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.

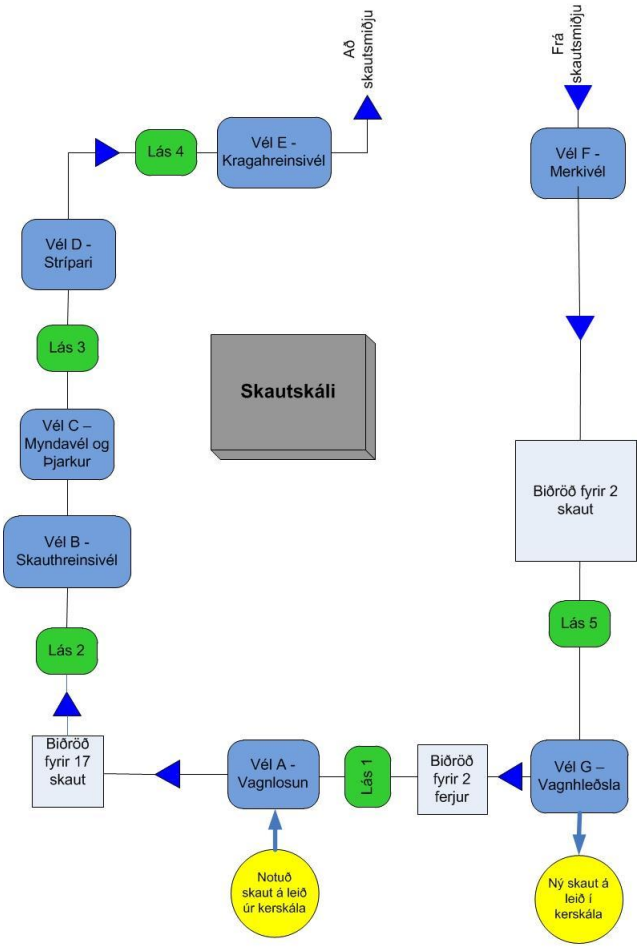
Efnisyfirlit

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



Mynd 1: Ferli skautskála

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.

2 Niðurstöður

Í grein 3.2 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 bilanna 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 3 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 3.3)

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 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.

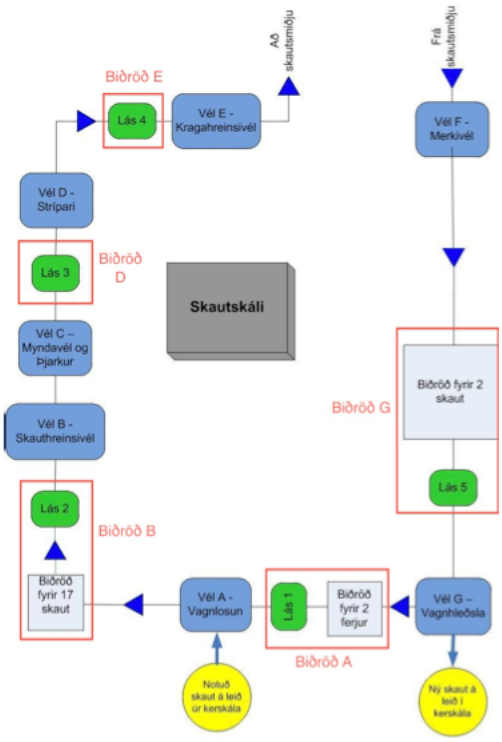
3.3 Einingar og undirkerfi

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 2. Vélar B og C geta unnið tvö skaut í einu. [1]

Þegar Mynd 2 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



Mynd 2: Einingar líkans

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.

3.4 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

3.5 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).

