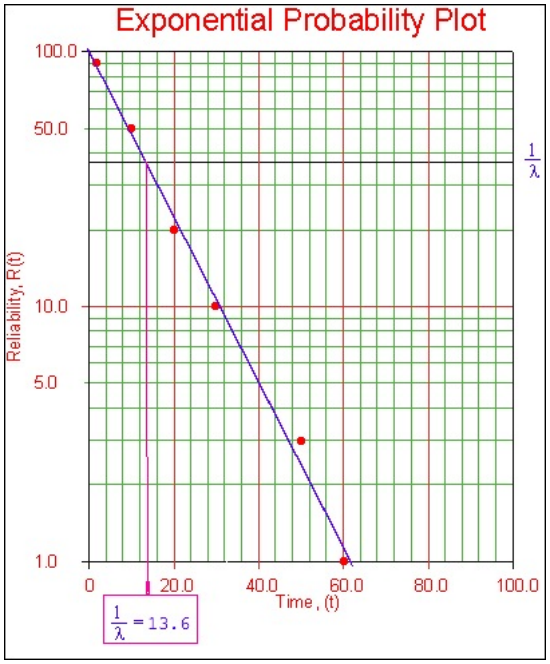
3.1 Umskipting skauta og afköst Skautskála

- Fyrir straumhækkun þá þarf að skipta um skaut í skálum 1 og 2 á 30 daga fresti, og í skála 3 á 26 daga fresti. Það gerir $\frac{24 \cdot 160}{26} + \frac{24 \cdot 160}{30} + \frac{24 \cdot 160}{30} = 403.69$ skaut á dag, þar sem að allir skálar hafa 24 skaut í hverju keri og 160 ker eru í hverjum skála. Nefnararnir í formúlunni eru endingardagar skauta í viðeigandi kerskála. Sú tala er námunduð upp í 404 skaut á dag og eru það lágmarksafköst skautskála.
- Eftir straumhækkun þarf að skipta um skaut í öllum skálum á 26 daga fresti. Það gerir $\frac{24 \cdot 160}{26} + \frac{24 \cdot 160}{26} + \frac{24 \cdot 160}{26} = 443.07$ skaut á dag. Sú tala er námunduð upp í 444 skaut á dag og er það lágmarks afkastageta skautskála eftir straumhækkun.
- Fræðileg hámarksafköst skála eru 52 skaut á klst, eða $16 \cdot 62 = 832$ skaut á sólarhring (Skautskálinn starfar aðeins í tvær vaktir, eða 16 klukkustundir á sólarhring).
- Raunveruleg hámarksafköst skála eru 40 skaut á klst, eða $16 \cdot 40 = 640$ skaut á sólarhring (Skautskálinn starfar aðeins í tvær vaktir, eða 16 klukkustundir á sólarhring).

3.2 Dreifing fyrir viðgerðartíma

Við völdum exponential distribution til að ákvarða viðgerðartíma. Ástæðan fyrir því er sú að fallið er frekar einfalt, stærðfræðilega séð, og að það virtist smellpassa við þau gögn sem við settum upp. Einnig gerum við ekki grein fyrir því að vélar bili oftar með tímanum(degrade/wear out) því þurfum við ekki flóknari dreifingu en veldisdreifingu.

Við settum upp töflur fyrir viðgerðartíma útfrá því sem okkur var sagt í ferðinni. Taflan hér að neðan sýnir viðgerðartíma og áreiðanleika. Þar að segja er stuttur viðgerðartími líklegri en lengri. Við munum bæta 5 mínútum við alla viðgerðartíma þar sem það er sá lágmarkstími sem tekur að gera við vél. Við gerum ekki mun á milli véla, allar vélar hafa sömu líkur á því að bila(komum að



Mynd 2: Inntak fyrir veldisdreififall miðað við núverandi lengd viðgerða

því síðar).

Alvarleiki Bilanna / Viðgerðartími (+ 5 mín)	Áreiðanleiki Metið %
2	100-10 = 90
10	100-50 = 50
20	100-70 = 30
30	100-90 = 10
50	100-97 = 3
60	100-99 = 1

Til að finna rétt inntak í formúluna til að fá þessa dreifingu þurfum við að plotta þessa punkta á mynd sem er sett upp fyrir Veldisdreifingu(sjá Mynd 2.)

Þegar búið er að plotta þessa punkta þá er dregin lína í gegnum þá, með góðri nálgun. Þar sem línan mun skera 36.8 inntakið fyrir fallið á X-ásnum.

Talan 36.8% er fengin með

$$R(t) = e^{(-\lambda \cdot t)}$$

$$R(t) = e^{(-\lambda \cdot \frac{1}{\lambda})} \text{ því að t = m 1/lambda}$$

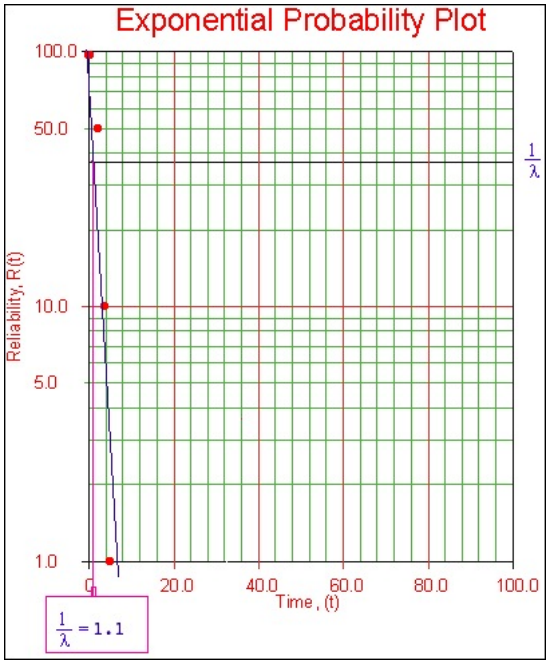
$$R(t) = e^{-1} \text{ styttist út í}$$

$$e^{-1} = 0.368 \text{ eða } 36.8\%$$

Bilanatíðni véla er random þar sem við fengum engin gögn um tíðni vélanna. En bilanir véla geta átt sér stað á sama tíma, þar að segja 2 vélar geta bilað á svipuðum eða sama tí ma. Einnig getur sama vélin ekki bilað tvisvar á sama tímapunkti.

Á Mynd 2 er búið að plotta á myndina og út úr því fékkst 13.9 sem inntak í veldisdreifinguna. Þetta miðast við núverandi bilanalengd (sjá 5.2).

