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.

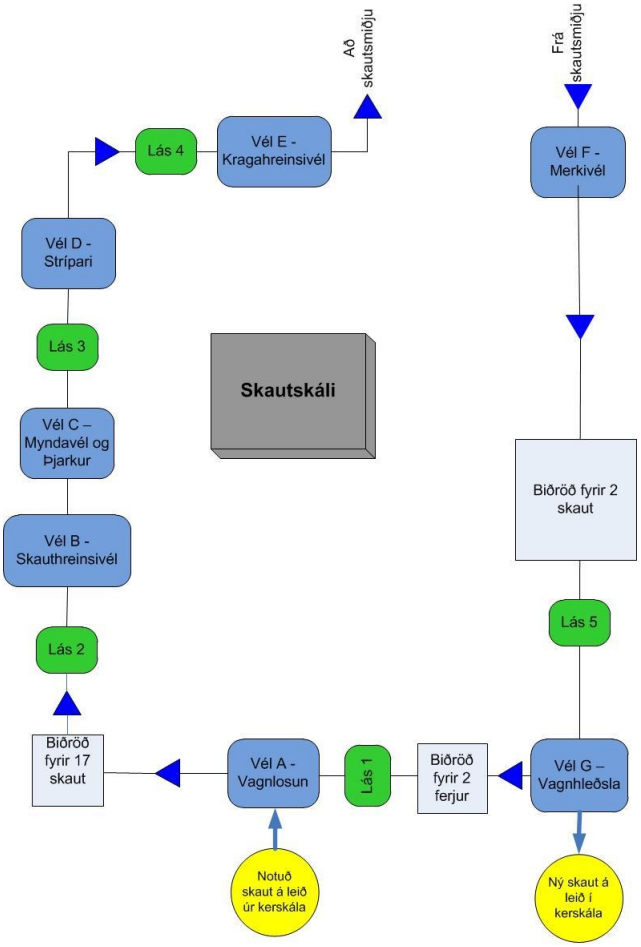
Skautið er tekið inn í ferlið þannig að það er hengt á ferju sem að er dregin áfram af keðju, sem að fer í gegnum allan skautskálann og inn í skautsmiðjuna og til baka. Ferlið er raðgengt svo ef vél er að afgreiða skaut þarf skautið á eftir að bíða þangað til að vélin hefur lokið sér af. Til að stýra 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 á sólarhring**
A	0.517888	*A*	497.663670
B	0.140782	*B*	0.004016
C	0.089724	*C*	*-engin röð-*
D	0.467316	*D*	1.119780
E	0.448905	*E*	0.971277

Kvarði	**Tími í sekúndum á sólarhring**
Meðalbið í röðum í sekúndum	99.952977
Versta biðtilfelli skauts í gegnum kerfi	9499.287679
Besta biðtilfelli skauts í gegnum kerfi	0.0
Meðal tími skauts í gegnum kerfi	1118.785573
Minnsti tími skauts í gegnum kerfi	536.020005

Fjöldi skauta í gegnum kerfið á þessu 90 daga tímabili er 65082 skaut, sem eru 45 skaut á klukkustund. Sjá keyrsluskýrslu í grein 5.3. 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

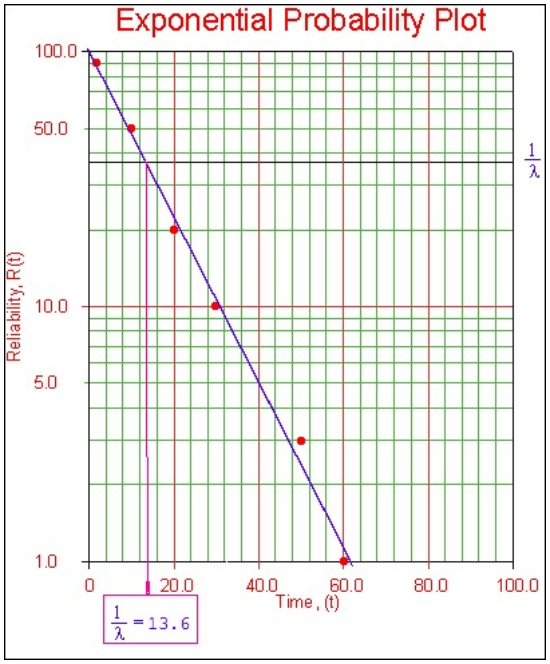
Til að komast að niðurstöðum smíðuðum við líkan sem að hermir eftir ferli skautskála. Í næstu undirgreinum er forsendum líkansins lýst.