3.6 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 3.2 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 20.0

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;
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;


int sampst_delays, throughput_time; // variable for queue delays and throughput
     time
time_t dummy;
unsigned int skaut_id, stream, failure_nr;
int queue_size[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   float ble = 4332.692383;
159   int q = 5;
160   float t = ble / q;
161   printf("DGB %f\n", t);
162
163     // initialize arrays and variables
164     if((fail_list = malloc(sizeof(breakdown)*max_no_failures))==NULL) {
165   printf("Allocation Error\n");
166   exit(1);
167     }
168
169
170
171     int b;
172 /*     for (b=1; b <= number_of_machines; b++) {
173   printf("transfer_time[%d] = %f\n", b,transfer_time[b] );
174   printf("busy %d broken %f \n",is_machine_busy[b],machine_broken[b]);
175   }*/
176     // We perform simulation for "a few" failures per day
177
178     for (failure_nr = min_no_failures; failure_nr<= max_no_failures; failure_nr
          ++) {
179   stream = (unsigned int)time(NULL) % 100;
180
181   memset( is_machine_busy,0, NUM_MACHINES +1 );
182   memset( machine_broken,0, NUM_MACHINES +1);
183   memset( fail_list ,0, sizeof(breakdown)*NUM_MACHINES+1);
184   fail_index = 0;
185   skaut_throughput = 0;
186   sampst_delays = number_of_machines +1;
187   throughput_time = number_of_machines +2;
188
189   skaut_id = 1;
190   skaut_throughput = 0;
191
192
193   // Initialize rndlib
194   init_twister();
195
196   // Initialize simlib
197   init_simlib();
```

```
198
199     maxatr = 6; // how many attributes do we need?
200
201     /* Schedule first wagen arrival */
202     event_schedule( 10.0, EVENT_WAGEN_UNLOAD_ARRIVAL );
203
204     /* Schedule end of warmup time */
205     event_schedule( end_warmup_time, EVENT_END_WARMUP );
206     event_schedule(end_warmup_time, EVENT_GENERATE_FAILURES );
207     /* Schedule simulation termination */
208     event_schedule(end_simulation_time , EVENT_END_SIMULATION );
209
210     next_event_type = 0;
211
212
213
214     while (next_event_type != EVENT_END_SIMULATION) {
215
216         timing();
217         /*                      printf("event_type = %d, transfer[3] = %f\n",
              next_event_type, transfer[3]);
218             int k;
219             for (k = 1; k <= number_of_machines; k++)
220             printf("Items in machines/queues %d:  %d, %d\n", k, list_size[k],
                    list_size[number_of_machines +k]);
221             printf("\n");
222         */
223
224
225         switch (next_event_type) {
226         case EVENT_WAGEN_UNLOAD_ARRIVAL:
227       wagen_unload_arrival();
228       break;
229         case EVENT_SKAUT_ARRIVAL:
230       skaut_arrival();
231       break;
232         case EVENT_SKAUT_DEPARTURE:
233       skaut_departure();
234       break;
235         case EVENT_MACHINE_FAILURE:
236       machine_failure();
237       break;
238         case EVENT_MACHINE_FIXED:
239       machine_fixed();
240       break;
241         case EVENT_END_WARMUP:
242       end_warmup();
243       break;
244         case EVENT_END_SIMULATION:
245       report();
246       break;
247         case EVENT_GENERATE_FAILURES:
248       create_machine_fail_events();
249       break;
250
251         }
252     }
253       }
254 }
255
256
257
258 void wagen_unload_arrival()
259 {
260
261       int i;
262       int current_unit = 0;
263       float wagen_arrival_zeit = unirand((mean_wagen_arrival-std_wagen_arrival)
              *60.0,(mean_wagen_arrival+std_wagen_arrival)*60.0,stream);
264
```

