Faculteit Elektrotechniek, Wiskunde en Informatica

Bacheloropdracht TW 2009

Operating a Multi-Speed Production Facility

Sjoerd van Willigen
Koen Dijkstra

Begeleiders:
Werner Scheinhardt
Denis Miretskiy

Inhoudsopgave

1 Inleiding

We beschouwen een wachtrijsysteem van twee servers in serie, waarbij de eerste server kan werken op twee verschillende snelheden. Op welke van de twee snelheden deze server werkt is afhankelijk van de hoeveelheid werk in de tussenliggende buffer. Door deze snelheid aan te passen kan de hoeveelheid werk in deze buffer beïnvloed worden, om zo te voorkomen dat deze buffer vol of juist leeg raakt.

Als praktische representatie hiervan beschouwen we een productielijn van twee machines. Door de snelheid van de eerste machine aan te passen kan de hoeveelheid werk die in de wachtrij staat voor de tweede machine beïnvloed worden. Als deze hoeveelheid werk nul zou worden, dan moet de 2e machine worden stilgezet (*starving*), wat kosten met zich meebrengt. Als de hoeveelheid een bepaald maximum bereikt, dan is de buffer vol en zal de eerste machine moeten worden stilgezet (*blocking*), wat ook kosten veroorzaakt.

Onze voornaamste interesse is het bepalen van de bedieningsstrategie waarbij de verwachte kosten minimaal zijn. We bepalen hiervoor een kostenfunctie die kosten toekent aan de volgende vier aspecten van ons systeem: wachttijd (opslagkosten), *blocking*, *starving* en het werken op verschillende bedieningssnelheden (*running* kosten).

Tandem-wachtrijen met blocking zijn veelvuldig besproken in de literatuur. Brandwajn en Jow [3] benaderen evenwichtsgegevens m.b.v. marginale kansverdelingen en conditionele kansen. In Cheng en Yao [4] wordt een algemeen blocking schema bepaald, waarbij ook blocking van informatie wordt beschouwd.

Konheim en Reiser [8] bespreken een twee-server systeem met een eindige tussenliggende wachtruimte met *feedback* en *blocking*. Hier worden producten na behandeling bij de tweede server met een bepaalde kans weer teruggestuurd naar de eerste server. Ook wordt de eerste server geblockt als er M producten in de tussenliggende wachtruimte zijn. Vervolgens wordt het systeem geanalyseerd onder assumpties van exponentialiteit, en wordt m.b.v. generende functies een algoritme bepaald voor het berekenen van de evenwichtsverdeling.

In Grassman en Drekic [7] wordt de evenwichtsverdeling voor een twee-server systeem met blocking bepaald met behulp van gegeneraliseerde eigenvectoren. Specifieker, een formule met daarin een cotangens wordt afgeleid. De periodiciteit van deze cotangens wordt vervolgens gebruikt om de meeste eigenwaarden te bepalen. Als de eigenwaarden eenmaal zijn gevonden kunnen de eigenvectoren recursief worden bepaald. Deze methode heeft een relatief lage complexiteit.
Er wordt ook vermeld dat deze methode tevens gebruikt kan worden voor het geval van twee mogelijke snelheden bij de eerste server, maar dit wordt niet verder uitgewerkt.

Systemen met aanpasbare bedieningssnelheden worden besproken in Bekker en Boxma [1] en

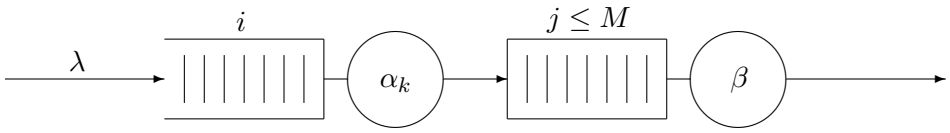
in Bekker, Boxma en Resing [2], maar in beide artikelen wordt aangenomen dat er alleen informatie is over de tweede wachtrij op aankomstmomenten bij de eerste wachtrij. Dit heeft dus tot gevolg dat de snelheid van de eerste server alleen aangepast kan worden bij aankomsten, en niet bij vertrekken van de eerste of tweede server. In Foreest, Mandjes, v. Ommeren en Scheinhardt [6] wordt ook een twee-server systeem bekeken. Zij richten zich voornamelijk op stabiliteit en de snelheid waarmee de kansen op grote rijlengtes afnemen (*decay rate*).

In Tijms [9] wordt een kostenfunctie bepaald voor een systeem met één server en twee mogelijke snelheden. De snelheid waarop de server werkt hangt af van het aantal producten in het systeem, er wordt gewerkt met zowel een *slowdown-treshold* als een *speedup-treshold*. De gebruikte kostenfunctie bevat opslagkosten, bedieningskosten, en *switch-over* kosten. Er worden echter geen kosten toegekend aan blocking of starving, waar in ons onderzoek juist de meeste nadruk op wordt gelegd.

Dit verslag is als volgt opgebouwd. In sectie 2 presenteren we ons model en bespreken we enkele aannames die gemaakt zijn. Ook bespreken we hier stabiliteit van het systeem en de kostenfunctie die gebruikt wordt. In sectie 3 wordt een uitgebreide afleiding van de analytische oplossing gegeven. In sectie 4 wordt een simulatie-programma van het systeem besproken. Sectie 5.1 bevat resultaten, en in sectie 5.2 worden conclusies geformuleerd.

2 Modelbeschrijving

2.1 Het wachtrijsysteem



Figuur 1: Tandem-wachtrij met eindige buffer.

We beschouwen een netwerk van twee wachtrijen in serie, zoals weergegeven in figuur 1. Het systeem wordt gevuld door een Poisson proces met intensiteit λ. Producten komen dan terecht in een oneindig grote wachtruimte voor de eerste server. Ze worden vervolgens behandeld bij de eerste server, die kan werken op twee verschillende snelheden. Bij beide snelheden zijn de behandeltijden exponentieel verdeeld met intensiteiten α_1 en α_2 ($\alpha_1 > \alpha_2$).
Na behandeling komen de producten in de tussenliggende wachtruimte voor de tweede server terecht. Deze wachtruimte heeft een maximale capaciteit M. Als deze wachtruimte vol raakt, dan houdt de eerste server op met bedienen, dit effect heet *blocking*. De eerste server zal dan weer verder gaan met bedienen zo gauw het aantal producten in de tweede wachtrij weer gedaald is tot een bepaald aantal r ($0 < r < M$). Vervolgens worden de producten behandeld door de tweede server, die ook een exponentiële bedieningsduur heeft met intensiteit β. Na behandeling bij de tweede server verlaten de producten het systeem.
De snelheid waarop de eerste server werkt, wordt bepaald door het aantal producten in de tweede wachtrij. Als dit aantal stijgt tot een zeker aantal k ($k < M$) en de eerste server op hoge snelheid α_1 werkt, dan wisselt de server naar de lagere snelheid α_2. Als dit aantal vervolgens weer daalt tot een bepaalde grens s ($s < k$), dan wisselt de server weer terug naar de hogere snelheid α_1.
Door het wisselen naar de lagere snelheid kan *blocking* worden voorkomen, aangezien er dan minder snel producten in de tussenliggende wachtruimte terecht komen. Het wisselen naar de hogere snelheid zal er juist voor zorgen dat er sneller producten in de tussenliggende wachtruimte terecht komen, dit kan *starving* voorkomen. In figuur 2 is een mogelijke keuze voor de parameters s, k en r schematisch weergegeven.

Figuur 2: Een mogelijke configuratie.

Het eenvoudigste geval treedt op als er voor elk mogelijk aantal producten in de tweede rij

maar één snelheid (α_1, α_2 of 0) van de eerste server mogelijk is. Hiervoor moet gelden:

$$s = k - 1 \tag{1}$$

$$r = M - 1 \tag{2}$$

Als aan (1) en (2) is voldaan, wordt de wachtrij beschreven door de twee stochastische processen $X_1(t)$ ($X_1(t) \geq 0$) en $X_2(t)$ ($0 \leq X_2(t) \leq M$), die het aantal producten in de eerste en tweede rij op tijdstip t representeren. De toestandsruimte is dan te schrijven als

$$S_1 = \{(i,j) \mid i \geq 0,\ 0 \leq j \leq M\}.$$

Het systeem is op tijdstip t in toestand (i,j) als $X_1(t) = i$ en $X_2(t) = j$.

Lastiger wordt het als (1) en/of (2) niet gelden, omdat $X_1(t)$ en $X_2(t)$ de toestand van het systeem dan niet altijd vastleggen. Er is in dit geval immers minimaal één j waarvoor de eerste server op minimaal twee van de drie mogelijke snelheden (α_1, α_2, 0) kan werken als $X_2(t) = j$. We hebben dus een derde dimensie in de toestanden nodig, die aangeeft met welke snelheid de eerste server werkt. We definiëren een stochastisch proces $V(t)$ ($V(t) \in \{0, \alpha_1, \alpha_2\}$) die de bedieningssnelheid van de eerste server op tijdstip t aangeeft. De toestandsruimte is dan te schrijven als

$$S_2 = \{(i,j,v) \mid i \geq 0,\ 0 \leq j \leq M,\ v \in \{0, \alpha_1, \alpha_2\}\}.$$

Het systeem is in toestand (i,j,v) als $X_1(t) = i$, $X_2(t) = j$ en $V(t) = v$. Merk echter op dat een aantal van deze toestanden niet bereikbaar zijn. Als $X_2(t) \leq s$ kan immers niet gelden dat $V(t) = \alpha_2$. Evenmin kan $V(t) = \alpha_1$ zijn als $X_2(t) \geq k$ en kan de eerste server nooit geblokkeerd zijn als $X_2(t) \leq r$.

Helaas lijken (1) en (2) vaak geen logische keuze. Als we bijvoorbeeld aan *blocking* relatief hoge kosten toekennen (omdat het bijvoorbeeld erg duur is om de eerste machine na het blokkeren weer op te starten), maar het niet uitmaakt hoe lang de machine geblokkeerd is, lijkt de keuze $r = M - 1$ niet verstandig. De eerste server zal dan weliswaar zo snel mogelijk herstarten, maar doet dit met nog maar één vrije plek in de tweede rij, zodat de kans vrij groot zal zijn dat de eerste server binnen korte tijd weer blokkeert.