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.



Mynd 2: Inntak fyrir veldisdreififall miðað við núverandi lengd viðgerða

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 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

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

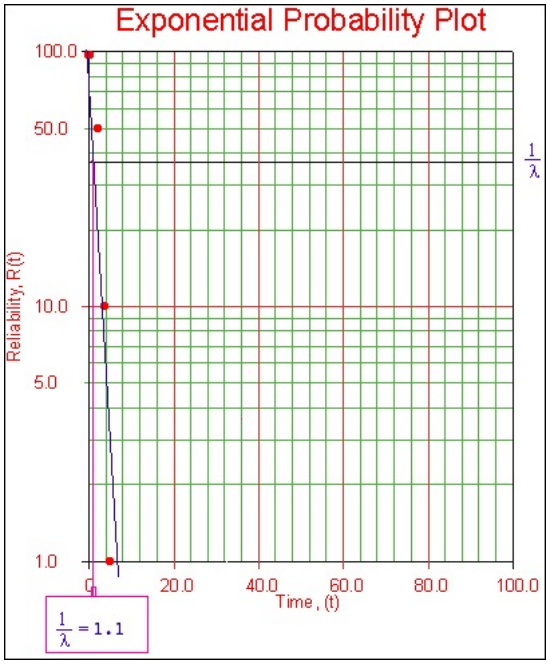
$$R(t) = e^{(-\lambda \cdot \frac{1}{\lambda})} \text{ því að t = m } 1/\text{lambda}$$

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

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

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: Inntak fyrir veldisdreififall ef hámarksviðgerð hverrar vélar er 10 mínútur

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.

3.3 Bilanir

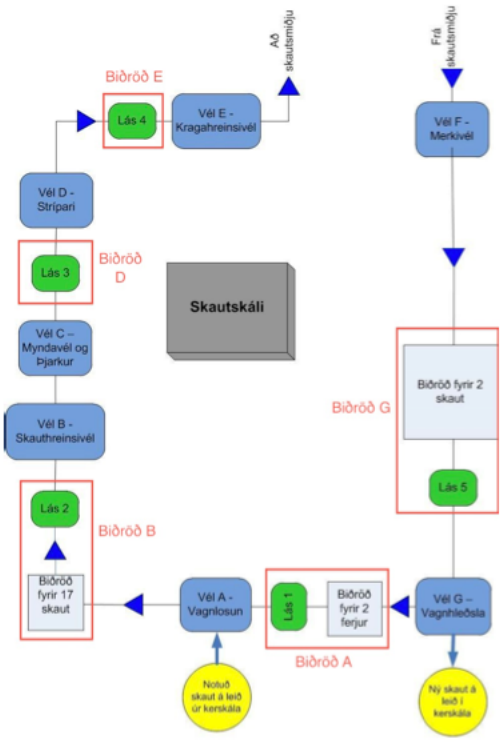
Samkvæmt verkefnislýsingunni [1] er það nokkuð slembið hvaða vélar bila, að vél F undanskilinni, og að bilanir eiga sér stað nokkrum sinnum á sólarhring. Gildið á tölunni *nokkrum sinnum* er illa skilgreint en höfundar sammæltust um töluna 8. Fyrir flestar bilanir er viðgerðartíminn 5 til 30 mínútur. 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



Mynd 4: Einingar líkans

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. 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
- Búa til bilanir

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.

Búa til bilanir

Á hverjum degi er settur af stað atburður sem býr til *Vél bilar* byggða á inntaksskrá. Þegar hann er ræstur býr hann til bilanir fyrir hvern dag sem eru byggðar á eftirfarandi gögnum.

- Fjöldi véla.
- Lengd dags.
- Fjölda bilanna þá vakt.
- Inntak í veldisdreifingarfall fyrir lengd bilanna.

Þessi atburður býr til nokkra *Vél bilar* atburði, ásamt því að senda í atburðinn upplýsingar um viðgerðartíma/downtime og vélanúmer. Þegar þessu er lokið þá býr hann til nýjan *Búa til bilanir* atburð sem verður ræstur í byrjun næsta dags.