Mynd 3 sýnir inntak í veldisdreififall ef starfsmenn skautskála koma biluðum vélum í gang eins fljótt og þeir geta. Hámarks viðgerðartími getur þá numið 10 mínútum.



Mynd 3: Inntak fyrir veldisdreififall ef hámarksviðgerð hverrar vélar er 10 mínútur

3.3 Bilanir

Samkvæmt verkefnislýsingunni [1] er það nokkuð slembið hvaða vélar bila, að vél F undanskilinni, og að bilanir eiga sér stað nokkrum sinnum á sólarhring. Gildið á tölunni *nokkrum sinnum* er illa skilgreint en höfundar sammæltust um töluna 8. Fyrir flestar bilanir er viðgerðartíminn 5 til 30 mínútur. Í einhverjum tilfellum þarf að ræsa út viðgerðarmann ef um stórar bilanir er að ræða og slíkar bilanir geta varað í nokkrar klukkustundir. Engin önnur gögn liggja fyrir um bilanir eða tíðni þeirra og þar sem að gögnin eru ekki nákvæmari voru eftirfarandi forsendur gefnar:

- Allir viðgerðartímar liggja á bilinu 5 til 30 mínútur.
- Viðgerðartímar eru veldisdreyfðir þannig að mestar líkur eru á viðgerð taki 5 mínútur og minnstar líkur eru á 30 mínútuna viðgerð. Þar sem að bilanir og tafir vegna skakkra skauta í vélum má flokka undir sama hatt þykir höfundum líklegast að um slíkar tafir sé að ræða frekar en vélræna bilun.
- Tímasetningar bilana á sólarhring eru uniform dreifðar.
- Ef að vél A eða G bila eru engar ferðir farnar frá Skautskála til Kerskálanna meðan á viðgerð stendur.

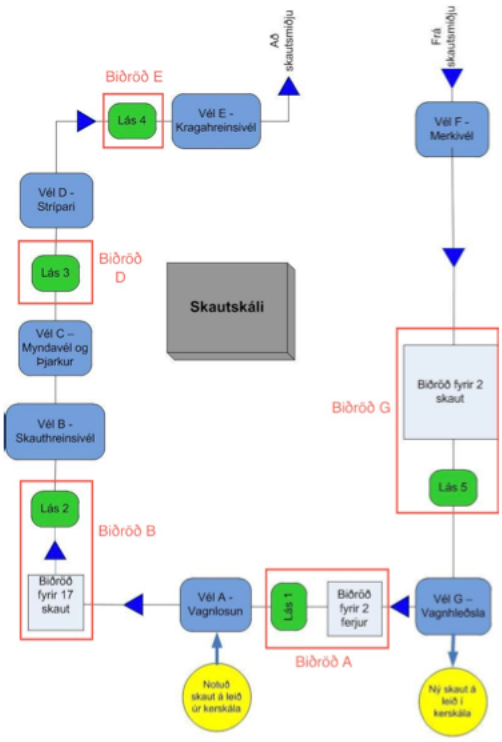
3.4 Einingar og undirkerfi

Þættir skautskála eru dregnar saman í undirkerfi eins og sjá má á eftirfarandi töflu:

Eining	**Þættir**	**Hlutverk**
A	Vél A, 14 skauta biðröð í formi vagna	Vagnlosun
B	Vél B, biðröð og lás sem geyma 17 skaut	Skauthreinsivél
C	Vél C	Myndavél og Þjarkur
D	Vél D, einn lás	Strípari
E	Vél E, einn lás	Kragahreinsivél
F	Vél F, einn lás	Merkivél
G	Vél G, biðröð fyrir 2 skaut	Vagnhleðsla

Þessu er lýst á Mynd 4. Vélar B og C geta unnið tvö skaut í einu. [1]

Þegar Mynd 4 er skoðuð má sjá að hægt er að skipta Skautskála upp í tvo helminga sem hefur hvor sitt inntak og sitt úttak. Inntak í vinstri helming kemur frá vél A, og úttak hans fer frá vél E.



Mynd 4: Einingar líkans

Inntak í hægri helming kemur frá vél F, en sú nálgun er gerð að þar sé ávallt nóg af nýjum skautum að taka, og úttak þess helmings er vél G, sem að hleður nýju skautunum á vagna. Við heimsókn í Ísal [3] kom fram að bilanir og tafir megi sjaldnast rekja til hægri helmingsins. Þar eru aðeins tvær vélar meðan vinstri hliðin hefur fimm vélar sem að vinna flóknari verk. Af þeim ástæðum er vél F undanskilin hermun. Vél G getur bilað, og ef það gerist stöðvar lestun og losun skauta um þann tíma sem það tekur að gera við bilunina.

Þegar vagn kemur með skautaleyfar til vélar A bíður hann á meðan leyfarnar eru hýfðar af honum. Því næst er hann er hann hlaðinn með nýjum skautum [1]. Gert er ráð fyrir því að fjöldi skautaleyfða sem hýfðar eru af vagni og fjöldi nýrra skauta sem lestuð eru á vagn sé um það bil sá sami.

3.5 Vélar og vinnslutímar

Vinnslutími vélar er sá tími sem líður milli þess að skaut kemur að lausri vél og fer frá vélinni aftur. Færslutími er sá tími sem líður milli þess að skaut fer frá vél og kemur að næstu vél. Þeir eru reiknaðir út samkvæmt gagnaskjali, [2]. Þar sem að vélar B og C geta unnið tvö skaut í einu er vinnslutími þeirra helmingaður. Færslutími vélar E er 0 af því að skaut sem fer frá þeirri vél fer úr skautskála.

Vél	Vinnslutími	Færslutími
A	70	129.83
B	21.798	122.83
C	12.7	18.98
D	67.41	22.74
E	69.75	0

3.6 Upphitunartími

Engin gögn liggja fyrir um upphafsstöðu skautskála þegar ein vakt tekur við af annari. Sú forsenda er því gerð að þegar ein vakt tekur við af annari eru skaut nú þegar í ferlinu, vakt kemur ekki að tómum skála. Hvar skaut eru í ferlinu eru slembið og ekki fasti í líkaninu en þegar hermiforrit er ræst eru engin skaut í skálanum. Þess vegna þarf upphitunartíma, tíma þar sem að hermun er framkvæmd til þess að fá skaut í kerfið en engum gögnum safnað af því að þau eru ómarktæk. Til að byrja

með eru engin skaut í röðum, og ef að sú staða væri tekin með í reikninginn við gerð skýrslu þá skekkir hún niðurstöðurnar. Eftir að upphitunartíma er lokið er hægt að safna marktækum gögnum. Upphitunartíma má lesa úr inntaksskrá (sjá grein 5.2).

