ISSN 0280-5316
ISRN LUTFD2/TFRT--5806--SE

Batch-produktion av biologiska ämnen: Terminologi, metoder och verktyg för förbättrad batch-strategi vid Novozymes Biopharma AB

Magnus Lindhult

Institutionen för Reglerteknik
Lunds universitet
December 2007

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
December 2007

Document Number
ISRN LUTFD2/TFRT--5806--SE

Author(s)
Magnus Lindhult

Supervisor
Christian Cimander, Novozymes Biopharma AB, Lund
Charlotta Johansson, Inst för Reglerteknik, Lund (Examinator)

Sponsoring organization

Title and subtitle
Batch-produktion av biologiska ämnen: Terminologi, metoder och verktyg för förbättrad batch-strategi vid Novozymes Biopharma AB (Batch Production of Biological Substances: Terminology, Methods, and Tools for Improved Batch Strategy at Novozymes Biopharma AB)

Abstract
In every kind of industry there is a wish to use the resources available in the best way possible. If this can be achieved, both time and money can be saved and profit will be maximized. The problem is to know in which way this can be done. When is a resource used in an optimal way? This question was in focus when Novozymes Biopharama AB was about to expand their batch production. In order to contribute with general guidlines and to measure the capacity of the new facility, this thesis evaluates a scheduling method based upon Dijkstra's algorithm and it tests two comercial scheduling software packages.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
Swedish

Number of pages
55

Recipient's notes

Security classification

Förord

Detta examensarbete utfördes på Novozymes Biopharma AB:s produktionsanläggning i Lund mellan november och maj 2006/2007. Jag vill tacka Charlotta Johnsson på institutionen för reglerteknik, LTH för god handledning, samt Christian Cimander på Novozymes för stöd och hjälp. Jag vill även passa på att tacka båda för att jag fick tillfälle att utföra detta examensarbete.

Innehållsförteckning

Inledning

För företag som sysslar med någon form av produktion, finns det oftast stora kostnader associerade med de maskiner och personal som krävs. Mycket pengar, tid och resurser kan sparas genom att vara noggrann med att använda allt så effektivt som möjligt. Frågan är bara hur man gör detta?

Novozymes Biopharma AB kommer under år 2006 att bygga ut sin produktionsanläggning i Lund. Den färdiga anläggningen ska givetvis användas så effektivt som möjligt och för att ta reda på hur detta görs, har det startats ett examensarbete som går ut på att optimera och schedulera en av processcellerna i fabriken. Den utvalda processcellen blev en cell för buffertberedning.

Detta examensarbete är uppdelat enligt följande:

Det första kapitlet handlar om uppgiften och beskriver frågeställningarna. Standarden ISA-88s struktur samt innehåll tas upp i kapitel två.

I kapitel 3 ges en beskrivning av företaget Novozymes Biopharma AB och kopplingar mellan ISA-88 och produktionsanläggningen görs.

Grunder och strategier gällande schedulering följer sedan i kapitel fyra, varpå en egen lösning presenteras i kapitel fem.

Sedan följer svar på uppgifterna till kapitel ett (kapitel sex), samt en sammanfattning av examensarbetet (kapitel sju).

Slutligen kommer en sammanställningen av terminologin, samt referenser och bilagor.

1. Uppgiften

Målet med arbetet är att få fram terminologi, metoder, verktyg och riktlinjer för en förbättrad batchproduktion på Novozymes. Arbetet ska stämma överens med ISA-88 standarden för batch kontroll.

Som hjälp för att komma igång med scheduleringen ska följande tre scenarion utredas.

Scenario 1
Vad är den maximala mängden buffert som kan göras på 24 timmar? (Volym och antal) .

Scenario 2
Vad är det maximala antalet buffertberedningar a 2300 liter, som kan blandas per dygn?

Scenario 3
Om följande åtta buffertar ska tillredas, hur blandar man dessa snabbast och hur lång tid tar det?
Buffert H har högst prioritet och måste vara klar först av buffertarna.

A – 600 liter
B – 2700
C – 980
D – 200
E – 600
F – 400
G – 400
H – 5500 *

Utöver dessa scenarion, finns en mängd andra frågeställningar som också är intressanta:

- Vad är den normala cykel tiden (tiden som behövs för en batch)?
- Hur ska produktionen planeras för att maximal mängd buffert ska kunna tillverkas?
- Finns det några generella riktlinjer och regler som kan öka effektiviteten?
- Vilka optimeringsmetoder finns, som kan hjälpa till vid schedulering?
- Finns det färdiga verktyg eller program som kan vara användbara för Novozymes planering?
- Går det att få testlicenser till dessa så att det går att göra en utvärdering på anläggningen?
- En sammanställning och definiering av terminologin, som används eller borde användas på Novozymes, ska konstrueras.

2. ISA-88

2.1 Vad är ISA-88?

ISA står för ”Instrumentation, Systems and Automation Association”. ISA-88 är en standard för styrning av satsvisa processer (batch control). Standarden innehåller modeller och terminologi och beskriver hur man ska strukturera produktionsprocessen. Fördelarna med att ha en gemensam standard är flera. Kommunikationen mellan inblandade parter blir mer effektiv med en gemensam terminologi och risken för missförstånd minskar. Modellerna som ingår kan appliceras på automatiserade, semiautomata och manuella produktionsprocesser. De är bland annat designade för att reducera kostnaden för automatisering, hjälpa användaren att identifiera sina behov, samt underlätta och bidra med en rättfram metod vid tillverkning och konvertering mellan olika recept-typer. Det mesta i ISA-88 är presenterat i en hierarkisk struktur där man kan zooma in och ut till olika detaljnivåer.

2.2 Fysisk hierarki

För att beskriva de fysiska delarna av ett företags verksamhet gör ISA-88 följande uppdelning, se bild 1. En snabb förklaring ges till vänster.

Enterprise: Företaget. Ansvarar för vilka produkter som skall tillverkas. Både var och hur.

Site: Anläggning. En fysisk, geografisk eller logisk indelning som bestäms av företaget.

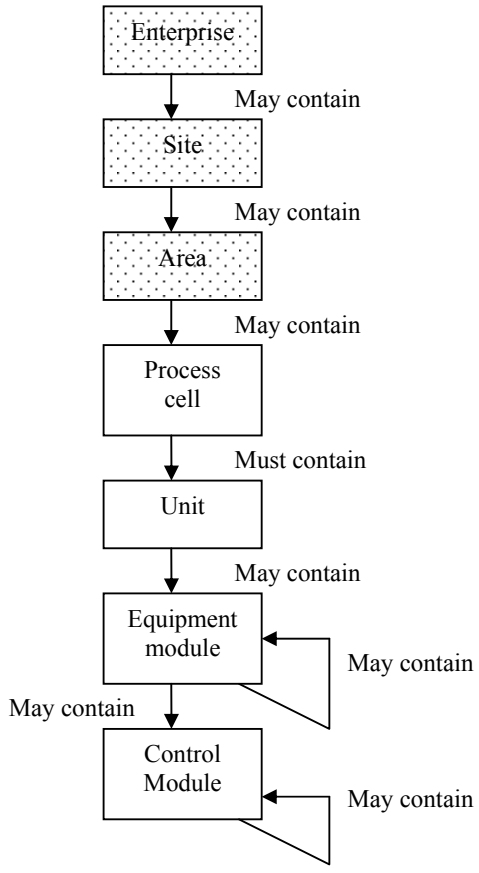
Area: Område. Mindre indelning av anläggningen. Även denna indelning bestäms av företaget.

Process cell: Process cell. Logisk indelning av den utrustning som behövs för att tillverka en eller flera batcher.

Unit: Enhet. Kan innehålla både control moduler och equipment moduler. En processaktivitet måste ske i en enhet (tex en tank).

Equipment module: Utrustningsmodul. Utför mindre sysslor i en enhet. Tex omrörare i en tank.

Control module: Kontrollmodul. Sensorer, regulatorer, aktuatorer eller kombinationer av dessa kallas för kontrollmoduler.



Figur 1 – Ett företags fysiska delar enligt ISA-88.

De tre översta rutorna i figur 1 är fyllda pga att de inte ingår i ISA-88. Hur dessa ska struktureras beskrivs ej i standarden.

2.3 Recept

För att beskriva hur en produkt skall tillverkas använder man sig av recept. Det finns fyra olika typer av recept och dessa beskrivs i ISA-88. Oavsett vilken typ av recept det handlar om, så innehåller det fem kategorier av information. Dessa är: dokumenthuvud, formel, krav på utrustning, recept procedur och övrig information. Ett recept innehåller endast processrelaterad information för en produkt. Ingen schedulering finns med i receptet.

2.3.1 Recept-typer

De fyra typer av recept som beskrivs i ISA-88 är ”General recipe”, ”Site recipe”, ”Master recipe” och ”Control recipe”. Dessa recept beskriver en tillverkningsprocess olika noggrant. Ett företag kan själv bestämma vilka typer av recept de vill använda.

General recipe: Det generella receptet
Det generella receptet skapas på företagsnivå utan specifik kunskap om vilken utrustning som finns tillgänglig vid tillverkningsplatsen. Låt oss tex låtsas att ett visst företag skall tillverka apelsinjuice. I det generella receptet hade det då typiskt stått saker som att man behöver apelsiner, dessa ska skalas och sedan pressas. Två tankar på 2000 liter behövs och en av dessa måste gå att hetta upp till 212 Fahrenheit så att juicen kan homogeniseras. Låt oss anta att detta recept är skrivet på engelska.

Site recipe: Anläggnings-recept
Detta recept är anpassat och specifikt för en viss anläggning. Oftast är det tillverkat från det generella receptet, men det kan även vara skapat utan det. Låt oss anta att vi nu är i Sverige. Det generella receptet har blivit översatt till svenska och mer specificerat. Det står att tillverkningen ska ske i Lund och att apelsiner från Portugal skall användas eftersom dessa apelsiner är de bästa i området och färdsträckan är rimlig. Temperaturen för homogeniseringen har blivit ändrad till grader Celsius.

Master recipe: Huvud recept
Nu har receptet blivit ännu mer inriktat på ett område. Specifika processceller blir utvalda att hantera tillverkningen. Det står saker som ”Tank T512 skall användas vid uppvärmningen” och exakt hur många apelsiner som skall användas.

Detta recept kan antingen skapas från de två ovanstående eller helt oberoende. Det är dock ett nödvändigt recept för att produktionen skall fungera och utan detta recept blir det inga batchar.

Control recipe: Kontroll recept
Ett kontroll recept är en kopia på huvudreceptet. Skillnaden är att kontrollreceptet kan innehålla tillfälliga förändringar. Anta till exempel att man har märkt att denna månads omgång med apelsiner är surare än normalt och att man bestämmer sig för att öka mängden socker. Tyvärr har även tank T512 gått sönder och Tank T321 får hoppa in som vikarie. Denna typ av förändringar finns i kontrollreceptet.

2.4 Recept procedur

Det nämns i föregående avsnitt att alla recept skall innehålla fem kategorier av information: Dokumenthuvud, formel, krav på utrustning, recept procedur och övrig information.

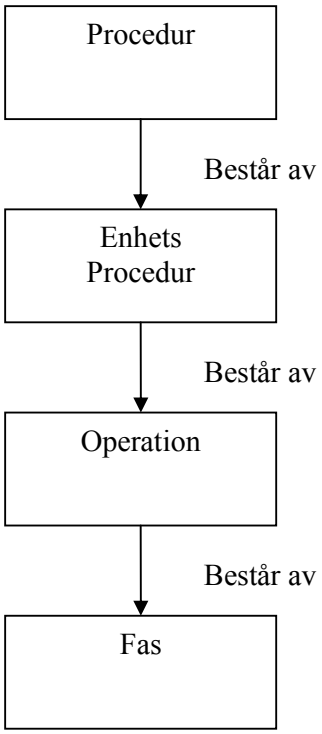
Den del som en operatör använder sig av och som har mest praktisk betydelse i en processcell är recept proceduren. All information om hur batchen skall tillverkas finns i denna procedur. När det är en procedur som styr tillverkningen så talas det om så kallad ”procedure control”. Det är denna typ av reglering som gör det möjligt för utrustningen att tillverka batcherna. En procedur har en underhierarki som består av ”enhets procedurer”, ”operationer” och ”faser”.

Procedure: Procedur. Styr samordningen av cellen. Låt säga att vi tittar på en procedur som heter ”Gör apelsinjuice”. Består av en mängd enhetsprocedurer.

Unit procedure: Enhetsprocedur. Styr en enhet i processcellen. Kan tex. vara ”skala apelsin”. Består av en mängd operationer.

Operation: Operation. Styr en ”equipment module” i en enhet eller alternativt en hel enhet. Tex. en kniv i enheten där ”skala apelsin” utförs. Operationen kallas kanske för ”rör kniv”. En operation består av en mängd faser.

Phase. Fas. Ger information till en ”control module”, equipment modul eller enhet. Operationen ”rör kniv” består kanske av faserna ”lyft kniv” och ”sänk kniv”.



Figur 2 – Hierarkin i en procedur.

Det är inte alltid som alla dessa finns med. Ett företag kan tex. bygga upp sin enhetsprocedur direkt av en mängd faser, utan att samordna dessa i så kallade operationer. Det blir dock bättre ordning och mer översikt om man bygger upp sina procedurer som ovan.

3. Novozymes

I syfte att ge en bättre förståelse för den uppgift som detta examensarbete försöker lösa, kommer följande avsnitt handla om företaget Novozymes, produktionsanläggningen i Lund samt den aktuella processcellen. En koppling till föregående kapitel om ISA-88 kommer att göras i det fall det är möjligt.

3.1 Företaget

Novozymes A/S är en bioteknologibaserad världsledare inom enzymer och mikroorganismer. Huvudkontoret är beläget i Köpenhamn och företaget har ungefär 4000 personer anställda. Ungefär hälften av dessa jobbar i Danmark och med inte mindre än 600 produkter är Novozymes teknologi inblandad i diverse slutprodukter från andra företag, alltifrån kläder och mat till mediciner.

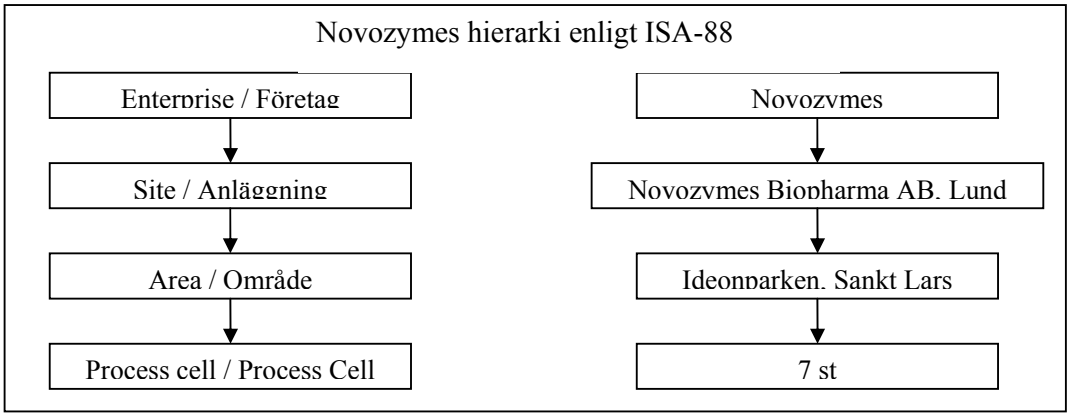
3.2 Anläggningen

Med ”anläggningen” syftas det på produktionsanläggningen i Lund. Det är på denna anläggning som examensarbetet bedrivs och denna produktionsanläggning tillhör Novozymes A/S. Produktionen i Lund går under namnet Novozymes Biopharma AB.