2.2 Stabiliteit

In [6] bespreken de auteurs een systeem dat voldoet aan (1) en (2) en waarvoor geldt dat

$$\lambda < \beta < \alpha_2 < \alpha_1. \tag{3}$$

Zij tonen aan dat dit systeem stabiel is dan en slechts dan als

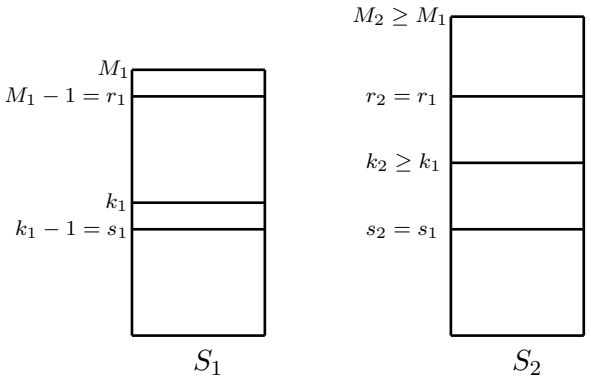
$$\lambda < \frac{\alpha_1(1-b_1^k)(1-b_2) + \alpha_2 b_1^k(1-b_1)(1-b_2^{M-k})}{(1-b_1^k)(1-b_2) + b_1^k(1-b_1)(1-b_2^{M-k+1})} \tag{4}$$

met $b_1 = \frac{\alpha_1}{\beta}$ en $b_2 = \frac{\alpha_2}{\beta}$.

Op grond van numerieke experimenten geloven wij dat (4) ook geldt als niet is voldaan aan (3). Door de limieten $b_1 \to 1$ ($\alpha_1 \to \beta$) en $b_2 \to 1$ ($\alpha_2 \to \beta$) van (4) te bepalen verkrijgen we de stabiliteitscondities in de gevallen $\alpha_1 = \beta$ en $\alpha_2 = \beta$:

$$\lambda < \frac{\alpha_1 k(1-b_2) + \alpha_2(1-b_2^{M-k})}{1 + k(1-b_2) - b_2^{M-k+1}}, \quad \alpha_1 = \beta,$$
$$\lambda < \frac{\alpha_1(1-b_1^k) + \alpha_2 b_1^k(1-b_1)(M-k)}{1 - b_1^k(b_1 - (1-b_1)(M-k))}, \quad \alpha_2 = \beta.$$

Hieruit kunnen wij conclusies trekken voor systemen die niet aan (1) en (2) voldoen. Als een systeem S_1 met aankomstintensiteit λ en bediengssnelheden α_1, α_2 en β dat aan (1) en (2) voldoet stabiel is voor zekere *slowdown threshold* k_1 ($s_1 = k_1 - 1$) en buffercapaciteit M_1 ($r_1 = M_1 - 1$), dan zijn alle systemen S_2 met dezelfde aankomstintensiteit en bedieningssnelheden en met $s_2 = s_1$ ($k_2 \geq k_1$) en $r_2 = r_1$ ($M_2 \geq M_1$) zeker ook stabiel (zie figuur 3). Met andere woorden: Een stabiel systeem kan niet instabiel worden door de buffercapaciteit M en/of de *slowdown threshold* k te verhogen. De eerste server werkt in systeem S_2 bij een gegeven aantal producten in de tweede rij immers minimaal even snel als in systeem S_1.



Figuur 3: Systemen S_1 en S_2.

2.3 De kostenfunctie

We veronderstellen dat er kosten zijn verbonden aan de volgende gebeurtenissen: *blocking* van de eerste server en *starving* van de tweede server. Wij gaan ervanuit dat het geld kost om producten in een van de twee wachtruimtes op te slaan en dat deze opslagkosten lineair zijn. Bovendien is het duurder om de eerste machine op hoge snelheid te laten werken dan op lage.

Als we uitgaan van een lineaire kostenfunctie kunnen we de gemiddelde kosten per tijdseenheid schrijven als:

$$C = c_B \cdot E[B] + c_S \cdot E[S] + c_R \cdot E[R] + c_F \cdot E[F] + c_L \cdot E[L] \tag{5}$$

met:

$B \mathrel{\widehat{=}}$ Aantal keer dat *blocking* in stationariteit per tijdseenheid optreedt

$S \mathrel{\widehat{=}}$ Aantal keer dat *starving* in stationariteit per tijdseenheid optreedt

$R \mathrel{\widehat{=}}$ Aantal producten in stationariteit in het systeem

$F \mathrel{\widehat{=}}$ Percentage van de tijd dat de eerste server in stationariteit op hoge snelheid werkt

$L \mathrel{\widehat{=}}$ Percentage van de tijd dat de eerste server in stationariteit op lage snelheid werkt

$c_B \mathrel{\widehat{=}}$ *Blocking* – kosten per keer dat de eerste machine *blockt*

$c_S \mathrel{\widehat{=}}$ *Starving* – kosten per keer dat de tweede machine *starvt*

$c_R \mathrel{\widehat{=}}$ Opslagkosten per product per tijdseenheid

$c_F \mathrel{\widehat{=}}$ Kosten om de eerste machine een tijdseenheid op hoge snelheid te laten werken

$c_L \mathrel{\widehat{=}}$ Kosten om de eerste machine een tijdseenheid op lage snelheid te laten werken

We kijken dus alleen naar het aantal keren dat *blocking* en *starving* optreden en zijn niet geïnteresseerd in de tijdsduur van deze gebeurtenissen. De kostenfunctie bevat geen inkomsten omdat, als we uitgaan van een stabiel systeem, de intensiteit waarmee producten het systeem verlaten op lange termijn altijd gelijk is aan de intensiteit van het aankomstproces, ongeacht de configuratie van het systeem. De inkomsten zijn dus niet afhankelijk van k, s en r.

2.4 Doelstelling

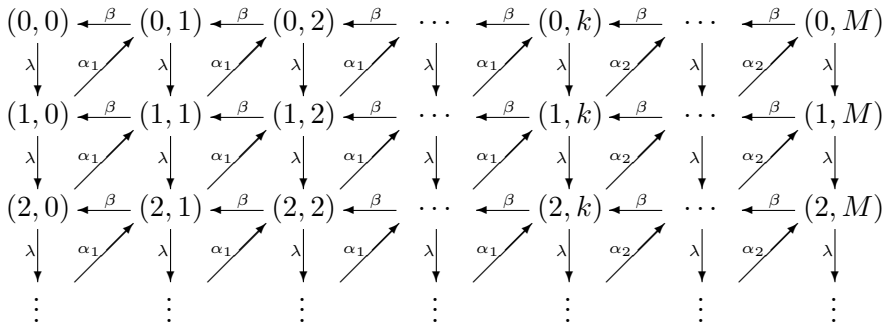
Ons doel is om, gegeven een systeem met buffercapaciteit M, aankomstintensiteit λ, bedieningssnelheden α_1, α_2 en β en bijbehorende kosten c_B, c_S, c_R, c_F en c_L, de parameters k, s en r zo te kiezen dat de verwachte kosten per tijdseenheid minimaal zijn. Als bij de herstartwaarde r de eerste server zowel met snelheid α_1 als ook met met snelheid α_2 zou kunnen werken, moeten we bovendien kiezen met welke van de twee mogelijke snelheden we de eerste server herstarten.

3 Analytische oplossing

We zijn op zoek naar de verwachtingswaarden van B, S, R, F en L zoals gedefinieerd in sectie 2.3. Al deze grootheden zijn uit te drukken in de evenwichtskansen van de toestanden.
Als aan (1) en (2) wordt voldaan, kunnen we de evenwichtskansen definiëren als

$$\pi_{i,j} = \lim_{t\to\infty} P(X_1(t) = i,\ X_2(t) = j).$$

We noemen dit geval voortaan het *eenvoudigste geval*.
Figuur 4 toont de toestandsruimte met bijbehorende overgangsintensiteiten:



Figuur 4: Toestandsruimte en overgangsintensiteiten in het eenvoudigste geval.

Er geldt:

$$E[B] = \alpha_2 \sum_{i=1}^{\infty} \pi_{i,M-1} \tag{6a}$$

$$E[S] = \beta \sum_{i=0}^{\infty} \pi_{i,1} \tag{6b}$$

$$E[R] = \sum_{i=0}^{\infty} \sum_{j=0}^{M} (i+j)\pi_{i,j} \tag{6c}$$

$$E[F] = \sum_{i=0}^{\infty} \sum_{j=0}^{k-1} \pi_{i,j} \tag{6d}$$

$$E[L] = \sum_{i=0}^{\infty} \sum_{j=k}^{M-1} \pi_{i,j} \tag{6e}$$

Als (1) en/of (2) niet gelden definiëren we de evenwichtskansen als

$$\pi_{i,j,v} = \lim_{t\to\infty} P(X_1(t) = i,\ X_2(t) = j,\ V(t) = v).$$

Voor de niet bereikbare toestanden is de evenwichtskans natuurlijk gelijk aan nul.
Er geldt:

$$E[B] = \sum_{v} v \sum_{i=1}^{\infty} \pi_{i,M-1,v} = \alpha_2 \sum_{i=1}^{\infty} \pi_{i,M-1,\alpha_2} \tag{7a}$$

$$E[S] = \beta \sum_{v} \sum_{i=0}^{\infty} \pi_{i,1,v} \tag{7b}$$

$$E[R] = \sum_{v} \sum_{i=0}^{\infty} \sum_{j=0}^{M} (i+j)\pi_{i,j,v} \tag{7c}$$

$$E[F] = \sum_{i=0}^{\infty} \sum_{j=0}^{k-1} \pi_{i,j,\alpha_1} \tag{7d}$$

$$E[L] = \sum_{i=0}^{\infty} \sum_{j=s+1}^{M-1} \pi_{i,j,\alpha_2} \tag{7e}$$

Dit geval noemen we vooraan het *algemene geval*.
We laten nu zien hoe de evenwichtskansen bepaald kunnen worden.

3.1 Het eenvoudigste geval

Grassmann en Drekic [7] bepalen de evenwichtskansen voor een tandem-wachtrij met *blocking*. Hun methode kan met kleine aanpassingen ook worden gebruikt als de bedieningssnelheid van de eerste server een functie is van het aantal producten in de tweede rij, zoals in ons systeem als voldaan is aan (1) en (2).