3.7 Atburðir og kjarnavirkni líkans

Líkanið er atburðadrifið: einhver atburður á sér stað sem að getur skrásett annan atburð og þannig koll af kolli þangað til að keyrslu er lokið. Kjarni líkansins er atburðavinnslan sjálf, hvernig bregðast skal við þeim atburðum sem að skilgreindir eru. Eftirfarandi atburði skal skilgreina:

- Vagn kemur með brunnin skaut að vél A
- Skaut kemur að vél
- Skaut fer frá vél
- Vél bilar
- Vél löguð
- Endir upphitunartíma
- Endir hermunar

Næstu undirgreinar útskýra hvernig bregðast þarf við þessum viðburðum.

Vagn kemur með brunnin skaut

Vagnar sem koma með hrein skaut geta flutt 12 til 14 skaut saman lagt og er það uniform dreifð slembitala. Fyrir hvert skaut þarf að framkalla atburðinn *skaut kemur að vél*, þar sem að vélin er vél *A*. Hver slíkur atburður þarf að innihalda eftirfarandi gögn:

- Tímasetningin þegar atburðurinn á sér stað
- Staðsetning skauts í ferlinu
- Tímasetningin þegar skautið kemur fyrst í kerfið
- Raðnúmer skauts

Loks þarf að skrásetja annan *vagn kemur með brunnin skaut* atburð. Þar sem að vagnar koma á um það bil 32 mínútna fresti að meðaltali á dag, meðan unnið er í Skautskála [1], er eðlilegt að koma þeirra sé uniform slembitala milli 28 og 30. Ef að vélar *A* eða *G* eru bilaðar þarf að fresta komu næsta vagns um þann tíma sem það tekur við að gera við vélarnar af því að vagnlosunin og lestun eru samtengd ferli [1].

Skaut kemur að vél

Þegar skaut kemur að vél þarf að huga að ýmsu.

- **Er vélin upptekin?**
  Ef að vélin er laus skal merkja að skautið hafi fengið þjónustu umsvifalaust. Svo skal skrásetja *Skaut fer frá vél* atburð sem að inniheldur tímasetningu brottfarar og vél sem að farið er frá. Annars skal vista komutíma skauts og setja það í röð þeirrar einingar sem skautið kemur að, ef einhver er.
- **Hefur vélin röð og ef svo er, er röðin full?**
  Ef að vélin hefur enga röð eða röðin er full þarf að fresta komu þessa skauts, lásinn sem að heldur því má í rauninni ekki sleppa því þangað til að það rúmast til í röðinni.
- **Er vélin biluð?**
  Ef að vélin er biluð þarf að fresta þessum atburði um þann tíma sem að samsvarar viðgerðartímanum.

Skaut fer frá vél

Þegar *skaut fer frá vél* atburður er meðhöndlaður hefur vél unnið sitt verk á skautinu.

- Ef að vélin sem skautið fer frá er biluð þarf að fresta atburðinum um þann tíma sem það tekur að gera við vélina.

- Ef að vélin sem skautið fer frá er vél E þarf að hækka teljara sem telur hversu mörg skaut hafa farið frá vél A til E. Sú forsenda var gerð að fjöldi skautaleyfa sem losuð eru af vögnum við A sé um það bil sá sami og fjöldi nýrra skauta sem lestuð eru við G má hækka þennan teljara um tvo. Annars skal skrásetja *Skaut kemur að vél* atburð frá núverandi vél sem á að meðhöndla eftir færslutímann að næstu vél.
- Ef að röð vélarinnar sem farið er frá er ekki tóm þarf að vinna fremsta skautið í vélinni, halda utan um hve lengi skautið þurfti að bíða eftir þjónustu og skrásetja þá *Skaut fer frá vél* atburð eftir vinnslutíma vélarinnar.

Vél bilar

Þegar að vél bilar þarf að merkja hana bilaða, áætla viðgerðartíma fyrir hana og skrásetja *Vél löguð* atburð eftir viðgerðartímann. Sjá undirgrein 3.3 fyrir umfjöllun um bilanir að ofan.

Vél löguð

Þegar að biluð vél er löguð þarf að merkja vélina lagaða, svo að hún valdi ekki lengur töfum í ferlinu.

Endir upphitunartíma

Nú þarf að endurstilla skautateljara og ýmsar tölfræðibreytur sem að halda utan um tafir raða og nýtni véla.

Endir hermunar

Nú þarf að prenta út skýrslu á skjá eða í skjal með niðurstöðum hermunar.

4 Sannreyning Líkans

Höfundar beittu fjórum aðferðum til að sannreyna það að líkanið sé gott og gilt fyrir gögnin sem þeim voru gefin.

Samanburður við þekktar niðurstöður

Fræðileg hámarks framleiðslugeta skautskála eru 52 skaut á klukkustund [1], en vegna bilanna eru raunafköst nær 44 skautum á klukkustund [3] vegna bilana. Þegar frumgerð líkans var keyrð upphaflega, áður en rökfræði sem snýr að bilunum far smíðuð, náðu bestu afköst líkansins að meðaltali 52.9 skautum á klukkustund. Þar er vissulega skekkja um 0.9 skaut á klukkustund en hvort sú skekkja stafar af mistökum í líkanagerð eða dræmum gögnum [2] um vinnslutíma véla og færslutíma skauta á milli þeirra er erfitt að segja til um. Hinsvegar eru 52.9 skaut / klst nokkuð nákvæmt og gefur það til kynna að líkanið og gögnin séu ásættanlega lýsandi fyrir ferlið sjálft.

Eftir að rökfræði bilana var smíðuð eru afköst um það bil 43.49 skaut / klst, miðað við 8 bilanir á sólarhring, sem er 4 skautum meira en raun ber vitni um. [1]. Niðurstöðurnar eru samt sem áður nærri lagi og þykja því bera vitni um áreiðanleika líkansins og gagnana sem það notar.

Engin þörf er á því að skala kerfið, svosem með auknum töfum til að það gangi upp.

Ferlið rakið

Þegar á þróun líkansins stóð var staða véla og raða prentuð í skrá við hvern atburð sem var meðhöndlaður. Þannig var hægt að sjá það skref fyrir skref hvernig skaut ferðuðust milli véla og biðraða til að ganga úr skugga um að allt væri samkvæmt settum reglum.

Mismunandi inntaksgögn

Ferlið var hermt fyrir mismunandi margar bilanir á sólarhring. Útkoman var eins og við var að búast; bein tenging er á milli fjölda bilanna og afkastagetu á skautskála klukkustund.