Hela Novozymes produktionsverksamhet är uppbyggd efter ISA-88 och produktionsanläggningen i Lund består av ett produktionsområde innehållande sju processceller. Dessa är:

- Buffertberedning
- Substratberedning
- Huvud fermentation
- Ymp fermentation
- Avdödningstankar
- Upparbetning
- Chromotografi

Kopplingen till ISA-88 syns i figur 3 nedan. Denna bild ger även en förståelse för företagets uppbyggnad.



Figur 3. Koppling mellan ISA-88 och Novozymes Biopharma AB.

Novozymes Biopharma AB har bara verksamhet på ett område i Lund och det är i Ideonparken vid Sankt Lars. Det finns dock företag som kan ha verksamhet på mer än ett ställe i samma stad. Därför sätts ”Lund” in under anläggningsnivån och områdesnivån blir Ideonparken vid Sankt Lars.

Tillverkningen av en produkt utförs genom en rad batchproduktioner där produkten passerar mellan olika processceller. Dessa batcher beskrivs av recept, som även dessa är utformade efter ISA-88. Denna anläggning tillämpar så kallad GMP, ”Good Manufacturing Practice”. Detta betyder att fabriken synas med jämna mellanrum och uppfyller en mängd regler och lagar vilka medför att tillverkningen och slutprodukten av aktiva läkemedel blir säker. Detta är viktigt bland annat eftersom Novozymes Biopharma AB får hantera patogener[1] av klass 2, men även eftersom kunder till Novozymes förväntar sig att få exakt det ämne de beställer. GMP försäkrar en validerad produktion av ett läkemedel som tillverkas efter en förutbestämd specifikation.

För tillfället (2006) byggs anläggningen ut och designas om eftersom en önskan om större produktionskapacitet har uppstått. Det är denna ombyggnad som ger upphov till ett behov av schedulering, så att den maximala produktionen på den nya anläggningen kan bestämmas och utföras.

3.3 Processcellen - Buffertberedning

Buffertberedningscellen var den processcell som blev utvald till examensarbetet. Anledningen var att denna cell delar ut buffert till hela produktionsanläggningen och att en förbättring i denna cell syns i hela produktionen. Cellen ansågs även lagom komplicerad för att kunna angripas under de 20 veckor som ett examensarbete tar. Här nedan beskrivs cellen.

Novozymes Biopharma AB följer ISA-88 och har delat in processcellen i en mängd enheter (units) enligt följande:

3 stycken blandningstankar med volymerna 1500 liter, 2500 liter samt 3000 liter.

- K1500C
- K2500B
- K3000A

En fyllningsledning för tillsättning av purified water (PW).

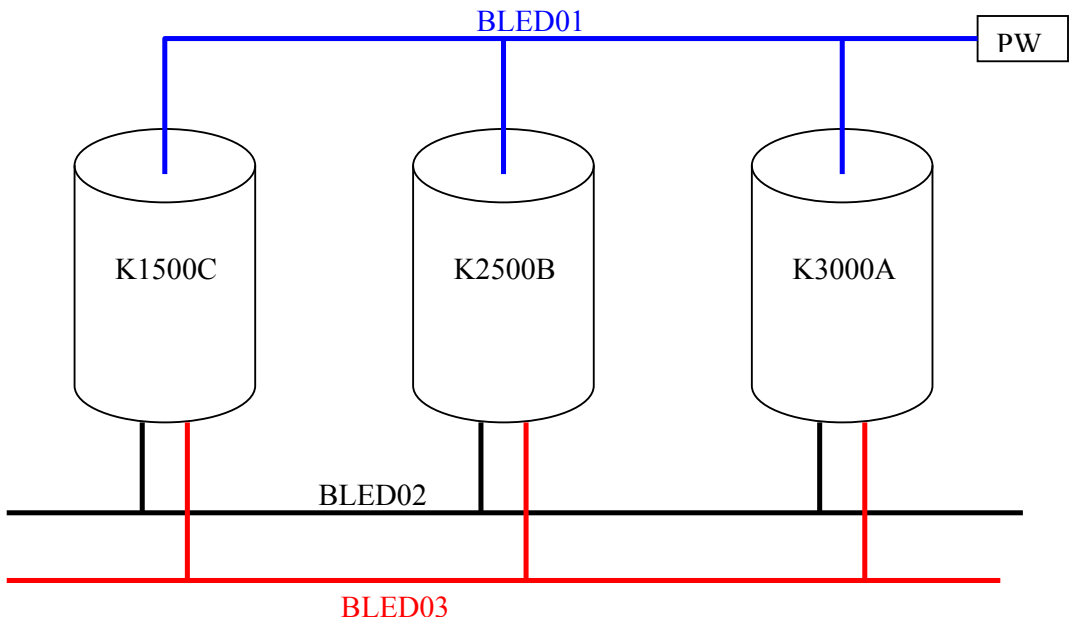
- BLED01

2 stycken transferledningar för att möjliggöra transport av bufferten från buffertberedningscellen till andra celler. I denna uppsats antas endast en ledning användas av buffertberedningscellen.

- BLED02
- BLED03

[1] En patogen är en mikroorganism (oftast en bakterie eller ett virus) som kan framkalla sjukdom hos en värdorganism. För att det inte ska förekomma någon risk för exempelvis personal vid hantering av dessa, krävs hög säkerhet.

Nedan visas en förenkling av processcellen, där endast enheterna är inritade.



Figur 4. Förenkling av buffertberedningscellen.

Det finns en del regler och begränsningar för hur systemet får och kan användas.

- Fyllningsledningen för PW (BLED01) har som maxkapacitet 6m^3/h. Dessutom är det endast möjligt att fylla på en tank i taget.

- Transferledningar (BLED02 och BLED03) har som maxkapacitet 5m^3/h och endast en blandningstank får använda en transferledning åt gången. Däremot kan två olika blandningstankar använda vars en transferledning samtidigt, men en hålltank i en annan processcell kan endast fyllas av en transferledning åt gången.

- Blandningstankarna fylls aldrig helt utan ligger som maximalt 200 liter under maxvolym. Detta betyder att K1500C rymmer 1300 liter, K2500B rymmer 2300 liter och K3000A rymmer 2800 liter. Det är dessa volymer som menas när det pratas om att en tank är full.

För att få lite koll på vilka tider som är inblandade, räknas fyllnadstiden fram med hjälp av maxkapaciteten för fyllningsledningen av PW och volymerna av tankarna. BLED01 levererar 6000 liter PW / h, så 1000 liter tar 10 min att fylla. Detta ger att:

- K1500C tar 13 minuter att fylla.
- K2500B tar 23 minuter att fylla.

- K3000A tar 28 minuter att fylla.

3.3.1 Operationer

Inom processcellen kan ett antal operationer utföras. Operationerna finns i en version för respektive blandningstank. Dessa beskrivs nedan.

Uppstart
Kontrollerar att systemet är klart för att börja användas.

Blandning med recirkulationsloop
Blandar en buffert med stora mängder salt med hjälp av en intern recirkulationsslinga. Slingan är till för att smälta ett saltblock som tillsätts av operatören.

Blandning utan recirkulationsloop (bara i kärlet)
Blandar en buffert med små mängder salt. Blandningen sker med en omrörare i kärlet.

Transfer av buffert, sköljning av ledning
Finns i en version för respektive tank som bufferten kan överföras till (15 st). Operationen överför bufferten i en bestämd blandningstank till någon annan tank i en annan processcell. Efter detta rengörs transferledningen.

Sköljning av tank och recirkulationsloop.
Denna operation ska användas efter varje blandning. Behövs dock ej om samma buffert skall tillverkas i samma tank ännu en gång. Operationen rengör tanken och recirkulationsloopen.

En noggrannare beskrivning av operationerna fås i bilaga 1.

3.4 Schedulering idag

Novozymes Biopharma AB har hittills inte använt sig av schedulering eller någon form av optimering. Hur tillverkningen av batcherna bör ske har med tiden framkommit rent empiriskt. Det viktiga har varit att ett fungerande system, uppbyggt på två skift, har funnits och detta uppkom med lite erfarenhet och ”trial and error”. Om Novozymes, pga den nya situationen, tvingas övergå till treskift uppstår det problem. Fler operatörer måste anställas vilket drar ner effektiviteten och lönsamheten. Därför finns ett önskemål om ett resultat, med den nya scheduleringen, där personalen fortsätter jobba i tvåskift.

Före ombyggnaden av fabriken fanns inte ett lika stort behov av schedulering. När mängden batcher per dygn ökar, ställs det även högre krav på buffertcellen att leverera tillräckliga mängder buffer. Även om också denna del av fabriken byggts ut, så krävs det att tillverkningen blir optimal för att allt ska gå ihop i slutändan. Ur företagets synpunkt finns det självklart ekonomiska vinningar att göra genom att se till att alla resurser används så bra som möjligt.

4. Schedulering

Med schedulering menas i detta sammanhang ett verktyg för att effektivisera tillverkningen. På svenska kan man säga schemaläggning. Begränsar man sig ännu mer till så kallad batch-scheduling kan man tänka sig att man ser en anläggnings enheter som svarta boxar med ingångar och utgångar. Man behöver inte bry sig speciellt mycket om vad som händer i en box. Bara om hur länge den är upptagen och hur många som får använda boxen samtidigt. Schedulering handlar om hur man effektivt ska besluta hur boxarna ska användas för att få ut så mycket som möjligt ur en anläggning, i ett visst avseende, samtidigt som inga begränsningar bryts. För att lyckas få fram en produkt måste en mängd jobb utföras. För att utföra jobbet krävs resurser som tex. material, utrustning, maskiner, personal och sist men inte minst tid. Eftersom det inte finns oändligt av dessa resurser, och de dessutom kostar pengar, vill man använda dem så effektivt som möjligt. Låter man en schedulering dela ut dessa resurser till jobben, är det meningen att dessa ska ha delats ut på bästa sätt om man ser på de villkor man vill ska vara uppfyllda.

Man kan dela in schedulering i två grupper. De som utförs i realtid, det vill säga algoritmer som bevakar en anläggning samt dess resurser och direkt ändrar om produktionen till den mest optimala vid varje punkt i tiden. Till exempel om en tank går sönder. Denna typ av schedulering kostar mycket pengar, är omständlig att driva igenom och det uppkommer en mängd pappersarbete för Novozymes om de skulle valt att implementera denna lösning. Eftersom de tillverkar läkemedel är kraven väldigt hårda på tillverkningen. När realtids-scheduleringen väl är installerad är det däremot en enastående lösning.

Den andra typen av schedulering är den som görs ”offline”. Dvs. endast en gång och bara för en typ av uppsättning inom tillverkningen. Denna schedulering ger ut en manual för hur resurserna ska användas och när. Följer man denna manual gör man tillverkningen så effektiv som möjligt. Denna metod är mycket billigare än den förra, eftersom inga fysiska förändringar behöver göras i anläggningen. Det är endast operatörerna som får tips om hur tillverkningen ska gå till.

4.1 Komplexitet

Schedulering är klassat som ett NP-svårt problem. Detta betyder att problemet tillhör en klass av problemställningar inom matematiken, som för närvarande inte kan lösas med någon känd metod. Man måste helt enkelt ta fram varje möjlig lösning och jämföra dem, vilket tar orimligt lång tid. NP står för ”Nondeterministic algorithm in Polynomial time”. Att få fram en lösning tar exponentiellt växande tid och mycket minne, i förhållande till storleken på problemet. Det uppstår en så kallad kombinatorisk explosion. På grund av detta kan en optimal schedulering vara svår, eller till och med omöjlig, att hitta. Låt oss ta några exempel.

- Om vi har en fabrik med m steg av n maskiner och vill låta en batch passera, så får denna batch en valmöjlighet på $\mathbf{n^m}$ olika vägar.

- Om vi har n batcher på en enhet får vi **n!** stycken sekvenser.

- Har vi n batcher på m enheter fås $\mathbf{(n!)^m}$ stycken olika scheduleringar. Detta betyder att om man har 4 maskiner och 5 batcher fås $(5!)^4 = 2.1 * 10^8$ olika scheduleringar.

Nästan alla typer av schedulering fungerar på följande sätt.

- Modellera verkligheten
- Optimera modellen i valfritt avseende
- Ta hänsyn till osäkerheter på lämpligt sätt

4.2 Modellering

Denna del av scheduleringsprocessen är väldigt viktig. Scheduleringen kan endast ta hänsyn till saker som inkluderas i modellen. På grund av detta vill man ha modellen så komplett som möjligt. Det är viktigt att alla begränsningar finns med, så att inte ett schema som egentligen ej är realiserbart konstrueras. Uppfyller modellen detta benämns den som ”korrekt”.

Saknad information kan leda till att den optimala lösningen ej kan genereras. En maskin som enligt modellen endast kan hantera ett jobb samtidigt, när den i själva verket kan hantera tre, blir ju en stor onödig flaskhals. Uppfyller modellen att all information finns med, kallas modellen ”fullständig”.

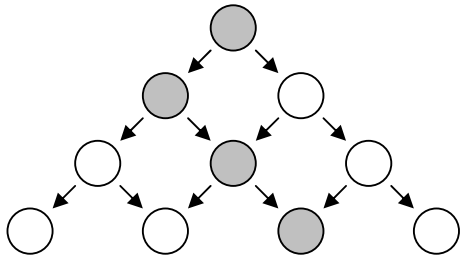
4.3 Optimering

Detta är huvudproblemet vid schedulering. Den algoritm som bestämmer en schedulering löser ett optimeringsproblem. Målet är att hitta en lösning som är både genomförbar och optimal. Tyvärr är det i de flesta fall omöjligt att hitta ett globalt optimum och de algoritmer som endast hittar en genomförbar lösning brukar därför också, i detta fall, ingå under begreppet ”optimerare”.

När det gäller schedulering handlar det om att göra en kombinatorisk optimering och söka genom ett träd med alternativ. Detta kan göras på en mängd sätt. Metoderna brukar delas upp efter hur de tar sina beslut.

4.3.1 Sökstrategier

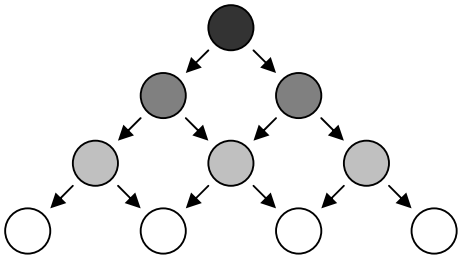
Ett träd med alternativ kan genomsökas på olika sätt. Antingen görs en så kallad ”djup först”-sökning, vilket betyder att algoritmen letar igenom en lämplig rutt hela vägen till löven, dvs de sista tillstånden. Med lämplig väg menas att algoritmen följer någon form av regel som har stor sannolikhet att generera ett bra resultat. Efter detta är klart kan man ”backtracka” och börja jämföra andra grenar. Fördelen med det här är att så fort den nya sammanlagda vägen blir större än den vägen som räknades ut allra först, kan man kapa hela grenen med underliggande tillstånd och på så sätt slippa testa en mängd vägar. (Man antar att alla kostfunktioner är positiva). Denna typ av genomsökning kräver ganska lite minne av en dator, men för att den skall fungera bra, krävs att någon tänkt till och kommit på ett smart sätt att välja ut den första vägen. Om man vid ”backtrackingen” hittar en bättre väg än den första skall givetvis detta värde användas för den



Figur 5. Illustration av ”djup först”-sökning.

fortsatta jämförningen av vägar. I det fallet då man har väldigt stora sökträd blir dock en bra första väg mycket viktig.

