دانشگاه صنعتی شریف

دانشکده‌ی مهندسی صنایع

گزارش پروژه‌ی شبیه‌سازی گسسته پیشامد رستوران

استاد درس

دکتر نفیسه صدقی

دستیاران درس

روزبه آذرگشاسبی

حمیدرضا محمدیها

نگارش

حامد علی اکبری

هستی قادرآزاد

بهار- تابستان ۱۳۹۹

فهرست

شرح سیستم شبیه‌سازی

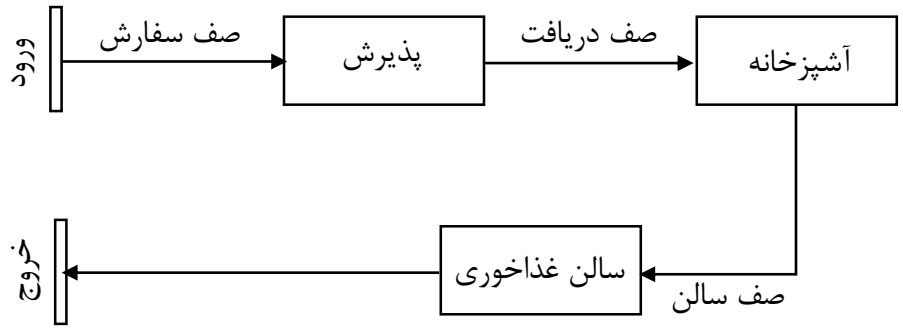
هدف از این مسئله، شبیه‌سازی یک رستوران فست‌فود است. اطلاعات مربوط به صورت زیر ارائه گردیده‌است:

- طول مدت شبیه‌سازی از ساعت ۱۰ صبح تا ۳ بعد از ظهر است. افراد به سه صورت به رستوران می‌رسند: پیاده، به وسیله‌ی خودرو و اتوبوس.
- پیاده‌ها با فاصله‌های زمانی که از توزیع نمایی منفی با میانگین ۳ دقیقه پیروی می‌کند، وارد رستوران می‌شوند. اولین شخص با توزیع مذکور بعد از ساعت ۱۰ وارد رستوران می‌شود.
- تعداد سرنشینان هر خودرو ۱، ۲، ۳ یا ۴ نفر است که احتمال هر یک به ترتیب ۰,۲، ۰,۳، ۰,۳، ۲.۰ است.
- فاصله‌های زمانی بین دو ورود خودرو از توزیع نمایی منفی با میانگین ۵ دقیقه پیروی می‌کند. اولین خودرو در زمان با توزیع مذکور بعد از ساعت ۱۰ وارد رستوران می‌شود.
- یک دستگاه اتوبوس بین ساعت ۱۱ تا ۱۳ ظهر به رستوران می‌رسد که توزیع زمان رسیدن آن به طور یکنواخت است. تعداد سرنشینان این اتوبوس از توزیع پواسون با میانگین ۳۰ نفر تبعیت می‌کند.
- زمانی که شخصی وارد رستوران می‌شود، مستقل از نوع ورودش، به سمت قسمت سفارش غذا حرکت می‌کند. در این بخش مشتری ابتدا غذا را سفارش می‌دهد که مدت زمان این فعالیت به صورت توزیع مثلثی با پارامترهای ۱-۲-۴ است. بلافاصله بعد از سفارش غذا مشتری پول را می‌پردازد که این فرایند نیز از تابع مثلثی با پارامترهای ۱-۲-۳ پیروی می‌کند.
- پس از سفارش غذا، مشتری به قسمت دریافت غذا می‌رود که در این قسمت غذای وی طی زمانی که از توزیع یکنواخت بین ۳۰ ثانیه تا ۲ دقیقه برخوردار است، حاضر می‌شود.
- پس از دریافت غذا، مشتری به سمت سالن غذاخوری حرکت می‌کند. در سالن غذاخوری ۳۰ عدد صندلی وجود دارد. مدت زمان صرف غذا از توزیع مثلثی با پارامترهای ۱۰-۲۰-۳۰ برخوردار است. پس از صرف غذا مشتری به طرف درب خروجی رستوران حرکت می‌کند.

- مدت زمان حرکت بین هر یک از بخش‌ها از توزیع نمایی منفی با میانگین ۳۰ ثانیه برخوردار است. سیستم صف در هر یک از بخشها از قانون فایفو تبعیت می‌کند. بعد از صرف غذا، مشتری با سرعت کمتری که تابع توزیع آن نمایی با میانگین ۱ دقیقه است، رستوران را ترک می‌کند.
- خدمت‌دهندگان دو میز پذیرش و دریافت غذا، دارای چند زمان استراحت هستند که به طور تصادفی بینشان تقسیم می‌شود. به طور دقیق‌تر، در ساعت‌های ۱۰:۵۰، ۱۱:۵۰، ۱۳:۵۰و ۱۴:۵۰ از هر میز یک نفر به استراحت می‌رود و پس از ۱۰ دقیقه برمی‌گردد. اگر خدمت‌دهنده در زمان استراحت در حال کار باشد، پس از اتمام سرویس به استراحت خواهد رفت. تعداد خدمت‌دهنده‌ها در میز پذیرش ۵ و در میز دریافت غذا ۲ است.

مدل‌سازی

مدل‌سازی مساله در قالب شبیه‌سازی به همراه معرفی پیشامدها، فعالیت‌ها و متغیرهای حالت (توصیف ایستا).



توصیفات استاتیک مدل

در این بخش به معرفی اجزای مدل شبیه‌سازی می‌پردازیم.

متغیرهای حالت[1]

- طول صف سفارش (LQo)

این متغیر نشان می‌دهد در زمان t چند نفر در صفی که برای سفارش غذا و پرداخت پول تشکیل شده‌است منتظر هستند. این متغیر با LQo(t) نمایش داده می‌شود.

[1] State variables

- طول صف دریافت غذا (LQr)

این متغیر نشان میدهد در زمان t چند نفر در صفی که برای دریافت غذا در قسمت آشپزخانه تشکیل شده‌است منتظر هستند. این متغیر با LQr(t) نمایش داده می‌شود.

- تعداد صندلی‌های خالی (ES)

این متغیر تعداد صندلی‌های خالی در سالن در زمان t را نشان می‌دهد و به صورت ES(t) نمایش داده می‌شود.

- طول صف انتظار سالن (LQc)

این متغیر نشان می‌دهد در زمان t چند نفر در صفی که برای نشستن بر روی صندلی در سالن تشکیل شده‌است منتظر هستند. این متغیر با LQc(t) نمایش داده می‌شود.

- تعداد کارمند در میز پذیرش در حال استراحت (nRr)

این متغیر یک متغیر ۰ و ۱ است که اگر فردی از پرسنل پذیرش در حال استراحت باشد مقدار ۱ به آن تعلق می‌گیرد و با nRr(t) نمایش داده می‌شود.

- تعداد کارمند در آشپزخانه در حال استراحت (nRk)

این متغیر یک متغیر ۰ و ۱ است که اگر فردی از پرسنل آشپزخانه و تحویل غذا در حال استراحت باشد مقدار ۱ به آن تعلق می‌گیرد و با nRk(t) نمایش داده می‌شود.

- تعداد کارمند پذیرش آزاد (Lr)

این متغیر نشان می‌دهد در زمان t چه تعداد از کارمندان بخش پذیرش آزاد هستند و با Lr(t) نمایش داده می‌شود.

- تعداد کارمند آشپزخانه آزاد (Lk)

این متغیر نشان می‌دهد در زمان t چه تعداد از کارمندان بخش آشپزخانه آزاد هستند و با Lk(t) نمایش داده می‌شود.

- تعداد کارمند پذیرش در انتظار استراحت (WRr)

هنگامی که وقت استراحت یک کارمند پذیرش فرا می‌رسد اما مشغول به کار است، بعد از اتمام کارش باید به استراحت برود. این متغیر تعداد کارمندان پذیرش که در زمان t دچار چنین وضعیتی هستند را نشان می‌دهد و مقدار ۰ یا ۱ می‌گیر و با WRr(t) نشان داده می‌شود.

- تعداد کارمند آشپزخانه در انتظار استراحت (WRk)

هنگامی که وقت استراحت یک کارمند آشپزخانه فرا می‌رسد اما مشغول به کار است، بعد از اتمام کارش باید به استراحت برود. این متغیر تعداد کارمندان آشپزخانه که در زمان t دچار چنین وضعیتی هستند را نشان می‌دهد و مقدار ۰ یا ۱ می‌گیر و با WRk(t) نشان داده می‌شود.

- تعداد مشتریان حاضر در رستوران (LS)

این متغیر تعداد کل مشتری‌های حاضر در بخش‌های مختلف رستوران در زمان t را نشان می‌دهد و با LS(t)

نمایش داده می‌شود.

نهادها[۲]

- مشتریان
- کارمندان بخش پذیرش
- کارمندان بخش آشپزخانه

پیشامدها[۳]

- ورود به رستوران به صورت پیاده و شروع سفارش‌دهی (Aw)
- ورود به رستوران به وسیله ماشین و شروع سفارش‌دهی (Ac)
- ورود به رستوران به وسیله اتوبوس و شروع سفارش‌دهی (Ab)
- اتمام سفارش‌دهی و پرداخت (rE)
- شروع فرایند دریافت غذا (Er)
- دریافت غذا (R)
- شروع صرف غذا (E)
- پایان صرف غذا (En)
- خروج از رستوران (Ex)
- شروع استراحت کارمندان پذیرش (Srr)
- اتمام استراحت کارمند پذیرش (Err)
- شروع استراحت کارمندان آشپرخانه (Srk)
- اتمام استراحت کارمند آشپزخانه (Erk)
- پیشامد اتمام شبیه‌سازی (End)

اعلام پیشامد[۴]

- (Ac,t)
- (Ab,t)
- (Aw,t)
- (rE,t)
- (Er,t)
- (R,t)

Entities[۲]
Events[۳]
Event notice[۴]

- (Erk,t)
- (E,t)
- (Ex,t)
- (En,t)
- (Srr,t)
- (Err,t)
- (Srk,t)
- (End,t)

فعالیت‌ها

- زمان بین ورود مشتری پیاده (WInter)

زمان بین ورود دو مشتری در حالت پیاده را نشان می‌دهد و از توزیع نمایی منفی با میانگین ۳ دقیقه پیروی می‌کند.

- زمان بین ورود مشتری با ماشین (CInter)

زمان بین ورود دو ماشین را نشان می‌دهد و از توزیع نمایی منفی با میانگین ۵ دقیقه پیروی می‌کند.

- زمان بین ورود مشتری با اتوبوس (BInter)

زمان بین ورود دو اتوبوس را نشان می‌دهد و از توزیع یکنواخت پیروی می‌کند.