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

```
1  /*
2   *  isal.c
3   *
4   *
5   *  Created by Gunnarr Baldursson & Ragnar Gisli Olafsson on 4/18/11.
6   *  Copyright 2011 Haskoli Islands. All rights reserved.
7   *
8   */
9
10 #include <stdio.h>
11 #include <string.h>
12 #include <stdlib.h>
13 #include <math.h>
14 #include <time.h>
15 #include "simlib/rndlib.h"
16 #include "simlib/simlib.h"
17
18 // EVENTS
19 #define EVENT_WAGEN_UNLOAD_ARRIVAL   1
20 #define EVENT_WAGEN_UNLOAD_DEPARTURE 2
21 #define EVENT_SKAUT_ARRIVAL 3
22 #define EVENT_SKAUT_DEPARTURE 4
23 #define EVENT_MACHINE_FAILURE 5
24 #define EVENT_MACHINE_FIXED 6
25 #define EVENT_END_SIMULATION 7
26 #define EVENT_END_WARMUP   8
27 #define EVENT_GENERATE_FAILURES 9
28
29 // STREAMS
30 #define STREAM_WAGEN_ARRIVAL 1
31
32 //Other constants
33 #define NUM_MACHINES 7
```

```
34 #define SHIFT_LENGTH 57600.0;
35 #define WAGEN_LOAD 14
36 #define MACHINES_ON_THE_LEFT_SIDE 5
37 #define MACHINES_ON_THE_RIGHT_SIDE 2
38 #define OPTIMAL_THROUGHPUT 52
39 #define ACTUAL_THROUGHPUT 40
40 #define TRANSFER_ARRAY_LENGTH 11
41 #define PREP_TIME 0.0
42
43 typedef struct
44 {
45     float failtime;
46     float downtime;
47     int machine_nr;
48 } breakdown;
49
50
51 //#define LOADING_TIME_PER_SKAUT
52
53 // Global variables
54 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     for (failure_nr = min_no_failures; failure_nr<= max_no_failures; failure_nr
          ++) {
165   stream = (unsigned int)time(NULL) % 100;
```

```
166
167     memset( is_machine_busy,0, NUM_MACHINES +1 );
168     memset( machine_broken,0, NUM_MACHINES +1);
169     memset( queue_max_lengths,0, NUM_MACHINES +1);
170     memset( fail_list,0, sizeof(breakdown)*(NUM_MACHINES+1));
171     fail_index = 0;
172     skaut_throughput = 0;
173     sampst_delays = number_of_machines +1;
174     throughput_time = number_of_machines +2;
175
176     skaut_id = 1;
177     skaut_throughput = 0;
178
179
180     // Initialize rndlib
181     init_twister();
182
183     // Initialize simlib
184     init_simlib();
185
186     maxatr = 6; // how many attributes do we need?
187
188     /* Schedule first wagen arrival */
189     event_schedule( 10.0, EVENT_WAGEN_UNLOAD_ARRIVAL );
190
191     /* Schedule end of warmup time */
192     event_schedule( end_warmup_time, EVENT_END_WARMUP );
193     event_schedule(end_warmup_time, EVENT_GENERATE_FAILURES );
194     /* Schedule simulation termination */
195     event_schedule(end_simulation_time , EVENT_END_SIMULATION );
196
197     next_event_type = 0;
198
199
200     while (next_event_type != EVENT_END_SIMULATION) {
201
202         timing();
203
204         switch (next_event_type) {
205         case EVENT_WAGEN_UNLOAD_ARRIVAL:
206       wagen_unload_arrival();
207       break;
208         case EVENT_SKAUT_ARRIVAL:
209       skaut_arrival();
210       break;
211         case EVENT_SKAUT_DEPARTURE:
212       skaut_departure();
213       break;
214         case EVENT_MACHINE_FAILURE:
215       machine_failure();
216       break;
217         case EVENT_MACHINE_FIXED:
218       machine_fixed();
219       break;
220         case EVENT_END_WARMUP:
221       end_warmup();
222       break;
223         case EVENT_END_SIMULATION:
224       report();
225       break;
226         case EVENT_GENERATE_FAILURES:
227       create_machine_fail_events();
228       break;
229
230         }
231     }
232
233       }
234 }
235
```

```
236
237
238 void wagen_unload_arrival()
239 {
240
241     int i;
242     int current_unit = 0;
243     float wagen_arrival_zeit = unirand((mean_wagen_arrival-std_wagen_arrival)
            *60.0,(mean_wagen_arrival+std_wagen_arrival)*60.0,stream);
244
245     for (i = 1; i<NUM_MACHINES+1; i++) {  //delay unload of skaut by the time
            it takes to repair
246   if (machine_broken[i] > 0.0) {
247       event_schedule(sim_time + machine_broken[i], EVENT_WAGEN_UNLOAD_ARRIVAL);
248       return;
249   }
250     }
251
252     if (list_size[number_of_machines + 1] != 0) { // ef allt er enn fullt Ã¿Ãą
            koma meÃř nÃęsta vagn eftir uÃ¿b hÃąlftÃŋma
253   event_schedule(sim_time + wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
254   return;
255     }
256
257     int vagn_magn =  WAGEN_LOAD-((int)unirand(0.0,3.0,stream));   //12 - 14
            skaut Ãą hverjum vagni
258     for (i=1; i <= vagn_magn; i++) {
259
260   transfer[3]=1.0;
261   transfer[4] = sim_time + (i * 0.01); // skaut entering system time
262   transfer[6] = (float) skaut_id++;