Na een transitie kan het aantal producten in de eerste rij met één toe- of afgenomen, of gelijk gebleven zijn. We definiëren Q_{+1} als de matrix met de intensiteiten van alle gebeurtenissen waardoor de rijlengte bij de eerste server stijgt, Q_{-1} als de matrix met de intensiteiten van alle gebeurtenissen waardoor de rijlengte bij de eerste server daalt en Q_0 en Q_{00} als de matrices met de intensiteiten van alle gebeurtenissen waardoor de rijlengte bij de eerste server gelijk blijft en groter dan nul is, dan wel gelijk aan nul is. Als $X_1(t) = i$ bekend is kan het systeem in $M+1$ mogelijke toestanden verkeren, dus zijn de matrices vierkant met dimensie $M+1$. De (oneindige) generator-matrix P kan geschreven worden als

$$\mathsf{P} = \begin{bmatrix} \mathsf{Q}_{00} & \mathsf{Q}_{+1} & \mathsf{0} & \mathsf{0} & \cdots \\ \mathsf{Q}_{-1} & \mathsf{Q}_{0} & \mathsf{Q}_{+1} & \mathsf{0} & \cdots \\ \mathsf{0} & \mathsf{Q}_{-1} & \mathsf{Q}_{0} & \mathsf{Q}_{+1} & \cdots \\ \mathsf{0} & \mathsf{0} & \mathsf{Q}_{-1} & \mathsf{Q}_{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{8}$$

met de $(M+1)\times(M+1)$ matrices gegeven door

$$\mathsf{Q}_{00} = \begin{bmatrix} -\lambda & & & \\ \beta & -q & & \\ & \ddots & \ddots & \\ & & \beta & -q \end{bmatrix},$$

$$\mathsf{Q}_0 = \begin{bmatrix} -\lambda-\alpha_1 & & & & & & & \\ \beta & -p_1 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & \beta & -p_1 & & & & \\ & & & \beta & -p_2 & & & \\ & & & & \ddots & \ddots & & \\ & & & & & \beta & -p_2 & \\ & & & & & & \beta & -q \end{bmatrix},$$

$$\mathsf{Q}_{-1} = \begin{bmatrix} 0 & \alpha_1 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 0 & \alpha_1 & & & & \\ & & & 0 & \alpha_2 & & & \\ & & & & \ddots & \ddots & & \\ & & & & & 0 & \alpha_2 & \\ & & & & & & 0 & \end{bmatrix},$$

$$\mathsf{Q}_{+1} = \lambda \mathsf{I}_{M+1},$$

met $q = \lambda+\beta$, $p_1 = \lambda+\alpha_1+\beta$, $p_2 = \lambda+\alpha_2+\beta$ en I_{M+1} de $(M+1)\times(M+1)$ identiteitsmatrix. In Q_{-1} staat in rij 1 t/m rij k een α_1 en in rij $k+1$ t/m rij M een α_2. Net zo staat in Q_0 in rij 2 t/m rij k een $-p_1$ en in rij $k+1$ t/m rij M een $-p_2$.

Als we de evenwichtsvectoren $\boldsymbol{\pi_i}$ definiëren als

$$\boldsymbol{\pi_i} = \begin{bmatrix} \pi_{i,0} & \pi_{i,1} & \dots & \pi_{i,M} \end{bmatrix}$$

worden de evenwichtsvergelijkingen gegeven door

$$\mathbf{0} = \boldsymbol{\pi_0}\mathsf{Q}_{00} + \boldsymbol{\pi_1}\mathsf{Q}_{-1} \tag{9}$$

$$\mathbf{0} = \boldsymbol{\pi_{i-1}}\mathsf{Q}_{+1} + \boldsymbol{\pi_i}\mathsf{Q}_0 + \boldsymbol{\pi_{i+1}}\mathsf{Q}_{-1}, \quad i > 0 \tag{10}$$

(10) heeft oplossingen van de vorm

$$\boldsymbol{\pi_i} = \boldsymbol{g}x^i, \ \boldsymbol{g} = \begin{bmatrix} g_0 & g_1 & \dots & g_M \end{bmatrix} \neq \mathbf{0}. \tag{11}$$

Er moet gelden dat $x < 1$ omdat een oplossing met $x \geq 1$ niet zou convergeren. (11) in (10) substitueren levert

$$\boldsymbol{g}\mathsf{Q}(\mathsf{x}) = 0 \tag{12}$$

met

$$\mathsf{Q}(\mathsf{x}) = \begin{bmatrix} \lambda - (\lambda+\alpha_1)x & \alpha_1 x^2 & & & & & & & \\ \beta x & \lambda - p_1 x & \alpha_1 x^2 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \beta x & \lambda - p_1 x & \alpha_1 x^2 & & & & \\ & & & \beta x & \lambda - p_2 x & \alpha_2 x^2 & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \beta x & \lambda - p_2 x & \alpha_2 x^2 & \\ & & & & & & \beta x & \lambda - qx \end{bmatrix} \tag{13}$$

We zijn dus op zoek naar de waarden van x waarvoor de matrix $\mathsf{Q}(\mathsf{x})$ een eigenwaarde gelijk aan nul heeft. Konheim en Reiser [8] laten zien dat er precies $M+1$ van deze waarden in de eenheidscirkel liggen. Elk van deze $M+1$ waarden vormt samen met zijn bijbehorende eigenvector $\boldsymbol{g}$ een oplossing van (10). Om de evenwichtskansen $\pi_{i,j}$ uit te rekenen bepalen we een lineaire combinatie van deze $M+1$ oplossingen zodat voldaan wordt aan (9).

We beginnen met het bepalen van de eigenvectoren. Vanwege (13) kunnen we (12) schrijven als

$$0 = g_0(\lambda - (\lambda+\alpha_1)x) + g_1\beta x, \tag{14}$$

$$0 = g_{j-1}\alpha_1 x^2 + g_j(\lambda - p_1 x) + g_{j+1}\beta x, \quad j = 1, 2, \ldots, k-1, \tag{15}$$

$$0 = g_{k-1}\alpha_1 x^2 + g_k(\lambda - p_2 x) + g_{k+1}\beta x, \tag{16}$$

$$0 = g_{j-1}\alpha_2 x^2 + g_j(\lambda - p_2 x) + g_{j+1}\beta x, \quad j = k+1, k+2, \ldots, M-1, \tag{17}$$

$$0 = g_{M-1}\alpha_2 x + g_M(\lambda - qx). \tag{18}$$

Als er geldt dat $x = 0$ is de vector $\boldsymbol{g}$ gelijk aan nul, dus moet x ongelijk nul zijn. Om dezelfde reden moet er gelden dat $g_0 \neq 0$, dus kunnen we $g_0 = 1$ kiezen. Hieruit en uit (14) volgt dat

$$g_1 = \frac{\lambda + \alpha_1 - \lambda/x}{\beta}.$$

De waarden g_2 t/m g_k kunnen worden bepaald met behulp van (15):

$$g_j = g_{j-1}\frac{p_1 - \lambda/x}{\beta} - g_{j-2}\frac{\alpha_1}{\beta}x, \quad j = 2, 3, \ldots, k.$$

Met (16) kan nu g_{k+1} berekend worden:

$$g_{k+1} = g_k\frac{p_2 - \lambda/x}{\beta} - g_{k-1}\frac{\alpha_1}{\beta}x.$$

De resterende g_j worden door (17) bepaald:

$$g_j = g_{j-1}\frac{p_2 - \lambda/x}{\beta} - g_{j-2}\frac{\alpha_2}{\beta}x, \quad j = k+2, k+3, \ldots, M.$$

Met behulp van de g_j kunnen we nu de $M+1$ verschillende waarden van x bepalen waarvoor de matrix $\mathsf{Q}(\mathsf{x})$ een eigenwaarde gelijk aan nul heeft. Dit zijn de waarden van x waarvoor voldaan wordt aan (18). We nummeren deze met $x_0, x_1, \ldots, x_M$ met $x_0 > x_1 > \cdots > x_M$. Elk van deze waarden x_n heeft een bijbehorende eigenvector $\boldsymbol{g^{(n)}}$.

We hebben nu $M+1$ verschillende oplossingen voor (10):

$$\boldsymbol{\pi_i^{(n)}} = \boldsymbol{g^{(n)}} x_n^i.$$

Elke lineaire combinatie van deze oplossingen is ook een oplossing, dus zijn alle mogelijke oplossingen te schrijven als

$$\boldsymbol{\pi_i} = \sum_{n=0}^{M} c_n \boldsymbol{\pi_i^{(n)}} = \sum_{n=0}^{M} c_n \boldsymbol{g^{(n)}} x_n^i.$$

Hierin zijn de parameters c_n willekeurige constantes. We bepalen nu de c_n zodat voldaan wordt aan (9):

$$\begin{aligned} \mathbf{0} &= \boldsymbol{\pi_0}\mathsf{Q}_{00} + \boldsymbol{\pi_1}\mathsf{Q}_{-1} = \sum_{n=0}^{M} c_n \boldsymbol{g^{(n)}}(\mathsf{Q}_{00} + x_n\mathsf{Q}_{-1}) \\ \Leftrightarrow \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} &= \begin{bmatrix} \boldsymbol{g^{(0)}}(\mathsf{Q}_{00} + x_0\mathsf{Q}_{-1}) \\ \boldsymbol{g^{(1)}}(\mathsf{Q}_{00} + x_1\mathsf{Q}_{-1}) \\ \vdots \\ \boldsymbol{g^{(M)}}(\mathsf{Q}_{00} + x_M\mathsf{Q}_{-1}) \end{bmatrix}^T \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_M \end{bmatrix}. \end{aligned} \tag{19}$$

Grassmann en Drekic [7] laten zien dat x_0 de enige mogelijke waarde van x is waarvoor geldt

$$x_0 > max(\frac{\lambda}{\alpha_1}, \frac{\lambda}{\beta}).$$

Hieruit volgt dat x_0 deel uit moet maken van de oplossing omdat het systeem anders beter zou presteren dan een systeem met maar één server met exponentiële bedieningssnelheid gelijk aan het minimum van α_1 en β. We kunnen dus $c_0 = 1$ kiezen, (19) wordt dan

$$\begin{bmatrix} \boldsymbol{g^{(1)}}(\mathsf{Q}_{00} + x_1\mathsf{Q}_{-1}) \\ \boldsymbol{g^{(2)}}(\mathsf{Q}_{00} + x_2\mathsf{Q}_{-1}) \\ \vdots \\ \boldsymbol{g^{(M)}}(\mathsf{Q}_{00} + x_M\mathsf{Q}_{-1}) \end{bmatrix}^T \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} = -(\mathsf{Q}_{00} + x_0\mathsf{Q}_{-1})^T \begin{bmatrix} g_0^{(0)} \\ g_1^{(0)} \\ \vdots \\ g_M^{(0)} \end{bmatrix}.$$

Hiermee kunnen we alle c_n uitrekenen. Als laatste stap moeten we de c_n nog met een factor K vermenigvuldigen, zodat de som van alle evenwichtskansen gelijk is aan 1. Er moet dus gelden:

$$\sum_{i=0}^{\infty}\sum_{j=0}^{M}\pi_{i,j} = \sum_{i=0}^{\infty}\sum_{n=0}^{M}\sum_{j=0}^{M}Kc_n g_j^{(n)}x_n^i = K\sum_{n=0}^{M}\sum_{j=0}^{M}\frac{c_n g_j^{(n)}}{1-x_n} = 1.$$

Dit is natuurlijk alleen het geval als

$$K = \left(\sum_{v=0}^{M}\sum_{w=0}^{M}\frac{c_v g_w^{(v)}}{1-x_v}\right)^{-1}.$$

De evenwichtskansen zijn nu gegeven door

$$\pi_{i,j} = K\sum_{n=0}^{M}c_n g_j^{(n)}x_n^i.$$

3.2 Het algemene geval

In het algemene geval legt het aantal producten in de tweede rij de snelheid waarmee de eerste machine werkt niet meer altijd vast. Er ontstaan 'dubbele' of zelfs 'driedubbele' toestanden waarvoor de eerste machine voor een gegeven aantal producten in de tweede rij op twee of zelfs drie (α_1, α_2 of 0) verschillende snelheden kan werken.