```
265     for (i = 1; i<NUM_MACHINES+1; i++) {  //delay unload of skaut by the time
            it takes to repair
266     if (machine_broken[i] > 0.0) {
267       event_schedule(sim_time + machine_broken[i], EVENT_WAGEN_UNLOAD_ARRIVAL);
268       return;
269     }
270     }
271
272     if (list_size[number_of_machines + 1] != 0) { // ef allt er enn fullt Ã¿Ãą
            koma meÃř nÃęsta vagn eftir uÃ¿b hÃąlftÃąma
273   event_schedule(sim_time + wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
274   return;
275     }
276
277     int vagn_magn =  WAGEN_LOAD-((int)unirand(0.0,3.0,stream));   //12 - 14
            skaut Ãą hverjum vagni
278     for (i=1; i <= vagn_magn; i++) {
279
280   transfer[3]=1.0;
281   transfer[4] = sim_time + (i * 0.01); // skaut entering system time
282   transfer[6] = (float) skaut_id++;
283   //printf("tr4 in wagen: %f\n", transfer[4]);
284   event_schedule( sim_time + ( i* 0.01),  EVENT_SKAUT_ARRIVAL);
285     }
286
287     event_schedule(sim_time+wagen_arrival_zeit, EVENT_WAGEN_UNLOAD_ARRIVAL);
288 }
289
290
291 void skaut_arrival()
292 {
293     push_array();
294     int current_unit = (int)transfer[3];
295     int i;
296
297     for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if there is a
            broken machine before current one
298   if (machine_broken[i] > 0.0) {
299       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
300     event_schedule(sim_time +  machine_broken[i] + work_time[current_unit],
          EVENT_SKAUT_ARRIVAL); //also if next queue is full then delay it.
301     return;
302     }
303   }
304     }
305
306     // check if machine is not busy
307     if (list_size[current_unit] == 0 && machine_broken[current_unit] == 0.0) {
308   sampst(0.0, sampst_delays);
309   sampst(0.0, current_unit);
310
311   list_file(FIRST, current_unit); // last := first here because there are only
          to be 0 or 1 items in machine
312
313   // schedule departure after machine processing time
314   pop_array();
315   event_schedule(PREP_TIME + sim_time + work_time[current_unit],
          EVENT_SKAUT_DEPARTURE);
316     } else {
317
318   if (list_size[number_of_machines + current_unit] == queue_size[current_unit])
          {
319
320       event_schedule(PREP_TIME + sim_time + work_time[current_unit],
            EVENT_SKAUT_ARRIVAL); //also if queue is full then delay it.
321
322   } else {
323       transfer[5] = sim_time;
```

```
324         list_file(LAST, number_of_machines + current_unit);
325         //printf("puting skaut in queue: %d\n", current_unit);
326     }
327 
328       }
329 
330 }
331 
332 void skaut_departure()
333 {
334       push_array();
335       int current_unit = (int) transfer[3];
336       int i = 0;
337       for (i = NUM_MACHINES; i>=current_unit; i--) {  //add delay if machine is
              broken or there is a broken machine before current one
338     if (machine_broken[i] > 0.0) {
339         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.
340       event_schedule(sim_time + machine_broken[i], EVENT_SKAUT_DEPARTURE); //also
              if next queue is full then delay it.
341       return;
342         }
343 //        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]);
344         break;
345     }
346       }
347 
348       if (current_unit == MACHINES_ON_THE_LEFT_SIDE) {
349     skaut_throughput += 2;
350     sampst(sim_time - transfer[4], throughput_time);
351     list_remove(FIRST,current_unit);
352       } else {
353     list_remove(FIRST,current_unit);
354     pop_array();
355     transfer[3]++;
356     event_schedule(PREP_TIME + sim_time + transfer_time[(int)(transfer[3])-1],
          EVENT_SKAUT_ARRIVAL);
357       }
358 
359 
360       if (list_size[number_of_machines + current_unit] != 0) {
361     pop_array();
362 
363     list_file(FIRST,current_unit); // first equals last because size should only
          be 1
364     pop_array();
365 
366     list_remove(FIRST, number_of_machines + current_unit);
367     pop_array();
368 
369     sampst(sim_time - transfer[5], sampst_delays);
370     sampst(sim_time - transfer[5], current_unit);
371     event_schedule(PREP_TIME + sim_time + work_time[current_unit],
          EVENT_SKAUT_DEPARTURE);
372       }
373 }
374 
375 
376 void parse_input(char inputfile_data[], char inputfile_time[])
377 {
378 
379 
380       if ((infile = fopen (inputfile_data, "r")) == NULL) {
381     printf("Could not open file %s\n",inputfile_data);
382       }
383 
384       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);
385     fclose(infile);
386
387
388     if ((infile = fopen (inputfile_time, "r")) == NULL) {
389   printf("Could not open file %s\n",inputfile_time);
390     }
391     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);
392
393     int counter = 1;
394     while (!feof(infile)) {
395   fscanf(infile, "%f %d %f", &transfer_time[counter], &queue_size[counter], &
        work_time[counter] );
396   printf("%f %d %f\n", transfer_time[counter], queue_size[counter], work_time[
        counter] );
397   counter++;
398     }
399     fclose(infile);
400
401 }
402
403 void end_warmup()
404 {
405     sampst(0.0, 0);
406     timest(0.0, 0);
407     skaut_throughput = 0;
408 }
409
410 void report()
411 {
412
413     int i;
414     float total_downtime = 0.0;
415     printf("\n**************************************************\n");
416     printf("Report for %d failures per day\n",failure_nr);
417
418     for (i=0; i <NUM_MACHINES; i++) {
419   printf("--Breakdown nr %d--\n", i+1);
420   printf("Number of fails\t Downtime \t\n");
421   printf("\t %d\t", fail_list[i].machine_nr);
422   printf("%.3f sec / %.3f min\t", fail_list[i].downtime,fail_list[i].downtime
        /60.0);
423   printf("\n");
424   total_downtime+=fail_list[i].downtime;
425     }
426     printf("\n\n");
427
428
429     printf("Total downtime was %.3lf seconds or %.3lf minutes\n",total_downtime
          , total_downtime/60.0);
430
431     printf("----------------\nMachine load\n----------------\n");
432     for (i=1; i <= number_of_machines; i++) {
433   printf("Machine %d\t", i);
434     }
435     printf("\n");
436     for (i=1; i <= number_of_machines; i++) {
437   printf("%f\t", filest(i) );
438     }
439     printf("\n\n");
440
441     printf("------------------------\nAverage delay in queues\n
          ------------------------\n");
442     for (i=1; i <= number_of_machines; i++) {
443   printf("Queue %d \t", i);
444     }
```