263   //printf("tr4 in wagen: %f\n", transfer[4]);
264   event_schedule( sim_time + ( i* 0.01),  EVENT_SKAUT_ARRIVAL);
265     }
266
267     event_schedule(sim_time+wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
268 }
269
270
271 void skaut_arrival()
272 {
273     push_array();
274     int current_unit = (int)transfer[3];
275     int i;
276
277     for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if there is a
            broken machine before current one
278
279   if (machine_broken[i] > 0.0) {
280       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
281     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.
282     return;
283       }
284   }
285     }
286
287     // check if machine is not busy
288     if (list_size[current_unit] == 0 && machine_broken[current_unit] == 0.0) {
289   sampst(0.0, sampst_delays);
290   sampst(0.0, current_unit);
291
292   list_file(FIRST, current_unit); // last := first here because there are only
          to be 0 or 1 items in machine
293
294   // schedule departure after machine processing time
295   pop_array();
```

```
296     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
            EVENT_SKAUT_DEPARTURE);
297       } else {
298
299     if (list_size[number_of_machines + current_unit] == queue_size[current_unit])
            {
300
301         event_schedule(PREP_TIME + sim_time + work_time[current_unit],
                EVENT_SKAUT_ARRIVAL); //also if queue is full then delay it.
302
303     } else {
304         transfer[5] = sim_time;
305         list_file(LAST, number_of_machines + current_unit);
306         if(list_size[current_unit] > queue_max_lengths[number_of_machines +
                current_unit]) {
307       queue_max_lengths[current_unit] = list_size[number_of_machines +
              current_unit];
308         }
309     }
310
311       }
312
313 }
314
315 void skaut_departure()
316 {
317       push_array();
318       int current_unit = (int) transfer[3];
319       int i = 0;
320       for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if machine is
              broken or there is a broken machine before current one
321
322     if (machine_broken[i] > 0.0) {
323         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.
324       event_schedule(PREP_TIME+sim_time + machine_broken[i],
              EVENT_SKAUT_DEPARTURE); //also if next queue is full then delay it.
325       return;
326         }
327 //        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]);
328       break;
329     }
330       }
331
332       if (current_unit == MACHINES_ON_THE_LEFT_SIDE) {
333     skaut_throughput += 2;
334     sampst(sim_time - transfer[4], throughput_time);
335     list_remove(FIRST,current_unit);
336       } else {
337     list_remove(FIRST,current_unit);
338     pop_array();
339     transfer[3]++;
340     event_schedule(PREP_TIME + sim_time + transfer_time[(int)(transfer[3])-1],
            EVENT_SKAUT_ARRIVAL);
341       }
342
343
344       if (list_size[number_of_machines + current_unit] != 0) {
345     pop_array();
346
347     list_file(FIRST,current_unit); // first equals last because size should only
            be 1
348     pop_array();
349
350     list_remove(FIRST, number_of_machines + current_unit);
351     pop_array();
352
353     sampst(sim_time - transfer[5], sampst_delays);
```

```
354     sampst(sim_time - transfer[5], current_unit);
355     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
            EVENT_SKAUT_DEPARTURE);
356       }
357 }
358
359
360 void parse_input(char inputfile_data[], char inputfile_time[])
361 {
362
363
364       if ((infile = fopen (inputfile_data, "r")) == NULL) {
365     printf("Could not open file %s\n",inputfile_data);
366       }
367
368       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);
369       fclose(infile);
370
371
372       if ((infile = fopen (inputfile_time, "r")) == NULL) {
373     printf("Could not open file %s\n",inputfile_time);
374       }
375       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);
376
377       int counter = 1;
378       while (!feof(infile)) {
379     fscanf(infile, "%f %d %f", &transfer_time[counter], &queue_size[counter], &
            work_time[counter] );
380     printf("%f %d %f\n", transfer_time[counter], queue_size[counter], work_time[
            counter] );
381     counter++;
382       }
383       fclose(infile);
384
385 }
386
387 void end_warmup()
388 {
389       sampst(0.0, 0);
390       timest(0.0, 0);