- سفارش غذا (Ost)

مدت زمانی که سفارش غذا به صورت مستقل طول می‌کشد و از توزیع مثلثی با پارامترهای ۱-۲-۴ پیروی می‌کند.

- پرداخت پول (Pay)

مدت زمانی که فرآیند پرداخت پول به صورت مستقل طول می‌کشد و از توزیع مثلثی با پارامترهای ۱-۲-۳ پیروی می‌کند.

- سرویس‌دهی بخش آشپزخانه_دریافت غذا (Kst)

مدت زمانی که طول می‌کشد غذای افراد حاضر شود که از توزیع یکنواخت پیروی می‌کند.

- مدت زمان صرف غذا (eat)

مدت زمانی که طول می‌کشد مشتریان غذای خود را صرف کنند که از توزیع مثلثی با پارامترهای ۱۰-۲۰-۳۰ پیروی می‌کند.

- حرکت از بخش پذیرش به سمت آشپزخانه (t1)

مدت زمانی که پیمودن مسافت بین بخش پذیرش و آشپزخانه طول می‌کشد که از توزیع نمایی منفی با میانگین ۳۰ ثانیه برخوردار است.

- حرکت از بخش آشپزخانه به سمت سالن غذاخوری (t2)

مدت زمانی که پیمودن مسافت بین آشپزخانه و سالن طول می‌کشد که از توزیع نمایی منفی با میانگین ۳۰ ثانیه برخوردار است.

- حرکت به سمت خروج از رستوران (t3)

مدت زمانی که پیمودن مسافت بین سالن و خروجی طول می‌کشد که از توزیع نمایی با میانگین ۱ دقیقه برخوردار است.

- استراحت کارمند پذیرش (Rr)

۱۰ دقیقه

- استراحت کارمند آشپزخانه (Rk)

۱۰ دقیقه

تأخیر[۵]

- انتظار در صف پذیرش

مدت زمان انتظار مشتری تا خالی شدن سرور پذیرش

- انتظار در صف دریافت غذا

مدت زمان انتظار مشتری تا خالی شدن سرور آشپزخانه

- انتظار در صف سالن غذاخوری

مدت زمان انتظار مشتری تا خالی شدن صندلی

- انتظار برای استراحت کارمند بخش پذیرش

مدت زمان بین ساعتی که کارمند بخش پذیرش به استراحت می‌رود و ساعت استانداردی که برای استراحت در نظر گرفته شده بود. منظور از ساعت استاندارد ۱۰:۵۰، ۱۱:۵۰، ۱۳:۵۰ و ۱۴:۵۰ است.

- انتظار برای استراحت کارمند بخش آشپزخانه

مدت زمان بین ساعتی که کارمند بخش آشپزخانه به استراحت می‌رود و ساعت استانداردی که برای استراحت در نظر گرفته شده بود.

آماره‌های تجمعی

آماره‌های تجمعی

در راستای انجام شبیه‌سازی نیازمند تعریف برخی از آماره‌های تجمعی هستیم که باید از آنها در اجرای فرایند استفاده کنیم. ذکر این نکته ضروری است که تمامی آماره‌های زیر تجمعی هستند و به این همین علت، جهت پیشگیری از تکرار، از آوردن واژه‌ی تجمعی در کنار آنها پرهیز شده‌است. این آماره‌ها عبارت‌اند از:

- طول صف انتظار پذیرش
- طول صف سالن صرف غذا
- طول صف دریافت غذا
- مدت زمان انتظار در صف پذیرش

[۵] Delay

- مدت زمان انتظار در صف دریافت غذا
- مدت زمان انتظار در صف سالن صرف غذا
- مدت زمان انتظار جهت دریافت غذا
- تعداد مشتریان رستوران
- زمان فعالیت کارکنان بخش پذیرش
- زمان فعالیت کارکنان بخش آشپزخانه
- مدت زمان حضور مشتریان در رستوران

ساعت[۶]

زمان شبیه‌سازی را به ما نشان می‌دهد. این زمان از ساعت ۱۰ صبح الی ۳ بعد از ظهر خواهد بود.

ساختاربندی لیست پیشامدهای آتی[۷]

لیست پیشامدهای آتی مجموعه‌ای از اعلام پیشامدها می‌باشد که با استفاده از آنها باید زمان شبیه‌سازی خود را جلو ببریم. در مسئله‌ی مورد بررسی ما به صورت کلی سیزده پیشامد منحصر به فرد وجود دارد که پس از ورود مشتریان و مشغول شدن تمام بخش‌های سیستم به طور معمول دست‌کم سیزده پیشامد تعریف شده را خواهیم داشت. به عبارت دیگر بعد از عبور از زمان سرد سیستم، لیست پیشامدهای آتی ما به طور معمول شامل حداقل سیزده اعلام پیشامد خواهد بود. با توجه به اینکه تعداد کارمندان بخش پذیرش پنج و کارمندان آشپزخانه دو نفر می‌باشد پیشامدهای مربوط به اتمام فعالیت این افراد می‌تواند بیشتر از یک مورد و تکراری باشد و این امر کاملا عادی می‌باشد. به همین دلیل از واژه‌ی دست‌کم سیزده پیشامد استفاده شده‌است. البته شایان ذکر است که در حالت‌هایی استثنایی مانند شروع شبیه‌سازی، اعضای لیست مدنظر ما کمتر خواهد بود.

کمینه مقدار زمان از این لیست انتخاب و زمان شبیه‌سازی به آن منتقل می‌شود. پس از این اقدام اعلام پیشامد مذکور از لیست حذف می‌شود. همچنین با توجه به شرایط خاصی که برای مسئله تعریف شده‌است پس از ورود مشتران به وسیله‌ی اتوبوس دیگر پیشامد تولید نخواهد شد و بنابراین به لیست باز نمی‌گردد.

نکته‌ی دیگر در مورد وجود پیشامدهای مربوط به اتمام استراحت‌ها می‌باشد که در ابتدا باید پیشامد شروع استراحت اتفاق بیافتد تا بتوان برای اتمام آن برنامه‌ریزی کرد.

برای مثال می‌توانیم لیست زیر را به عنوان نمونه‌ای از لیست پیشامدهای آتی معرفی کنیم که در حالت عادی رخ خواهد داد:

$FEL = \{(Ac,t_1), (Ab,t_2), (Aw,t_3), (rE,t_4), (Er,t_5), (R,t_6), (Erk,t_7), (E,t_8), (Ex,t_9), (En,t_{10}), (Srr,t_{11}), (Err,t_{12}), (Srk,t_{13})\}$

---

[۶] Clock

[۷] FEL

در مثال بالا هنوز مشتریان مربوط به اتوبوس به رستوران نرسیده‌اند؛ بنابراین پیشامد ورود آنها همچنان در لیست وجود دارد. همچنین تنها یک پیشامد برای اتمام فرایند سفارش‌دهی و دریافت غذا در نظر گرفته شده‌است که به طور معمول اعلام‌های مربوط به این پیشامدها در هر لحظه بیش از یک مورد خواهد بود اما در حال حاضر برای سهولت در خوانایی متن گزارش به معرفی یک مورد بسنده شده‌است.

در نهایت زمان شبیه‌سازی با توجه به زمان‌های لیست مثال فوق پیش خواهد رفت. حال لیست پیشامدهای آتی را برای لحظه شروع سیستم و صفر زمان به عنوان مثال ذکر می‌کنیم:

FEL = {(Ac,۱۰:۰۵), (Ab,۱۲:۰۰), (Aw,۱۰:۰۳), (Srr,۱۰:۵۰), (Srk,۱۰:۵۰)}

همانطور که مشاهده می‌شود در حالت اولیه سیستم تنها پنج پیشامد وجود دارد که زمان شبیه‌سازی و حالت سیستم بر اساس کمینه مقدار زمان موجود در این لیست تغییر خواهد کرد.

فرضیات و ساده‌سازی‌ها

۱) بین ورودی رستوران و بخش پذیرش فاصله ناچیز است. به همین دلیل فرض می‌شود شروع سفارش و ورود به رستوران یکی است.
۲) عملیات سفارش‌دهی و پرداخت پول هر دو در بخش پذیرش انجام می‌شوند. در واقع سروری که سفارش را ثبت می‌کند به پرداخت نیز رسیدگی می‌کند و بین سفارش و پرداخت تغییری در سیستم رخ نمی‌دهد.
۳) سرورهای پذیرش یکسان در نظر گرفته شده‌اند و تفاوتی با هم ندارند. به همین دلیل اولویتی در استراحت وجود ندارد.
۴) کارمندان آشپزخانه یکسان در نظر گرفته شده‌اند و تفاوتی با هم ندارند. به همین دلیل اولویتی در استراحت وجود ندارد.
۵) منظور از آشپرخانه همان بخش دریافت غذا می‌باشد.

معیارهای ارزیابی عملکرد سیستم

معیارهای ارزیابی عمکلرد سیستم از مهمترین مواردی است که مدیران باید همواره به آن توجه داشته باشند. محاسبه برخی از این معیارها ساده و برخی دیگر نیازمند تلاش بیشتر است، به این دلیل که به دست آوردن اطلاعات لازم برای آنها دشوار است. در این قسمت از پروژه ما ابتدا سه معیار را به عنوان شاخص‌های مرتبه اول معرفی می‌کنیم که در محاسبه شش معیار مهم مرتبه دوم ارزیابی عملکرد سیستم از آنها استفاده خواهیم کرد. در پایان نیز شاخص‌های منتخب جهت ازریابی سیستم شبیه‌سازی شده را گزارش خواهیم داد. لازم به ذکر است که کلیه‌ی میانگین‌ها به ازای تمامی مشتریان در زمان شبیه‌سازی محاسبه خواهد شد.

معیارهای ارزیابی عملکرد مرتبه اول

- زمان خدمت‌دهی[۸]

اولین مورد زمان خدمت‌دهی می‌باشد که به خودی خود می‌تواند بینش مناسبی از سیستم را برای ما فراهم آورد. این شاخص به صورت زیر محاسبه می‌شود:

زمان خدمت‌دهی = (میانگین مدت زمان انتظار در صف سفارش + میانگین زمان سفارش غذا) + (میانگین مدت زمان انتظار در صف دریافت غذا + میانگین زمان دریافت غذا)

برای محاسبه‌ی قسمت اول این شاخص نیازمند استفاده از فلوچارت‌های «پیشامد ورود به صورت پیاده»، «پیشامد ورود به وسیله‌ی ماشین» و «پیشامد ورود به وسیله‌ی اتوبوس» هستیم. در این فلوچارت‌ها زمان فعالیت سفارش غذا تولید می‌شود و باید از داده‌های آن استفاده کنیم و همچنین باتوجه به تأخیری که مشتریان در صف دریافت غذا متحمل می‌شوند میانگین مدت زمان انتظار آنها را نیز می‌توانیم محاسبه کنیم. البته شایان ذکر است که برای محاسبه‌ی زمان انتظار به فلوچارت «پیشامد اتمام سفارش‌دهی»، بخش کاهش طول صف هم نیازخواهیم داشت که جزئیات اجرای آن متناسب با شرایط کد برنامه‌نویسی خواهد بود.