Ett annat alternativ är att man genomsöker trädet via så kallad ”bredd först”-sökning. Som namnet antyder är detta lite av motsatsen till det förra alternativet. Man kollar helt enkelt alla vägar från nivå till nivå och när man väl nått löven så är alla vägar redan utvärderade och den optimala vägen finns direkt till hands. Denna typ av sökning tar mycket minne och tid och används inte lika ofta som ”djup först”-sökningen. Ingen ”smart” regel eller heuristik används, men finns det en lösning kommer denna typ av sökning alltid att hitta den.



Figur 6. Illustration av ”bredd först”-sökning.

4.4 Hantering av osäkerheter

Det finns inte många algoritmer som tar hänsyn till denna aspekt av scheduleringsproblemet. Detta beror på att schedulering utan osäkerheter är ett så pass komplext problem att utökningen av osäkerhetsaspekten gör problemet i princip omöjligt att lösa. Trotts detta är det ett väldigt viktigt ämne som ej bör bortses helt ifrån vid användning av schedulering och rent förnuft kan eliminera några av osäkerheterna. I fallet med off-line schedulering spelar detta ämne mindre roll än vid realtids-schedulering.

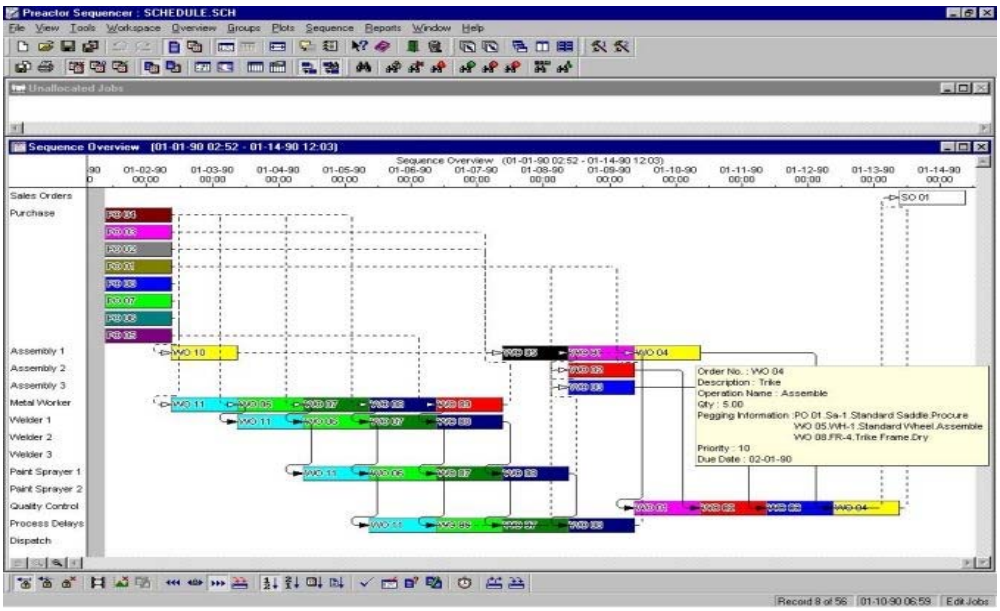
4.5 Verktyg

Under utförandet av detta exjobb letades det väldigt mycket efter färdiga program som kunde optimera en batchproduktion, men tyvärr kunde inget hittas. Det fanns verktyg för schedulering, men inget av dem hade någon form av algoritm för optimering. Endast ”constraints” eller begränsningar kunde modelleras. Ofta hade de en bra grafisk illustration av förloppet, men händelseföljden var tvungen att modifieras efter en människas förmåga, vilket kan vara väldigt svårt vid långa händelseförlopp med många enheter. Det verkar väldigt märkligt att dessa optimeringsprogram inte finns, med tanke på hur många företag som skulle kunna ha nytta av dem. Möjligtvis hålls program som dessa, privata inom företagen för att få ett övertag om konkurrenterna, eller så har mitt letande inte varit tillräckligt bra. Överlag verkar det inte finnas någon färdig metod eller algoritm för hur man ska gå till väga för att bekämpa dessa typer av optimeringsproblem. Det finns många lösa teorier i olika böcker, men inga handlingskraftiga, praktiska förslag på hur problemet skall lösas.

Ganska tidigt i examensarbetet uppkom en egen idé om hur problemet skulle kunna lösas i just detta fall (se kap ”Egen lösning”). Denna idé ansågs så pass bra att det var värt att jobba vidare med den. På grund av detta lades efter ett tag mer tid på den egna idén, än på att fortsätta leta efter färdiga program. De program som hittades och ansågs vara tillräckligt intressanta för att tas upp var ”Preactor” från ”Preactor International” och ”Schedule Pro” från ”Intelligen Inc”. Dessa program kommer kort att beskrivas nedan. Endast ”Preactor” har testats personligen då Novozymes inte ansåg det vara någon idé med att skaffa en testlicens till ”Schedule Pro”, efter vad som angavs av informationen på ”Intelligen Inc”:s hemsida. Försök till att ange några för och nackdelar med varje program görs i slutet av de följande kapitlen.

4.5.1 Preactor

Detta program görs av ”Preactor International” som är beläget i England. Företaget satsar mycket på forskning och utveckling och har många stora kunder inom bland annat process- och tillverkningsindustrin. Programmet benämns som ett schedulings-verktyg och bygger på en metod som kallas ”Finite Capacity Scheduling”. Som namnet antyder är detta en metod för att ta hänsyn till att man har begränsad kapacitet vid sin schedulering. Det finns inte enbart ett accepterat sätt att utföra ”Finite Capacity Scheduling” på och vissa av metoderna är företagshemligheter. Oftast tar metoderna hänsyn till så kallade ”constraints”, det vill säga, programmet varnar om man försöker planera in två jobb på samma maskin samtidigt, när maskinen enbart kan hantera ett jobb åt gången. Ett exempel på hur programmet ser ut grafiskt kan ses nedan. Bilden visar vyn som beskriver jobbens placering på de olika resurserna. Tidsaxeln är horisontell och riktad åt höger. De olika färgerna gör det lätt att följa ett jobbs väg över resurserna, i tiden. Det går även att markera ett jobb och få en ännu tydligare bild på hur detta jobb är planerat.



Figur 7. Vy från scheduleringsverktyget Preactor.

En annan finess som metoden kan ta hänsyn till är prioriteter. Tyvärr kan det uppstå onödiga tider mellan jobb, då resurserna inte används. Metoden utför alltså inte någon form av optimering utan lägger enbart på jobben på resurserna enligt vissa regler. Det börjar pratas om att nya så kallade genetiska algoritmer ska kunna integreras i metoden för att till exempel minimera den totala tiden, men än så länge är dessa angreppspunkter på metoden i väldigt tidiga stadier. Preactor utför alltså inte någon form av optimering. Däremot ger programmet en bra överblick och kan hjälpa planeraren att undgå onödiga fel. Man ser även klart och tydligt när arbetet kommer att vara färdigt enligt den befintliga scheduleringen och saker som material och personal kan tas med i beräkningen. Tyvärr krävs det mycket tid av scheduleraren att prova sig fram till en så bra lösning som möjligt, men relativt ofta letar antagligen företag efter en tillräckligt bra lösning istället för den bästa lösningen. Givetvis skulle företagen i så fall kunna spara mycket tid, pengar och resurser på att ändra till den bästa

lösningen, men än så länge klarar Preactor inte av att leverera denna. Användarvänligheten är sådär och det tar ett tag innan man kommer in i programmet. Ska man göra avancerade saker behövs antagligen någon form av kurs.

Fördelar:

- Tydlig översikt över scheduleringen.
- Hjälp till scheduleraren.
- Möjlighet till prioriteter på jobben.
- Kräver ej mycket datorkraft.
- Möjlighet att ändra vissa visuella aspekter samt menyer för att få ett mer personligt program.

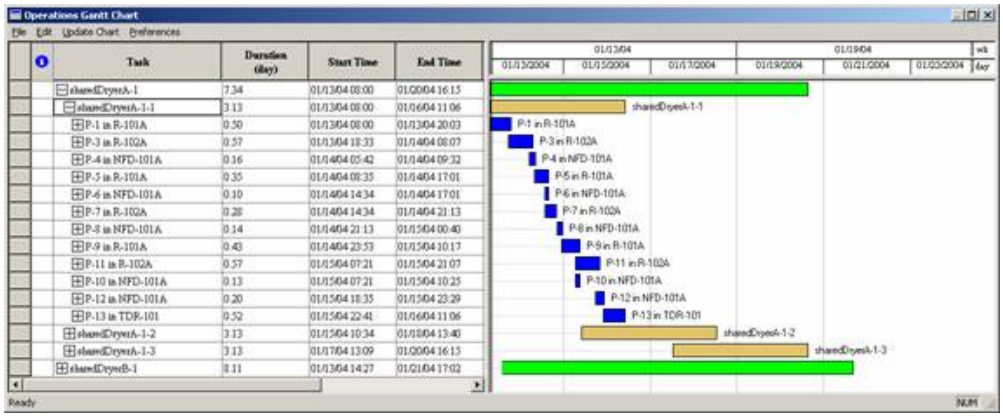
Nackdelar:

- Ganska rörigt att använda.
- Krävs en inlärningsperiod för att lära sig programmet.
- Optimerar ej.

Man kan läsa mer om finite capacity scheduling på http://www.ifm.eng.cam.ac.uk/dstools/process/fcs.html och mer information om Preactor finns på http://www.preactor.com.

4.5.2 Schedule Pro och Superpro Designer

Dessa program är skapat av ett företag vid namn ”Intelligen Inc.”. Grundarna har alla Ph.D från MIT i Michigan och företaget skapades 1992 med målet att bidra med ett bra hjälpmedel till de företag som hade behov av schedulering i sin verksamhet. Även dessa program bygger på metoder för ”Finite Capacity Scheduling” och har ingen optimering inbyggd. Det finns möjligheter att konstruera så kallade Gantt-charts. Det vill säga stapeldiagram för illustrativ visning av hur resurserna används (se http://en.wikipedia.org/wiki/Gantt_chart). Även möjligheter för simulering av processflöden, samt utföring av ”debottlenecking” finns inbyggt. I stort sett är detta program ganska likt Preactor, med skillnaden att det finns lite mer möjligheter att designa och simulera processer om man införskaffar det stora paketet med både Schedule Pro och Superpro Designer. Användarvänligheten kan tyvärr ej betygsättas då programmet inte testats personligen. Vill man ha fullständiga licenser för både Schedule Pro och Superpro Designer, kostar det 16995$ per uppsättning (Mars 2007) och priset per kopia blir något billigare om man beställer fler kopior.



Figur 8. Vy ur scheduleringsverktyget Schedule Pro.

Fördelar:

- Finns möjlighet för avancerade användare att förbättra scheduleringen genom att ändra i scheduleringsalgoritmen.
- Genererar mycket statistik.
- Möjlighet till simulering av processer.

Nackdelar:

- Optimerar ej.

4.5.3 Slutsats

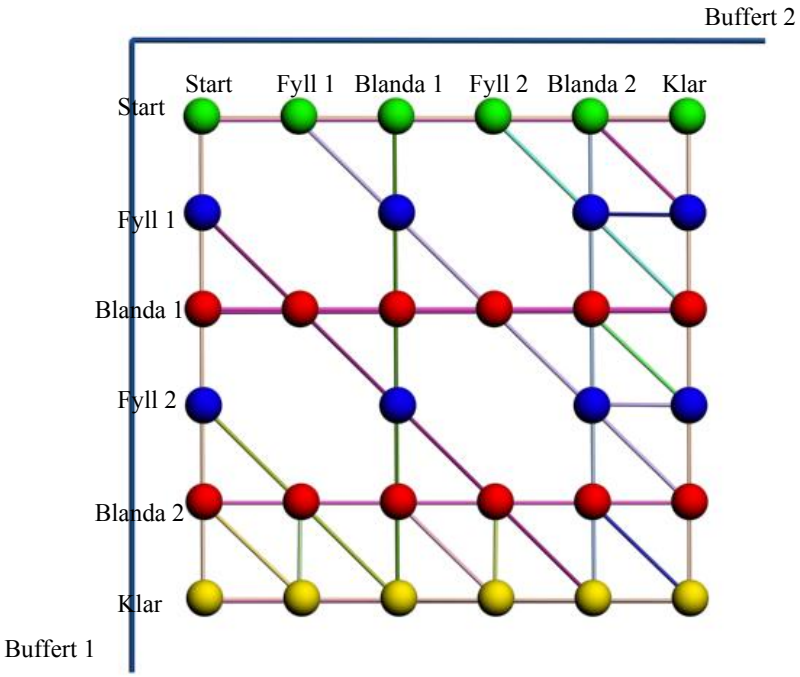
Novozymes Biopharma AB kan säkerligen ha nytta av program som Preactor och Schedule Pro. Tyvärr har inget av dem förmågan att optimera tillverkningen. Trots detta kan de vara värdefulla verktyg till personen som planerar och schedulerar. Vilket av programmen man bör välja är svårt att säga, eftersom programmen är ganska lika gällande specifikationerna. Båda företagen satsar mycket på forskning och utveckling och båda erbjuder mycket support och kurser. Det går att få tag i testversioner till båda programmen, men det var lättast att få tag i en testversion av Preactor.

5. Egen lösning

På grund av svårigheter att hitta lämpliga program och metoder för att lösa uppgiften, gjordes ett försök på egen hand genom en egen idé.

5.1 Att beskriva cellen

Början till idén uppkom efter ett föredrag av Avenir Kobetski från Chalmers. Föredraget handlade om optimering av tillverkningsceller genom diskreta händelsemodeller. En sådan modell är den så kallade ”finita automata”-modellen. Vad som beskrevs var ett antal tillverkningsrobotar som rörde sig i samma rum. I vissa områden av rummet fanns det risk för att mer än en robot befann sig samtidigt och därmed uppstod även en viss möjlighet att dessa robotar skulle kollidera. Detta var dåligt och skulle förhindras och modellerades därför med hjälp av en automata. Snabbt beskrivet är en automata en samling tillstånd förbundna med riktade pilar (övergångar). En tid efter föredraget uppstod idén att man kanske skulle kunna beskriva buffertberedningscellen på samma sätt som robotarna. Vi befann oss i ett begränsat rum, där mer än en buffert tillverkades samtidigt. Dessa kunde befinna sig i samma område (bland annat tankarna) och krocka, vilket inte var önskvärt. I buffertberedningscellen tillverkades alla buffertar på samma sätt. Det enda som kunde skilja var lite olika tider (tex påfyllnadstider och blandningstider). Genom att dela upp tillverkningsstadierna för en buffert i lämpliga, generella delar som tex. fyllning 1, blandning 1, fyllning 2 osv, kunde en karta med tillstånd tillverkas, om man satte tillstånden för de olika buffertarna enligt kartesiska koordinater (buffert1,buffert2,buffert3). Till exempel skulle (start,f1,b1) kunna beskriva att buffert 1 befinner sig i starttillståndet, buffert 2 håller på med sin första fyllning och buffert 3 håller på med sin första blandning. Ritade man upp detta grafiskt för två tankar såg det ut såhär, dvs det som egentligen har uppstått är en graf.