Extreme Programming

Höfundar smíðuðu líkan í *Extreme Programming* stíl sem stuðlar að hraðvirkari uppgötvun villa.

Heimildir

[1] Starfsmaður Ísal, *HermunIsal_2011_r2.pdf*. 2011.

[2] Starfsmaður Ísal, *Millitimar.xls*. 2011.

[3] *Heimsókn til Ísal*. 21. mars 2011.

5 Viðauki

5.1 Frumgerð í forritunarmálinu C

```c
/*
 *  isal.c
 *
 *
 *  Created by Gunnarr Baldursson & Ragnar Gisli Olafsson on 4/18/11.
 *  Copyright 2011 Haskoli Islands. All rights reserved.
 *
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "simlib/rndlib.h"
#include "simlib/simlib.h"

// EVENTS
#define EVENT_WAGEN_UNLOAD_ARRIVAL  1
#define EVENT_WAGEN_UNLOAD_DEPARTURE 2
#define EVENT_SKAUT_ARRIVAL 3
#define EVENT_SKAUT_DEPARTURE 4
#define EVENT_MACHINE_FAILURE 5
#define EVENT_MACHINE_FIXED 6
#define EVENT_END_SIMULATION 7
#define EVENT_END_WARMUP  8
#define EVENT_GENERATE_FAILURES 9

// STREAMS
#define STREAM_WAGEN_ARRIVAL 1

//Other constants
#define NUM_MACHINES 7
#define SHIFT_LENGTH 57600.0;
#define WAGEN_LOAD 14
#define MACHINES_ON_THE_LEFT_SIDE 5
#define MACHINES_ON_THE_RIGHT_SIDE 2
#define OPTIMAL_THROUGHPUT 52
#define ACTUAL_THROUGHPUT 40
#define TRANSFER_ARRAY_LENGTH 11
#define PREP_TIME 0.0

typedef struct
{
    float failtime;
    float downtime;
    int machine_nr;
} breakdown;


//#define LOADING_TIME_PER_SKAUT

// Global variables
int number_of_machines, min_productivity, min_no_failures, max_no_failures,
    skaut_throughput;
```

```
55 float mean_wagen_arrival, std_wagen_arrival, mean_failures, std_failures,
       min_machine_repair_time, max_machine_repair_time, end_warmup_time,
       end_simulation_time;
56
57
58 int sampst_delays, throughput_time; // variable for queue delays and throughput
       time
59 time_t dummy;
60 unsigned int skaut_id, stream, failure_nr;
61 int queue_size[NUM_MACHINES +1], queue_max_lengths[NUM_MACHINES + 1];
62 float machine_broken[NUM_MACHINES +1];
63 breakdown *fail_list;
64 int fail_index;
65
66 int is_machine_busy[NUM_MACHINES +1],
67     queue_size[NUM_MACHINES +1];
68
69 float work_time[NUM_MACHINES + 1],
70     transfer_time[NUM_MACHINES +1]; // +1 is the less preferable simlib
          indexing scheme
71
72
73 float temp_transfer[TRANSFER_ARRAY_LENGTH];
74
75 FILE *infile, *outfile;
76
77 /* Function signatures */
78
79 // Usage: create_machine_fail_events()
80 // Pre:   init_twister must be called for random number generation
81 // Post:  scheduled events have been created for machines
82 void create_machine_fail_events();
83
84
85 // Usage: push_array();
86 // Pre:   we expect that correct values are in transfer array
87 // Post:  our temp_transfer array now has the values in transfer_array
88 void push_array();
89
90 // Usage: pop_array();
91 // Pre:   we expect that correct values are in transfer_temp array
92 // Post:  our transfer array now has the values in transfer_temp
93 void pop_array();
94
95 // Usage: wagen_arrival();
96 // Pre:   EVENT_WAGEN_UNLOAD_ARRIVAL is the next event to be processed
97 // Post:  14 EVENT_SKAUT_ARRIVAL events are next to be processed on the event
       list.
98 void wagen_unload_arrival();
99
100 // Usage: skaut_arrival();
101 // Pre:   EVENT_SKAUT_ARRIVAL is the next event to be processed
102 // Post:  a skaut has been processed by a machine or put in it's queue.
103 //      subsequent events may have been scheduled
104 void skaut_arrival();
105
106 // Usage: skaut_departure();
107 // Pre:   EVENT_SKAUT_DEPARTURE is the next event to be processed
108 // Post:
109 void skaut_departure(); // do we need an event for departure?
110
111 // Usage: machine_failure();
112 // Pre:   EVENT_MACHINE_FAILURE is the next event to be processed
113 // Post:
114 void machine_failure();
115
116 // Usage: machine_fixed();
117 // Pre:   EVENT_MACHINE_FIXED is the next event to be processed
118 // Post:
119 void machine_fixed();
```

```
120
121 // Usage: end_warmup();
122 // Post:  SIMLIB statistical variables have been cleared
123 void end_warmup();
124
125 // Usage: parse_input(input_filename_data,input_filename_time);
126 // Pre:   input_filename_data,input_filename_time of type char[],
127 //        global variables from the input file exist.
128 // Post:  the global variables were assigned values from input_filename,
129 //
130 void parse_input(char[] ,char[]);
131
132 // Usage: x = N(muy, sigma, stream);
133 // Pre:   muy and sigma are of type float
134 //        stream is of type int
135 // Post:  x is a random gaussian distributed variable of type float
136 //        with mean muy and std sigma
137 float N(float muy, float sigma, int stream);
138
139 // Usage: report("the_report.out");
140 // Pre:   the values to be reported have values
141 // Post:  a report on program values and simlib statistics
142 //        have been APPENDED to "the_report.out"
143 void report();
144
145 // Usage: schedule_failures(i);
146 // Pre:   the global variable end_simulation_time has a value, i is of type int
147 // Post:  i failures have been scheduled uniformly on machines
148 //        with ?random? repair times on the interval [min_machine_repair_time,...
        max_machine_repair_time]
149 //        uniformly distributed over the interval 0...end_simulation_time
150 void schedule_failures(int i);
151
152
153 int main()
154 {
155     // load datafiles
156     parse_input("adal_inntak.in","velar_og_bidradir.in");
157
158     // initialize arrays and variables
159     if((fail_list = malloc(sizeof(breakdown)*NUM_MACHINES+1))==NULL) {
160   printf("Allocation Error\n");
161   exit(1);
162     }
163
164
165
166     int b;
167 /*    for (b=1; b <= number_of_machines; b++) {
168       printf("transfer_time[%d] = %f\n", b,transfer_time[b] );
169       printf("busy %d broken %f \n",is_machine_busy[b],machine_broken[b]);
170       }*/
171     // We perform simulation for "a few" failures per day
172
173     for (failure_nr = min_no_failures; failure_nr<= max_no_failures; failure_nr
          ++) {
174   stream = (unsigned int)time(NULL) % 100;
175
176   memset( is_machine_busy,0, NUM_MACHINES +1 );
177   memset( machine_broken,0, NUM_MACHINES +1);
178   memset( queue_max_lengths,0, NUM_MACHINES +1);
179   memset( fail_list,0, sizeof(breakdown)*(NUM_MACHINES+1));
180   fail_index = 0;
181   skaut_throughput = 0;
182   sampst_delays = number_of_machines +1;