Er zijn k verschillende lengtes van de tweede rij waarvoor de eerste server snel zou kunnen werken $(0, 1, \ldots, k-1)$, $M-s-1$ lengtes waarbij de eerste machine langzaam zou kunnen werken $(s+1, s+2, \ldots, M-1)$ en $M-r$ lengtes waarvoor de eerste geblokkeerd zou kunnen zijn $(r+1, r+2, \ldots, M)$. Als $X_1(t) = i$ bekend is kan het systeem dus in $2M+k-(s+r+1)$ mogelijke toestanden verkeren.
Er geldt ook hier dat het aantal producten in de eerste rij na een transitie alleen met één toe- of afgenomen, of gelijk gebleven kan zijn. Dus ook voor dit systeem kan de (oneindige) generator-matrix P geschreven worden zoals in (8). De matrices Q_{00}, Q_0, Q_{-1} en Q_1 zijn dus vierkant met dimensie $2M+k-(s+r+1)$. Om de matrices compact weer te geven definiëren we met $\mathsf{S}_{\mathsf{d}}^{(\mathsf{n})}$ de vierkante matrix van dimensie n die we verkrijgen door in de eenheidsmatrix alle diagonaal elementen d plaatsen naar rechts te verschuiven. Alle elementen van $\mathsf{S}_{\mathsf{d}}^{(\mathsf{n})}$ zijn dus nul behalve de elementen d posities rechts van de diagonaal die één zijn. Er geldt:

$$\mathsf{Q}_{00} = \begin{bmatrix} \mathsf{Q}_{00}^{(\mathsf{fast})} & 0 & 0 \\ \mathsf{Q}_{\mathsf{speedup}} & \mathsf{Q}_{00}^{(\mathsf{slow})} & 0 \\ \mathsf{Q}_{\mathsf{restart}}^{(\mathsf{fast})} & \mathsf{Q}_{\mathsf{restart}}^{(\mathsf{slow})} & \mathsf{Q}_{00}^{(\mathsf{zero})} \end{bmatrix},$$

$$\mathsf{Q}_{0} = \begin{bmatrix} \mathsf{Q}_{0}^{(\mathsf{fast})} & 0 & 0 \\ \mathsf{Q}_{\mathsf{speedup}} & \mathsf{Q}_{0}^{(\mathsf{slow})} & 0 \\ \mathsf{Q}_{\mathsf{restart}}^{(\mathsf{fast})} & \mathsf{Q}_{\mathsf{restart}}^{(\mathsf{slow})} & \mathsf{Q}_{0}^{(\mathsf{zero})} \end{bmatrix},$$

$$\mathsf{Q}_{-1} = \begin{bmatrix} \mathsf{Q}_{-1}^{(\mathsf{fast})} & \mathsf{Q}_{\mathsf{slowdown}} & 0 \\ 0 & \mathsf{Q}_{-1}^{(\mathsf{slow})} & \mathsf{Q}_{\mathsf{block}} \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathsf{Q}_{+1} = \lambda \mathsf{I}_{2\mathsf{M}+\mathsf{k}-(\mathsf{s}+\mathsf{r}+1)},$$

met

$$\begin{aligned}
\mathsf{Q}_{00}^{(\mathsf{fast})} &= \beta \mathsf{S}_{-1}^{(\mathsf{k})} - \lambda \mathsf{S}_{0}^{(\mathsf{k})} - \beta \mathsf{S}_{-1}^{(\mathsf{k})} \mathsf{S}_{1}^{(\mathsf{k})}, \\
\mathsf{Q}_{00}^{(\mathsf{slow})} &= \beta \mathsf{S}_{-1}^{(\mathsf{M}-\mathsf{s}-1)} - (\lambda+\beta) \mathsf{S}_{0}^{(\mathsf{M}-\mathsf{s}-1)}, \\
\mathsf{Q}_{00}^{(\mathsf{zero})} &= \beta \mathsf{S}_{-1}^{(\mathsf{M}-\mathsf{r})} - (\lambda+\beta) \mathsf{S}_{0}^{(\mathsf{M}-\mathsf{r})}, \\
\mathsf{Q}_{0}^{(\mathsf{fast})} &= \beta \mathsf{S}_{-1}^{(\mathsf{k})} - (\lambda+\alpha_1) \mathsf{S}_{0}^{(\mathsf{k})} - \beta \mathsf{S}_{-1}^{(\mathsf{k})} \mathsf{S}_{1}^{(\mathsf{k})}, \\
\mathsf{Q}_{0}^{(\mathsf{slow})} &= \beta \mathsf{S}_{-1}^{(\mathsf{M}-\mathsf{s}-1)} - (\lambda+\alpha_2+\beta) \mathsf{S}_{0}^{(\mathsf{M}-\mathsf{s}-1)}, \\
\mathsf{Q}_{0}^{(\mathsf{zero})} &= \beta \mathsf{S}_{-1}^{(\mathsf{M}-\mathsf{r})} - (\lambda+\beta) \mathsf{S}_{0}^{(\mathsf{M}-\mathsf{r})}, \\
\mathsf{Q}_{-1}^{(\mathsf{fast})} &= \alpha_1 \mathsf{S}_{1}^{(\mathsf{k})}, \\
\mathsf{Q}_{-1}^{(\mathsf{slow})} &= \alpha_2 \mathsf{S}_{1}^{(\mathsf{M}-\mathsf{s}-1)}.
\end{aligned}$$

$\mathsf{Q}_{\mathsf{speedup}}$, $\mathsf{Q}_{\mathsf{slowdown}}$ en $\mathsf{Q}_{\mathsf{block}}$ hebben elk maar één niet-nul element. $\mathsf{Q}_{\mathsf{speedup}}$ is een $(M-s-1)\times k$ matrix met een β op positie $(1,s)$. $\mathsf{Q}_{\mathsf{slowdown}}$ is een $k\times(M-s-1)$ matrix net een α_1 op positie $(k,k-s)$. $\mathsf{Q}_{\mathsf{block}}$ is een $(M-s-1)\times(M-r)$ matrix met een α_2 op positie $(M-s-1,M-r)$. Als er geldt dat $r \leq s$, wordt de eerste machine nadat hij geblokkeerd is geweest op hoge snelheid herstart; als $r \geq k$ op lage. Als er geldt dat $s < r < k$ zou de eerste machine snel of langzaam herstart kunnen worden en moet er dus een keuze worden gemaakt.
Als de eerste server op hoge snelheid herstart wordt is $\mathsf{Q}_{\mathsf{restart}}^{(\mathsf{fast})}$ een $(M-r)\times k$ matrix met maar één niet-nul element op positie $(1,r)$. Dit element is β en er geldt $\mathsf{Q}_{\mathsf{restart}}^{(\mathsf{slow})} = 0$.
Als de eerste server op lage snelheid herstart wordt is $\mathsf{Q}_{\mathsf{restart}}^{(\mathsf{fast})} = 0$ en $\mathsf{Q}_{\mathsf{restart}}^{(\mathsf{slow})}$ een $(M-r)\times(m-s-1)$ matrix met maar één niet-nul element (β) op positie $(1,r-s)$.

Eigenlijk zouden we nu de evenwichtskansen op dezelfde manier als in sectie 3.1 moeten kunnen bepalen. Het lukt ook om $2M+k-(s+r+1)$ verschillende oplossingen van (10) te vinden. Het lukt echter niet om een niet-triviale lineaire combinatie van deze oplossingen te bepalen zodat voldaan is aan (9). Het is ons niet gelukt te achterhalen waar het precies fout gaat. We hebben de evenwichtskansen $\pi_{i,j,v}$ daarom op de volgende manier benadert: Als het aantal producten in de eerste rij begrensd is tot een zeker aantal n is het aantal toestanden en daarmee dus ook de generator matirx P eindig. De evenwichtskansen voor dit systeem kunnen worden dan gegeven door de *nullspace* van P te bepalen en op één te normeren. Voor grote n is deze benadering heel erg nauwkeurig zolang het systeem niet 'bijna onstabiel' is.

4 Simulatie

Naast de analytische oplossing hebben we ook een simulatie programma geschreven. Dit programma maakt gebruik van Discrete-Event Simulatie, wat we hier eerst uiteenzetten. Vervolgens bespreken we in het kort ons specifieke programma. Er wordt in dit verslag echter niet op het volledige programma ingegaan, omdat het alleen gebruikt wordt als hulpmiddel en verder geen toegevoegde waarde heeft voor ons onderzoek.

4.1 Discrete-Event Simulatie

Discrete-Event Simulatie houdt in dat het systeem beschouwd wordt als een chronologische opeenvolging van gebeurtenissen (events). Elke gebeurtenis doet zich voor op een bepaald punt in de tijd, en brengt veranderingen in het systeem.

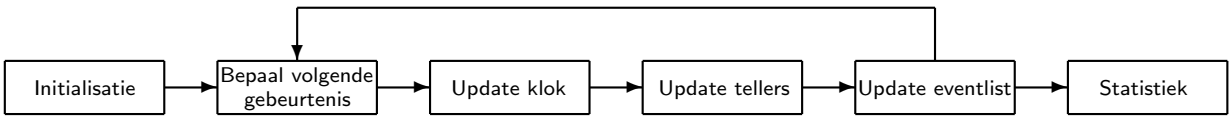
Belangrijke aspecten van Discrete-Event Simulatie zijn:

Simulatie-Klok: Deze houdt bij hoeveel tijd er verstreken is binnen de simulatie. Deze tijd is niet continu maar springt steeds naar het tijdstip van de eerstvolgende gebeurtenis.

Tellers: Dit zijn parameters die data uit de simulatie bijhouden. Welke data bijgehouden wordt is afhankelijk van wat onderzocht wordt, in ons geval o.a. de gemiddelde wachttijd en het aantal keer dat de eerste server geblockt wordt.

Event Lijst: Deze lijst houdt aankomende gebeurtenissen bij. Per mogelijke gebeurtenis staat in deze lijst het eerstvolgende tijdstip waarop deze optreedt.

Het simulatieprogramma bepaalt welke van de aankomende gebeurtenissen als eerste optreedt, en verzet de klok naar het tijdsip waarop dit gebeurt. Vervolgens worden de tellers bijgewerkt. Hoe deze precies veranderen hangt af van de huidige staat van het systeem en welke gebeurtenis er zojuist is gesimuleerd. Hierna wordt bepaald wanneer de zojuist gesimuleerde gebeurtenis weer optreedt, en wordt de event lijst hiermee bijgewerkt.
Dit proces blijft zichzelf herhalen totdat het systeem wordt gestopt. Dit kan bijvoorbeeld als er een bepaalde hoeveelheid tijd is verstreken of als een bepaalde hoeveelheid producten het systeem verlaten hebben. Schematisch ziet het er als volgt uit:



Figuur 5: Blokschema van de simulatie.