Figur 9. Graf som beskriver tillstånden i en cell med två tankar.

Tillståndet (f1,f1) försvinner pga att endast en tank kan fyllas i taget, så denna händelse kan ej ske. Likaså för alla andra händelser som fyller två eller fler tankar samtidigt. Sträcken mellan händelserna har en riktning som ej är utritad. Man kan endast hoppa vidare så länge man ökar värdet på någon av axlarna, dvs höger, nedåt eller snett nedåt höger. Sträcken mellan tillstånden kan tilldelas en kostfunktion i form av tid, så dessa kan ses som en sträcka mellan tillstånden. En tredje buffert kan introduceras längst den tredje axeln och en kub av tillstånd uppstår. Avenirs behandling av sitt robot-problem genom en automata ledde alltså fram till att jag försökte beskriva ”min” cell med tillstånd förbundna med riktade pilar. Man kan dock ej kalla min modell för en automata, utan det bästa sättet att beskriva den är som en graf med tillhörande grafteori. Hur som helst var detta ett stort steg framåt då jag plötsligt fick en modell av cellen som gick att bearbeta och jobba med. Tack Avenir!

5.2 Hur fungerar en gps?

När väl grafen hade uppstått började funderingarna kring hur man skulle kunna hitta den snabbaste vägen från översta vänstra hörnet (start,start) ner till lägre högra hörnet (klar,klar). Jag fick för mig att man skulle kunna se tillstånden som städer förbundna med vägar av olika längd. Vilken var den kortaste vägen? Detta är ju precis vad programvaran i en gps-mottagare räknar ut. Så efter mycket letande och lite tur hittade jag ”Dijkstras algoritm” som är en av de viktiga algoritmer som finns inom grafteori. Med hjälp av denna algoritm hittar man den snabbaste vägen (minimal kostfunktion) mellan två punkter. Enda kravet är att ingen av delvägarna har en negativ kostfunktion. Det fanns givetvis fler algoritmer för detta ändamål och det kan hända att någon annan algoritm är snabbare. Dijkstras algoritm försäkrar dock att den snabbaste vägen alltid hittas och algoritmen kommer att beskrivas närmare senare i detta kapitel. De flesta gps:er bygger inte enbart på denna algoritm. De räknar både från start och från mål och försöker hitta en väg som ”krockar” på mitten. Detta fungerar pga att man har gett olika vägar olika prioritet. T.ex. har en motorväg högre prioritet än en mindre väg. Några stycken bra vägar utvärderas och den bästa väljs. Detta betyder att inte alla vägar behöver hittas, men även att det inte är säkert att den bästa vägen är med bland slut-urvalet. Hur som helst är det kanske rimligt att försöka hitta en lösning med denna metod istället om man försöker schedulera processer med väldigt stort urval.

5.3 Programmet

När väl idén hade uppstått kvarstod utmaningen att skriva ett lämpligt program. Att hitta fakta och hjälp om Dijkstras algoritm visade sig inte vara speciellt svårt och programmet kunde göras relativt enkelt. Graferna skrevs från början för hand genom att göra en textfil med startdestination, slutdestination och kostnad för alla linjer som förband punkter. Detta var inte heller så svårt, ända tills det blev dags att generera grafen för tre tankar. Detta skulle bli en kub med omkring 120 punkter och varje punkt kunde gränsa till sju andra som mest. Det var helt enkelt dags att skriva ett program för genereringen av graferna också. Slutligen skulle alltså programmet bestå av två delar.

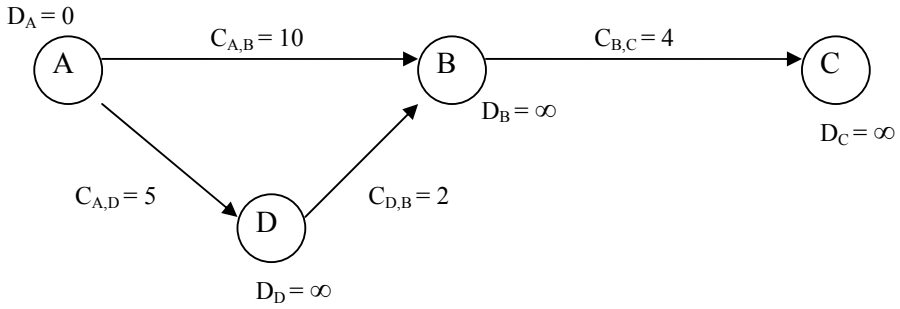
- **Autogen** - En del för att generera graferna till buffertcellen om man antog att tillverkningsstegen för de olika buffertarna var identiska.

- **Buffer** – Programmet som körde Dijkstras algoritm på den genererade grafen från Autogen.

Programmen skrevs i Java i programmeringsverktyget Eclipse.

5.3.1 Dijkstras algoritm

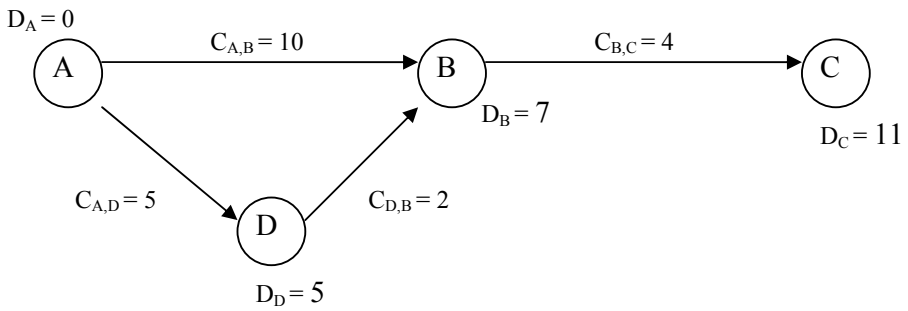
Denna algoritm tar sig igenom en hel graf och har koll på vad den kortaste vägen från en punkt, till alla andra punkter, är. Ett tillstånd kallas för en nod och vägen emellan två noder kallas för en länk. Alla länkar är riktad och försedda med en länkkostnad $C \geq 0$. Den kortaste vägen från en nod till en annan har en kostnad som benämns $D_{slutnod}$. Den nod man väljer som startnod får D = 0. Alla andra börjar med D = ∞. Låt säga att vi vill titta på följande graf och att vi vill ta oss från A till C. Då ser det ut såhär i början av algoritmen.



Figur 10. Beskrivning av Dijkstras Algoritm.

Kostnaderna C är kända och alla D är satta till ∞, utom för startpunkten A där D_A=0. Avståndet från A till A är ju noll. För att lättare kunna beskriva hur algoritmen fungerar tänker vi oss en ögonglob som rullar runt och tittar i de olika noderna. Vi tänker oss även att varje nod har en variabel som kallas scratch. Scratch = 1 om ögongloben har varit i noden och 0 om den inte varit där än.

Algoritmen börjar titta i A och ser efter vilka noder som är intilliggande. I detta fall syns nod D och B. Dessa två noder får nu sin totala kostnad D ändrad till $D_A + C_{A,noden}$ om denna kostnad är mindre än den nuvarande D (i detta fallet ∞). Detta ger att $D_B = 10$ och $D_D = 5$. Båda dessa noder läggs till i en så kallad prioritetskö efter länkarnas totala kostnad D. Minst överst, vilket ger att D har högre prioritet än B eftersom $D_D = 5$ och $D_B = 10$. Scratch för nod A sätts till 1. Nu rullar ögongloben till D eftersom den är överst i kön. De intilliggande tillstånden till D (alltså B) identifieras och D_B sänks till $D_D + C_{D,B}$ eftersom detta värde (7) är lägre än det nuvarande (10). B läggs till ännu en gång i prioritetskön. Överst även denna gång eftersom 7 < 10. Scratch för D sätts till 1 och ögongloben rullar från D till B. Det intilliggande tillståndet är C och D_C sätts till $D_B + C_{B,C}$ eftersom 11 < ∞. C läggs till sist i listan eftersom 11 > 10. Scratch för B sätts till 1 och nästa nod i prioritetskön är B igen. Eftersom scratch för B är 1 plockas denna nod bort direkt ur kön och C blir nästa valda nod. Det finns inga angränsande tillstånd, så inget läggs till i kön och inga slutliga kostnader finns att ändra. Scratch sätts till 1 och nästa nod i kön väljs. Eftersom det inte finns någon nästa nod är algoritmen klar. Algoritmen kör på så länge antalet besökta noder < totala antalet noder och under förutsättning att det finns noder i prioritetskön. Den nya grafen ser ut såhär.



Figur 11. Beskrivning av Dijkstras algoritm.

Den slutliga kostnaden D beskriver nu den minimala kostnaden för att ta sig från startnoden till den befintliga noden. Alla noder har vars en D-kostnad och följer man hela tiden den minsta D-kostnaden kommer man få den optimala vägen. Vill man till exempel ha kortaste vägen från A till C vet man direkt att den är 11, men för att se vilken denna väg är får man titta från början. När man är i A kan man välja mellan $D_B = 7$ och $D_D = 5$. Eftersom 5 är det minsta hoppar vi till D. Eftersom det inte finns fler valmöjligheter är sedan $D_B = 7$ det minsta och sist $D_C = 11$. Vi ska alltså gå från A till D till B till C och kostnaden är 11.

Eftersom storleken på en prioritetskö kan vara lika stor som antalet länkar |E| och det max kommer finnas |E| insättningar och borttagningar av länkar i kön blir körtiden av storleken O(|E|log(|E|)).

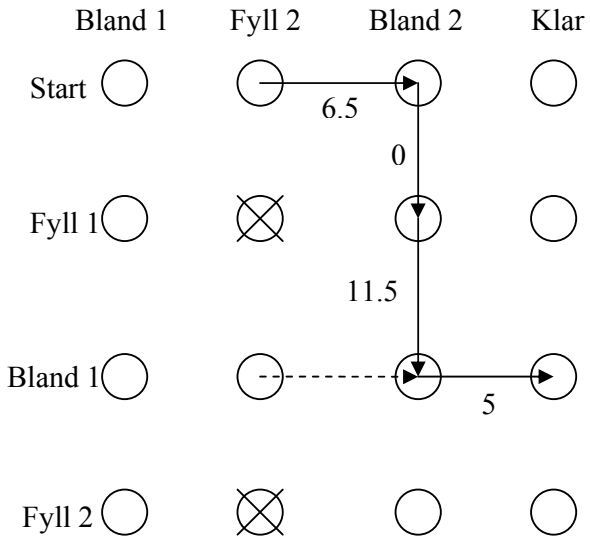
Nackdelen med Dijkstras är att den söker igenom hela grafen, vilket tar tid.
Fördelen är att man kan vara säker på att den optimala vägen har hittats.

5.4 Problem med modellen

Teorin i sin helhet verkar fungera väldigt bra. En tid in i exjobbet hittades dock brister i teorin bakom modellen av buffertberedningscellen. Förhoppningen är att det finns en smart lösning som eliminerar alla felaktigheter på en och samma gång, då de egentligen är sammanhängande.

5.4.1 Iakttagelse 1

Problemet som uppstår är att tiderna mellan tillstånden är beskrivna som konstanter. Detta visar sig efter lite klurande vara fel. Beroende på från vilket håll man kommer till det aktuella tillståndet, kan det ibland var tvunget att ändra på tiden därifrån. Vi tar ett exempel.
Nedan syns ett utdrag ur en av modellerna. Grafen beskriver en del av samverkandet mellan buffert 1 och buffert 2. Vissa tillstånd är kryssade på grund av att de är förbjudna. Anta att vi rör oss enligt pilarna och att siffrorna intill anger tiden det tar (kostfunktionen) för att få använda pilen. Den nedersta pilen visar att det tar 5 minuter att gå horisontellt ifrån (Bland1,Bland2) till (Bland1,Klar). Detta gäller generellt för att få hoppa höger ifrån den lodräta ”Bland 2” stapeln. Om vi går enligt pilarna visar det att vi hoppar in i ”Bland 2”-stapeln ifrån ”Fyll 2”-stapeln efter 6.5 minut.



Figur 12. Illustration av problem i tillståndsbeskrivningen.

Det tar sedan ingen tid att starta fyllningen av buffert 1, dvs hoppa nedåt till ”Fyll 1”-raden, men för att gå ifrån ”Fyll 1”-raden till ”Bland 1”-raden måste vi betala 11.5 minut. Detta betyder att när vi kommer fram till (Bland 1, Bland 2) måste vi ha befunnit oss i ”Bland 2”-stapeln i 11.5 minuter. Eftersom blandning 2 endast tar 5 minuter att göra måste den ha hunnit bli klar under de 11.5 minuterna vi har fyllt på buffer 1! Det ska alltså inte kosta något att ta ett steg till höger från (Bland 1, Bland 2). Hade vi däremot kommit ifrån (Bland 1, Fyll 2) rakt höger till (Bland 1, Bland 2) så har vi inte befunnit oss någon tid i ”Bland 2”-stapeln och ytterligare ett steg åt höger bör ta 5 minuter (streckade pilen).

5.4.2 Iakttagelse 2

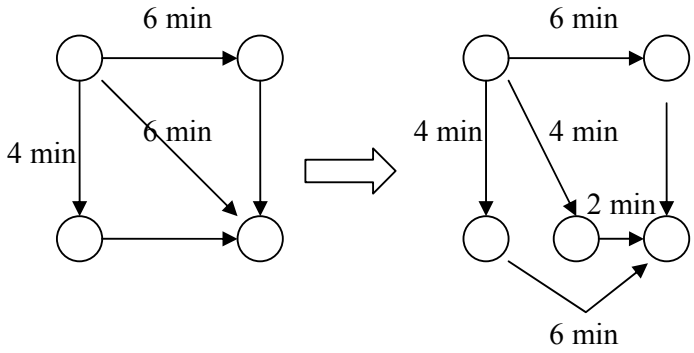
Låt oss anta att problemet med konstanta tider blir löst. Då kvarstår fortfarande ett annat problem. Även om kostnaden blir ändrad i förhållande till ingångsvägen, kommer ”Bland 2”-stapeln vara låst av ”Fyll 1”-raden. Blandningen är klar efter 5 minuter, men fyllningen av buffer 1 pågår i 11.5 minut. Egentligen räcker det alltså inte att iaktta att ”Bland 2”-stapeln är klar och att ett steg åt höger borde kosta 0 minuter. Vi vill ju kunna gå vidare åt höger direkt när blandningen är klar. Annars måste ju blandningen vänta i 11.5 - 5minuter = 6 minuter på att fyllningen ska bli klar. Likadant blir det för alla sneda pilar. Kostnaden för att ta en sned väg är den dyraste av de två vägar som bygger upp den. Den dyraste vägen kommer alltså alltid att hindra den billigaste från att gå vidare.

5.5 Förslag på lösning

5.5.1 Flytande upplösning

För att lösa problemen ovan, speciellt problemet i ”iakttagelse 2”, var målet att dela upp sträckorna (kostfunktionerna) i lika delar så att sneda hopp inte gjorde att något tillstånd skulle behöva vänta. Om man tänker sig en tvådimensionell modell där ett hopp åt höger kostar 6 minuter och ett hopp nedåt kostar 4 minuter, skulle man kunna dela upp den längsta vägen i två bitar. En bit som är 4 minuter och en resterande bit på 2 minuter. Då skulle hoppet

nedåt inte behöva vänta på hoppet åt höger eftersom det lagts in en extra punkt efter 4 minuter. Det blir lättare att förstå om det illustreras med en bild.