391       skaut_throughput = 0;
392 }
393
394 void report()
395 {
396
397       int i;
398       float total_downtime = 0.0;
399       printf("\n**************************************************\n");
400       printf("Report for %d failures per day\n",failure_nr);
401
402       for (i=0; i <NUM_MACHINES; i++) {
403     printf("--Breakdown in machine nr %d--\n", i+1);
404     printf("Number of fails\t Downtime \t\n");
405     printf("\t %d\t", fail_list[i].machine_nr);
406     printf("%.3f sec / %.3f min\t", fail_list[i].downtime,fail_list[i].downtime
            /60.0);
407     printf("\n");
408     total_downtime+=fail_list[i].downtime;
409       }
410       printf("\n\n");
411
412
```

```
413     printf("Total downtime was %.3lf seconds or %.3lf minutes\n",total_downtime
            , total_downtime/60.0);
414
415     printf("----------------\nMachine load\n----------------\n");
416     for (i=1; i <= number_of_machines; i++) {
417   printf("Machine %d\t", i);
418     }
419     printf("\n");
420     for (i=1; i <= number_of_machines; i++) {
421   printf("%f\t", filest(i) );
422     }
423     printf("\n\n");
424
425     printf("-------------------------\nAverage delay in queues\n
            -------------------------\n");
426     for (i=1; i <= number_of_machines; i++) {
427   printf("Queue %d \t", i);
428     }
429
430     printf("\n");
431
432     for (i=1; i <= number_of_machines; i++) {
433   printf("%f\t", sampst(0.0, -i));
434     }
435     printf("\n\n");
436     printf("Average queue delay: %f\n", sampst(0.0, -sampst_delays));
437
438     printf("Worst case queue delay: %f\n", transfer[3]);
439     printf("Best case queue delay: %f\n", transfer[4]);
440
441     printf("System throughput: %d\n", skaut_throughput );
442     printf("Average throughput time: %f\n", sampst(0.0, -throughput_time));
443     printf("Min throughput time: %f\n", transfer[4]);
444     printf("Random seed: %d\n\n", stream);
445
446     int l;
447     int sum_q_lenths =0;
448     int number_of_queues =0;
449     for (l = 1; l <= number_of_machines; l++) {
450   if (queue_size[l] < 1) continue;
451   printf("Maximum length of queue %d: %d\n", l, queue_max_lengths[l]);
452   sum_q_lenths += queue_max_lengths[l];
453   number_of_queues++;
454     }
455
456     printf("Average maximum length of queues: %f\n\n", (float) sum_q_lenths / (
            float) number_of_queues);
457 }
458
459 void push_array() {
460
461     memcpy(temp_transfer,transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
462 }
463
464 void pop_array() {
465     memcpy(transfer,temp_transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
466 }
467
468 void create_machine_fail_events() {
469     int i;
470     float a[20],shift_length;
471     shift_length = (float)SHIFT_LENGTH;
472     int n = failure_nr;
473     memset(a,0,20*sizeof(float));
474     float span = shift_length / (float)n+1.0;  //max time between machine
            failures
475     float current_span = 0.0;
476     int machine;
477     float repair_time ;
478     float breakdown_time;
```

```
479
480     for (i = 0;i<n;i++) {
481
482   current_span+=span;
483   machine = (int)unirand(1,number_of_machines+1,stream);
484   breakdown_time = unirand(0.0,current_span,stream);
485   repair_time =(min_machine_repair_time + expon(max_machine_repair_time ,stream
          ))*60.0;
486   if (a[machine]<breakdown_time) {  //
487       a[machine] = breakdown_time+repair_time;
488   }
489   else { // if breakdown_time clashes with the same machine then let the
          breakdown happen after the machine goes up again
490       breakdown_time = a[machine] + 1.0;
491       a[machine] = breakdown_time+repair_time;
492   }
493   transfer[3] = repair_time;
494   transfer[4] = (float)machine;
495   fail_list[machine-1].downtime+= repair_time;
496   fail_list[machine-1].machine_nr++;
497   event_schedule(sim_time + breakdown_time, EVENT_MACHINE_FAILURE );
498     }
499
500     event_schedule(sim_time + shift_length, EVENT_GENERATE_FAILURES );
501 }
502
503 void machine_failure(){
504     float repair_time = transfer[3];
505     int   machine     = (int)transfer[4];
506     machine_broken[machine] = repair_time;
507     //    printf(" Machine %d broke down and it takes %f to repair\n", machine,
            repair_time/60.0);
508
509     event_schedule(sim_time + repair_time, EVENT_MACHINE_FIXED);
510 }
511
512 void machine_fixed(){
513
514     int   machine     = (int)transfer[4];
515     machine_broken[machine] = 0.0;
516 }
```