برای محاسبه‌ی قسمت دوم نیازمند استفاده از فلوچارت «پیشامد شروع فرایند دریافت غذا» به این دلیل که زمان فعالیت فرایند دریافت غذا در آن تولید می‌شود. همچنین برای محاسبه‌ی میانگین مدت زمان انتظار در صف دریافت غذا نیازمند به استفاده از فلوچارت «پیشامد اتمام دریافت غذا» نیز هستیم به این دلیل که صف در آنجا تغییر حالت می‌دهد.

- تعداد کل مشتریان سیستم در بازه‌ی مورد نظر

این شاخص می‌تواند دید مناسبی از لحاظ کارایی سیستم و شرایط بازاریابی را به ما بدهد. برای محاسبه این شاخص به صورت زیر عمل می‌کنیم.

تعداد کل مشتریان سیستم در بازه‌ی مورد نظر = تعداد کل مشتریانی که از رستوران در بازه‌ی مورد نظر خدمت‌گرفته و خارج شده‌اند

با توجه به شرایط سیستم که مشتری ناراضی در آن وجود ندارد، برای محاسبه‌ی این شاخص تنها از فلوچارت «پیشامد خروج از رستوران» استفاده می‌کنیم. البته لازم به ذکر است که برای محاسبه‌ی این شاخص باید در

---

[۸] Service Time

کدنویسی از یک متغیر کمکی استفاده کنیم چون تعریف متغیر حالت کمکی به حل بهتر مسئله‌ی شبیه‌سازی نمی‌کند.

- میانگین مدت زمان حضور مشتری در رستوران

این شاخص به‌طور پیش‌فرض دید مناسبی در مورد مشتریان ما و عملکرد سیستم را به ما می‌دهد. برای محاسبه این معیار به صورت زیر عمل می‌کنیم:

میانگین مدت زمان حضور مشتری در رستوران = زمان خدمت‌دهی + میانگین زمان حرکت بین بخش‌های رستوران + میانگین زمان صرف غذا + میانگین زمان انتظار در صف سالن صرف غذا

نحوه‌ی محاسبه زمان خدمت‌دهی که پیش‌تر به طور کامل ذکر شد. برای محاسبه‌ی زمان حرکت بین بخش‌های رستوران از فلوچارت‌های «پیشامد اتمام صرف غذا»، «پیشامد اتمام سفارش‌دهی» و «پیشامد اتمام دریافت غذا» استفاده می‌کنیم به این دلیل که زمان فعالیت‌های مدنظر ما در دید رو به جلوی آنها محاسبه شده‌است و تنها باید در یک متغیر کمکی ذخیره شود. برای محاسبه‌ی میانگین زمان صرف غذا و میانگین زمان انتظار در صف سالن در فلوچارت‌های «پیشامد شروع صرف غذا» و «پیشامد اتمام صرف غذا» استفاده می‌کنیم. زمان فعالیت مورد نظر و شرایط حالت صف در این دو فلوچارت تولید و تغییر می‌کند.

حال با استفاده از شاخص‌های مرتبه اول ذکر شده به معرفی شش معیار اصلی ارزیابی عملکرد سیستم تحت عنوان شاخص‌های مرتبه دوم می‌پردازیم.

شاخص‌های مرتبه‌ی دوم

- نسبت‌های پذیرش

$$\text{نسبت اول} = \frac{\text{میانگین زمان سفارش غذا}}{\text{زمان خدمت دهی}}$$

$$\text{نسبت دوم} = \frac{\text{میانگین مدت زمان انتظار در صف سفارش}}{\text{زمان خدمت دهی}}$$

محاسبه‌ی اجزای این شاخص همگی در شاخص‌های مرتبه اول انجام شده‌است. بازه‌ی این نسبت‌ها بین صفر و یک می‌باشد. هرچه این مقدار بزرگتر باشد سیگنالی ایجاد می‌کند که زمان فرایند سفارش‌دهی زیاد است و باید اصلاحاتی صورت گیرد. به همین دلیل مطلوب ما این است که تا حد ممکن این شاخص‌ها را کمینه کنیم. طبیعی

است که مقدار آن تا حدی می‌تواند کمتر شود و در حالت بهینه‌ی سیستم بازهم مقداری قابل توجه را خواهد داشت.

- نسبت‌های دریافت

$$\text{نسبت اول} = \frac{\text{میانگین زمان دریافت غذا}}{\text{زمان خدمت دهی}}$$

$$\text{نسبت دوم} = \frac{\text{میانگین مدت زمان انتظار در صف دریافت غذا}}{\text{زمان خدمت دهی}}$$

این دسته از معیارها نیز بسیار مشابه دسته‌ی قبلی می‌باشد، با این تفاوت که در معیارهای به ارزیابی عملکرد بخش آشپزخانه می‌پردازیم. هرچه این نسبت‌ها بزرگتر باشد می‌تواند این مفهوم را برساند که فعالیت بخش آشپزخانه در حالت بهینه قرار ندارد و بهتر است که شرایط کاری آنها مورد ارزیابی بیشتر قرار گیرد. عناصر این شاخص نیز در معیارهای مرتبه اول همگی محاسبه شده‌است.

- نسبت زمان خدمت‌دهی به زمان کل

این شاخص به صورت کلی بیانگر این امر است که چه بخشی از زمان حضور مشتری در رستوران صرف فعالیت‌های مربوط به سفارش‌دهی و دریافت غذا شده‌است. هرچه این مقدار کمتر باشد به این مفهوم است که بخش زیادی از زمان حضور مشتری در رستوران به صرف غذا و استراحت پرداخته شده‌است و نه فعالیت‌های آزاردهنده‌ای مانند انتظار، بنابراین برای ما بهتر است. این شاخص به صورت زیر به دست می‌آید که محاسبه عناصر آن پیش‌تر ذکر شده‌است.

$$\frac{\text{زمان خدمت‌دهی}}{\text{میانگین مدت زمان حضور مشتری در رستوران}}$$

- نسبت گردش[۹]

[۹] Turn over

نسبت‌های گردش همواره یکی از مهمترین شاخص‌های هوشمند ارزیابی عملکلرد سیستم می‌باشند. کوچکتر بودن این معیار بیانگر سرعت و عملکرد مناسب سیستم است و بزرگتر بودن آن از مقدارهای استاندارد بیانگر سکون و لختی سیستم است. این شاخص را به صورت زیر تعریف می‌کنیم:

$$\frac{\text{زمان کل شبیه‌سازی}}{\text{تعداد کل مشتریان سیستم در بازه‌ی مورد نظر}}$$

برای رستوران خود ابتدا باید استانداردی را تعریف کنیم و سپس این شاخص را با آن بسنجیم. اگر نتیجه از مقدار تعریف بزرگتر بود یعنی مشتریان زمان زیادی را در سیستم سپری می‌کنند و می‌تواند به شرایط بازاریابی ما آسیب بزند. اگر که این شاخص از استاندارد کوچکتر باشد به معنای عدم رضایت مشتری و خروج سریع آنها می‌باشد. بنابراین باید تلاش کنیم تا حول استاندارد عمل کنیم. زمان کل شبیه‌سازی ورودی مسئله می‌باشد و نحوه‌ی محاسبه‌ی مخرج کسر در شاخص‌های مرتبه‌ی اول ذکر شده‌است.

- نسبت تعداد مشتریان به کارمندان

این معیار یکی از بهترین شاخص‌های ازریابی عملکرد کارمندان می‌باشد. به طور کلی هر صنعت و حرفه‌ای استانداردی برای تعداد کارمندان به ازای تعداد مشتریان دارد و ما هم برای بررسی رستوران خود باید این مورد را در نظر بگیریم. محاسبه‌ی این شاخص به صورت زیر می‌باشد:

$$\frac{\text{تعداد کل مشتریان سیستم در بازه‌ی مورد نظر}}{\text{تعداد کارمندان رستوران}}$$

برای بررسی این شاخص نیز نیازمند استاندارد تعریف‌شده هستیم. اگر نتیجه از استاندارد بزرگتر باشد یعنی تعداد کارمندان ما کم است و در خدمت‌دهی عملکرد مناسبی نداریم. کوچکتر بودن هم به این معنی است که تعداد کارمندان ما زیاد و بهره‌وری سیستم پایین می‌باشد. بنابراین باید در ابتدا استانداردی برای این شاخص تعریف کنیم و در ادامه تلاش کنیم که از این مقدار استاندارد انحرافی را نداشته باشیم. مخرج کسر جزء اطلاعات ورودی مسئله‌ی شبیه‌سازی و صورت کسر جزء شاخص‌های مرتبه‌ی اول می‌باشد.

- میانگین طول صف سالن غذاخوری در حالت پر بودن سالن

این معیار شاخص بسیار مناسبی برای براورد کمبود صندلی و ظرفیت سالن در زمان اوج[1] حضور مشتریان است. برای محاسبه‌ی این شاخص باید از فلوچارت‌های «شروع فرایند صرف غذا» و «اتمام فرایند صرف غذا» استفاده کنیم. البته لازم به ذکر است که چالش محاسبه‌ی این شاخص در کدنویسی آن به علت شرطی بودن حالت می‌باشد.

بزرگ بودن این معیار می‌تواند زنگ خطری برای رضایت و تجربه مشتریان باشد و باید سریعاً در راستای بهبود آن اقدامی را انجام دهیم.