Figur 13. Förslag på lösning till problemet med tillståndsbeskrivningen.

Nu är det möjligt att nå fram till den nedre raden efter 4 minuter samtidigt som en förflyttning i sidled ändå har skett. Tyvärr uppstår ett nytt problem. Det är dumt att vara bunden till ursprungspunkterna för att kunna hoppa vidare. Självklart måste man kunna hoppa vidare direkt ifrån det nybildade tillståndet till ett annat tillstånd längre ner. Annars har ingen vinst blivit gjord eftersom man ändå inte kan fortsätta neråt förrän de 2 extra minuterna har gått och tillståndet längst ner till höger, i bilden ovan, är ockuperat. Det nya tillståndets ”från”-vägar måste alltså beräknas. Det är bara det att det måste kunna finnas vägar till nybildade tillstånd längre ner, som inte är bildade än vid beräkningen av ”från”vägarna. Dessutom är kanske den nya lilla 2 minuters vägen åt höger i bilden ovan, längre än nästa hopp nedåt och en ny uppdelning måste ske. Det kommer att bildas enormt många fler punkter och kombinationer av hopp. Tidsförskjutningen kommer att förskjutas framåt ända tills det sista tillståndet. Om inte summan av tiderna horisontellt är en heltalsmultipel av summan av tiderna lodrätt, kommer det dessutom inte att gå jämnt ut.

Denna lösning verkar svår att implementera programmeringsmässigt.

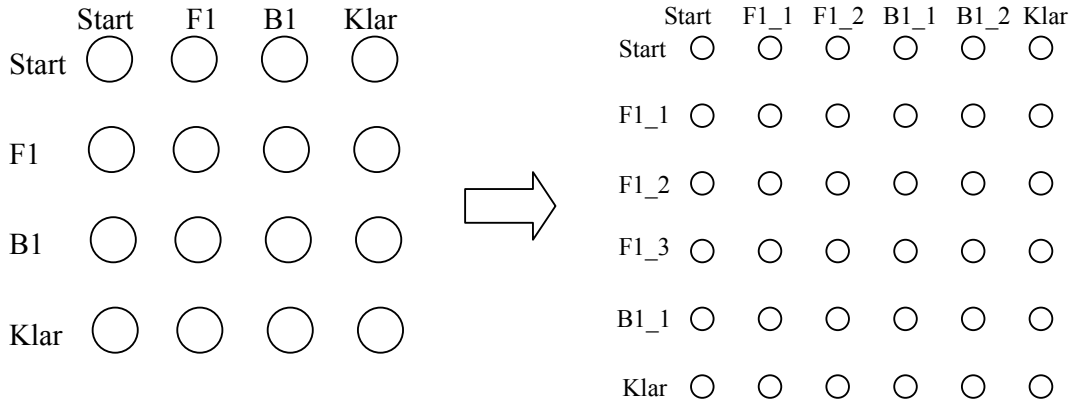
5.5.2 Fast upplösningar

Ett sätt att kringgå problemen som uppstår i den ”flytande” metoden ovan, är att bestämma sig för en fast upplösning. Hur noggrant kommer man att beskriva tiderna som ingår i modellen? Bestämmer man sig för att beskriva varje del av processen på en minut när, finns det ingen anledning för modellen att vara noggrannare än så. Varje hopp sätts då till en fast kostnad som är lika stor som alla andra. Till exempel en minut (om det är upplösningen man bestämt sig för). Har man en fyllning F1 på 4 minuter bildas alltså fyra punkter med en minuts mellanrum. Dessa blir F1_1, F1_2, F1_3 och F1_4. På detta sätt blir även alla diagonala hopp 1 minut och ingen behöver vänta på någon annan. Vill man ha det noggrannare ändrar man upplösningen till 30 sekunder, men detta gör att man får dubbelt så många punkter i varje riktning och eftersom vi har en tredimensionell ”kub” av tillstånd kommer det bli väldigt många fler vägar att gå. Alla problem i iakttagelserna ovanförsvinner dock.

Exempel:
Två dimensioner. Processen går: Start, F1, B1, Klar

Kostnader i minuter:

Horisontellt	0,	2,	2,	ingen
Lodrätt	0,	3,	1,	ingen



Figur 14. Förslag till lösning av problemet med tillståndsbeskrivningen.

Eftersom alla hopp i den nya grafen är lika stora, kan man sätta kostnaden för den till 1 och sedan multiplicera svaret med den riktiga kostfunktionen, till exempel 30 sekunder. Då kan man använda en snabbare metod än Dijkstras algoritm för att hitta vägen. Det är en metod som endast räknar antalet förflyttningar för att ta sig till det önskade tillståndet. Kostnaden för att ta sig från ”start” är dock 0, eftersom det i Novozymes fall inte tar någon tid att sätta igång processen. Detta måste korrigeras för.

Denna metod implementerades och testades. Det visade sig att det tog under en sekund för Dijkstras algoritm att räkna på en kub med runt 19000 tillstånd. Det som tog tid var att generera alla tillstånd och vägar för den modifierade versionen, av det egna programmet Autogen. För detta program tog det runt 10 sekunder att bli klar med modellen. Det finns dock mycket av det programmeringsmässiga som går att förbättra och det borde gå att använda den här metoden för schedulering, så länge man inte har allt för höga krav på tidsupplösningen i kombination med stora tider.

6. Svar på uppgifterna

De svar som presenteras här har tagits fram av det egenkonstruerade programmet. Detta program har inte hunnit testas fullt ut och det finns i detta stadium fortfarande risk för kvarliggande buggar. Det kan även hända att tiderna som används för uträkningen måste korrigeras. Dessa tider har tagits fram med hjälp av tiderna som presenteras i Control Draw, samt med hjälp av intervjuer med de operatörer som normalt sköter buffertberedningscellen. De fakta och de värden som används presenteras i varje uppgift.

6.1 Scenario 1

Vad är den maximala mängden buffert som kan göras på 24 timmar? (Volym och antal)

6.1.1 Fakta

I beräkningen av scenario 1 har tillvägagången för tillverkningen beskrivits enligt punkterna nedan. Två ”fyllning 1”, två ”fyllning 2” eller en blandning av ”fyllning 1” och ”fyllning 2” får inte ske samtidigt, eftersom det endast är möjligt att leverera PW till en tank i taget. ”Transfer” använder också PW och kan således inte heller köras samtidigt som någon fyllning. Endast en ”Transfer” kan köras åt gången. Dels på grund av PW, men även på grund av ett antagande att det endast finns en tillgänglig transferledning. Den sista punkten, ”Rengöring”, använder PW och måste alltså köras när ingen annan process använder PW.

Punkt ett till och med fem, ingår i recept proceduren 5.2 eller 5.3 (bilaga 1), beroende på vilken metod man använder till blandningarna. Sedan följer punkt sex, ”Transfer”, som är beskriven i 5.4 (bilaga 1) och slutligen ”Rengöring”, som finns med i 5.5 (bilaga 1).

- Start
- Fyllning 1 – 60% av (tankens slutvolym – 200 liter). Påfyllnadshastighet 6m^3/timme.
- Blandning 1
- Fyllning 2 – 40% av (tankens slutvolym – 200 liter). Påfyllnadshastighet 6m^3/timme.
- Blandning 2
- Transfer
- Rengöring (CIP)
- Klar

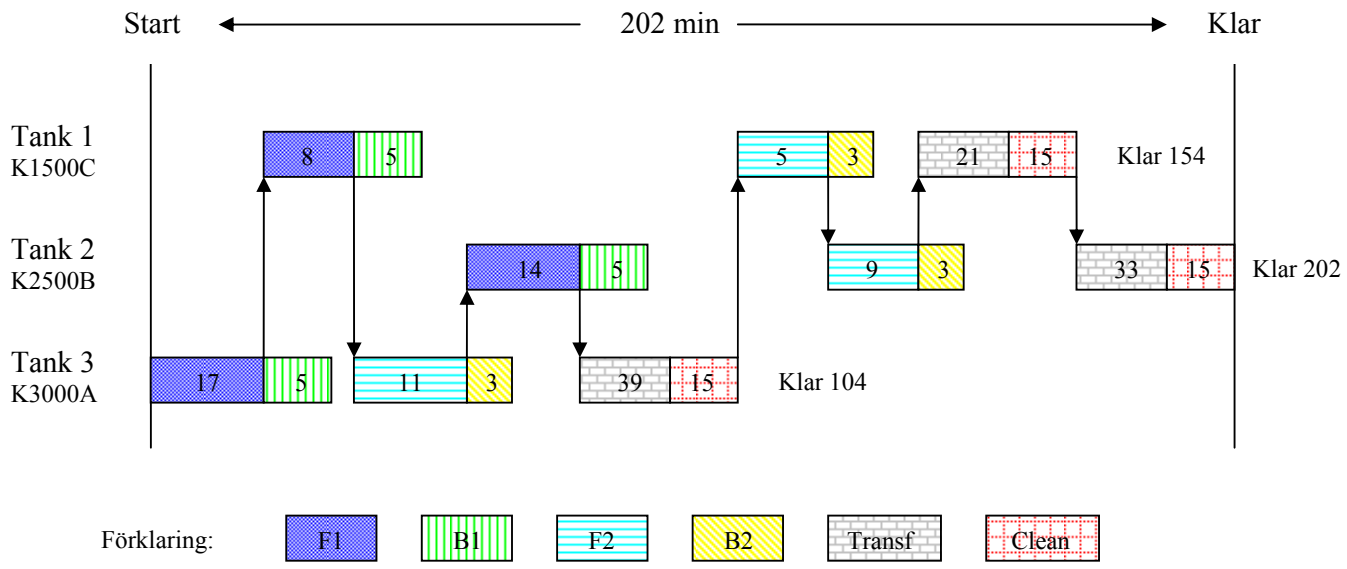
6.1.2 Tider [minuter]

Tiderna nedan anges i minuter och är avrundade uppåt. Det är viktigt att här finns en marginal som gör att operatörerna slipper stressa. Om operatörerna inte får den tid som de behöver, så ökar risken för ett mänskligt misstag och att fel uppstår.

Tillstånd	Buffert 1 (K1500C)	Buffert 2 (K2500B)	Buffert 3 (K3000A)
Start	0	0	0
F1	8	14	17
B1	5	5	5
F2	5	9	11
B2	3	3	3
Transf	21	33	39
Clean	15	15	15
Klar	-	-	-

6.1.3 Schedule

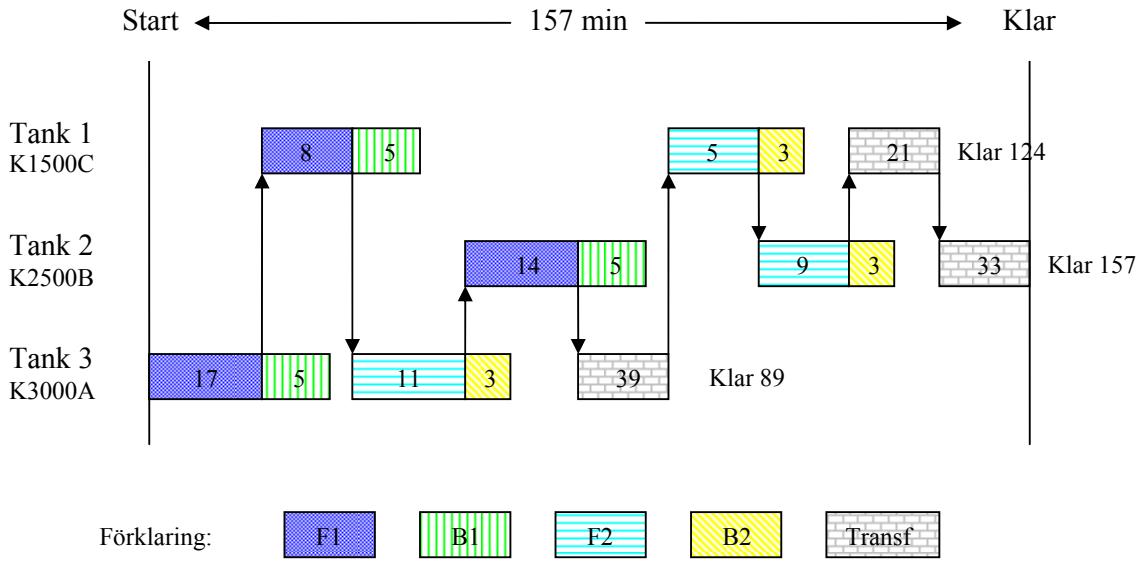
Följande schedulering skapas av programmet. I illustrationen nedan är blocken ej skalenliga.



Figur 15. Illustration av scheduleringen för scenario 1 (med CIP).

Pilarna visar hur PW förflyttar sig. I bilden syns att den begränsade resursen (PW) används optimalt, dvs hela tiden. Denna resurs väntetid ska vara noll vid optimal användning. Det syns även att den stora tanken blir klar först (efter 104 minuter), sedan den lilla tanken (154 minuter) och sist mellantanken (202 minuter). Totalt tar det 202 minuter att köra alla tre tankarna. Ska samma process köras gång på gång är det bara att köra ovanstående schema gång på gång. Inga förskjutningar kan ske eftersom den begränsade resursen används kontinuerligt.

Om det finns möjlighet att göra många likadana buffertar i rad så kan sista rengöringsoperationen ”Clean” hoppas över. Då blir scheduleringen korrigerad enligt nedan.



Figur 16. Illustration av scheduleringen i scenario 1 (utan CIP).

På detta sparas totalt 45 minuter, om ”clean” tar en kvart. Ska beredningen köras många gånger blir det ganska mycket tid som sparas in.

6.1.4 Svar:

Om man måste köra ”Clean” i slutet så tar en omgång buffert (tre tankar) 202 minuter. Totalt på 24h finns det 24*60 minuter = 1440 minuter. Det går således att köra
$\frac{1440}{202} = 7{,}1287 \approx 7$ stycken buffertar. De 0,1287 buffertar man går miste om motsvarar i tid
$0{,}1287 * 202 \approx 26$ minuter. Dessa kan fördelas som 3 till 4 minuter extra tid på de övriga 7 buffertarna för att göra dessa mindre stressiga. Det finns moment som görs för hand av operatörerna och som kan variera i tid. En omgång buffert bildar 1300 + 2300 + 2800 = 6400 liter buffert. Totalt fås alltså 7 * 6400 liter = 44800 liter buffert. Hoppar man över ”Clean”, eller CIP (clean in place) som det också kallas, så sparar man 45 minuter och den totala tiden blir 157 minuter.
$\frac{1440}{157} = 9{,}17 \approx 9$ stycken buffertar. Detta blir i liter 9 * 6400 liter = 57600 liter.
Det är alltså en ökning på mer än 12000 liter per dygn, jämfört med om man har med rengöringen i slutet.

Annan viktig information som kan ses ur scheduleringen är efter hur lång tid som en viss tank är redo för en transfer. I fallet när ”Clean” körs, så är tredje tanken redo för en transfer efter 50 minuter. En transfer består egentligen av två delar. Först skickas bufferten från tanken i buffertberedningscellen till en tank i någon annan cell. Till detta behövs inte PW. Först efter detta är gjort sköljs ledningen med PW. (En uträkning med uppdelade transfertillstånd finns med i bilaga 2).