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 0 10 30.000000 2.000000 5.000000 13.900000 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
```

```
0.100000 1 69.750000
0.100000 0 81.849998
0.100000 2 70.330002
0.000000 0 0.000000
DGB 866.538452

**************************************************
Report for 0 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    0  0.000 sec / 0.000 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    0  0.000 sec / 0.000 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    0  0.000 sec / 0.000 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    0  0.000 sec / 0.000 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    0  0.000 sec / 0.000 min
--Breakdown for machine nr 6--
Number of fails  Downtime
    0  0.000 sec / 0.000 min
--Breakdown for machine nr 7--
Number of fails  Downtime
    0  0.000 sec / 0.000 min


Total downtime was 0.000 seconds or 0.000 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.505353  0.157318  0.091695  0.486720  0.503629  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
422.573567  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000

Average queue delay: 84.507036
Worst case queue delay: 913.844595
Best case queue delay: 0.000000
System throughput: 74864
Average throughput time: 957.850792
Min throughput time: 536.020005
Random seed: 92

**************************************************
Report for 1 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    14 17408.664 sec / 290.144 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    14 14729.754 sec / 245.496 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    17 21813.156 sec / 363.553 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    12 11268.964 sec / 187.816 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    16 18231.232 sec / 303.854 min
--Breakdown for machine nr 6--
```

```
Number of fails   Downtime
    5   4622.723 sec / 77.045 min
--Breakdown for machine nr 7--
Number of fails   Downtime
    12 15270.830 sec / 254.514 min


Total downtime was 103345.328 seconds or 1722.422 minutes
-----------------
Machine load
-----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.506194   0.155238   0.090343   0.482215   0.497180   0.000000   0.000000

--------------------------
Average delay in queues
--------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
431.438618   0.000000   0.000000   0.114245   0.157482   0.000000   0.000000

Average queue delay: 86.334101
Worst case queue delay: 4401.822073
Best case queue delay: 0.000000
System throughput: 73618
Average throughput time: 975.335583
Min throughput time: 536.020005
Random seed: 92

**************************************************
Report for 2 failures per day
--Breakdown for machine nr 1--
Number of fails   Downtime
    18 29180.676 sec / 486.345 min
--Breakdown for machine nr 2--
Number of fails   Downtime
    25 26656.006 sec / 444.267 min
--Breakdown for machine nr 3--
Number of fails   Downtime
    34 33937.508 sec / 565.625 min
--Breakdown for machine nr 4--
Number of fails   Downtime
    32 34680.711 sec / 578.012 min
--Breakdown for machine nr 5--
Number of fails   Downtime
    25 30137.250 sec / 502.288 min
--Breakdown for machine nr 6--
Number of fails   Downtime
    25 28437.951 sec / 473.966 min
--Breakdown for machine nr 7--
Number of fails   Downtime
    21 23484.295 sec / 391.405 min


Total downtime was 206514.406 seconds or 3441.907 minutes
-----------------
Machine load
-----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.509502   0.152950   0.091803   0.483113   0.489960   0.000000   0.000000

--------------------------
Average delay in queues
--------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
438.621380   0.000000   0.000000   0.194889   0.252850   0.000000   0.000000

Average queue delay: 87.814796
Worst case queue delay: 6815.956869
Best case queue delay: 0.000000
System throughput: 72386
```

```
Average throughput time: 991.017875
Min throughput time: 536.020005
Random seed: 93

**************************************************
Report for 3 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    28 32704.871 sec / 545.081 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    34 37312.844 sec / 621.881 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    39 42175.816 sec / 702.930 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    50 52987.277 sec / 883.121 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    30 39646.910 sec / 660.782 min
--Breakdown for machine nr 6--
Number of fails  Downtime
    38 46417.207 sec / 773.620 min
--Breakdown for machine nr 7--
Number of fails  Downtime
    51 59002.086 sec / 983.368 min