4.2 Ons programma

Voor de simulatie hebben wij gekozen vor Matlab, omdat het een relatief simpele programmeertaal is met vrij uitgebreide mathematische functies.

Omdat we een systeem simuleren met twee wachtrijen, bestaat de event lijst uit 3 elementen:

- Tijdstip van eerstvolgende aankomst bij eerste server.
- Tijdstip van eerstvolgende vertrek bij eerste server.
- Tijdstip van eerstvolgende vertrek bij tweede server.

Als er geen producten in de eerste of tweede wachtrij staan wordt het bijbehorende element van de event lijst op oneindig gezet. Dit garandeert dat er geen 'onmogelijke' gebeurtenissen optreden.
Ook wordt er een grote hoeveelheid tellers gebruikt. Dit zorgt ervoor dat alle gegevens waar wij in geïnteresseerd zijn worden bijgehouden. Zo wordt er o.a. bijgehouden of de servers bezet zijn of niet en hoeveel producten er in de wachtrij staan, maar ook bijvoorbeeld percentage van de tijd dat de eerste server geblockt is geweest.
In de initialisatie worden alle tellers op hun nulwaarden gezet, en wordt de eerste aankomst in de event lijst gezet. Vervolgens begint de simulatie: er wordt bepaald welke van de 3 gebeurtenissen als eerste optreedt, ofwel wat het minimum is van de drie elementen in de event lijst. Deze gebeurtenis wordt vervolgens gesimuleerd en de tijd, tellers en event lijst worden geupdate. Dit proces blijft zichzelf herhalen tot een bepaald stop-criterium is bereikt. Wat dit criterium precies is hangt af van de gegevens die verzameld moeten worden. Als we geïnteresseerd zijn in bijvoorbeeld de gemiddelde wachttijd per product, dan simuleren we tot een bepaald aantal producten het systeem heeft verlaten. Zijn we geïnteresseerd in de gemiddelde rijlengte, dan simuleren we een bepaalde tijd.
Dit is essentiëel als meerdere simulatieruns met elkaar vergeleken worden. Als je bijvoorbeeld de gemiddelde wachttijd van een product wil bepalen en je simuleert een aantal keer een bepaalde tijdsduur dan zijn het aantal afgehandelde producten in een simulatierun en de gemiddelde wachttijd in een simulatierun gecorreleerd. Hiermee kan in de statistische analyse rekening gehouden worden maar het is eenvoudiger om een ander stop-criterium te kiezen. In de realiteit wordt dit echter vaak verwaarloosd, wat leidt tot incorrecte gegevens. Dit staat ook wel bekend als de 'boerenfluitjes methode'.

Als de simulatie eenmaal is beëindigd, kunnen alle relevante data worden bepaald. Omdat één enkele simulatie natuurlijk geen betrouwbare resultaten geeft, moet deze vaak worden herhaald. Per 'meting' die gedaan wordt simuleren we dan ook vele (duizenden) keren onafhankelijk van elkaar, waarbij de data elke keer worden opgeslagen in een matrix.
Na één simulatierun hebben we dus een matrix vol met de resultaten van vele onafhankelijke

simulaties. Per opgeslagen variabele bepalen we het gemiddelde van al deze metingen, dit is de waarde die wij vervolgens gebruiken. Omdat het echter gaat om een grote verzameling meetresultaten, zegt een gemiddelde statistisch gezien niet veel. Er zal altijd een betrouwbaarheidsinterval gegeven moeten worden.
Een betrouwbaarheidsinterval geeft een intervalschatting voor een parameter, terwijl een gemiddelde slechts een puntschatting geeft. Van alle realisaties van het interval zullen sommige de waarde van de geschatte parameter wel bevatten, maar sommige ook niet. Hoe groter de betrouwbaarheid, hoe 'vaker' het interval de parameter bevat. Voor het bepalen van de grenzen van het betrouwbaarheidsinterval is wel enig rekenwerk nodig.

Stel we hebben een rij van n meetresultaten, $X_0, X_1, \ldots, X_n$, met bijbehorend gemiddelde

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_n$$

en steekproefvariantie:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

De meetresultaten $X_0, X_1, \ldots, X_n$ zijn onderling onafhankelijk en gelijk verdeeld met (onbekende) verwachting μ en standaardafwijking σ. Volgens de centrale limiet stelling heeft het gemiddelde van al deze (stochastische) variabelen voor grote N bij benadering een normale verdeling met verwachting μ en standaardafwijking σ/n. Nu kunnen de grenzen van het betrouwbaarheidsinterval bepaald worden door te standaardiseren:

$$Z = \frac{\bar{X} - \mu}{S/\sqrt{n}}$$

Om vervolgens een 95% betrouwbaarheidsinterval te construeren moet er een waarde voor c gevonden worden zodanig dat

$$P\{-c < Z < c\} = 0.95$$

Deze waarde is op te zoeken in tabellen voor standaard normale verdeling en is 1.96. Nu is de bovenstaande vergelijking te herschrijven naar.

$$P\{-1.96 < \frac{\bar{X} - \mu}{S/\sqrt{n}} < 1.96\} = 0.95$$
$$\Leftrightarrow P\{\bar{X} - 1.96\frac{S}{\sqrt{n}} < \mu < \bar{X} + 1.96\frac{S}{\sqrt{n}}\} = 0.95$$

Het 95% betrouwbaarheidsinterval voor μ, de waarde die we willen schatten, wordt dus gegeven door

$$[\bar{X} - 1.96\frac{S}{\sqrt{n}}\ ,\ \bar{X} + 1.96\frac{S}{\sqrt{n}}].$$

5 Resultaten

5.1 Voorbeelden

In deze paragraaf zullen we enkele resultaten bespreken en analyseren. Elk aspect van de kosten zullen we hier in grafiekvorm weergeven, waarbij we de analytische resultaten vergelijken met de resultaten uit de simulatie. Bij deze laatste zijn ook de 95% betrouwbaarheidsintervallen weergegeven.

5.1.1 Voorbeeld 1: $\lambda = 1$, $\alpha_1 = 2$, $\alpha_2 = 1.1$, $\beta = 1.4$ en $M = 5$

Als eerste bekijken we een systeem met $\lambda = 1$, $\alpha_1 = 2$, $\alpha_2 = 1.1$, $\beta = 1.4$ en $M = 5$. We variëren k en r en kiezen s gelijk aan $k - 1$. De resultaten zijn weergegeven in figuur 7.

De eerste (blauwe) grafiek toont het verwachte aantal keer dat *blocking* per tijdseenheid optreedt. Wat hier opvalt is dat *blocking* bij $k = 1$ en $r = 4$ vaker optreed dan bij $k = 2$ en $r = 4$ (voor zowel $s = 1$ als $s = 0$). Dit lijkt op het eerste gezicht raar omdat de eerste server bij een hogere *slowdown treshold* vaker op hoge snelheid werkt en de buffer daarom gemiddeld voller zit. Je zou dus verwachten dat de eerste server om die reden vaker *blockt*. Dat dit niet het geval is ligt aan het feit dat er alleen *blocking* kan optreden als er minimaal één product bij de eerste server is, en dat dit bij een hogere *slowdown treshold* minder vaak het geval is. Met andere woorden: $\sum_{i=0}^{\infty} \pi_{i,4,\alpha_2}$ is weliswaar groter voor $k = 2$ dan voor $k = 1$,maar $\sum_{i=1}^{\infty} \pi_{i,4,\alpha_2}$ is kleiner. Verder gedraagt de $E[B]$ zich zoals verwacht. Voor een hogere r restart de eerste server met een vollere buffer, en is de kans dus groter dat er binnen korte tijd weer geblockt wordt. Eveneens voor hogere k werkt de eerste server vaker op hoge snelheid, dus zal er relatief vaker geblockt worden.

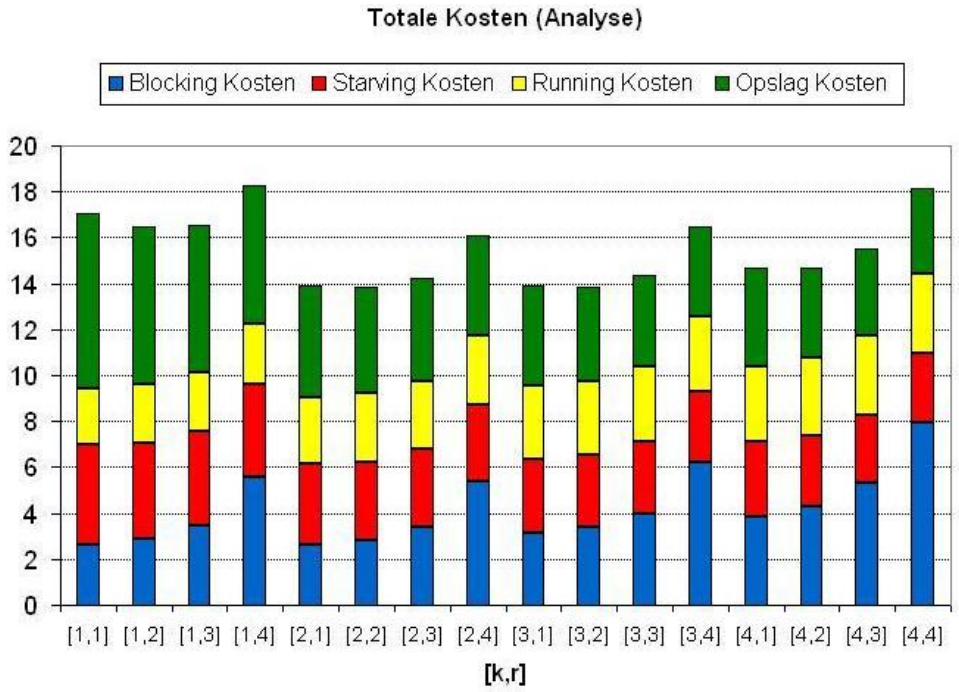
In de *starving* grafiek in figuur 7 is duidelijk te zien dat *starving* lang niet zo sterk beïnvloed wordt door de parameterkeuze als *blocking*. Voor een vaste waarde van k neemt $E[S]$ lichtelijk toe naarmate r kleiner wordt. Dit viel al wel te voorspellen; als er later wordt gerestart zullen relatief minder producten ophopen bij de tweede server en zal deze dus relatief vaker leeg zijn(*starving*).
Dat de $E[S]$ daalt naarmate k stijgt volgt logischerwijs uit het feit dat als de eerste server vaker op hogere snelheid werkt, deze dan regelmatiger producten aflevert bij de tweede server die dan dus minder vaak *starved*.