183   throughput_time = number_of_machines +2;
184
185   skaut_id = 1;
186   skaut_throughput = 0;
187
```

```
188
189   // Initialize rndlib
190   init_twister();
191
192   // Initialize simlib
193   init_simlib();
194
195   maxatr = 6; // how many attributes do we need?
196
197   /* Schedule first wagen arrival */
198   event_schedule( 10.0, EVENT_WAGEN_UNLOAD_ARRIVAL );
199
200   /* Schedule end of warmup time */
201   event_schedule( end_warmup_time, EVENT_END_WARMUP );
202   event_schedule(end_warmup_time, EVENT_GENERATE_FAILURES );
203   /* Schedule simulation termination */
204   event_schedule(end_simulation_time , EVENT_END_SIMULATION );
205
206   next_event_type = 0;
207
208
209   while (next_event_type != EVENT_END_SIMULATION) {
210
211       timing();
212
213       switch (next_event_type) {
214       case EVENT_WAGEN_UNLOAD_ARRIVAL:
215     wagen_unload_arrival();
216     break;
217       case EVENT_SKAUT_ARRIVAL:
218     skaut_arrival();
219     break;
220       case EVENT_SKAUT_DEPARTURE:
221     skaut_departure();
222     break;
223       case EVENT_MACHINE_FAILURE:
224     machine_failure();
225     break;
226       case EVENT_MACHINE_FIXED:
227     machine_fixed();
228     break;
229       case EVENT_END_WARMUP:
230     end_warmup();
231     break;
232       case EVENT_END_SIMULATION:
233     report();
234     break;
235       case EVENT_GENERATE_FAILURES:
236     create_machine_fail_events();
237     break;
238
239       }
240   }
241
242     }
243 }
244
245
246
247 void wagen_unload_arrival()
248 {
249
250     int i;
251     int current_unit = 0;
252     float wagen_arrival_zeit = unirand((mean_wagen_arrival-std_wagen_arrival)
          *60.0,(mean_wagen_arrival+std_wagen_arrival)*60.0,stream);
253
254     for (i = 1; i<NUM_MACHINES+1; i++) {  //delay unload of skaut by the time
          it takes to repair
255   if (machine_broken[i] > 0.0) {
```

```
256         event_schedule(sim_time + machine_broken[i], EVENT_WAGEN_UNLOAD_ARRIVAL);
257         return;
258     }
259       }
260
261       if (list_size[number_of_machines + 1] != 0) { // ef allt er enn fullt Ã¿Ãą
              koma meÃř nÃęsta vagn eftir uÃ¿b hÃąlftÃŋma
262     event_schedule(sim_time + wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
263     return;
264       }
265
266       int vagn_magn =  WAGEN_LOAD-((int)unirand(0.0,3.0,stream));   //12 - 14
              skaut Ãą hverjum vagni
267       for (i=1; i <= vagn_magn; i++) {
268
269     transfer[3]=1.0;
270     transfer[4] = sim_time + (i * 0.01); // skaut entering system time
271     transfer[6] = (float) skaut_id++;
272     //printf("tr4 in wagen: %f\n", transfer[4]);
273     event_schedule( sim_time + ( i* 0.01),  EVENT_SKAUT_ARRIVAL);
274       }
275
276       event_schedule(sim_time+wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
277 }
278
279
280 void skaut_arrival()
281 {
282       push_array();
283       int current_unit = (int)transfer[3];
284       int i;
285
286       for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if there is a
              broken machine before current one
287
288     if (machine_broken[i] > 0.0) {
289         if ((list_size[1+number_of_machines + current_unit] < queue_size[1+
                current_unit]) || queue_size[1+current_unit] == 0) {  // if current
                machine is broken then delay it.x
290       event_schedule(PREP_TIME+sim_time +  machine_broken[i] + work_time[
              current_unit], EVENT_SKAUT_ARRIVAL); //also if next queue is full then
              delay it.
291       return;
292         }
293     }
294       }
295
296       // check if machine is not busy
297       if (list_size[current_unit] == 0 && machine_broken[current_unit] == 0.0) {
298     sampst(0.0, sampst_delays);
299     sampst(0.0, current_unit);
300
301     list_file(FIRST, current_unit); // last := first here because there are only
              to be 0 or 1 items in machine
302
303     // schedule departure after machine processing time
304     pop_array();
305     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
              EVENT_SKAUT_DEPARTURE);
306       } else {
307
308     if (list_size[number_of_machines + current_unit] == queue_size[current_unit])
              {
309
310         event_schedule(PREP_TIME + sim_time + work_time[current_unit],
                EVENT_SKAUT_ARRIVAL); //also if queue is full then delay it.
311
312     } else {
313         transfer[5] = sim_time;
314         list_file(LAST, number_of_machines + current_unit);
```

```
315         if(list_size[current_unit] > queue_max_lengths[number_of_machines +
                current_unit]) {
316       queue_max_lengths[current_unit] = list_size[number_of_machines +
              current_unit];
317         }
318     }
319
320       }
321
322 }
323
324 void skaut_departure()
325 {
326       push_array();
327       int current_unit = (int) transfer[3];
328       int i = 0;
329       for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if machine is
              broken or there is a broken machine before current one
330
331     if (machine_broken[i] > 0.0) {
332         if ((i == current_unit)  || (list_size[1+number_of_machines +
                current_unit] < queue_size[1+current_unit])) {  // if current machine
                 is broken then delay it.
333       event_schedule(PREP_TIME+sim_time + machine_broken[i],
              EVENT_SKAUT_DEPARTURE); //also if next queue is full then delay it.
334       return;
335         }
336 //        printf("Size of next queue %d, limit of next queue %d\n",list_size[1+
      number_of_machines + current_unit], queue_size[1+current_unit]);
337         break;
338     }
339       }
340
341       if (current_unit == MACHINES_ON_THE_LEFT_SIDE) {
342     skaut_throughput += 2;
343     sampst(sim_time - transfer[4], throughput_time);
344     list_remove(FIRST,current_unit);
345       } else {
346     list_remove(FIRST,current_unit);
347     pop_array();
348     transfer[3]++;
349     event_schedule(PREP_TIME + sim_time + transfer_time[(int)(transfer[3])-1],
              EVENT_SKAUT_ARRIVAL);
350       }
351
352
353       if (list_size[number_of_machines + current_unit] != 0) {
354     pop_array();
355
356     list_file(FIRST,current_unit); // first equals last because size should only