```
445
446     printf("\n");
447
448     for (i=1; i <= number_of_machines; i++) {
449   printf("%f\t", sampst(0.0, -i));
450     }
451     printf("\n\n");
452     printf("Average queue delay: %f\n", sampst(0.0, -sampst_delays));
453   printf("Worst case queue delay: %f\n", transfer[3]);
454   printf("Best case queue delay: %f\n", transfer[4]);
455
456     printf("System throughput: %d\n", skaut_throughput );
457     printf("Average throughput time: %f\n", sampst(0.0, -throughput_time));
458     printf("Min throughput time: %f\n", transfer[4]);
459     printf("Random seed: %d\n", stream);
460
461
462 }
463
464 void push_array() {
465
466     memcpy(temp_transfer,transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
467 }
468
469 void pop_array() {
470     memcpy(transfer,temp_transfer,TRANSFER_ARRAY_LENGTH*sizeof(float));
471 }
472
473 void create_machine_fail_events() {
474     int i;
475     float a[20],shift_length;
476     shift_length = (float)SHIFT_LENGTH;
477     int n = failure_nr;
478     memset(a,0,20*sizeof(float));
479     float span = shift_length / (float)n+1.0;  //max time between machine
          failures
480     float current_span = 0.0;
481     int machine;
482     float repair_time ;
483     float breakdown_time;
484
485     for (i = 0;i<n;i++) {
486   current_span+=span;
487   machine = (int)unirand(1,number_of_machines+1,stream);
488   breakdown_time = unirand(0.0,current_span,stream);
489   repair_time =(5.0 + expon(log(max_machine_repair_time -
          min_machine_repair_time),stream))*60.0;
490   if (a[machine]<breakdown_time) {  //
491       a[machine] = breakdown_time+repair_time;
492   }
493   else { // if breakdown_time clashes with the same machine then let the
          breakdown happen after the machine goes up again
494       breakdown_time = a[machine] + 1.0;
495       a[machine] = breakdown_time+repair_time;
496   }
497   transfer[3] = repair_time;
498   transfer[4] = (float)machine;
499   fail_list[machine-1].downtime+= repair_time;
500   fail_list[machine-1].machine_nr++;
501   event_schedule(sim_time + breakdown_time, EVENT_MACHINE_FAILURE );
502     }
503
504     event_schedule(sim_time + shift_length, EVENT_GENERATE_FAILURES );
505 }
506
507 void machine_failure(){
508     float repair_time = transfer[3];
509     int   machine     = (int)transfer[4];
510     machine_broken[machine] = repair_time;
```