	Tid per omgång [minuter]	Antal omgångar på 24 h	Antal liter på 24 h
Med ”Clean”	202	7	44 800
Utan ”Clean”	157	9	57 600

6.2 Scenario 2

Vad är det maximala antalet buffertberedningar a 2300 liter, som kan blandas per dygn?

6.2.1 Hypoteser

En buffertberedning på 2300 liter kan endast blandas av två av de tre tankarna. Den minsta tanken är på 1500 liter och räcker inte till. I lösningen antas det att två tankar, som producerar samma typ av buffert, får lov att blanda sitt innehåll i en större tank. Detta gör att även den minsta tanken kan hjälpa till. Vi antar att de två större tankarna blandar en full omgång buffert varje gång och att den minsta tanken blandar halvor, dvs. 1150 liter. Alla tankar körs sedan parallellt och på två körningar har det således bildats 5 stycken 2300 liters buffertar. Vi antar att varannan buffert måste blandas med ett salt, vilket ger 3 minuters extra blandningstid på första blandningen. Varannan buffert kan blandas med en lösning, vilket ger 1 minuts kortare blandningstid på första blandningen (blandning 1). Tillvägagången ser annars ut som i scenario 1:

- Start
- Fyllning 1 – 60% av (tankens slutvolym minus 200 liter). Påfyllnadshastighet $6m^3$/timme.
- Blandning 1 - Denna tid modifieras beroende på om det är ett salt eller en lösning som ska blandas.
- Fyllning 2 – 40% av (tankens slutvolym minus 200 liter). Påfyllnadshastighet $6m^3$/timme.
- Blandning 2
- Transfer
- Rengöring (CIP)
- Klar

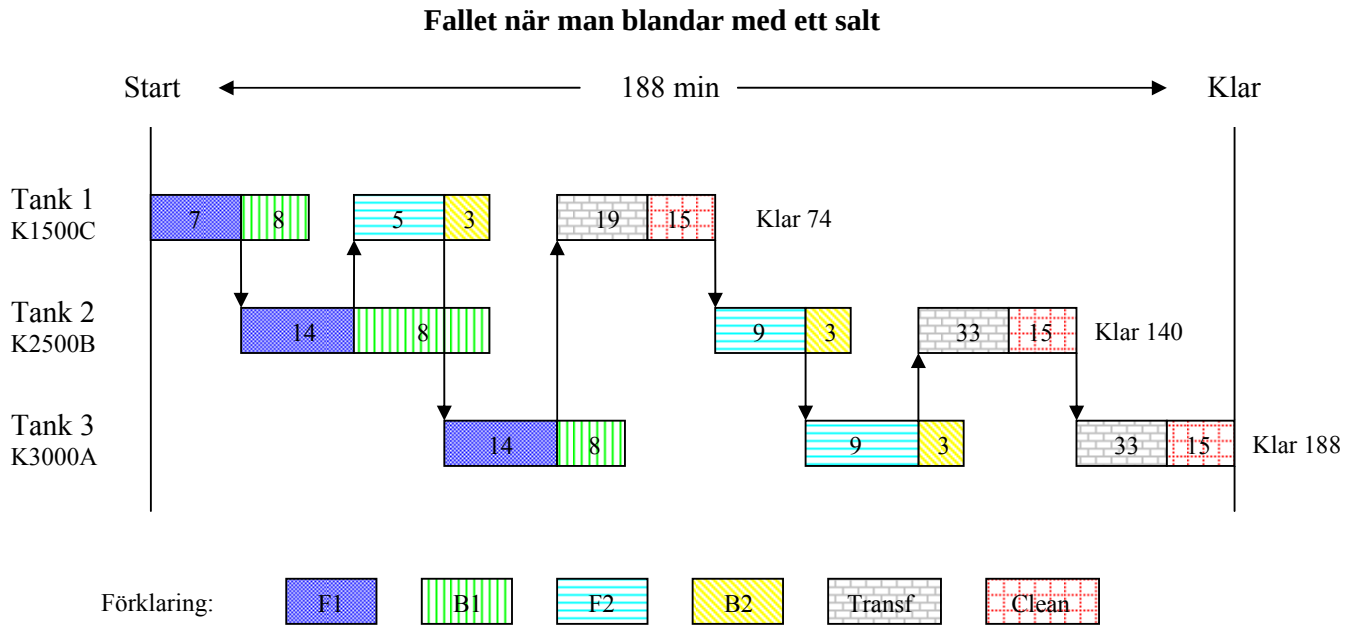
Även i detta fall kommer det att ändra avsevärt på tiden om man blandar alla likadana buffertar i rad, så att man kan slippa köra CIP mellan varje omgång.

6.2.2 Tider [minuter]

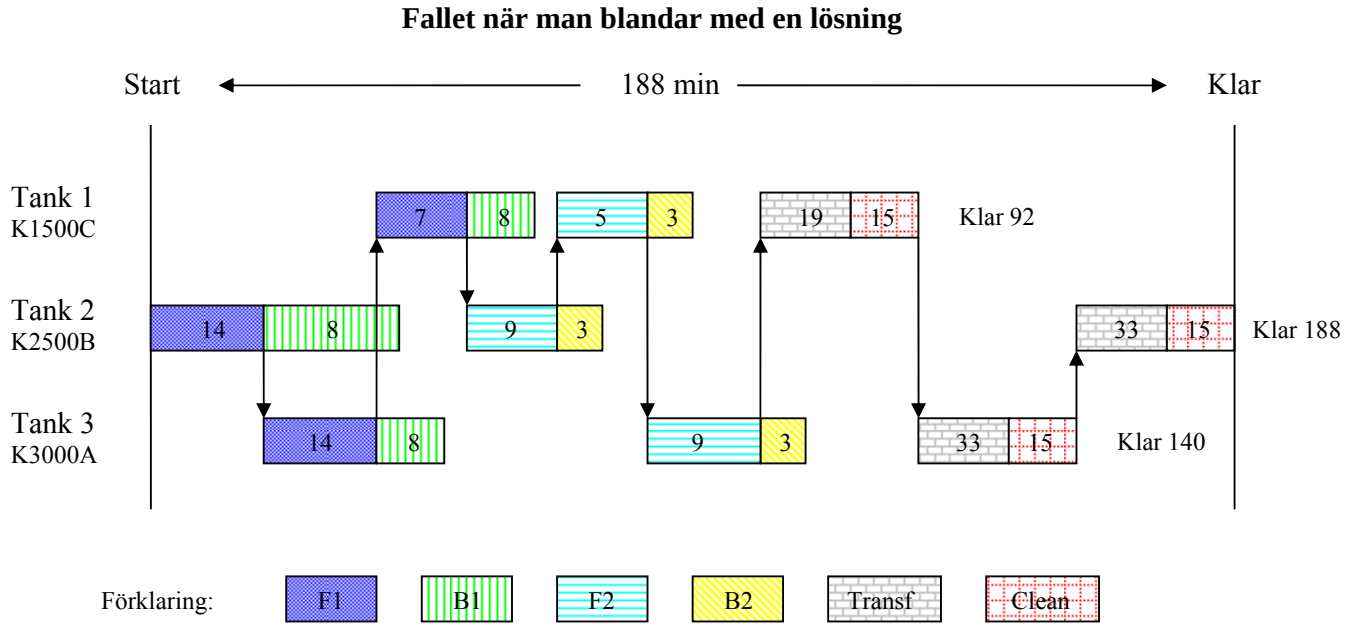
	K1500C	K2500B	K3000A
Storlek [liter]	1150	2300	2300
Start	0	0	0
F1	7	14	14
B1	8 | 4 (salt | lösning)	8 | 4 (salt | lösning)	8 | 4 (salt | lösning)
F2	5	9	9
B2	3	3	3
Transf	19	33	33
Clean	15	15	15
Klar	-	-	-

6.2.3 Schedule

Här nedan följer en schedule för om man blandar med ett salt och en schedule för om man blandar med en lösning. Det är endast den B1 tiderna som det blir skillnad på. Blockens storlek i scheduleringarna, står ej i proportion till blockens tid.



Figur 17. Schedule för fallet när man blandar med ett salt i scenario 2.



Figur 18. Schedule för fallet när man blandar med en lösning i scenario 2.

6.2.4 Svar

Det syns i scheduleringarna i figur 17 och figur 18 att den totala tiden blir densamma (188 minuter), oberoende om det gäller en blandning med salt eller en blandning med en lösning. Programmet gör dock en annan schedulering för blandningen med lösningen än för blandningen med saltet, men det går bra att använda vilken som helst av scheduleringarna i båda fallen. Det viktiga är att blandningen hinner bli klar under tiden som de övriga tankarna använder ”purified water”.

Om det tar 188 minuter att köra 2,5 stycken ”2300 liters” buffertar, kan man köra $\frac{24h \cdot 60\,\text{min}}{188\,\text{min}} = 7,66 st\ omgångar$ på ett dygn. Eftersom detta varken är ett heltal eller ett jämnt tal kan man sluta köra den minsta tanken efter 6 omgångar. Att bara köra 2300 liter i de två större tankarna tar 142 min.

6 * 188 minuter = 1128 minuter
24 * 60 minuter – 1128 minuter = 312 minuter.
Man hinner alltså köra de två större tankarna två gånger till om man struntar i den lilla tanken!

Slutsatsen är att om man kör alla tre tankar, varje omgång i sex omgångar och sedan kör de två större tankarna tillsammans två gånger, får man 19 stycken 2300 liters buffertar per 24 timmar. Den totala tiden för att göra detta är 1128 + 142 + 142 = 1412 minuter vilket är 23,53 timmar. Man har alltså i snitt drygt 1,5 minut extra tid, mellan varje omgång.

6.3 Scenario 3

Om följande åtta buffertar ska tillredas, hur blandar man dessa snabbast och hur lång tid tar det?
Buffert H har högst prioritet och måste vara klar först av buffertarna.

A – 600 liter
B – 2700
C – 980
D – 200
E – 600
F – 400
G – 400
H – 5500 *

6.3.1 Funderingar

Denna fråga var svårare att svara på. Om man antar att alla buffertar tillverkas på samma sätt, dvs med tillstånden ”start,f1,b1,f2,b2,transfer,clean,klar”, så är det endast fyllnadstiderna samt transfertiden som kan variera. Dessa tider beror på hur stor volym buffert man vill ha i en tank. Eftersom all tillverkning slutar med ”clean” och börjar med en fyllning, betyder det att det aldrig kan bli någon förskjutning av tankarna. Både ”clean” och fyllningen är beroende av den begränsade resursen PW.

Nedan benämns buffertarna enligt deras tillfälliga tillredningsvolym.

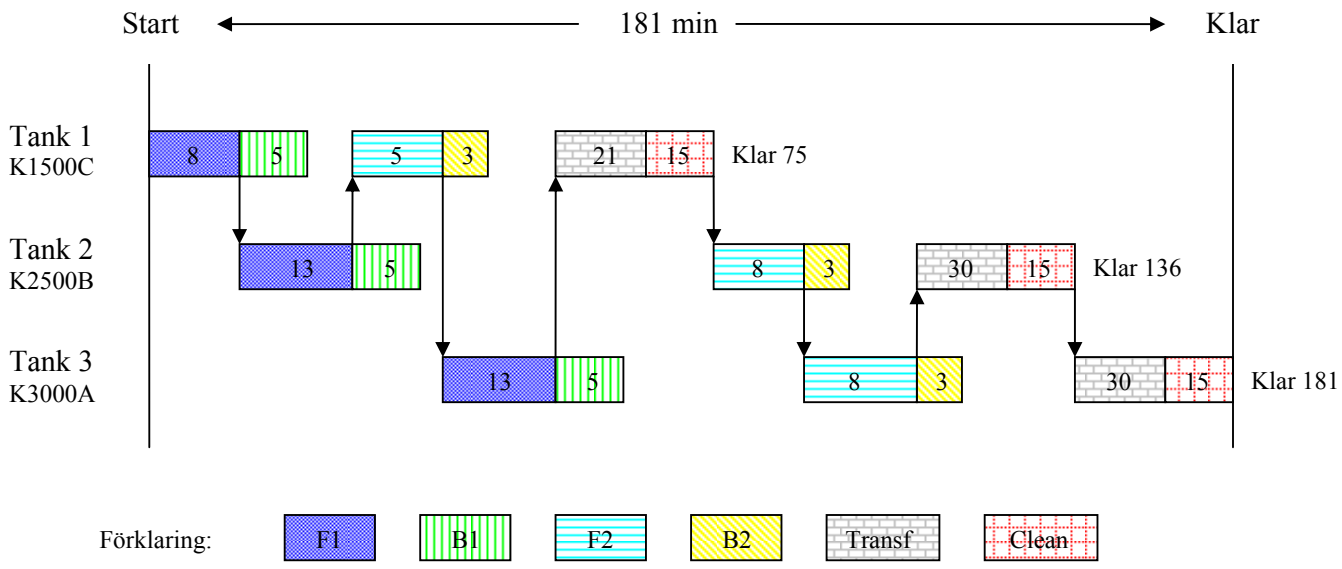
6.3.2 5500

En uträkning har gjorts, men den visar inte den optimala tillvägagången för att göra batcherna. Bufferten på 5500 liter är tvungen att använda mer än en tank eller samma tank flera gånger, vid sin tillverkning. Om vi antar att alla tankar hjälps åt för att göra denna tar det 181 minuter. Detta är om den lilla tanken används fullt ut och de två övriga delar lika på volymen som blir kvar.

6.3.3 Tider [minuter]

	K1500C	K2500B	K3000A
Storlek [liter]	1300	2100	2100
Start	0	0	0
F1	8	13	13
B1	5	5	5
F2	5	8	8
B2	3	3	3
Transf	21	30	30
Clean	15	15	15
klar	-	-	-

6.3.4 Schedule ”5500”



Figur 19. Illustration av schedulering för tillverkning av 5500 liter buffert.

Blocken ovan är inte korrekt skalade.

Bufferten på 5500 liter blir klar först och alla tankar är lediga igen. Då körs en ny buffert på varje tank.