به صورت خلاصه، معیارهای پیشنهادی موارد زیر هستند:

- نسبت خدمت مفید در پذیرش $= \frac{\text{میانگین زمان سفارش غذا}}{\text{زمان خدمت دهی}}$
- نسبت انتظار در پذیرش $= \frac{\text{میانگین مدت زمان انتظار در صف سفارش}}{\text{زمان خدمت دهی}}$
- نسبت خدمت مفید در آشپزخانه $= \frac{\text{میانگین زمان دریافت غذا}}{\text{زمان خدمت دهی}}$
- نسبت انتظار در آشپزخانه $= \frac{\text{میانگین مدت زمان انتظار در صف دریافت غذا}}{\text{زمان خدمت دهی}}$
- کل زمان به خدمت‌دهی زمان نسبت $= \frac{\text{زمان خدمت‌دهی}}{\text{میانگین مدت زمان حضور مشتری در رستوران}}$
- نسبت گردش $= \frac{\text{زمان کل شبیه‌سازی}}{\text{تعداد کل مشتریان سیستم در بازه‌ی مورد نظر}}$
- نسبت تعداد مشتریان به کارمندان $= \frac{\text{تعداد کل مشتریان سیستم در بازه‌ی مورد نظر}}{\text{تعداد کارمندان رستوران}}$

شاخص‌های منتخب جهت ارزیابی سیستم شبیه‌سازی شده

با توجه به توضیحاتی بخش‌های قبلی و اخذ نظر متخصصین، پنج شاخص زیر جهت ارزیابی سیستم شبیه‌سازی شده استفاده خواهد شد:

- میانگین مدت زمان حضور مشتری در سیستم
- میانگین مدت انتظار مشتری جهت دریافت غذا
- حداکثر و میانگین طول صف انتظار صرف غذا

[1] Peak

- میانگین بهره‌وری کارکنان قسمت پذیرش و دریافت غذا
- نسبت گردش

ذکر این نکته ضروری است که مقدار مبنا برای نسبت گردش با بررسی‌های انجام شده در این سیستم عدد ۱ تعریف می‌شود که توضیحات نحوه‌ی استفاده از این شاخص پیش‌تر گفته شده‌است.

توصیف پویا

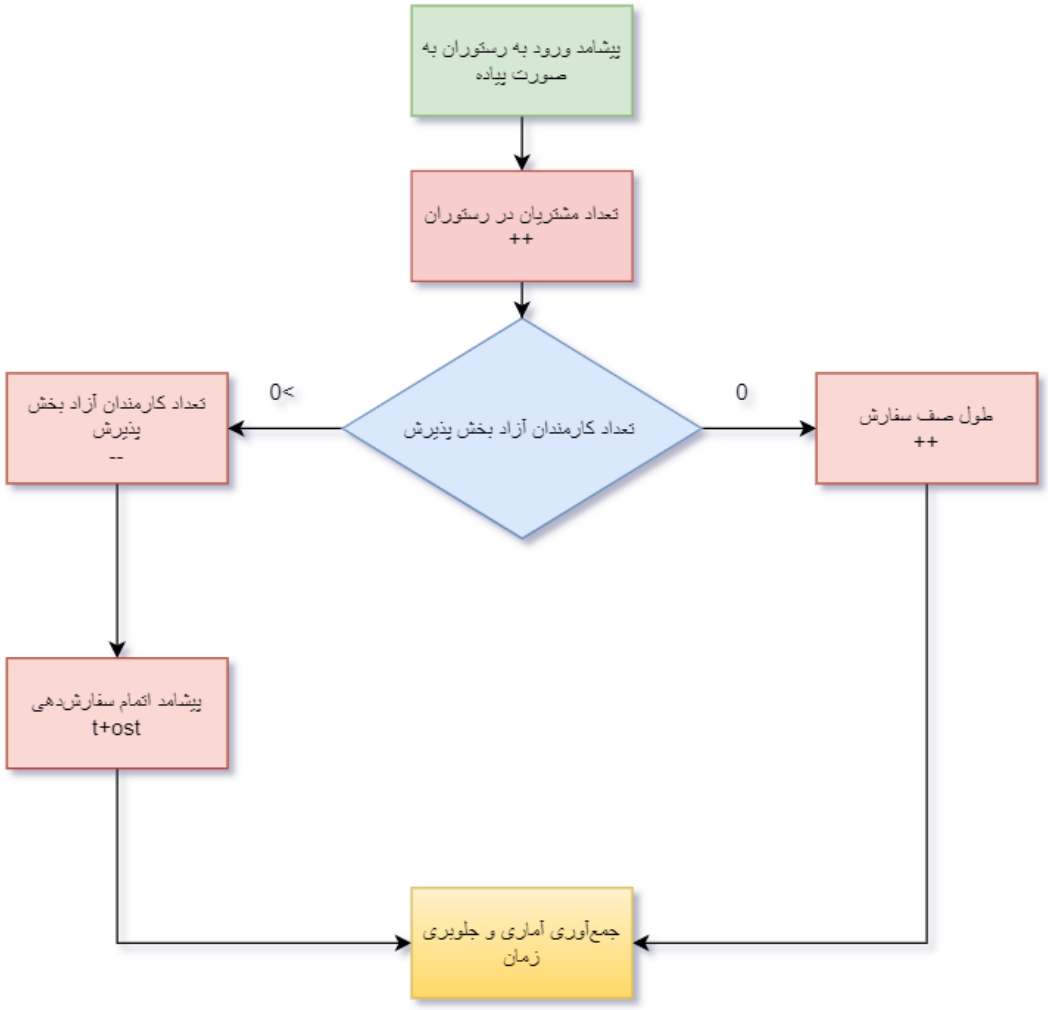
در این قسمت به بررسی فلوچارت‌های پیشامدهای موجود می‌پردازیم. در واقع قصد داریم به پاسخ سوالات زیر برسیم:

۱) چگونه هر پیشامد بر حالت سیستم، ویژگی‌های نهاد و محتوای مجموعه تاثیر می‌گذارد؟
۲) فعالیت‌ها چگونه تعریف می‌شود؟ قطعی هستند یا احتمالی؟کدام پیشامد معرف شروع یا پایان هر فعالیت است؟ آیا فعالیت می‌تواند صرف نظر از حالت سیستم شروع شود یا شروع آن مشروع به بودن سیستم در حالت خاصی است؟
۳) کدام پیشامدها آغاز و پایان هر نوع تاخیر را سبب می‌شود؟ هر تاخیر در کدام شرایط شروع یا تمام می‌شود؟
۴) حالت سیستم در زمان صفر چیست؟ در زمان صفر چه پیشامدهایی برای راه‌اندازی شبیه‌سازی باید تولید شود؟

در ابتدا باید اشاره کنیم که فرض ما در حالت صفر بر این است که هیچ مشتری در رستوران وجود ندارد. همچنین تمامی کارمندان رستوران نیز در حالت آزاد قرار دارند و مشغول نیستند. اولین پیشامد سیستم هم به طور طبیعی ورود یک مشتری خواهد بود که باعث راه‌اندازی سیستم (رستوران) خواهد شد.

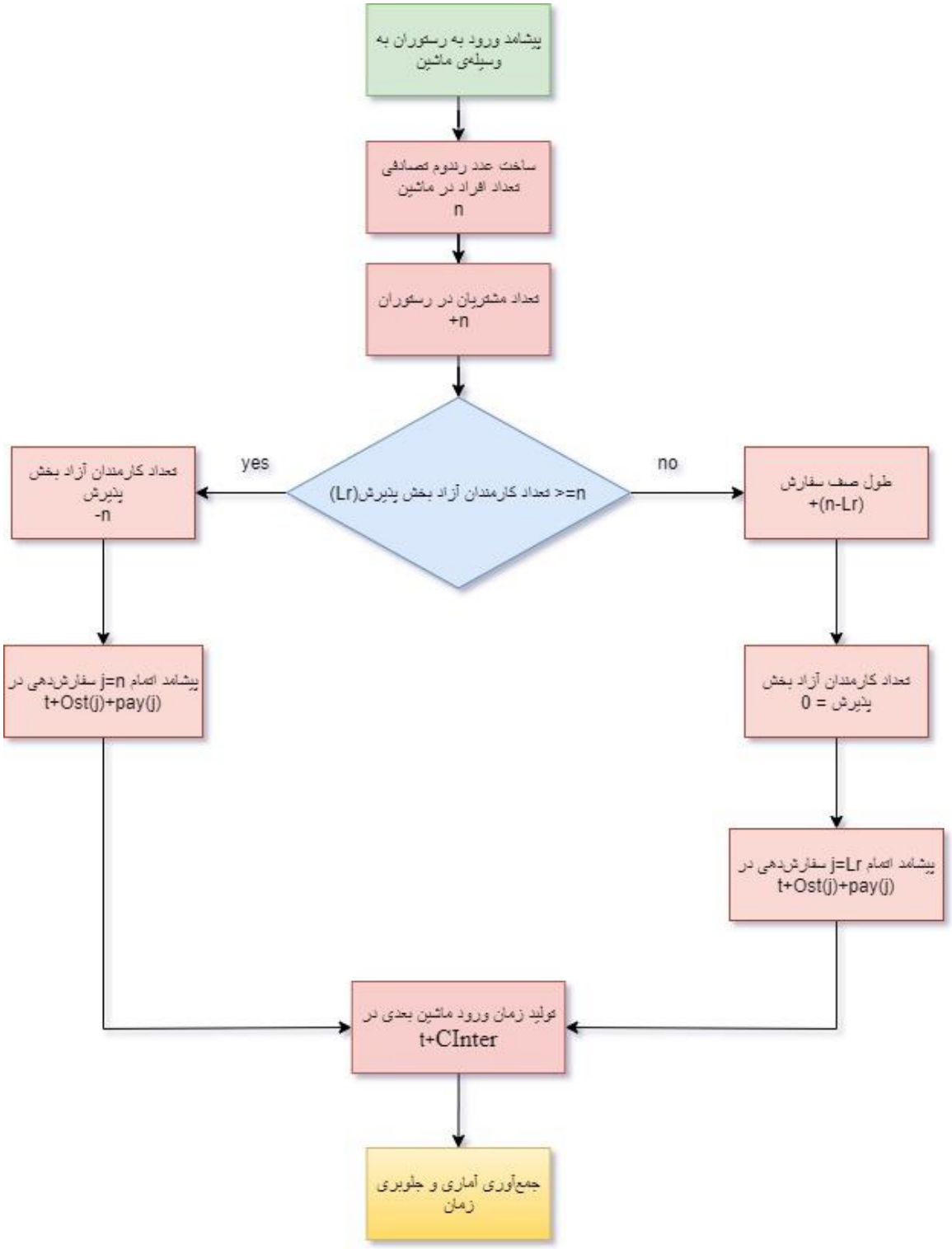
حال به معرفی نمودار جریان مربوط به هر پیشامد می‌پردازیم.