```
511 //     printf(" Machine %d broke down and it takes %f to repair\n", machine,
        repair_time/60.0);
512
513     event_schedule(sim_time + repair_time, EVENT_MACHINE_FIXED);
514 }
515
516 void machine_fixed(){
517
518     int    machine      = (int)transfer[4];
519     machine_broken[machine] = 0.0;
520 }
```

5.2 Inntaksgögn líkans

```
1 7 404 3 10 30.0 2.0 5.0 180.0 1000.0 5185000.0
2 num min min max mean std min max warmup simulationtime
3 prod- no no wagen wagen repair repair
4 uction fail- fail- arrival arrival time time
5 ures ures
```

```
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    41 26788.814 sec / 446.480 min
17 --Breakdown nr 2--
18 Number of fails  Downtime
19    45 24128.580 sec / 402.143 min
20 --Breakdown nr 3--
21 Number of fails  Downtime
22    33 17966.521 sec / 299.442 min
23 --Breakdown nr 4--
24 Number of fails  Downtime
25    37 22147.221 sec / 369.120 min
26 --Breakdown nr 5--
27 Number of fails  Downtime
28    49 29759.027 sec / 495.984 min
29 --Breakdown nr 6--
30 Number of fails  Downtime
31    32 19525.203 sec / 325.420 min
32 --Breakdown nr 7--
33 Number of fails  Downtime
34    33 20262.217 sec / 337.704 min
35
```

```
36
37 Total downtime was 160577.578 seconds or 2676.293 minutes
38 ---------------
39 Machine load
40 ---------------
41 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
42 0.652475   0.295655   0.235974   0.629209   0.637003   0.000000   0.000000
43
44 ---------------------------
45 Average delay in queues
46 ---------------------------
47 Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
48 556.128450   0.000000   0.000000   0.651664   0.813013   0.000000   0.000000
49
50 Average queue delay: 111.520452
51 Worst case queue delay: 2509.251272
52 Best case queue delay: 0.000000
53 System throughput: 73094
54 Average throughput time: 1307.654741
55 Min throughput time: 716.020005
56 Random seed: 2
57
58 ***************************************************
59 Report for 4 failures per day
60 --Breakdown nr 1--
61 Number of fails   Downtime
62     53 33950.348 sec / 565.839 min
63 --Breakdown nr 2--
64 Number of fails   Downtime
65     48 26518.125 sec / 441.969 min
66 --Breakdown nr 3--
67 Number of fails   Downtime
68     49 36245.836 sec / 604.097 min
69 --Breakdown nr 4--
70 Number of fails   Downtime
71     58 38801.020 sec / 646.684 min
72 --Breakdown nr 5--
73 Number of fails   Downtime
74     47 30412.449 sec / 506.874 min
75 --Breakdown nr 6--
76 Number of fails   Downtime
77     52 34193.930 sec / 569.899 min
78 --Breakdown nr 7--
79 Number of fails   Downtime
80     53 33250.887 sec / 554.181 min
81
82
83 Total downtime was 233372.609 seconds or 3889.543 minutes
84 ---------------
85 Machine load
86 ---------------
87 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
88 0.652833   0.291006   0.233113   0.621588   0.625639   0.000000   0.000000
89
90 ---------------------------
91 Average delay in queues
92 ---------------------------
93 Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
94 570.078344   0.000798   0.000000   0.831948   0.916519   0.000000   0.000000
95
96 Average queue delay: 114.362361
97 Worst case queue delay: 3951.289510
98 Best case queue delay: 0.000000
99 System throughput: 71814
100 Average throughput time: 1321.207520
101 Min throughput time: 716.020005
102 Random seed: 2
103
104 ***************************************************
105 Report for 5 failures per day
```