In de gele grafiek van figuur 7 zijn de verwachte *running*-kosten per tijdseenheid weergegeven. Omdat de totale *running*-kosten relevanter zijn dan de twee componenten hiervan (kosten voor hoge snelheid en kosten voor lage snelheid) zijn in figuur 8 de totale kosten U uitgezet: $E[U] := c_F \cdot E[F] + c_L \cdot E[L]$ met $c_F = 4$ en $c_L = 2.2$. We hebben voor lineaire *running*-kosten gekozen: de kosten om met een bepaalde snelheid te werken zijn evenredig aan de snelheid zelf.

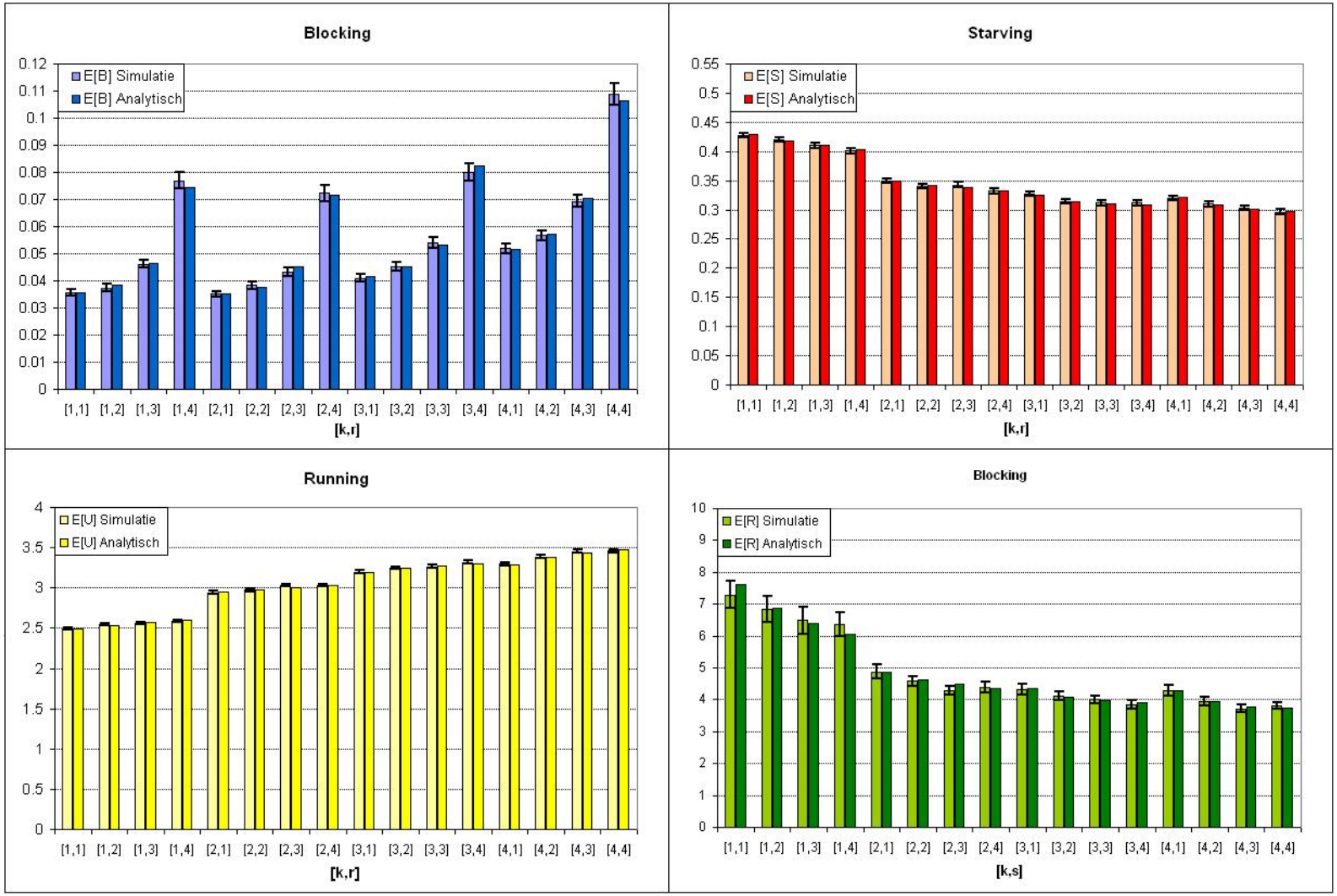
Deze kosten gedragen zich zoals verwacht, voor een toenemende k nemen ze toe, er wordt dan immers vaker op hoge snelheid gewerkt wat duurder is. Voor elk van de vaste waarden van k geldt dat $E[U]$ licht stijgt voor toenemende r, als de eerste server sneller herstart wordt er vaker snel gewerkt en zullen de *running*-kosten dus logischerwijs hoger liggen.

De groene grafiek in figuur 7 toont het gemiddelde aantal producten in het systeem. Voor een vaste waarde van k wordt $E[R]$ kleiner naarmate r toeneemt. Als er namelijk sneller wordt gerestart (een hoge r) is de eerste server relatief kort geblockt en zullen er minder producten ophopen dan wanneer er laat wordt gerestart (een lage r).
Voor een toemende k geldt dat de eerste server vaker snel werkt, en zal $E[R]$ dus afnemen. Ook zal voor grotere k er relatief iets vaker geblockt worden (wat voor een hogere $E[R]$ zorgt); dit weegt echter pas voor $k = 4$ op tegen de afname door hogere gemiddelde snelheid. Daar is namelijk te zien dat voor $r = 0$ en $r = 1$ de waarden voor $E[R]$ groter zijn dan die bij $k = 3$.

Voor het bepalen van de totale kosten moeten de coëfficienten voor de vier aspecten nog bepaald worden. Omdat de nadruk wordt gelegd op het voorkomen van *blocking* en *starving*, zullen deze kosten relatief zwaarder meewegen. Dit leidt tot de volgende keuze voor de coëfficienten: $c_B = 75$, $c_S = 10$, $c_R = 1$ en de al eerder bepaalde $c_F = 4$ en $c_L = 2.2$. Voor deze keuze worden de gemiddelde totale kosten per tijdseenheid gegeven in figuur 6.
De waarden uit deze figuur zijn de analytische, de uitkomsten van de simulatie komen hiermee overeen.

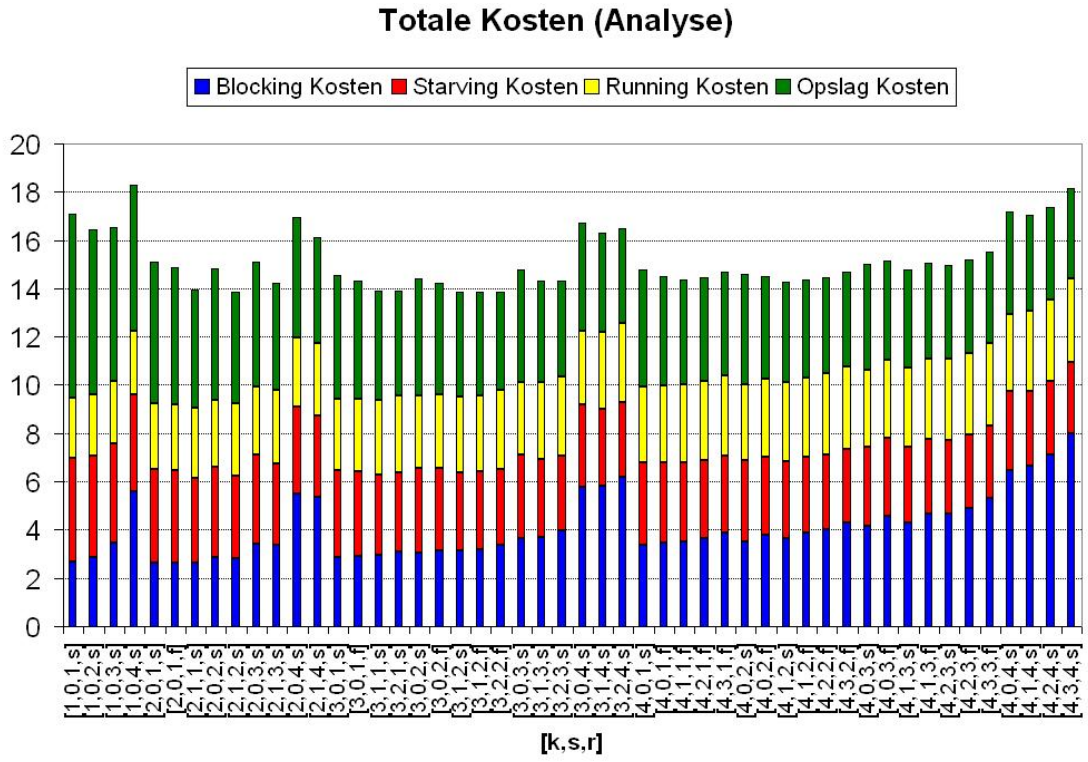


Figuur 6: Totale kosten in **Voorbeeld 1**. Het optimum ligt bij $k = 2$ en $r = 2$.



Figuur 7: Voorbeeld 1: $\lambda = 1$, $\alpha_1 = 2$, $\alpha_2 = 1.1$, $\beta = 1.4$ en $M = 5$. **Blauw:** Gem. aantal keer *blocking* per tijdseenheid. **Rood:** Gem. aantal keer *starving* per tijdseenheid. **Geel:** Gem. *running*-kosten per tijdseenheid. **Groen:** Gem. aantal producten in het systeem.

We hebben ook de kosten voor alle andere mogelijke configuraties van het systeem (dus s niet noodzakelijk gelijk aan $k-1$) bepaald. De resultaten staan in figuur 8. De letters in de vierkante haken onder de x-as geven aan of de eerste machine snel (f) of langzaam (s) herstart wordt.



Figuur 8: Totale kosten in **Voorbeeld 1** voor alle mogelijke configuraties.