- پیشامد ورود به رستوران به صورت پیاده



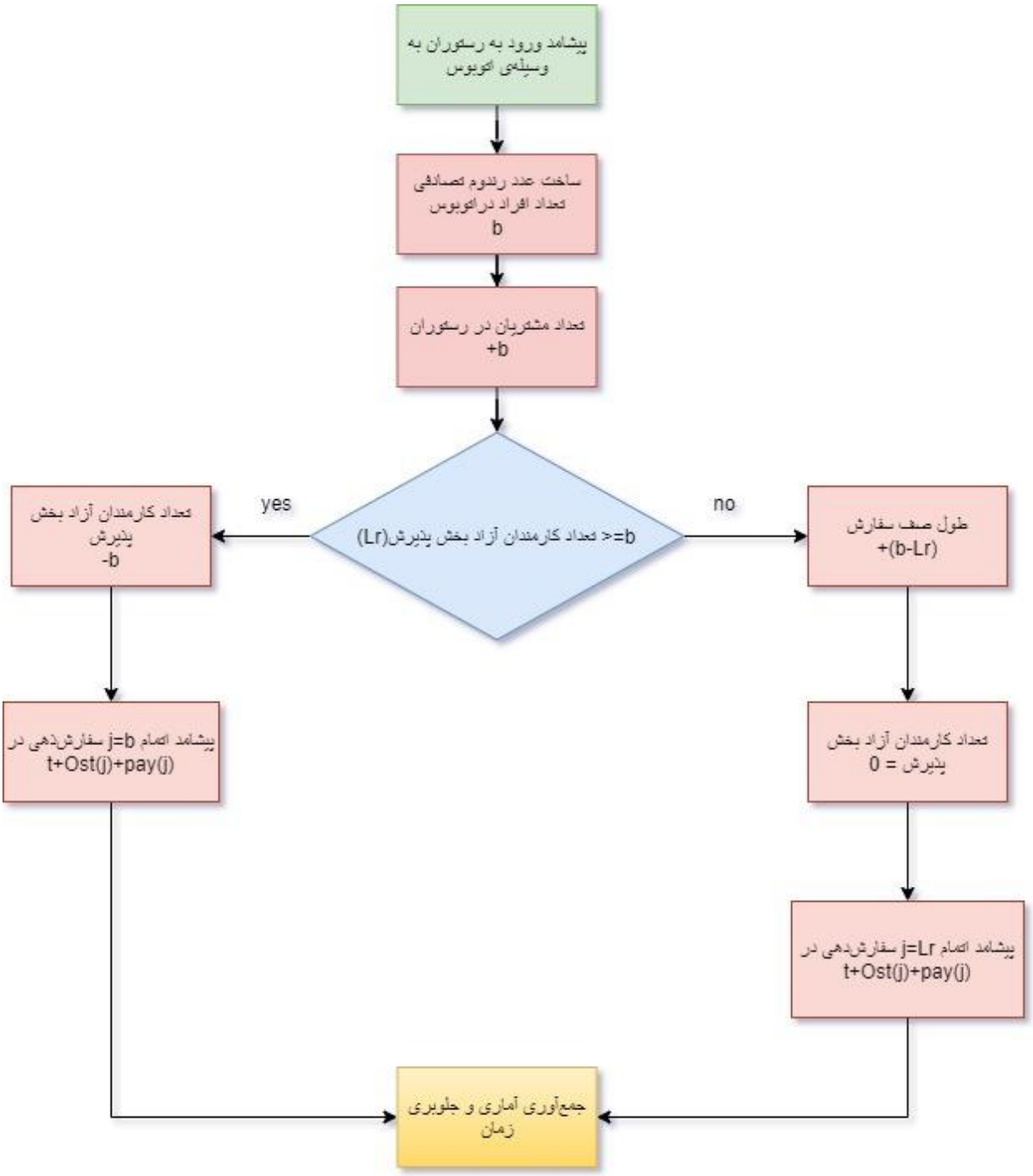
نمودار ۱ - پیشامد ورود به رستوران به صورت پیاده

- پیشامد ورود به رستوران به وسیله‌ی ماشین



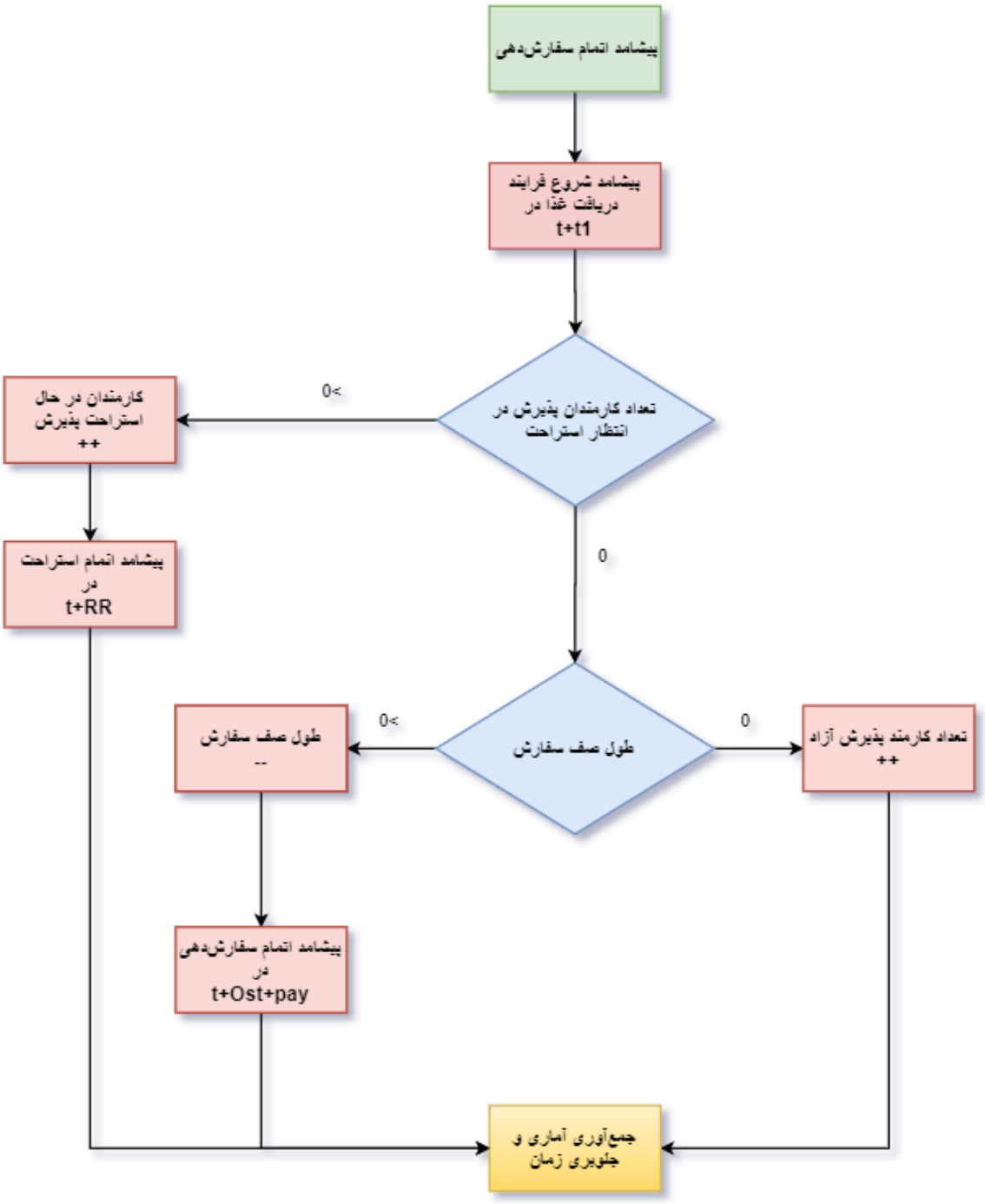
نمودار ۲ - پیشامد ورود به رستوران با ماشین

- پیشامد ورود به رستوران به وسیله‌ی اتوبوس



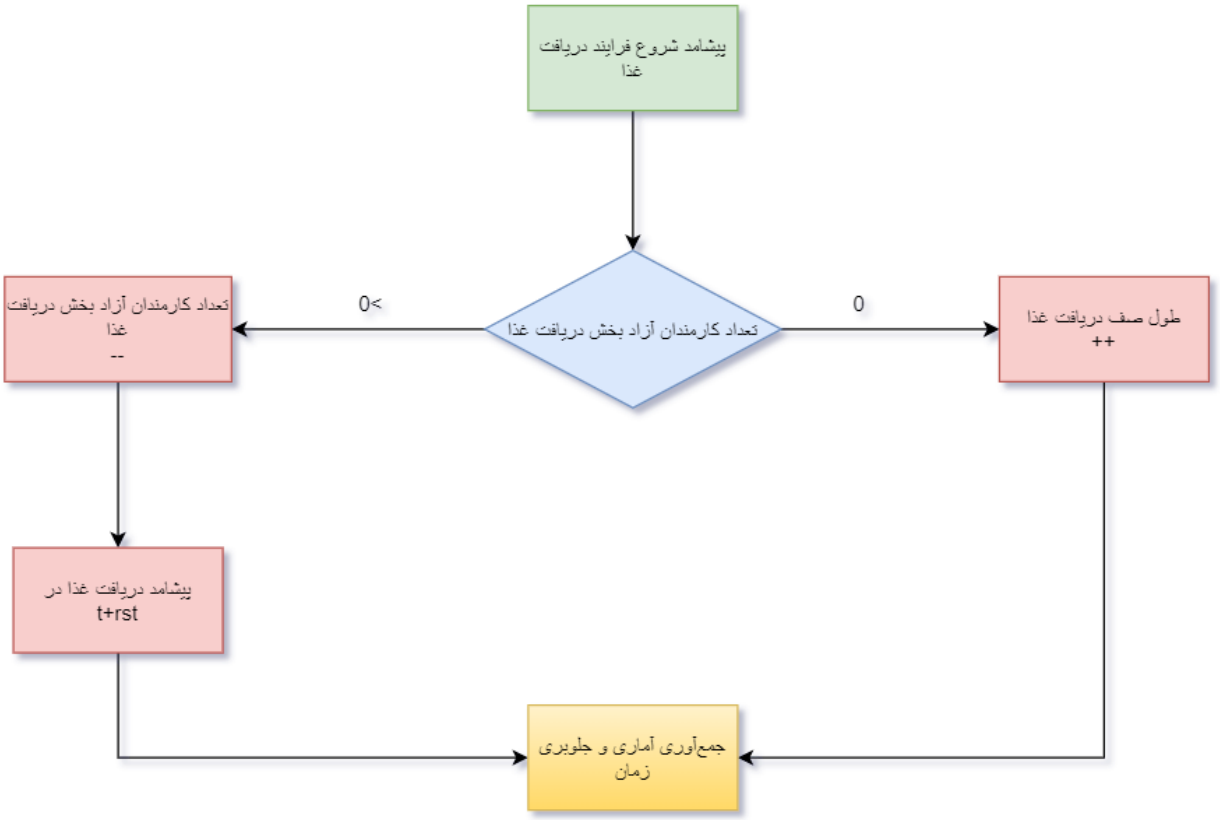
نمودار ۳ - پیشامد ورود به رستوران با اتوبوس