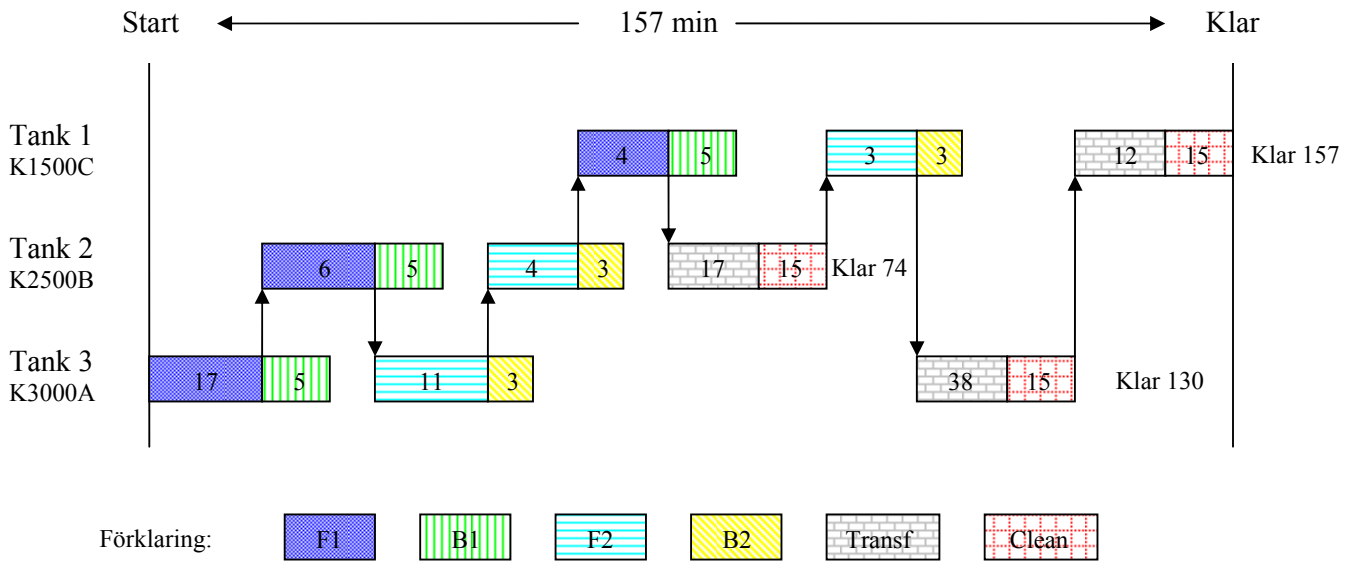
6.3.5 600, 980 och 2700

Fördelningen ser ut som nedan. Den lilla tanken tillverkar en 600 liters buffert, medeltanken en 980 liters buffert och den stora tanken en 2700 liters buffert.

6.3.6 Tider [minuter]

	K1500C	K2500B	K3000A
Storlek [liter]	600	980	2700
Start	0	0	0
F1	4	6	17
B1	5	5	5
F2	3	4	11
B2	3	3	3
Transf	12	17	38
Clean	15	15	15
klar	-	-	-

6.3.7 Schedule ”600, 980 och 2700”



Figur 20. Illustration av schedulering för tillverkning av 600 liter + 980 liter + 2700 liter buffert.

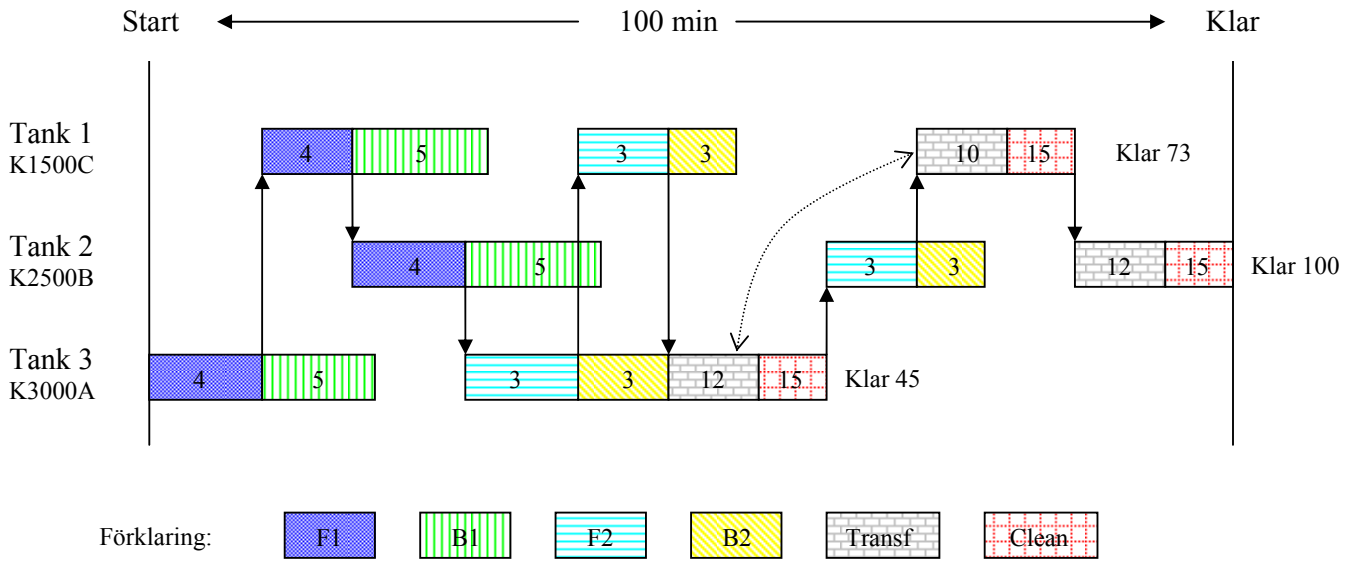
Om vi benämner tankarna efter deras tillfälliga tillverkningsvolym syns det ovan att 980 (tank 2) är klar först, sedan 2700 (tank 3) och sist 600 (tank 1). Nu är det dags för tre nya.

6.3.8 400, 600 och 600

6.3.9 Tider [minuter]

	K1500C	K2500B	K3000A
Storlek [liter]	400	600	600
Start	0	0	0
F1	4	4	4
B1	5	5	5
F2	3	3	3
B2	3	3	3
Transf	10	12	12
Clean	15	15	15
Klar	-	-	-

6.3.10 Schedule ”400, 600 och 600”



Figur 21. Illustration av schedulering för tillverkning av 400 liter + 600 liter + 600 liter buffert.

För första gången tar blandningen mer tid än fyllningen. Detta ger upphov till att vi måste köra buffertarna i ”trappsteg”. Om man efter varje fyllning väntar med att köra en ny fyllning, tills dess att båda de andra tankarna har gjort sin fyllning, så får man summan av två fyllningstider på sig att utföra blandningen. Eftersom alla tider är ganska lika och det enda som skiljer är transfer-tiden för tank 1 jämfört med de båda andra, kan man antagligen välja vilken buffert man vill ha klar först. Om transfer-blocket i tank 1 byter plats med transfer-blocket i tank 3, enligt den streckade pilen, kan man säga att tank 3 nu innehåller 400 liters bufferten och tank 1 en 600 liters buffert. Detta kan vara bra om man vill ha ”400” klar först.

6.3.11 200

Nu återstår endast en 200 liters buffert. Storleksmässigt kan den blandas i vilken tank som helst, men vi antar att vi använder tank 1 (på grund av sensorernas placering i tanken).

6.3.12 Tider [minuter]

(Volym 200 liter)	K1500C
Start	0
F1	2
B1	5
F2	1
B2	3
Transf	7
Clean	15
Klar	-

Detta ger en tid på: 33 minuter

6.3.13 Svar:

Om vi kör enligt scheduleringarna i scenario 3 kommer buffertarna bli klara i följande ordning.

1. 5500 (181 min)
2. 980 (255 min)
3. 2700 (311 min)
4. 600 (338 min)
5. 600 (383 min)
6. 400 (411 min)
7. 600 (438 min)
8. 200 (471 min)

Tiderna inom parentes visar tiden det tar innan bufferten är färdig. Den totala tiden, dvs. tiden det tar innan alla buffertar är klara, blir 471 minuter. Det motsvarar 7 h och 51 minuter. Ingen optimering är gjord i avseende på ordningen som buffertarna ska köras i. De enskilda scheduleringarna är dock optimerade.

7. Frågeställningar

7.1 Vad är den normala cykel tiden (tiden som behövs för en batch)?

Om endast en tank körs åt gången, så har denna tank full tillgång till alla resurser. Det betyder att det aldrig behöver bli någon väntetid för denna tank. Under dessa omständigheter tar det 1 timme och 19 minuter för tanken att producera en batch och sedan vara helt redo för en ny omgång. Denna tid är summan av deltiderna för de ingående operationerna och beror således på hur stor tanken är. Nedan syns uträkningen för medeltanken (K2500B).

Tillstånd	**Tid [minuter]**
Start	0
F1	14
B1	5
F2	9
B2	3
Transf	33
Clean	15
Klar	-
Summa:	**79 minuter = 1h 19 minuter**

För tank K1500C blir tiden = 57 minuter.
För tank K3000A blir tiden = 90 minuter = 1 timme och 30 minuter.

7.2 Hur ska produktionen planeras för att maximal mängd buffert ska kunna tillverkas?

Det beror på vilka resurser och tankar som ska användas. En ny schedulering måste göras för varje enskild typ av konfiguration.

7.3 Finns det några generella riktlinjer och regler som kan öka effektiviteten?

- Det är viktigt att alla begränsade resurser används oavbrutet. Om t.ex purified water måste stå och vänta innan det blir använt uppkommer en onödig dödtid. PW måste alltså användas hela tiden och de operationer som kan göras utan PW ska ske på mellantider, då PW är upptaget av en annan tank.

- Eftersom man vid transfern är beroende av att det finns en ledig mottagar-tank måste arbetet i buffertcellen synkas med den cell dit man skickar bufferten. Operatörerna måste hålla kontakt.

- Om det ska göras mer än en buffert av samma slag i samma tank, ska dessa göras efter varandra i den mån det går. Då sparar man tid på att slippa köra ”CIP” mellan bufferttillverkningarna.

7.4 Vilka optimeringsmetoder finns, som kan hjälpa till vid schedulering?

Om man angriper scheduleringsproblemet och försöker lösa det med trädstrukturer och grafer, så finns det en del optimeringsmetoder som kan hjälpa till.

- Dijkstras Algoritm
  Hittar den snabbaste vägen från en startpunkt till en slutpunkt i en riktad graf. Det enda kravet som finns är att ingen av delsträckorna har en negativ kostfunktion.

- Prim's Algoritm
  Denna algoritm spänner upp det minsta trädet som kan skapas mellan punkter i en viktad, oriktad graf. Vill man ta sig från en punkt till ett antal andra, så visar denna algoritm hur man gör detta snabbast.

- Kruskals Algoritm
  Fungerar på ungefär samma sätt som Prim's algoritm.

- A* (uttalas "A stjärna")
  Hittar den snabbaste vägen genom en graf. En kombination av girigt letande och Dijkstras algoritm. (En girig algoritm är en algoritm som i varje steg väljer det tillfälligt bästa alternativet utan att se framåt). Denna algoritm kan ges heuristik, dvs en tumregel för hur den skall göra sina val.

Det finns även andra metoder som antas kunna lösa problemet, men dessa har ej utvärderats pga att omfattningen av detta examensarbete i så fall skulle bli för stort. Metoder som skulle kunna undersökas är:

- Mixed integer linear programming – MILP

- Dynamic Progamming

7.5 Finns det färdiga verktyg eller program som kan vara användbara för Novozymes planering?

Inga program som optimerar tillverkningsprocessen har hittats under detta examensarbete. Däremot är Preactor och Schedule Pro två program som kan hjälpa till vid scheduleringsarbetet i buffertberedningscellen. Dessa program diskuteras under kapitlet "Schedulering".

7.6 Går det att få testlicenser till dessa så att det går att göra en utvärdering på anläggningen?

Begränsade versioner av både Preactor och Schedule Pro går att få tag i. Endast Preactor utvärderades och testades personligen. Mer information finns på hemsidorna.

http://www.preactor.com.
http://www.intelligen.com.

7.7 En sammanställning och definiering av terminologin, som används eller borde användas på Novozymes, ska konstrueras.

Se kapitel ”Terminologi”.

8. Terminologi

8.1 Physical hierarki – Fysisk hierarki

Enterprise: Företaget. Ansvarar för vilka produkter som skall tillverkas. Både var och hur.

Site: Anläggning. En fysisk, geografisk eller logisk indelning som bestäms av företaget.

Area: Område. Mindre indelning av anläggningen. Även denna indelning bestäms av företaget.

Process cell: Process cell. Logisk indelning av den utrustning som behövs för att tillverka en eller flera batchar.

Unit: Enhet. Kan innehålla både control moduler och equipment moduler. En processaktivitet måste ske i en enhet (tex en tank).

Equipment module: Utrustningsmodul. Utför mindre sysslor i en enhet. Tex omrörare i en tank.

Control module: Kontrollmodul. Sensorer, regulatorer, aktuatorer eller kombinationer av dessa kallas för kontrollmoduler.

8.2 Recipe – Recept

General recipe: Det generella receptet

Site recipe: Anläggnings-recept

Master recipe: Huvud recept

Control recipe: Kontroll recept

8.3 Recipe procedure – Recept procedur

Procedure: Procedur. Styr samordningen av cellen. Låt säga att vi tittar på en procedur som heter ”Gör apelsinjuice”. Består av en mängd enhetsprocedurer.

Unit procedure: Enhetsprocedur. Styr en enhet i processcellen. Kan tex. vara ”skala apelsin”. Består av en mängd operationer.

Operation: Operation. Styr en ”equipment module” i en enhet. Tex. en kniv i enheten där ”skala apelsin” utförs. Operationen kallas kanske för ”rör kniv”. En operation består av en mängd faser.

Phase. Fas. Ger information till en ”control module” anknuten till en ”equipment module”. Operationen ”rör kniv” består kanske av faserna ”lyft kniv” och ”sänk kniv”.

9. Sammanfattning

Detta arbete har fokuserat på att reda ut vilka metoder och verktyg som finns för att effektivisera den batchproduktion som bedrivs i buffertberedningscellen på Novozymes Biopharma AB i Lund. Arbetet har gjorts i enlighet med ISA-88 standarden för batch kontroll och rapporten innehåller en kort beskrivning av denna standard.

För att effektivisera batchproduktionen har två kommersiella schemaläggnings/scheduleringsverktyg testats och utvärderats; Preactor och Schedule Pro. Båda verktygen ansågs vara bra och lätta att arbeta med. De ansågs dock inte vara fullt tillräckliga eftersom det schema som verktygen föreslår inte med säkerhet kan sägas vara det optimala schemat för hur produktionen skall bedrivas.

Eftersom det var svårt att hitta lämpliga kommersiella schemaläggnings/scheduleringsverktyg så har algoritmer för ett nytt verktyg tagits fram och implementerats. Verktyget bygger på Dijkstras algoritm och har implementerats i Java. Verktyget har använts för att svara på de frågor som Novozymes sökte angående hur deras batchproduktion skall kunna effektiviseras.

10. Referenser

- **ISA-88**
  Kompendiet
  http://www.isa-88.com/

- **Batch Scheduling**
  Victor Terpstra
  ISBN 90-9009848-8

- **Optimization of manufacturing cells using discrete event models**
  Avenir Kobetski
  Thesis for the degree of licentiate of philosophy, Chalmers

- **The theory of finite automata**
  Kobrinskii Trakhtenbrot

- **Scheduling for production systems**
  Torbjörn Liljenvall
  Technical report, Chalmers

- **Batch processing systems engineering, fundamentals and applications for chemical engineering.**
  G. V. Reklaitis
  ISBN 3-540-59201-6 Springer-Verlag Berlin Heidelberg New York

- **Industrial Scheduling**
  Muth and Thompson

- **Wikipedia.com**
  Dijkstras Algorithm
  Patogen
  Scheduling
  GPS

11. Bilagor

1 Inledning

Denna specifikation beskriver övergripande de krav som finns för automationen för buffertberedningen.

Systemet skall användas för beredning av olika buffertlösningar till linje1 och linje2.

Kraven på buffertsystemet finns beskrivet i V-S155 (Användarspecifikation för buffertberedningens styrsystem).

Ritning på systemet finns beskrivet i A-40.

Till denna specifikation hör 11 bilagor.