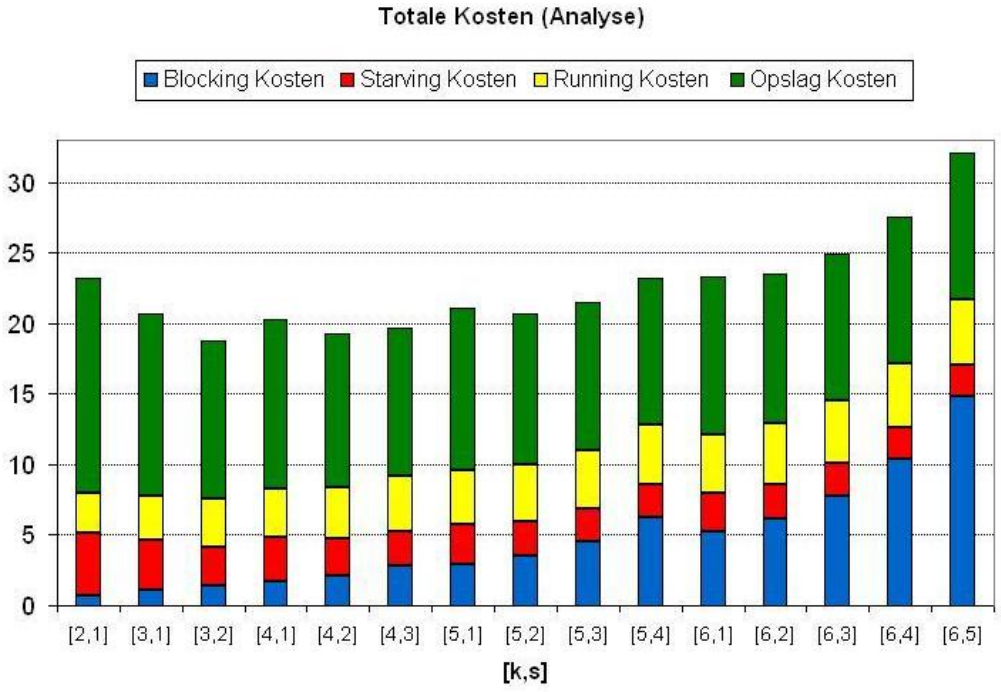
Het kiezen van een andere waarde voor s dan $k-1$ blijkt geen erg grote invloed op de totale kosten te hebben. De minimale kosten zijn in principe even hoog als bij een vaste $s = k-1$. Interessant is het ook om de resultaten de vergelijken die van een 'eenvoudige' machine zonder *slowdown* en keuzevrijheid van de herstart treshold r. Voor dit systeem geldt:

$$E[B] \approx 0.155$$
$$E[S] \approx 0.294$$
$$E[R] \approx 3.663$$
$$E[U] \approx 3.559$$

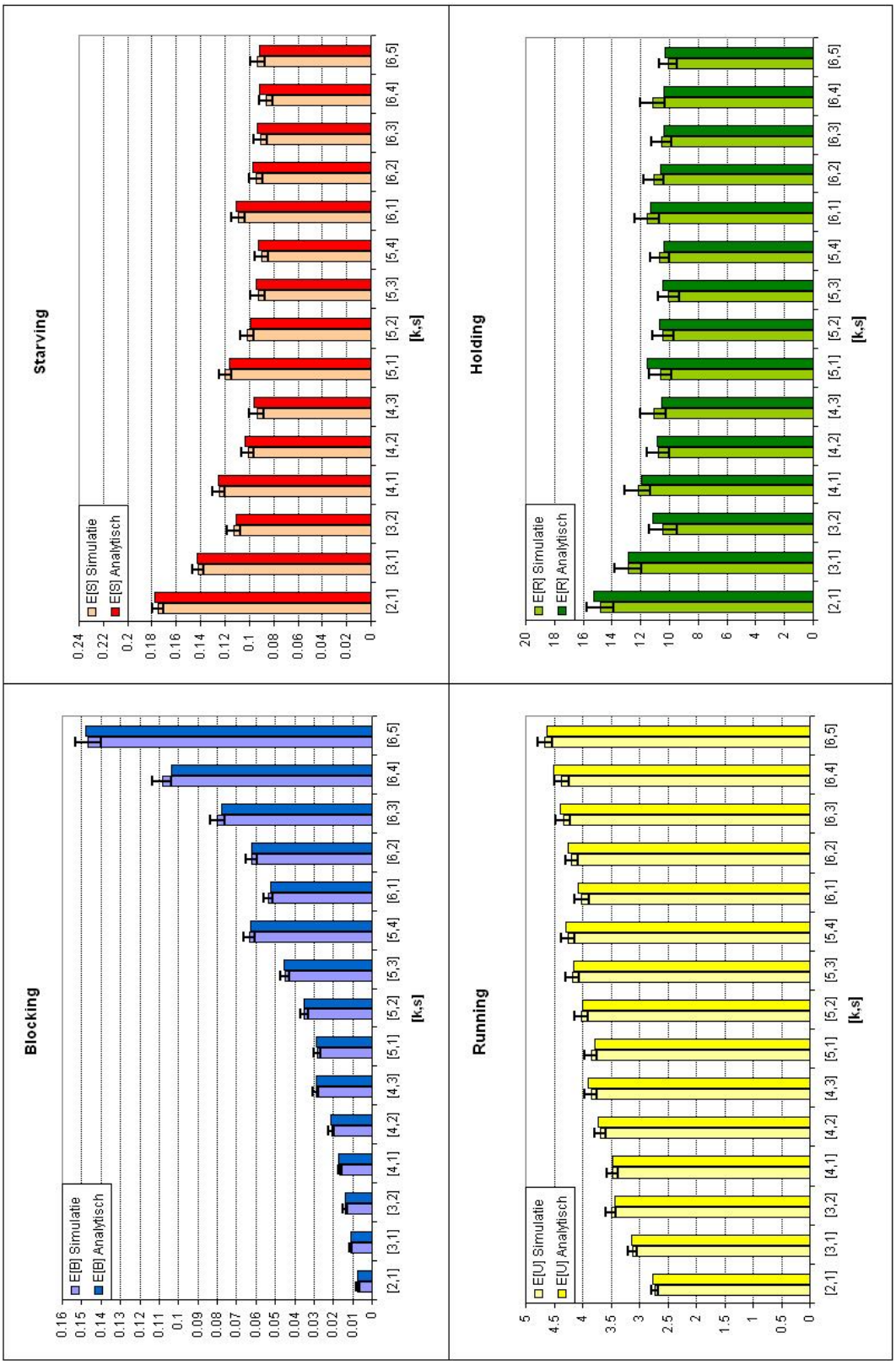
Het gemiddelde aantal keer dat de eerste server per tijdseenheid blokkeert is aanzienlijk groter dan bij een systeem met *slowdown*, terwijl E[S], E[U] en E[R] zijn bijna even groot als in de configuratie $k = 4$, $s = 3$ en $r = 4$.

5.1.2 Voorbeeld 2: $\lambda = 1$, $\alpha_1 = 4$, $\alpha_2 = 0.5$, $\beta = 1.1$ en $M = 7$

We beschouwen hier het systeem met $\lambda = 1$, $\alpha_1 = 4$, $\alpha_2 = 0,5$, $\beta = 1,1$ en $M = 7$. Het grote verschil tussen de twee mogelijke snelheden van server 1 zorgt ervoor dat deze haast werkt als een aan/uit functie. Als er wordt gewerkt op hoge snelheid zullen de producten extreem snel doorstromen naar de tweede server, als er wordt gewerkt op langzame snelheid wordt deze doorstroom zo goed als gestopt. Dit in combinatie met de lage snelheid bij de tweede server zorgt ervoor dat de tweede server hier duidelijk de *bottleneck* is. Aangezien de snelheid van de tweede server erg dicht bij de aankomstintensiteit ligt, zal het verwachte aantal mensen in het systeem ook relatief hoog zijn. Om deze reden is er gekozen voor een iets hogere waarde van M. We variëren k en s, en kiezen r altijd gelijk aan $M - 1$. Door de grotere buffercapaciteit, de hoge α_1 en de kleine α_2 kan in dit geval de hoeveelheid werk in de buffer beter beïnvloed worden. Dit is te zien in figuur 10, met een lage *slowdown treshold* kan *blocking* bijna helemaal worden voorkomen.
Uit (4) volgt dat het systeem voor $k = 1$ instabiel is. We kunnen dus stabiliteit in de gevallen $k > 1$, $s = 0$ niet garanderen.
Voor het bepalen van de totale kosten hebben we de volgende waarden voor de coëfficienten gekozen: $c_B = 100$, $c_S = 25$, $c_R = 1$,$c_F = 8$ en $c_L = 1$. Voor deze keuze worden de gemiddelde totale kosten per tijdseenheid gegeven in figuur 9.



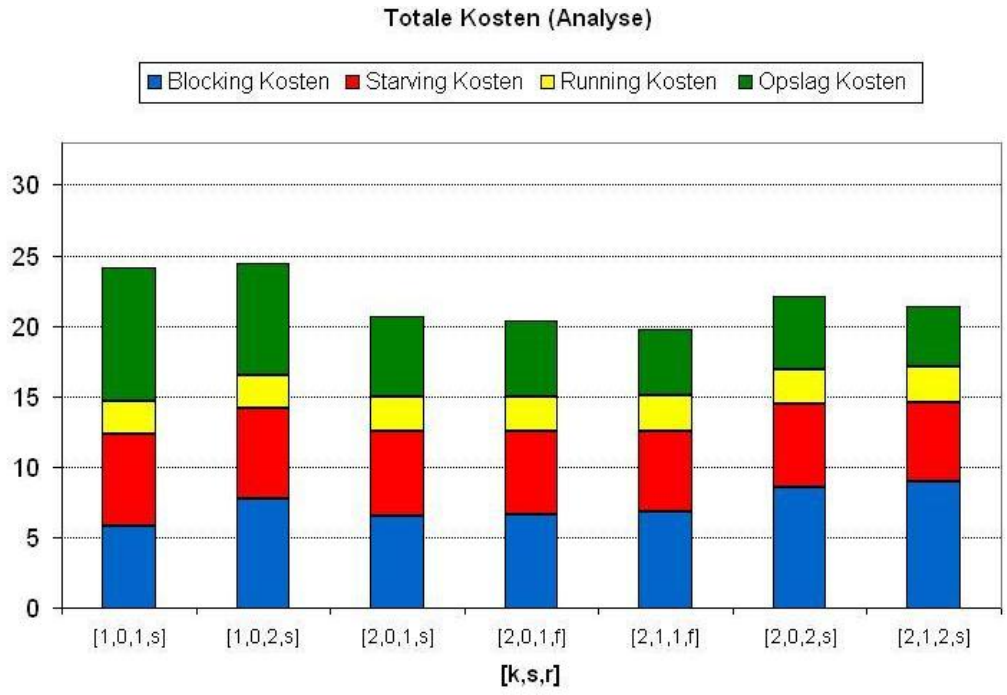
Figuur 9: Totale kosten in **Voorbeeld 2**.