- پیشامد اتمام سفارش‌دهی



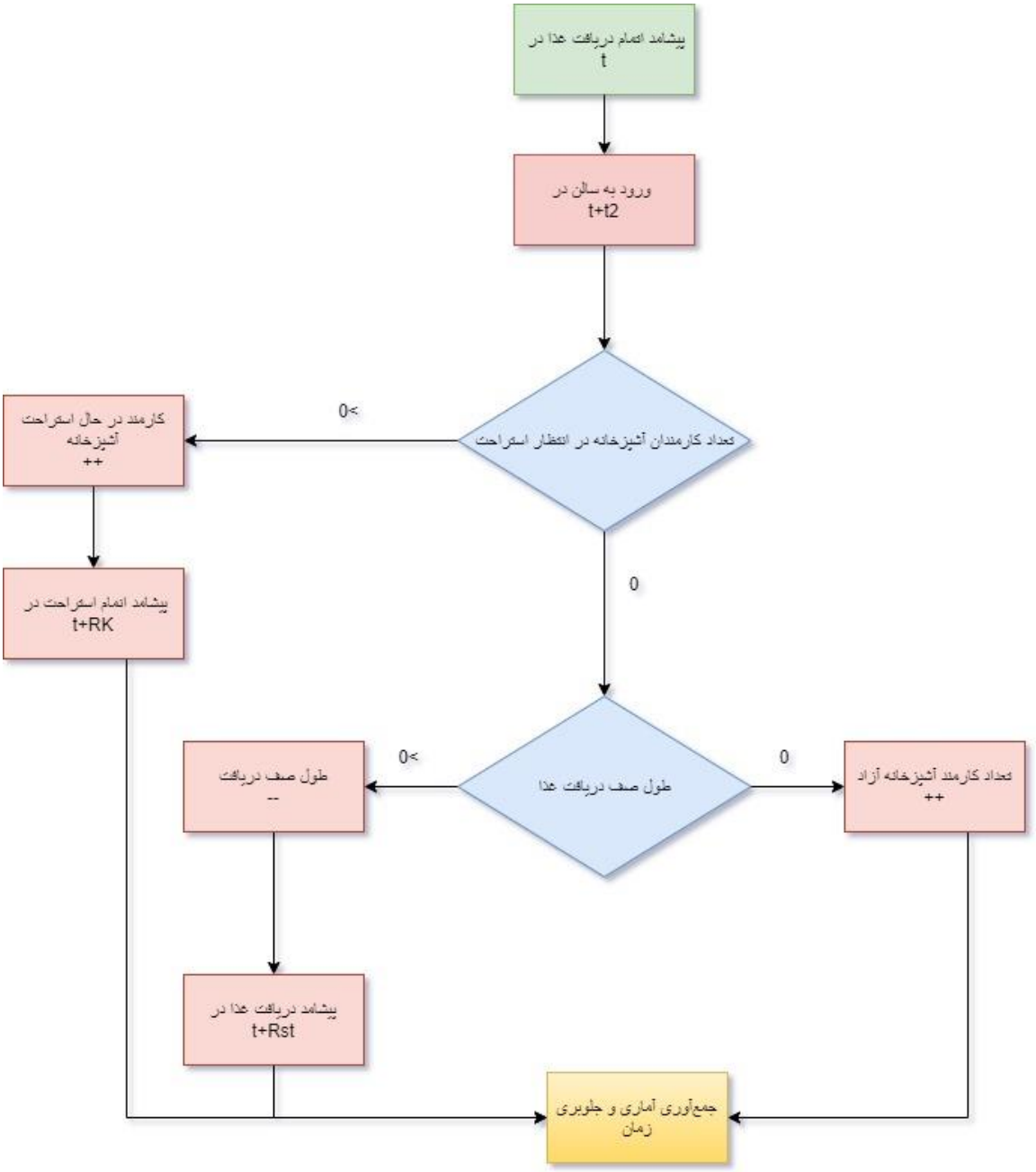
نمودار ۴ - پیشامد اتمام سفارش‌دهی

- پیشامد شروع فرایند دریافت غذا



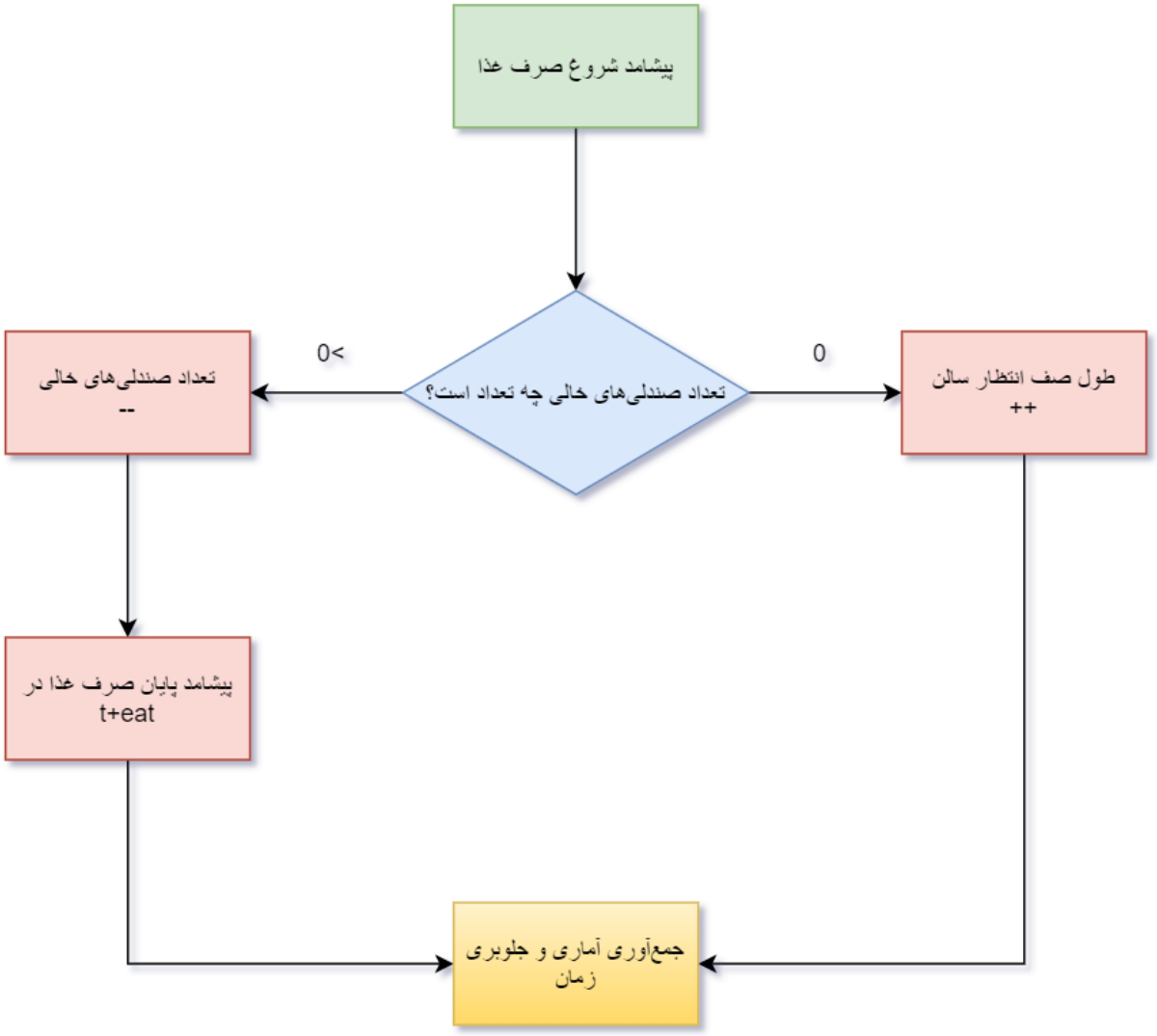
نمودار ۵ - پیشامد شروع فرآیند دریافت غذا