              be 1
357     pop_array();
358
359     list_remove(FIRST, number_of_machines + current_unit);
360     pop_array();
361
362     sampst(sim_time - transfer[5], sampst_delays);
363     sampst(sim_time - transfer[5], current_unit);
364     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
              EVENT_SKAUT_DEPARTURE);
365       }
366 }
367
368
369 void parse_input(char inputfile_data[], char inputfile_time[])
370 {
371
372
373       if ((infile = fopen (inputfile_data, "r")) == NULL) {
374     printf("Could not open file %s\n",inputfile_data);
```

```
375     }
376
377     fscanf (infile, "%d %d %d %d %f %f %f %f %f %f", &number_of_machines, &
            min_productivity, &min_no_failures, &max_no_failures, &
            mean_wagen_arrival, &std_wagen_arrival,  &min_machine_repair_time, &
            max_machine_repair_time, &end_warmup_time, &end_simulation_time);
378     fclose(infile);
379
380
381     if ((infile = fopen (inputfile_time, "r")) == NULL) {
382   printf("Could not open file %s\n",inputfile_time);
383     }
384     printf( "%d %d %d %d %f %f %f %f %f %f\n", number_of_machines,
            min_productivity, min_no_failures, max_no_failures, mean_wagen_arrival,
             std_wagen_arrival,  min_machine_repair_time, max_machine_repair_time,
            end_warmup_time, end_simulation_time);
385
386     int counter = 1;
387     while (!feof(infile)) {
388   fscanf(infile, "%f %d %f", &transfer_time[counter], &queue_size[counter], &
          work_time[counter] );
389   printf("%f %d %f\n", transfer_time[counter], queue_size[counter], work_time[
          counter] );
390   counter++;
391     }
392     fclose(infile);
393
394 }
395
396 void end_warmup()
397 {
398     sampst(0.0, 0);
399     timest(0.0, 0);
400     skaut_throughput = 0;
401 }
402
403 void report()
404 {
405
406     int i;
407     float total_downtime = 0.0;
408     printf("\n*****************************************************\n");
409     printf("Report for %d failures per day\n",failure_nr);
410
411     for (i=0; i <NUM_MACHINES; i++) {
412   printf("--Breakdown in machine nr %d--\n", i+1);
413   printf("Number of fails\t Downtime \t\n");
414   printf("\t %d\t", fail_list[i].machine_nr);
415   printf("%.3f sec / %.3f min\t", fail_list[i].downtime,fail_list[i].downtime
          /60.0);
416   printf("\n");
417   total_downtime+=fail_list[i].downtime;
418     }
419     printf("\n\n");
420
421
422     printf("Total downtime was %.3lf seconds or %.3lf minutes\n",total_downtime
          , total_downtime/60.0);
423
424     printf("----------------\nMachine load\n----------------\n");
425     for (i=1; i <= number_of_machines; i++) {
426   printf("Machine %d\t", i);
427     }
428     printf("\n");
429     for (i=1; i <= number_of_machines; i++) {
430   printf("%f\t", filest(i) );
431     }
432     printf("\n\n");
433
```

```
434     printf("-------------------------\nAverage delay in queues\n
            -------------------------\n");
435     for (i=1; i <= number_of_machines; i++) {
436   printf("Queue %d \t", i);
437     }
438
439     printf("\n");
440
441     for (i=1; i <= number_of_machines; i++) {
442   printf("%f\t", sampst(0.0, -i));
443     }
444     printf("\n\n");
445     printf("Average queue delay: %f\n", sampst(0.0, -sampst_delays));
446
447     printf("Worst case queue delay: %f\n", transfer[3]);
448     printf("Best case queue delay: %f\n", transfer[4]);
449
450     printf("System throughput: %d\n", skaut_throughput );
451     printf("Average throughput time: %f\n", sampst(0.0, -throughput_time));
452     printf("Min throughput time: %f\n", transfer[4]);
453     printf("Random seed: %d\n\n", stream);
454
455     int l;
456     int sum_q_lenths =0;
457     int number_of_queues =0;
458     for (l = 1; l <= number_of_machines; l++) {
459   if (queue_size[l] < 1) continue;
460   printf("Maximum length of queue %d: %d\n", l, queue_max_lengths[l]);
461   sum_q_lenths += queue_max_lengths[l];
462   number_of_queues++;
463     }
464
465     printf("Average maximum length of queues: %f\n\n", (float) sum_q_lenths / (
          float) number_of_queues);
466 }
467
468 void push_array() {
469
470     memcpy(temp_transfer,transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
471 }
472
473 void pop_array() {
474     memcpy(transfer,temp_transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
475 }
476
477 void create_machine_fail_events() {
478     int i;
479     float a[20],shift_length;
480     shift_length = (float)SHIFT_LENGTH;
481     int n = failure_nr;
482     memset(a,0,20*sizeof(float));
483     float span = shift_length / (float)n+1.0;  //max time between machine
          failures
484     float current_span = 0.0;
485     int machine;
486     float repair_time ;
487     float breakdown_time;
488
489     for (i = 0;i<n;i++) {
490
491   current_span+=span;
492   machine = (int)unirand(1,number_of_machines+1,stream);
493   breakdown_time = unirand(0.0,current_span,stream);
494   repair_time =(min_machine_repair_time + expon(max_machine_repair_time ,stream
          ))*60.0;
495   if (a[machine]<breakdown_time) {  //
496       a[machine] = breakdown_time+repair_time;
497   }
498   else { // if breakdown_time clashes with the same machine then let the
          breakdown happen after the machine goes up again
```

```
499         breakdown_time = a[machine] + 1.0;
500         a[machine] = breakdown_time+repair_time;
501     }
502     transfer[3] = repair_time;
503     transfer[4] = (float)machine;
504     fail_list[machine-1].downtime+= repair_time;
505     fail_list[machine-1].machine_nr++;
506     event_schedule(sim_time + breakdown_time, EVENT_MACHINE_FAILURE );
507       }
508
509       event_schedule(sim_time + shift_length, EVENT_GENERATE_FAILURES );
510 }
511
512 void machine_failure(){
513       float repair_time = transfer[3];
514       int   machine     = (int)transfer[4];
515       machine_broken[machine] = repair_time;
516       //     printf(" Machine %d broke down and it takes %f to repair\n", machine,
              repair_time/60.0);
517
518       event_schedule(sim_time + repair_time, EVENT_MACHINE_FIXED);
519 }
520
521 void machine_fixed(){
522
523       int   machine     = (int)transfer[4];
524       machine_broken[machine] = 0.0;
525 }
```