Total downtime was 310247.000 seconds or 5170.783 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.512001  0.151425  0.091938  0.485603  0.484706  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
452.238616  0.000640  0.000000  0.425210  0.433867  0.000000  0.000000

Average queue delay: 90.619165
Worst case queue delay: 6884.549948
Best case queue delay: 0.000000
System throughput: 71170
Average throughput time: 1019.672679
Min throughput time: 536.020005
Random seed: 93

**************************************************
Report for 4 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    47 54550.926 sec / 909.182 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    61 65808.180 sec / 1096.803 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    48 58941.898 sec / 982.365 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    49 59129.191 sec / 985.487 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    60 62343.688 sec / 1039.061 min
--Breakdown for machine nr 6--
Number of fails  Downtime
    47 49368.742 sec / 822.812 min
```

```
--Breakdown for machine nr 7--
Number of fails  Downtime
    48 55978.504 sec / 932.975 min


Total downtime was 406121.125 seconds or 6768.685 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.507544  0.147395  0.089774  0.471650  0.474397  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
454.429966  0.000493  0.000000  0.467445  0.451148  0.000000  0.000000

Average queue delay: 91.060953
Worst case queue delay: 6113.099844
Best case queue delay: 0.000000
System throughput: 69694
Average throughput time: 1023.175732
Min throughput time: 536.020005
Random seed: 94

**************************************************
Report for 5 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    72 75961.328 sec / 1266.022 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    63 72326.273 sec / 1205.438 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    64 81299.133 sec / 1354.986 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    61 65099.527 sec / 1084.992 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    70 84682.312 sec / 1411.372 min
--Breakdown for machine nr 6--
Number of fails  Downtime
    57 62940.590 sec / 1049.010 min
--Breakdown for machine nr 7--
Number of fails  Downtime
    63 68472.734 sec / 1141.212 min


Total downtime was 510781.875 seconds or 8513.031 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.508163  0.146118  0.089338  0.474031  0.467153  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
467.551200  0.000000  0.000000  0.518749  0.710669  0.000000  0.000000

Average queue delay: 93.746839
Worst case queue delay: 5011.952474
Best case queue delay: 0.000000
System throughput: 68376
Average throughput time: 1051.619674
Min throughput time: 536.020005
```

```
Random seed: 94

*************************************************
Report for 6 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    73 77181.312 sec / 1286.355 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    86 112307.180 sec / 1871.786 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    74 85593.086 sec / 1426.551 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    79 86854.828 sec / 1447.580 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    73 85323.086 sec / 1422.051 min
--Breakdown for machine nr 6--
Number of fails  Downtime
    80 94936.453 sec / 1582.274 min
--Breakdown for machine nr 7--
Number of fails  Downtime
    75 77037.805 sec / 1283.963 min


Total downtime was 619233.812 seconds or 10320.564 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.510509  0.145985  0.088430  0.465004  0.457288  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
474.432899  0.001618  0.000000  0.740779  0.693425  0.000000  0.000000

Average queue delay: 95.164149
Worst case queue delay: 10102.571829
Best case queue delay: 0.000000
System throughput: 67126
Average throughput time: 1066.333986
Min throughput time: 536.020005
Random seed: 94

*************************************************
Report for 7 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
    96 106166.305 sec / 1769.438 min
--Breakdown for machine nr 2--
Number of fails  Downtime
    100  136099.312 sec / 2268.322 min
--Breakdown for machine nr 3--
Number of fails  Downtime
    97 123498.727 sec / 2058.312 min
--Breakdown for machine nr 4--
Number of fails  Downtime
    87 95401.359 sec / 1590.023 min
--Breakdown for machine nr 5--
Number of fails  Downtime
    84 104042.820 sec / 1734.047 min
--Breakdown for machine nr 6--
Number of fails  Downtime
    93 101772.180 sec / 1696.203 min
--Breakdown for machine nr 7--
Number of fails  Downtime
```

```
   73 76167.812 sec / 1269.464 min


Total downtime was 743148.500 seconds or 12385.808 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.513300  0.141074  0.086087  0.452615  0.445818  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
495.838292  0.001619  0.000000  0.835708  0.807359  0.000000  0.000000

Average queue delay: 99.496596
Worst case queue delay: 7786.900594
Best case queue delay: 0.000000
System throughput: 65262
Average throughput time: 1099.123939
Min throughput time: 536.020005
Random seed: 95