```
--Breakdown nr 1--
Number of fails  Downtime
    59 38588.750 sec / 643.146 min
--Breakdown nr 2--
Number of fails  Downtime
    59 35094.961 sec / 584.916 min
--Breakdown nr 3--
Number of fails  Downtime
    70 44980.055 sec / 749.668 min
--Breakdown nr 4--
Number of fails  Downtime
    63 39424.258 sec / 657.071 min
--Breakdown nr 5--
Number of fails  Downtime
    76 46976.441 sec / 782.941 min
--Breakdown nr 6--
Number of fails  Downtime
    51 27134.527 sec / 452.242 min
--Breakdown nr 7--
Number of fails  Downtime
    72 39981.879 sec / 666.365 min


Total downtime was 272180.875 seconds or 4536.348 minutes
---------------
Machine load
---------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.655846  0.291449  0.234733  0.622672  0.625658  0.000000  0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1   Queue 2   Queue 3   Queue 4   Queue 5   Queue 6   Queue 7
571.223949  0.000033  0.000000  0.999569  1.378115  0.000000  0.000000

Average queue delay: 114.722883
Worst case queue delay: 3629.466635
Best case queue delay: 0.000000
System throughput: 71626
Average throughput time: 1345.000109
Min throughput time: 716.020005
Random seed: 3

**************************************************
Report for 6 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    58 34646.707 sec / 577.445 min
--Breakdown nr 2--
Number of fails  Downtime
    78 47186.781 sec / 786.446 min
--Breakdown nr 3--
Number of fails  Downtime
    82 52526.715 sec / 875.445 min
--Breakdown nr 4--
Number of fails  Downtime
    88 53314.695 sec / 888.578 min
--Breakdown nr 5--
Number of fails  Downtime
    81 52058.883 sec / 867.648 min
--Breakdown nr 6--
Number of fails  Downtime
    81 56321.199 sec / 938.687 min
--Breakdown nr 7--
Number of fails  Downtime
    72 41435.035 sec / 690.584 min


Total downtime was 337490.000 seconds or 5624.833 minutes
```

```
176 ----------------
177 Machine load
178 ----------------
179 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
180 0.654757   0.286137   0.236455   0.618815   0.616193   0.000000   0.000000
181
182 ---------------------------
183 Average delay in queues
184 ---------------------------
185 Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
186 579.281867   0.001669   0.000000   1.298685   1.452027   0.000000   0.000000
187
188 Average queue delay: 116.400965
189 Worst case queue delay: 4021.025900
190 Best case queue delay: 0.000000
191 System throughput: 70564
192 Average throughput time: 1362.773616
193 Min throughput time: 716.020005
194 Random seed: 3
195
196 ***************************************************
197 Report for 7 failures per day
198 --Breakdown nr 1--
199 Number of fails  Downtime
200     92 57198.012 sec / 953.300 min
201 --Breakdown nr 2--
202 Number of fails  Downtime
203     96 57828.371 sec / 963.806 min
204 --Breakdown nr 3--
205 Number of fails  Downtime
206     84 55161.184 sec / 919.353 min
207 --Breakdown nr 4--
208 Number of fails  Downtime
209     79 50026.469 sec / 833.774 min
210 --Breakdown nr 5--
211 Number of fails  Downtime
212     93 57617.109 sec / 960.285 min
213 --Breakdown nr 6--
214 Number of fails  Downtime
215     88 53010.113 sec / 883.502 min
216 --Breakdown nr 7--
217 Number of fails  Downtime
218     98 58190.375 sec / 969.840 min
219
220
221 Total downtime was 389031.625 seconds or 6483.860 minutes
222 ----------------
223 Machine load
224 ----------------
225 Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
226 0.651046   0.283728   0.231902   0.612669   0.609454   0.000000   0.000000
227
228 ---------------------------
229 Average delay in queues
230 ---------------------------
231 Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
232 586.887553   0.003761   0.000000   1.344610   1.446701   0.000000   0.000000
233
234 Average queue delay: 117.939218
235 Worst case queue delay: 5289.548269
236 Best case queue delay: 0.000000
237 System throughput: 69652
238 Average throughput time: 1367.822749
239 Min throughput time: 716.020005
240 Random seed: 3
241
242 ***************************************************
243 Report for 8 failures per day
244 --Breakdown nr 1--
245 Number of fails  Downtime
```