5.2 Inntaksgögn líkans

```
1 7 404 0 10    30.0  2.0    5.0    13.9  1000.0      5185000.0
2 num  min    min   max     mean     std    min    mean     warmup     simulationtime
3 prod-     no  no  wagen     wagen    repair  repair input
4 uction    fail-   fail-   arrival  arrival time    time    for expon
5 ures    ures                  distribution
```

```
1 129.83   14   70.0
2 122.82   17  21.79
3 18.98  0  12.70
4 22.74  1  67.41
5 0.1  1  69.75
6 0.1  0  81.85
7 0.1  2  70.33
```

5.3 Keyrsluskýrslur

```
1  7 404 3 10 30.000000 2.000000 5.000000 180.000000 1000.000000 5185000.000000
2  129.830002 14 70.000000
3  122.820000 17 21.790001
4  18.980000 0 12.700000
5  22.740000 1 67.410004
6  0.100000 1 69.750000
7  0.100000 0 81.849998
8  0.100000 2 70.330002
9  0.000000 0 0.000000
10 DGB 866.538452
11
12 **************************************************
13 Report for 3 failures per day
14 --Breakdown nr 1--
15 Number of fails   Downtime
16     36 21840.373 sec / 364.006 min
17 --Breakdown nr 2--
18 Number of fails   Downtime
```

```
    33 17649.326 sec / 294.155 min
--Breakdown nr 3--
Number of fails  Downtime
    44 26668.895 sec / 444.482 min
--Breakdown nr 4--
Number of fails  Downtime
    39 25308.217 sec / 421.804 min
--Breakdown nr 5--
Number of fails  Downtime
    46 26816.061 sec / 446.934 min
--Breakdown nr 6--
Number of fails  Downtime
    41 25011.408 sec / 416.857 min
--Breakdown nr 7--
Number of fails  Downtime
    31 18755.475 sec / 312.591 min


Total downtime was 162049.750 seconds or 2700.829 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.654318  0.296142  0.236139  0.628465  0.636727  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
560.780135  0.000000  0.000000  0.584707  0.693674  0.000000  0.000000

Average queue delay: 112.403735
Worst case queue delay: 3180.820822
Best case queue delay: 0.000000

System throughput: 73136
Average throughput time: 1304.970431
Min throughput time: 716.020005
Random seed: 61

Maximum length of queue 1: 12
Maximum length of queue 2: 0
Maximum length of queue 4: 1
Maximum length of queue 5: 0
Maximum length of queue 7: 0
Average maximum length of queues: 2.600000


**************************************************
Report for 4 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    54 33346.902 sec / 555.782 min
--Breakdown nr 2--
Number of fails  Downtime
    62 39069.430 sec / 651.157 min
--Breakdown nr 3--
Number of fails  Downtime
    47 28488.143 sec / 474.802 min
--Breakdown nr 4--
Number of fails  Downtime
    47 27977.752 sec / 466.296 min
--Breakdown nr 5--
Number of fails  Downtime
    59 36390.691 sec / 606.512 min
--Breakdown nr 6--
Number of fails  Downtime
    50 27826.713 sec / 463.779 min
--Breakdown nr 7--
Number of fails  Downtime
```

```
    41 25404.180 sec / 423.403 min


Total downtime was 218503.812 seconds or 3641.730 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.653198  0.292869  0.233496  0.622502  0.628993  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
566.252019  0.000000  0.000000  0.810736  0.970351  0.000000  0.000000

Average queue delay: 113.606621
Worst case queue delay: 3930.499258
Best case queue delay: 0.000000

System throughput: 72046
Average throughput time: 1321.957250
Min throughput time: 716.020005
Random seed: 62

Maximum length of queue 1: 11
Maximum length of queue 2: 0
Maximum length of queue 4: 1
Maximum length of queue 5: 0
Maximum length of queue 7: 0
Average maximum length of queues: 2.400000


**************************************************
Report for 5 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    70 40567.773 sec / 676.130 min
--Breakdown nr 2--
Number of fails  Downtime
    59 39782.234 sec / 663.037 min
--Breakdown nr 3--
Number of fails  Downtime
    78 45799.273 sec / 763.321 min
--Breakdown nr 4--
Number of fails  Downtime
    64 39138.762 sec / 652.313 min
--Breakdown nr 5--
Number of fails  Downtime
    65 37173.656 sec / 619.561 min
--Breakdown nr 6--
Number of fails  Downtime
    62 39064.641 sec / 651.077 min
--Breakdown nr 7--
Number of fails  Downtime
    52 26977.840 sec / 449.631 min


Total downtime was 268504.188 seconds or 4475.070 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.655376  0.290337  0.233495  0.618331  0.623694  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
573.853878  0.002043  0.000000  0.962555  1.069832  0.000000  0.000000
```

```
159
160 Average queue delay: 115.166766
161 Worst case queue delay: 4647.825765
162 Best case queue delay: 0.000000
163
164 System throughput: 71474
165 Average throughput time: 1339.667943
166 Min throughput time: 716.020005
167 Random seed: 62
168
169 Maximum length of queue 1: 12
170 Maximum length of queue 2: 0
171 Maximum length of queue 4: 1
172 Maximum length of queue 5: 0
173 Maximum length of queue 7: 0
174 Average maximum length of queues: 2.600000
175
176
177 ***************************************************
178 Report for 6 failures per day
179 --Breakdown nr 1--
180 Number of fails  Downtime
181    80 49088.027 sec / 818.134 min
182 --Breakdown nr 2--
183 Number of fails  Downtime
184    82 50090.105 sec / 834.835 min
185 --Breakdown nr 3--
186 Number of fails  Downtime
187    100  60332.621 sec / 1005.544 min
188 --Breakdown nr 4--
189 Number of fails  Downtime
190    68 42041.410 sec / 700.690 min
191 --Breakdown nr 5--
192 Number of fails  Downtime
193    68 39728.184 sec / 662.136 min
194 --Breakdown nr 6--
195 Number of fails  Downtime
196    74 44091.766 sec / 734.863 min
197 --Breakdown nr 7--
198 Number of fails  Downtime
199    68 41973.270 sec / 699.554 min
200
201
202 Total downtime was 327345.406 seconds or 5455.757 minutes
203 ----------------
204 Machine load
205 ----------------
206 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
207 0.656092  0.288621  0.233357  0.618043  0.620428  0.000000  0.000000
208
209 ------------------------
210 Average delay in queues
211 ------------------------
212 Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
213 582.002567  0.001562  0.000000  1.160263  1.253938  0.000000  0.000000