**************************************************
Report for 8 failures per day
--Breakdown for machine nr 1--
Number of fails  Downtime
   103  115247.594 sec / 1920.793 min
--Breakdown for machine nr 2--
Number of fails  Downtime
   90 107070.672 sec / 1784.511 min
--Breakdown for machine nr 3--
Number of fails  Downtime
   97 104376.836 sec / 1739.614 min
--Breakdown for machine nr 4--
Number of fails  Downtime
   97 114411.133 sec / 1906.852 min
--Breakdown for machine nr 5--
Number of fails  Downtime
   110  118845.734 sec / 1980.762 min
--Breakdown for machine nr 6--
Number of fails  Downtime
   113  117484.648 sec / 1958.077 min
--Breakdown for machine nr 7--
Number of fails  Downtime
   110  115256.328 sec / 1920.939 min


Total downtime was 792692.875 seconds or 13211.548 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.517888  0.140782  0.089724  0.467316  0.448905  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
497.663670  0.004016  0.000000  1.119780  0.971277  0.000000  0.000000

Average queue delay: 99.952977
Worst case queue delay: 9499.287679
Best case queue delay: 0.000000
System throughput: 65082
Average throughput time: 1118.785573
Min throughput time: 536.020005
Random seed: 95

```

```
426 ************************************************
427 Report for 9 failures per day
428 --Breakdown for machine nr 1--
429 Number of fails  Downtime
430    114  121955.883 sec / 2032.598 min
431 --Breakdown for machine nr 2--
432 Number of fails  Downtime
433    106  117085.289 sec / 1951.421 min
434 --Breakdown for machine nr 3--
435 Number of fails  Downtime
436    113  127078.883 sec / 2117.981 min
437 --Breakdown for machine nr 4--
438 Number of fails  Downtime
439    97 112215.234 sec / 1870.254 min
440 --Breakdown for machine nr 5--
441 Number of fails  Downtime
442    121  143202.188 sec / 2386.703 min
443 --Breakdown for machine nr 6--
444 Number of fails  Downtime
445    136  157542.328 sec / 2625.705 min
446 --Breakdown for machine nr 7--
447 Number of fails  Downtime
448    123  133715.547 sec / 2228.592 min
449
450
451 Total downtime was 912795.375 seconds or 15213.256 minutes
452 ---------------
453 Machine load
454 ---------------
455 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
456 0.514790  0.136965  0.091864  0.458324  0.438256  0.000000  0.000000
457
458 ------------------------
459 Average delay in queues
460 ------------------------
461 Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
462 505.305330  0.003893  0.000000  1.335710  1.057195  0.000000  0.000000
463
464 Average queue delay: 101.529608
465 Worst case queue delay: 9402.232382
466 Best case queue delay: 0.000000
467 System throughput: 63354
468 Average throughput time: 1145.054487
469 Min throughput time: 536.020005
470 Random seed: 95
471
472 ************************************************
473 Report for 10 failures per day
474 --Breakdown for machine nr 1--
475 Number of fails  Downtime
476    131  131869.453 sec / 2197.824 min
477 --Breakdown for machine nr 2--
478 Number of fails  Downtime
479    125  155051.344 sec / 2584.189 min
480 --Breakdown for machine nr 3--
481 Number of fails  Downtime
482    126  158219.750 sec / 2636.996 min
483 --Breakdown for machine nr 4--
484 Number of fails  Downtime
485    120  142989.531 sec / 2383.159 min
486 --Breakdown for machine nr 5--
487 Number of fails  Downtime
488    134  148801.031 sec / 2480.017 min
489 --Breakdown for machine nr 6--
490 Number of fails  Downtime
491    134  154658.062 sec / 2577.634 min
492 --Breakdown for machine nr 7--
493 Number of fails  Downtime
494    130  154041.078 sec / 2567.351 min
495
```

```
496
497 Total downtime was 1045630.250 seconds or 17427.171 minutes
498 ----------------
499 Machine load
500 ----------------
501 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
502 0.514304   0.133754   0.091676   0.452557   0.431712   0.000000   0.000000
503
504 ----------------------------
505 Average delay in queues
506 ----------------------------
507 Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
508 519.482689   0.007528   0.000000   1.308374   1.348414   0.000000   0.000000
509
510 Average queue delay: 104.423397
511 Worst case queue delay: 8779.974569
512 Best case queue delay: 0.000000
513 System throughput: 61998
514 Average throughput time: 1160.986552
515 Min throughput time: 536.020005
516 Random seed: 96
```