```
    115   67645.789 sec / 1127.430 min
--Breakdown nr 2--
Number of fails   Downtime
    98 56907.039 sec / 948.451 min
--Breakdown nr 3--
Number of fails   Downtime
    97 59712.578 sec / 995.210 min
--Breakdown nr 4--
Number of fails   Downtime
    101   57001.605 sec / 950.027 min
--Breakdown nr 5--
Number of fails   Downtime
    111   70068.227 sec / 1167.804 min
--Breakdown nr 6--
Number of fails   Downtime
    101   60606.320 sec / 1010.105 min
--Breakdown nr 7--
Number of fails   Downtime
    97 57413.035 sec / 956.884 min


Total downtime was 429354.594 seconds or 7155.910 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.658559   0.285551   0.234485   0.620156   0.613783   0.000000   0.000000

------------------------
Average delay in queues
------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
590.133777   0.002790   0.000000   1.595718   1.614850   0.000000   0.000000

Average queue delay: 118.660692
Worst case queue delay: 4149.571340
Best case queue delay: 0.000000
System throughput: 69988
Average throughput time: 1384.745498
Min throughput time: 716.020005
Random seed: 4

**************************************************
Report for 9 failures per day
--Breakdown nr 1--
Number of fails   Downtime
    126   70582.094 sec / 1176.368 min
--Breakdown nr 2--
Number of fails   Downtime
    96 55765.223 sec / 929.420 min
--Breakdown nr 3--
Number of fails   Downtime
    116   70053.008 sec / 1167.550 min
--Breakdown nr 4--
Number of fails   Downtime
    125   74211.961 sec / 1236.866 min
--Breakdown nr 5--
Number of fails   Downtime
    119   69861.961 sec / 1164.366 min
--Breakdown nr 6--
Number of fails   Downtime
    115   69705.078 sec / 1161.751 min
--Breakdown nr 7--
Number of fails   Downtime
    113   72078.836 sec / 1201.314 min


Total downtime was 482258.156 seconds or 8037.636 minutes
----------------
Machine load
```

```
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.662198   0.281631   0.230767   0.618631   0.606361   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
599.021005   0.005123   0.000000   2.019422   1.832545   0.000000   0.000000

Average queue delay: 120.565931
Worst case queue delay: 4225.243673
Best case queue delay: 0.000000
System throughput: 69064
Average throughput time: 1419.516327
Min throughput time: 716.020005
Random seed: 4

**************************************************
Report for 10 failures per day
--Breakdown nr 1--
Number of fails  Downtime
    119  72080.391 sec / 1201.340 min
--Breakdown nr 2--
Number of fails  Downtime
    134  76719.961 sec / 1278.666 min
--Breakdown nr 3--
Number of fails  Downtime
    114  69735.547 sec / 1162.259 min
--Breakdown nr 4--
Number of fails  Downtime
    135  76002.414 sec / 1266.707 min
--Breakdown nr 5--
Number of fails  Downtime
    142  79805.789 sec / 1330.096 min
--Breakdown nr 6--
Number of fails  Downtime
    122  75031.578 sec / 1250.526 min
--Breakdown nr 7--
Number of fails  Downtime
    134  86053.766 sec / 1434.229 min


Total downtime was 535429.438 seconds or 8923.824 minutes
----------------
Machine load
----------------
Machine 1 Machine 2 Machine 3 Machine 4 Machine 5 Machine 6 Machine 7
0.663127   0.280110   0.232150   0.613029   0.600534   0.000000   0.000000

-------------------------
Average delay in queues
-------------------------
Queue 1    Queue 2    Queue 3    Queue 4    Queue 5    Queue 6    Queue 7
606.704219   0.006387   0.000000   2.232793   2.070803   0.000000   0.000000

Average queue delay: 122.205674
Worst case queue delay: 4281.351339
Best case queue delay: 0.000000
System throughput: 68382
Average throughput time: 1429.879860
Min throughput time: 716.020005
Random seed: 4
```