2 Referenser

2.1 Dokument

Dokument som det refereras till i aktuellt dokument:

V-S155	Användarspecifikation för buffertberedningens styrsystem
A-40	Buffertberedning

2.2 Förkortningar

RP	Recipe Procedure
UP	Unit Procedure
PW	Purified Water
AI	Analog input
AO	Analog output
DI	Digital input
DO	Digital output
I/O	Input/Output
P&I	Process and instrument
ER	Electronisk Record

2.3 Definitioner

Common Resource	Common Resource är resurser så som ventiler och mätare och som inte tillhör någon enhet. Dessa resurser kan användas av olika enheter.
Process Cell	En grupp av "units". I en processcell kan ett RP-recept köras.
Unit	Enhet. Inom en enhet kan ett UP-recept köras. Endast ett UP-recept kan köras per enhet åt gången.
Receptparametrar	Parametrar som definieras i recept (ex RP, UP).

Equipmentparametrar Parametrar som är utrustningsberoende, exempelvis nivå för att stänga omrörare vid låg nivå.

Electronic Records Elektroniskt dokument enligt 21 CFR Part11.

3 Data

3.1 GMP-kritisk data

All GMP-kritisk data lagras på styrsystemet PCS02 och behandlas inom dess kvalificering.

Endast analoga signaler bedöms som kritiska.
De data som har bedömts som GMP-kritiska loggas av styrsystemet och finns noterade med inställningar i punkt 4.4 I/O-listorna för analoga signaler.

3.2 Electronic Records

All Electronic Records lagras på styrsystemet PCS02 och behandlas inom dess kvalificering.

Loggningen av de GMP-kritiska data enligt 3.1 anses vara ER.

4 Fysisk modell

P&I diagram finns i dokument A-40.

Ventil AV5, som är streckad på ritning A-40, tillhör fysiskt PW, men tillhör automationsmässigt buffertberedningen. All styrning av denna ventil finns beskrivet i denna funktionsbeskrivning.

4.1 Process cell

4.1.1 Buffertsystemet skall bestå av en egen processcell: IQ

- BBERED

4.2 Units

4.2.1 Systemet skall bestå av följande enheter: IQ

3 st blandningstankar

- K2500B
- K3000A
- K1500C

Fyllningsledning för tillsättning av PW

- BLED01

2 st transferledningar

- BLED02
- BLED03

4.3 Common Resource

4.3.1 Systemet har inga Common Resources.

4.4 I/O-lista

Alla I/O skall normalt ha en "scan rate" på 1 sekund och skall ha ett loggningsintervall på 5 sekunder. Om en signal skall ha en annan inställning skall detta noteras i I/O-listan.

Alla analoga signaler skall ha loggning startad och alla digitala signaler skall ha loggning avstängd. De signaler som avviker från standardvärdena skall noteras i I/O-listan.

4.4.1 DI-lista

4.4.1.1 Digitala insignaler finns definierade i bilaga 1. IQ

4.4.2 DO-lista

4.4.2.1 Digitala utsignaler finns definierade i bilaga 2. IQ

4.4.3 AI-lista

4.4.3.1 Analoga insignaler finns definierade i bilaga 3. IQ

4.4.4 AO-lista

4.4.4.1 Analoga utsignaler finns definierade i bilaga 4. IQ

4.4.5 Larm

4.4.5.1 Larm med larmtexter finns definierade i bilaga 5. OQ

4.4.6 Equipment parametrar

4.4.6.1 Följande equipmentparametrar skall finnas: OQ

- Maxtryck i BLED02.
- Maxtryck i BLED03.
- Minsta volym i K1500C.
- Minsta volym i K2500B.
- Minsta volym i K3000A.

5 Operationer

Följande operationer/recept skall vara möjliga. Lämpliga receptparametrar för att kunna utföra dessa operationer skapas i FDS.

5.1 Uppstart

5.1.1 Vid uppstart skall alla enheter vara Off/Stop/Deaktiverade.

5.2 Fyllning med PW i blandningstank, Blandning av buffert med recirkuleringsloop, Tillsatser till buffert

Programmet skall finnas i version för enhet K1500C, K2500B och K3000A. Denna operation är tänkt för blanding av buffertar med stora mängder salt där loopen behövs för att lösa saltet.

För att styra tillsatserna används ett lokalt system till uppfyllning och tömning av mätglas.

5.2.1	Prompt för fyllnivå 1.	OQ
5.2.2	Öppna ledningen för PW till avsedd tank.	OQ
5.2.3	Stäng ventil AV5 för PW och därefter ventilen närmast tanken vid inställd volym.	OQ
5.2.4	Prompt "Tillsätt komponenter enligt instruktion".	OQ
5.2.5	Starta omrörare.	OQ
5.2.6	Prompt för blandningstid 1.	OQ
5.2.7	Starta pump och öppna ledning för recirkulering.	OQ
5.2.8	Stäng pump efter inställd tid och stäng ledning för recirkulering.	OQ
5.2.9	Prompt "Kontrollera att salt har lösts" efter inställd tid.	OQ
5.2.10	Prompt för fyllnivå 2	OQ
5.2.11	Öppna ledningen för PW.	OQ
5.2.12	Stäng första ventilen på PW-ledningen vid inställd volym (fyllnivå 2).	OQ
5.2.13	Prompt ställ in tid blandning 2.	OQ
5.2.14	Rör om efter inställd tid.	OQ
5.2.15	Prompt för att kontrollera att syra/bas finns.	OQ
5.2.16	Öppna ledningen manuellt.	OQ
5.2.17	Manuell körning till rätt pH eller konduktivitet enligt 5.6.	OQ
5.2.18	Stäng ledningen manuellt.	OQ
5.2.19	Prompt för att godkänna rätt pH eller konduktivitet.	OQ
5.2.20	Stoppa omrörare.	OQ

5.3 Fyllning med PW i blandningstank, Blandning av buffert utan recirkuleringsloop (bara i kärlet), Tillsatser till buffert

Programmet skall finnas i version för enhet K1500C, K2500B och K3000A. Skillnad mellan blandningsprogrammet innan (med loop) är att det bara används omröraren för blandningen, loopen är inte igång. Denna operation är tänkt för blanding av buffertar med små mängder salt och buvudsakligen tillsats av flytande, koncentrerade lösningar.

För att styra tillsatserna används ett lokalt system till uppfyllning och tömning av mätglas.

5.3.1	Prompt för fyllnivå 1.	OQ
5.3.2	Öppna ledningen för PW till avsedd tank.	OQ
5.3.3	Stäng ventil AV5 för PW och därefter ventilen närmast tanken vid inställd volym.	OQ
5.3.4	Prompt "Tillsätt komponenter enligt instruktion".	OQ
5.3.5	Starta omrörare.	OQ
5.3.6	Prompt för blandningstid 1.	OQ
5.3.7	Prompt "Kontrollera att salt har lösts" efter inställd tid.	OQ
5.3.8	Prompt för fyllnivå 2	OQ
5.3.9	Öppna ledningen för PW.	OQ
5.3.10	Stäng första ventilen på PW-ledningen vid inställd volym (fyllnivå 2).	OQ
5.3.11	Stäng alla ventiler på ledningen efter tid.	OQ
5.3.12	Prompt ställ in tid för blandning 2.	OQ
5.3.13	Rör om efter inställd tid.	OQ
5.3.14	Prompt för att kontrollera att syra/bas finns.	OQ
5.3.15	Öppna ledningen manuellt.	OQ
5.3.16	Manuell körning till rätt pH eller konduktivitet enligt 5.6.	OQ
5.3.17	Stäng ledningen manuellt.	OQ
5.3.18	Prompt för att godkänna rätt pH eller konduktivitet.	OQ
5.3.19	Stoppa omrörare.	OQ

5.4 Transfer av buffert, sköljning av ledning

Programmet skall finnas i version för transfer från enhet K1500C, K2500B och K3000A.
Programmet skall finnas i version för transfer till tankar:
mobiltank i Ö009B, K2500C, K2500D, K5000B, K2200A, K1000F, K1000B, K1000G, K1000D, K800C, K1000E, mobiltank i Ö009A, mobiltank i Ö108A, mobiltank i 110B, mobiltank i Ö104.

5.4.1	Prompt för kontroll av filter.	OQ
5.4.2	Prompt öppna ledning till rätt tank.	OQ
5.4.3	Starta pump.	OQ
5.4.4	Prompt: Kontrollera att tanken är tom.	OQ
5.4.5	Stäng pump när blandningstanken är tom.	OQ

5.4.6	Öppna ledningen för dränering.	OQ
5.4.7	Stäng för dränering efter tid.	OQ
5.4.8	Öppna ledning till avlopp.	OQ
5.4.9	Öppna för PW.	OQ
5.4.10	Starta tidräknare när flödet är högre än inställt värde. Tidräknaren skall stoppas om flödet underskrider inställt värde.	OQ
5.4.11	Stäng för PW efter inställd tid.	OQ
5.4.12	Öppna ledning för dränering.	OQ
5.4.13	Stäng ledning efter inställd tid.	OQ

5.5 Sköljning av tank och recirkuleringsloop med kallvatten och PW

Programmet skall finnas i version för enheterna K1500C, K2500B och K3000A. Detta program är tänkt att användas efter alla blandningar (med och utan recirkuleringsloop).

5.5.1	Öppna ventil för KV via CIP-kula med bottenventil öppen. Omröraren är ej igång.	OQ
5.5.2	Starta pump och öppna ledning till avlopp en bestämd tid.	OQ
5.5.3	Fyll upp med KV via CIP-kula till bestämd volym med stängd bottenventil.	OQ
5.5.4	Öppna recirkuleringen, starta pump och starta omrörare en bestämd tid.	OQ
5.5.5	Stäng omrörare och recirkuleringsloop.	OQ
5.5.6	Dränera recirkuleringsloop och tank till avlopp en bestämd tid. Stäng pump efter dränering.	OQ
5.5.7	Öppna ventil för PW via CIP-kula med bottenventil öppen. Omröraren är ej igång.	OQ
5.5.8	Starta pump och öppna ledning till avlopp en bestämd tid.	OQ
5.5.9	Fyll upp PW med CIP-kula till bestämd volym med stängd bottenventil.	OQ
5.5.10	Öppna recirkuleringen, starta pump och starta omrörare en bestämd tid.	OQ
5.5.11	Stäng omrörare och recirkuleringsloop.	OQ
5.5.12	Dränera recirkuleringsloop och tank till avlopp en bestämd tid. Stäng pump efter dränering.	OQ

5.6 Manuella funktioner

Nedan beskrivs de funktioner som skall finnas för den manuella tillsatsen av lösningar.

5.6.1 Uppfyllning av mätglas

5.6.1.1	Sätt switch i läge för "Fyllning av mätglas".	OQ
5.6.1.2	Tryck in brytare tills mätglas är uppfyllt.	OQ
5.6.1.3	Sätt switch i läge "Neutral".	

5.6.2 Tömning av mätglas

5.6.2.1	Sätt switch i läge för "Tömning av mätglas".	OQ
5.6.2.2	Tryck in brytare tills rätt volym har tömts till blandningstank.	OQ
5.6.2.3	Sätt switch i läge neutral.	

5.6.3 Manuell tillsats till blandningstank

5.6.3.1	Tryck in brytare för förbikoppling av mätglas.	OQ

6 Förreglingar/interlocks/möjligheter

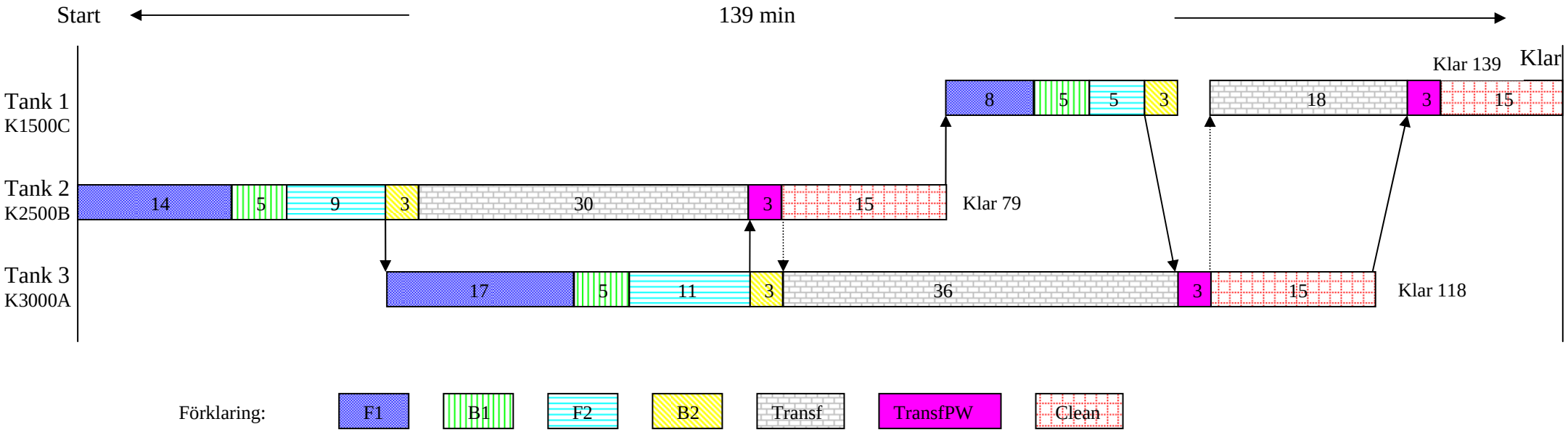
6.1 Förreglingar

6.1.1	Det skall endast gå att fylla PW till en tank eller till en ledning åt gången.	OQ
6.1.2	Endast en blandningstank skall kunna använda en transferledning åt gången.	OQ
6.1.3	Olika blandningstankar skall kunna använda olika transferledningar samtidigt.	OQ
6.1.4	Det skall endast gå att fylla en hålltank per transferledning.	OQ

6.2 Interlocks

6.2.1	Om locket på manluckan öppnas skall pump stoppas på aktuell tank.	OQ
6.2.2	Om trycket blir för högt i transferledning skall pumpen stoppas.	OQ
6.2.3	Om nivån i tank underskrider inställd volym skall omröraren stängas. Ställs in via parameter 4.4.6.1.	OQ

Schedulering med uppdelade transfer tillstånd



Ovan syns den schedulering som bildas om man delar upp transfertillståndet i två delar. Ett utan PW och ett med PW. De nya tiderna blir enligt tabellen nedan. De heldragna pilarna visar hur PW förflyttas och de prickade pilarna visar hur transferledningen fördelas. Endast en tank kan göra en transfer samtidigt och endast en tank får använda PW åt gången.

Den totala tiden blir i detta fall 139 minuter vilket ger upphov till att 10,36 omgångar kan köras. Antingen kör man 10 omgångar och använder de 0,36*139 = 50 minuterna som blir kvar som en marginal under dagen, eller så kör man en omgång ”ofulla” tankar som man beräknar ska ta runt 50 minuter.

10 omgångar motsvarar 64000 liter buffert.

Tider som används i ” Schedulering med uppdelade transfer tillstånd”

Tillstånd	Tank A (K1500C) [minuter]	Tank B (K2500B) [minuter]	Tank C (K3000A) [minuter]
Start	0	0	0
F1	8	14	17
B1	5	5	5
F2	5	9	11
B2	3	3	3
Transf	18	30	36
transfPW	3	3	3
clean	15	15	15
Klar	-	-	-

Tiderna beskriver hur lång tid varje händelse tar, dvs hur länge man befinner sig i det gällande tillståndet.