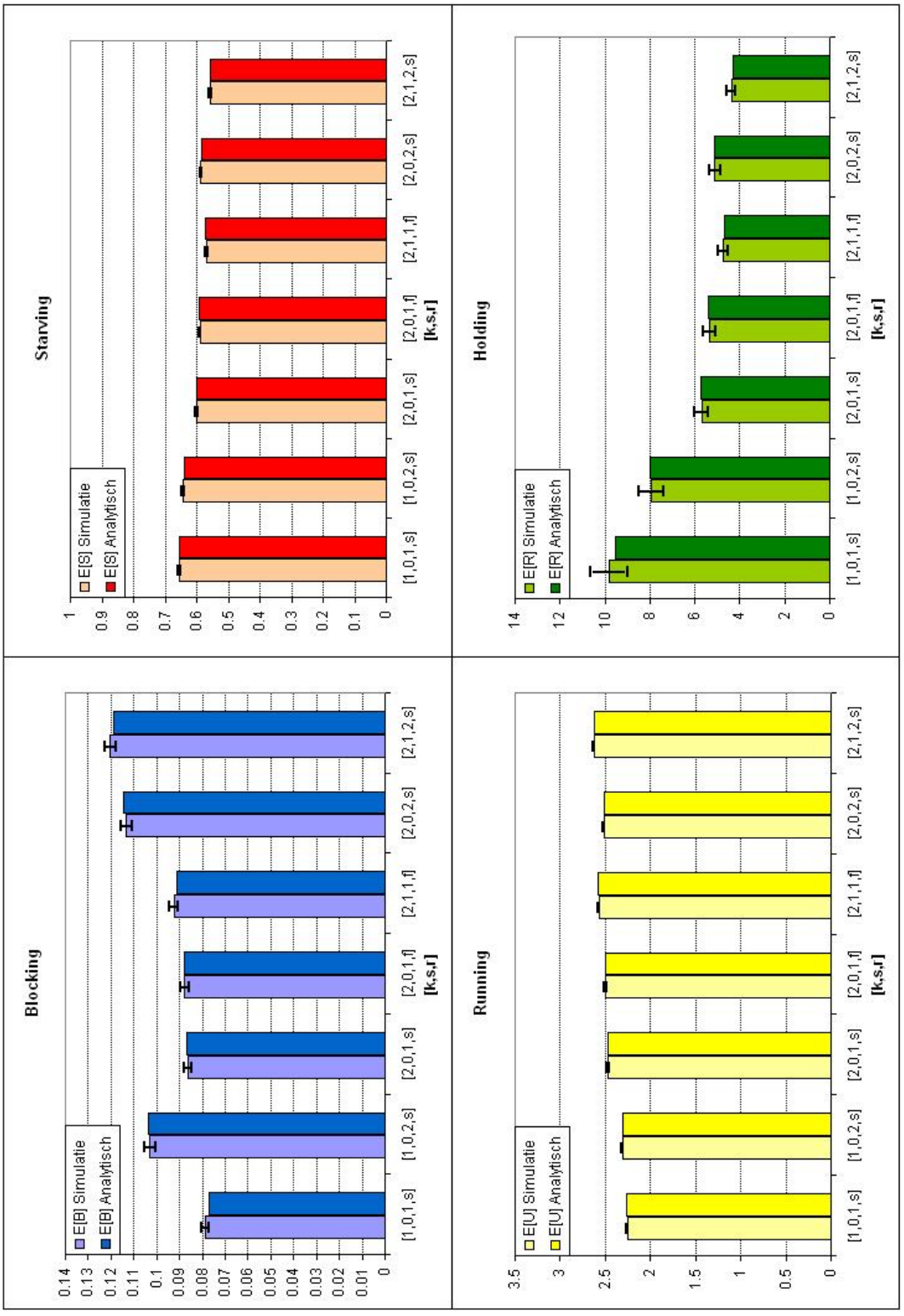
Figuur 10: Voorbeeld 2: $\lambda = 1$, $\alpha_1 = 4$, $\alpha_2 = 0.5$, $\beta = 1.1$ en $M = 7$. **Blauw:** Gem. aantal keer *blocking* per tijdseenheid. **Rood:** Gem. aantal keer *starving* per tijdseenheid. **Geel:** Gem. *running*-kosten per tijdseenheid. **Groen:** Gem. aantal producten in het systeem.

5.1.3 Voorbeeld 3: $\lambda = 1$, $\alpha_1 = 1.5$, $\alpha_2 = 0.9$, $\beta = 2$ en $M = 3$

We beschouwen hier het systeem met $\lambda = 1$, $\alpha_1 = 1.5$, $\alpha_2 = 0.9$, $\beta = 2$ en $M = 3$. Beide snelheden voor de eerste server zijn lager dan in voorbeeld 1 en de tweede server werkt altijd op hogere snelheid dan de eerste, er zullen dus minder producten in de buffer opstapelen. Om deze reden is hier ook gekozen voor $M = 3$, deze lagere waarde van de buffercapaciteit zorgt ervoor dat *blocking* nog steeds redelijk vaak optreedt.
De lagere snelheden bij de eerste server en de hogere snelheid bij de tweede zorgen ervoor dat de eerste server de *bottleneck* van het systeem wordt. We bekijken het systeem voor alle mogelijke waarden van k, s, r en de restart snelheid waar mogelijk. De resultaten voor al deze gevallen zijn te vinden in figuren 11 en 12.
Zoals altijd gedragen de *running*-kosten en *starving*-kosten zich vrij constant, de kleine fluctuaties die de twee hebben heffen elkaar zelfs op. Als er namelijk minder *starving*-kosten gemaakt moeten worden, zal de eerste server vaker op hoge snelheid moeten werken wat weer leidt tot hogere *running*-kosten.
De waarden van $E[B]$ varieren ook minder dan in eerdere voorbeelden, ze zijn wel allemaal vrij hoog. Dit wordt voornamelijk veroorzaakt door de lager gekozen waarde van M. Bij de *holding*-kosten is er wel een duidelijk verschil te merken, voornamelijk tussen de gevallen met $k = 1$ en de gevallen met $k = 2$.



Figuur 11: Totale kosten in **Voorbeeld 3**. $c_B = 75$, $c_S = 10$, $c_R = 1$,$c_F = 3$ en $c_L = 1.8$.



Figuur 12: Voorbeeld 3: $\lambda = 1$, $\alpha_1 = 1.5$, $\alpha_2 = 0.9$, $\beta = 2$ en $M = 3$. **Blauw:** Gem. aantal keer *blocking* per tijdseenheid. **Rood:** Gem. aantal keer *starving* per tijdseenheid. **Geel:** Gem. *running*-kosten per tijdseenheid. **Groen:** Gem. aantal producten in het systeem.

5.2 Conclusies

In dit verslag hebben wij onderzocht welk effect de verschillende parameterkeuzes hebben op het beschreven systeem, en welke hiervan de optimale zijn.

Uit vrijwel alle voorbeelden kwam naar voren dat *starving* het minst te beïnvloeden is door de parameterkeuze. Dit komt voornamelijk omdat *starving* grotendeels wordt vastgelegd door de waarden van $\lambda, \alpha_1, \alpha_2$ en β en minder door de keuze van k, s en r. Ook de *running*-kosten varieerden relatief weinig bij verschillende keuzes van de parameters. Zelfs in voorbeeld 2 waar de kosten voor snel en langzaam werken erg ver uit elkaar liggen, zijn de *running*-kosten redelijk constant. Er blijkt bovendien een sterke correlatie in de *starving*- en *running*-kosten te zitten, als de *starving*-kosten stijgen dalen de *running*-kosten en omgekeerd.
Interessanter zijn de gevonden resultaten voor de *blocking*-kosten en de *holding*-kosten. Deze bleken beide in hoge mate te beïnvloeden door de keuze van de parameters, en zijn voor het optimaliseren hiervan dus het meest van belang. Om de blocking kosten zo laag mogelijk te maken moet de eerste machine zo min mogelijk op hoge snelheid werken, en zo lang mogelijk wachten om weer te herstarten. Gebeurd dit niet, dan lopen deze kosten flink op, zoals mooi geïllustreerd wordt in voorbeeld 2.
Hier recht tegenover staan de *holding*-kosten. Om deze zo klein mogelijk te maken moet het systeem zo snel mogelijk werken, dat houdt in dat de eerste server zo kort mogelijk geblockt moet zijn en zo vaak mogelijk op hoge snelheid moet werken. Dit verband is in figuur 7 mooi te zien.

Mogelijkheden voor verder onderzoek liggen voornamelijk in uitbreiding van het systeem. Bijvoorbeeld door zowel de eerste als de tweede server met meerdere mogelijke snelheden te laten werken. Door *slowdown* van de tweede server mogelijk te maken, zou bijvoorbeeld het optreden van *starving* beter kunnen worden voorkomen. Een andere mogelijkheid zou zijn om meer dan twee verschillende snelheden toe te staan, of het verwisselen tussen snelheden niet alleen afhankelijk te maken van het aantal producten in de tweede wachtrij maar ook van het aantal producten in de eerste wachtrij. Ook zou het interessant zijn om te bekijken hoe het systeem werkt met niet-exponentiele tussenaankomst- en bedieningstijden.
Er moet bovendien nog verder onderzoek worden gedaan naar de stabilieitscondities voor systemen die niet aan (1) en (2) voldoen. Als laatste rest te achterhalen waar het fout gaat bij het bepalen van evenwichtskansen in het algemene geval. Hoewel er voldoende onafhankelijke eigenvectoren gevonden worden, is er geen lineaire combinatie die aan de randvoorwaarden voldoet.

6 Referenties

1. R. Bekker en O.J. Boxma. An M/G/1 Queue with Adaptable Service Speed. *Stochastic Models*, 23:373-396, 2007.

2. R. Bekker en O.J. Boxma en J.A.C. Resing. Queues with Service Speed Adaptations. *Statistica Neerlandica*, 62(4):441-457, 2008.

3. A. Brandwajn en Y.L. Jow. An Approximation Method for Tandem Queues with Blocking. *Operations Research*, 36(1):73-83, 1988.

4. D.W. Cheng en D.D. Yao. Tandem Queues with General Blocking: a Unified Model and Comparison Results. *Discrete Event Dynamic Systems: Theory and Applications*, 2:207-234, 1993.

5. J.W. Cohen. On the Optimal Switching Level for an M/G/1 Queueing System. *Stochastic Processes and Their Applications*, 4:297-316, 1976.

6. N.D van Foreest en M.R.H. Mandjes en J.C.W. van Ommeren en W.R.W. Scheinhardt. A Tandem Queue with Server Slow-down and Blocking. *Stochastic Models*, 21:695-724, 2005.

7. W.K. Grassman en S. Drekic. An Analytical Solution for a Tandem Queue with Blocking. *Queueing Systems*, 36:221.235, 2000.

8. A.G. Konheim en M. Reiser. A Queueing Model with Finite Waiting Room and Blocking. *Journal of the Association for Computing Machinery*, 23(2):328-341, 1976.

9. H.C Tijms. On a Switch-Over Policy for Controlling the Workload in a Queueing System with Two Constant Service Rates and Fixed Switch-Over Costs. *Zeitschrift für Operations Research*. 21:19-32, 1977.