- پیشامد اتمام دریافت غذا



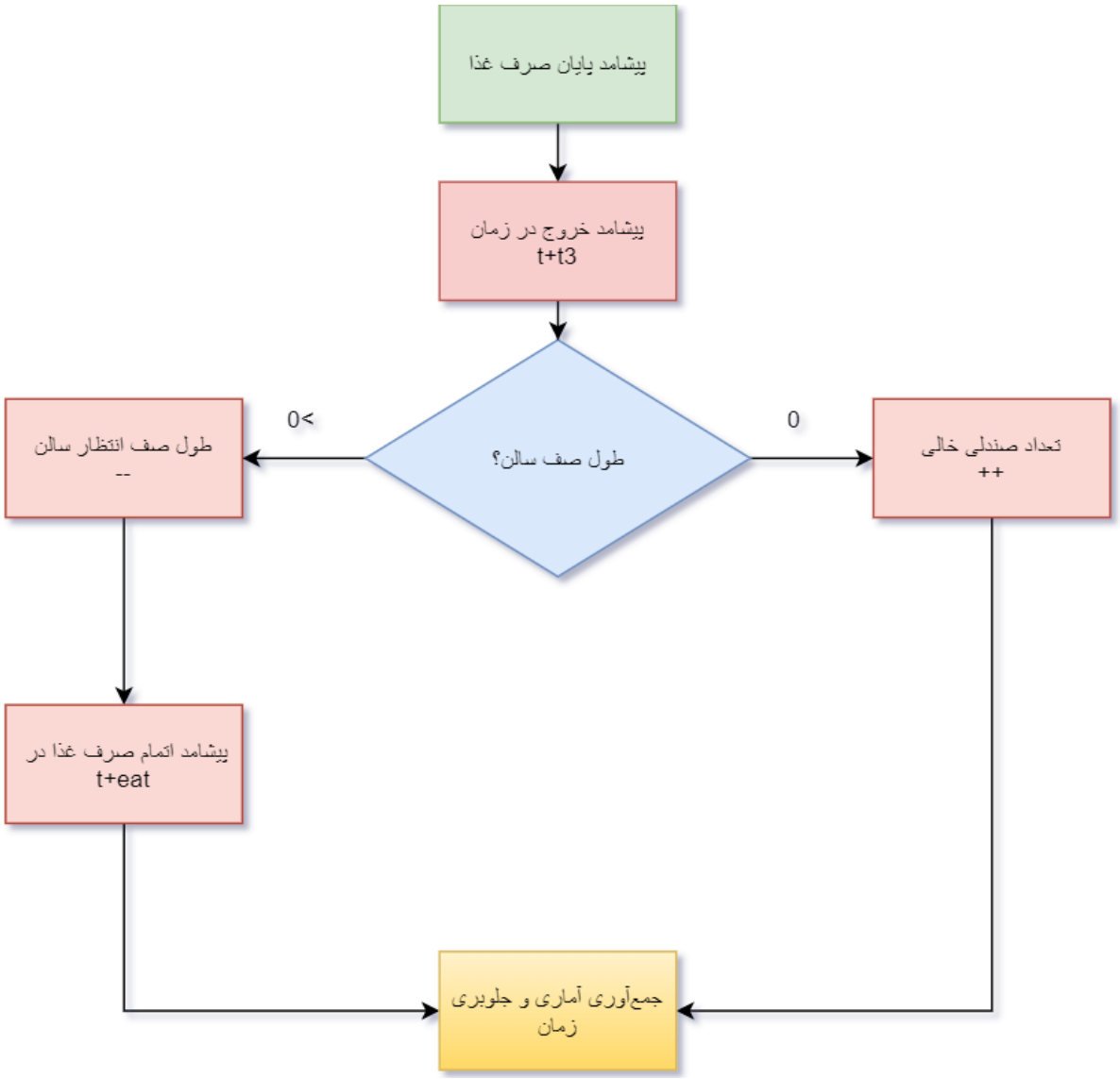
نمودار ۶ - پیشامد اتمام دریافت غذا

- پیشامد شروع صرف غذا



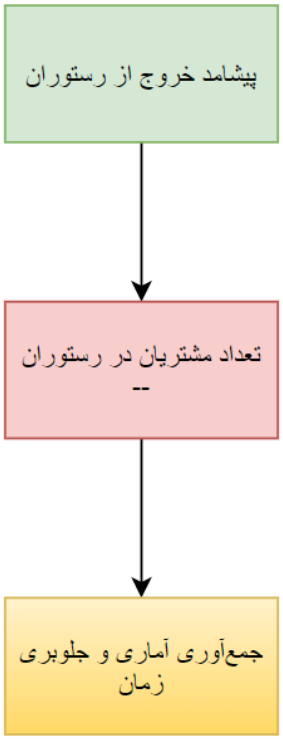
نمودار ۷ - پیشامد شروع صرف غذ

- پیشامد پایان صرف غذا



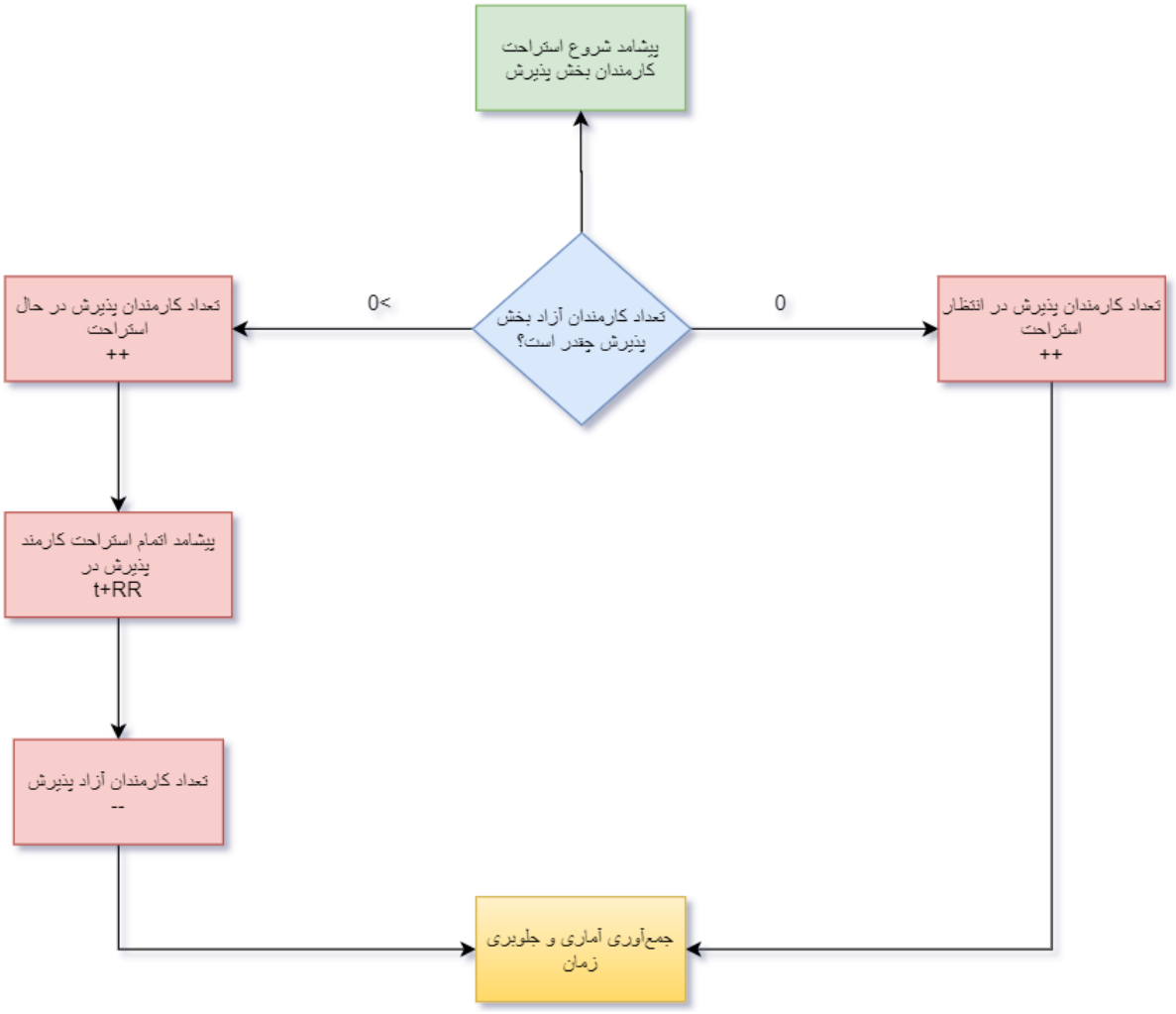
نمودار ۸ - پیشامد پایان صرف غذا

- پیشامد خروج از رستوران



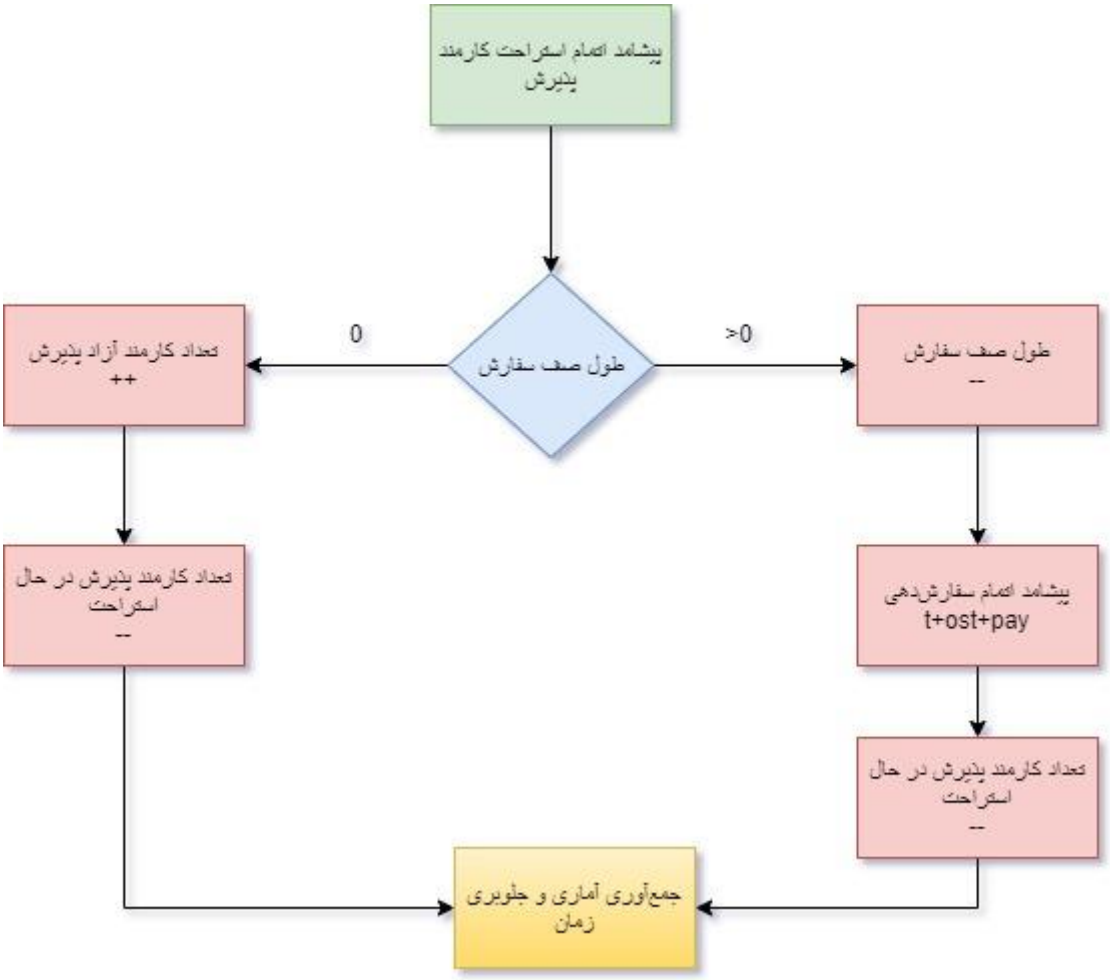
نمودار ۹ - پیشامد خروج از رستوران

- پیشامد شروع استراحت کارمندان بخش پذیرش



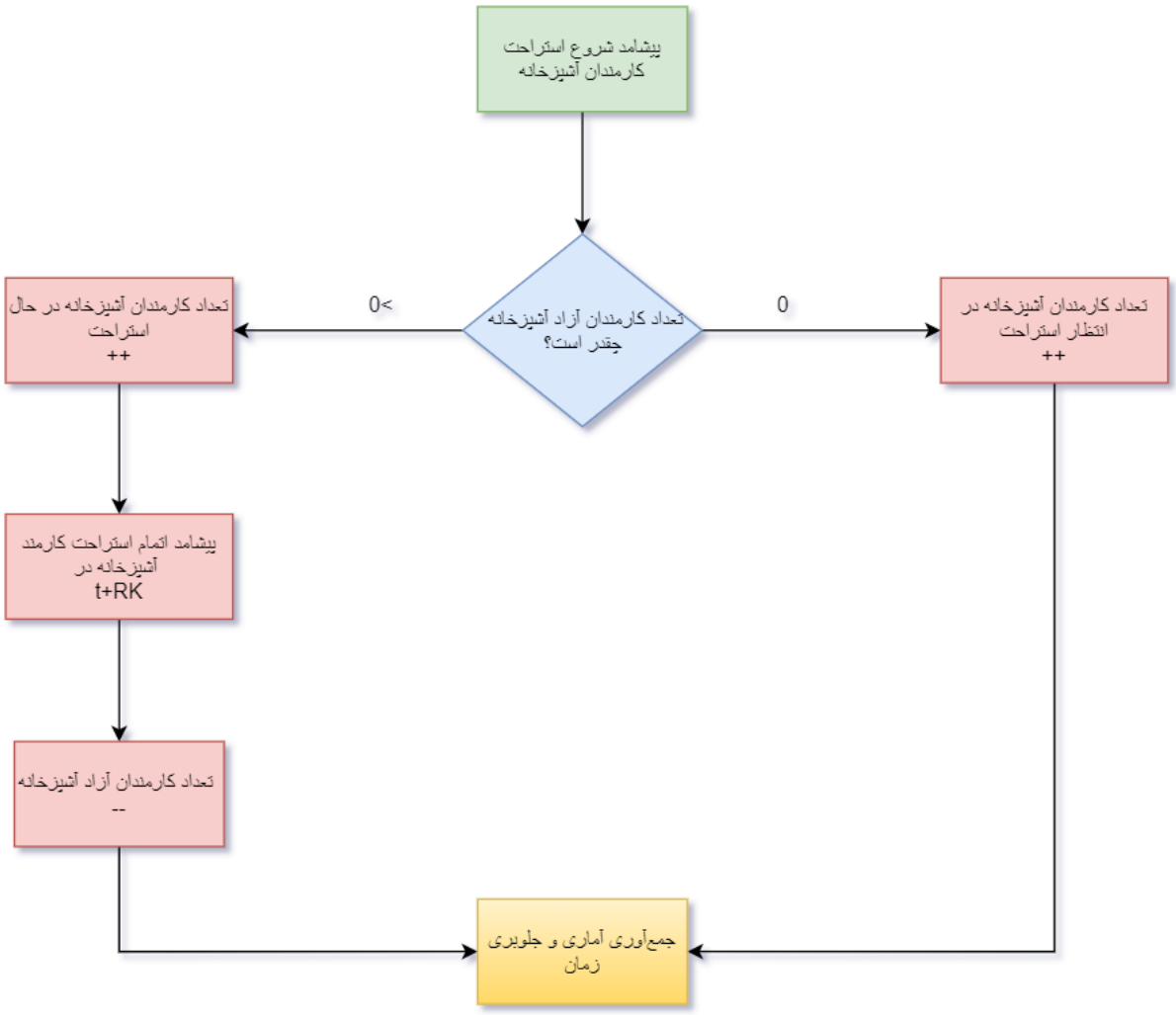
نمودار ۱۰ - پیشامد شروع استراحت کارمندان بخش پذیرش

- پیشامد اتمام استراحت کارمند پذیرش



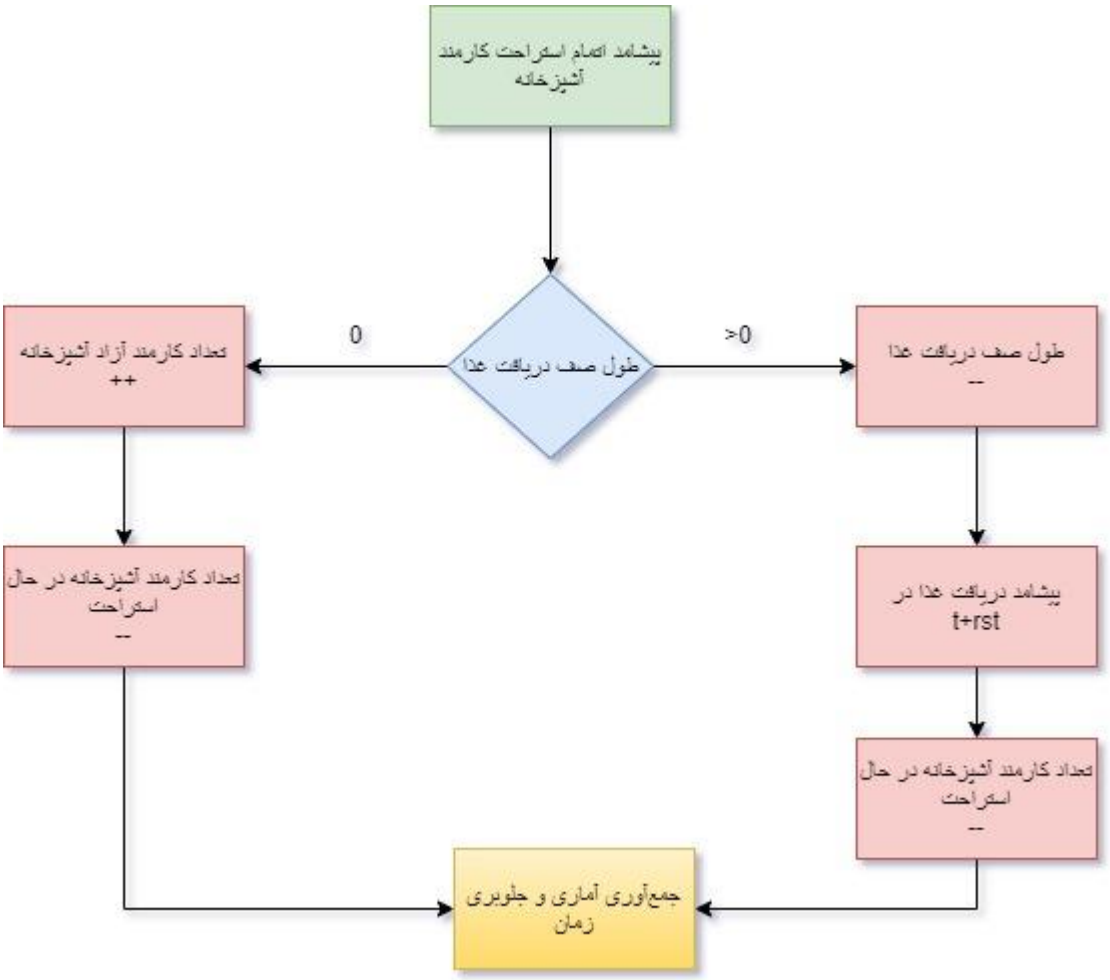
نمودار ۱۱ - پیشامد اتمام استراحت کارمند پذیرش

- پیشامد شروع استراحت کارمندان آشپزخانه



نمودار ۱۲ - پیشامد شروع استراحت کارمندان آشپزخانه

- پیشامد اتمام استراحت کارمند آشپزخانه



نمودار ۱۳ - پیشامد اتمام استراحت کارمند آشپزخانه

شبیه‌سازی گسسته پیشامد رستوران

در این بخش از اجرای شبیه‌سازی به کمک بستر کدزنی پایتون سیستم رستوران خود را مطابق با نمودارهای بخش قبل شبیه‌سازی و خروجی‌های آن را شناسایی کردیم. در انتهای زمان شبیه‌سازی دو رویکرد وجود دارد که می‌توان آن را اتخاذ کرد. رویکرد اول این است که در زمان پایان خروجی‌های شبیه‌سازی سیستم را در لحظه محاسبه کنیم و فرایندهای ناقص را در نظر نگیریم. رویکرد دوم نیز این است که پس از پایان زمان مدنظر ورودی‌های سیستم قطع شود و زمان تا اتمام تمامی فرایندها ادامه داشته باشد. بررسی‌های ما نشان داد که این دو خروجی تفاوت معنی‌داری با یک دیگر ندارند و با وجود اینکه کد طراحی شده هر دو خروجی را محاسبه می‌کند، جهت پیشگیری از اطاله‌ی کلام تنها خروجی‌های مربوط به رویکرد اول را در این سند مورد بررسی قرار