214
215 Average queue delay: 116.875824
216 Worst case queue delay: 4292.338857
217 Best case queue delay: 0.000000
218
219 System throughput: 71050
220 Average throughput time: 1359.820622
221 Min throughput time: 716.020005
222 Random seed: 62
223
224 Maximum length of queue 1: 12
225 Maximum length of queue 2: 0
226 Maximum length of queue 4: 1
227 Maximum length of queue 5: 0
228 Maximum length of queue 7: 0
```

```
Average maximum length of queues: 2.600000


**************************************************
Report for 7 failures per day
--Breakdown nr 1--
Number of fails  Downtime
   96 59138.098 sec / 985.635 min
--Breakdown nr 2--
Number of fails  Downtime
   88 52223.656 sec / 870.394 min
--Breakdown nr 3--
Number of fails  Downtime
   102  61826.266 sec / 1030.438 min
--Breakdown nr 4--
Number of fails  Downtime
   84 55065.039 sec / 917.751 min
--Breakdown nr 5--
Number of fails  Downtime
   89 50824.715 sec / 847.079 min
--Breakdown nr 6--
Number of fails  Downtime
   85 50948.812 sec / 849.147 min
--Breakdown nr 7--
Number of fails  Downtime
   86 46907.703 sec / 781.795 min


Total downtime was 376934.312 seconds or 6282.239 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.656857  0.285277  0.235948  0.614177  0.613190  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
586.436708  0.001679  0.000000  1.274650  1.606672  0.000000  0.000000

Average queue delay: 117.865947
Worst case queue delay: 3295.661001
Best case queue delay: 0.000000

System throughput: 70196
Average throughput time: 1373.005641
Min throughput time: 716.020005
Random seed: 63

Maximum length of queue 1: 11
Maximum length of queue 2: 0
Maximum length of queue 4: 1
Maximum length of queue 5: 0
Maximum length of queue 7: 0
Average maximum length of queues: 2.400000


**************************************************
Report for 8 failures per day
--Breakdown nr 1--
Number of fails  Downtime
   96 54253.984 sec / 904.233 min
--Breakdown nr 2--
Number of fails  Downtime
   114  72605.875 sec / 1210.098 min
--Breakdown nr 3--
Number of fails  Downtime
   103  60780.863 sec / 1013.014 min
--Breakdown nr 4--
```

```
Number of fails  Downtime
    105  59921.996 sec / 998.700 min
--Breakdown nr 5--
Number of fails  Downtime
    107  67782.305 sec / 1129.705 min
--Breakdown nr 6--
Number of fails  Downtime
    99 66643.648 sec / 1110.727 min
--Breakdown nr 7--
Number of fails  Downtime
    96 63453.926 sec / 1057.565 min


Total downtime was 445442.625 seconds or 7424.044 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.657351  0.283620  0.233861  0.614706  0.607624  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
591.317676  0.004835  0.000000  1.505407  1.647830  0.000000  0.000000

Average queue delay: 118.895150
Worst case queue delay: 4642.863577
Best case queue delay: 0.000000

System throughput: 69388
Average throughput time: 1392.042472
Min throughput time: 716.020005
Random seed: 63

Maximum length of queue 1: 11
Maximum length of queue 2: 0
Maximum length of queue 4: 1
Maximum length of queue 5: 0
Maximum length of queue 7: 0
Average maximum length of queues: 2.400000


**************************************************
Report for 9 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    120  71899.633 sec / 1198.327 min
--Breakdown nr 2--
Number of fails  Downtime
    103  61880.219 sec / 1031.337 min
--Breakdown nr 3--
Number of fails  Downtime
    116  74133.500 sec / 1235.558 min
--Breakdown nr 4--
Number of fails  Downtime
    109  64476.109 sec / 1074.602 min
--Breakdown nr 5--
Number of fails  Downtime
    120  72168.859 sec / 1202.814 min
--Breakdown nr 6--
Number of fails  Downtime
    128  81744.359 sec / 1362.406 min
--Breakdown nr 7--
Number of fails  Downtime
    114  65036.109 sec / 1083.935 min


Total downtime was 491338.812 seconds or 8188.980 minutes
---------------
```

```
Machine load
-----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.658101   0.279506   0.233829   0.609962   0.601260   0.000000   0.000000

--------------------------
Average delay in queues
--------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
599.193115   0.006786   0.000000   1.856088   1.921827   0.000000   0.000000

Average queue delay: 120.594190
Worst case queue delay: 5445.983511
Best case queue delay: 0.000000

System throughput: 68610
Average throughput time: 1402.723005
Min throughput time: 716.020005
Random seed: 63

Maximum length of queue 1: 11
Maximum length of queue 2: 0
Maximum length of queue 4: 1
Maximum length of queue 5: 0
Maximum length of queue 7: 0
Average maximum length of queues: 2.400000


**************************************************
Report for 10 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    142  85696.336 sec / 1428.272 min
--Breakdown nr 2--
Number of fails  Downtime
    112  66941.117 sec / 1115.685 min
--Breakdown nr 3--
Number of fails  Downtime
    130  78611.453 sec / 1310.191 min
--Breakdown nr 4--
Number of fails  Downtime
    143  88534.219 sec / 1475.570 min
--Breakdown nr 5--
Number of fails  Downtime
    124  65197.660 sec / 1086.628 min
--Breakdown nr 6--
Number of fails  Downtime
    125  75260.164 sec / 1254.336 min
--Breakdown nr 7--
Number of fails  Downtime
    124  70163.836 sec / 1169.397 min


Total downtime was 530404.750 seconds or 8840.079 minutes
-----------------
Machine load
-----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.659115   0.278222   0.230390   0.609487   0.597991   0.000000   0.000000

--------------------------
Average delay in queues
--------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
601.889850   0.007963   0.000000   2.080003   2.004042   0.000000   0.000000

Average queue delay: 121.184429
Worst case queue delay: 6167.243765
Best case queue delay: 0.000000

```

```
System throughput: 68266
Average throughput time: 1422.184285
Min throughput time: 716.020005
Random seed: 64

Maximum length of queue 1: 12
Maximum length of queue 2: 0
Maximum length of queue 4: 1
Maximum length of queue 5: 0
Maximum length of queue 7: 0
Average maximum length of queues: 2.600000
```