می‌دهیم. لازم به ذکر است که کد شبیه‌سازی سیستم و بخشی از خروجی فایل اکسل در پیوست‌های اول و دوم آورده شده‌است. در ادامه خروجی‌های مدل و تحلیل حساسیت این نتایج را مورد بررسی قرار می‌دهیم.

تحلیل حساسیت

در ابتدای امر حالت اولیه‌ی سیستم را بدون تغییر هیچ پارامتری، گزارش می‌دهیم. این حالت را به عنوان پایه در نظر می‌گیریم. سپس با تغییر پارامتر مدنظر حول مقدار اصلی آن، تحلیل حساسیت را انجام می‌دهیم. جهت سهولت در استفاده از نمودارها، تلاش بر این بوده‌است که به طور همزمان از دو محور عمودی استفاده شود.

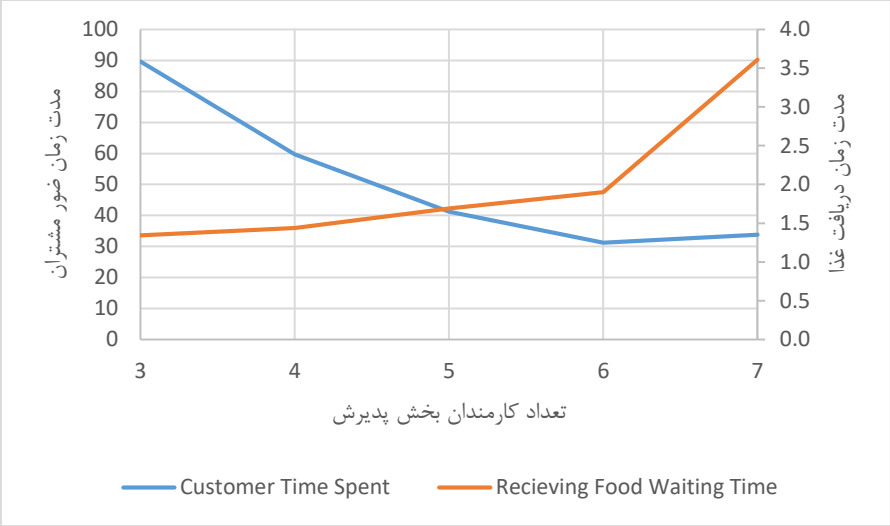
تغییر پارامتر تعداد کارمندان بخش پذیرش

در حالت پایه مقدار این پارامتر ۵ می‌باشد و جهت تحلیل مقدار آن را حول ۵ تغییر می‌دهیم. خروجی تحلیل حساسیت به شرح زیر است:

Receptionists	Customer Time Spent	Receiving Food Waiting Time	Seat Queue Length	Max Seat Queue	Receptionists Utilization	Kitchen Staff Utilization	Turnover Ratio
3	89.5740	1.3443	0.0000	0	0.9550	0.4302	1.0453
4	59.7246	1.4404	0.0000	0	0.9450	0.5566	0.9174
5	***41.2585***	***1.6922***	***0.0000***	***0***	***0.8569***	***0.6048***	***0.9677***
6	31.2675	1.8999	0.0462	4	0.6541	0.5717	1.0791
7	33.8137	3.6084	0.4238	8	0.5983	0.6230	1.0169

جدول ۱ – تحلیل حساسیت تغییر تعداد خدمت‌دهندگان بخش پذیرش

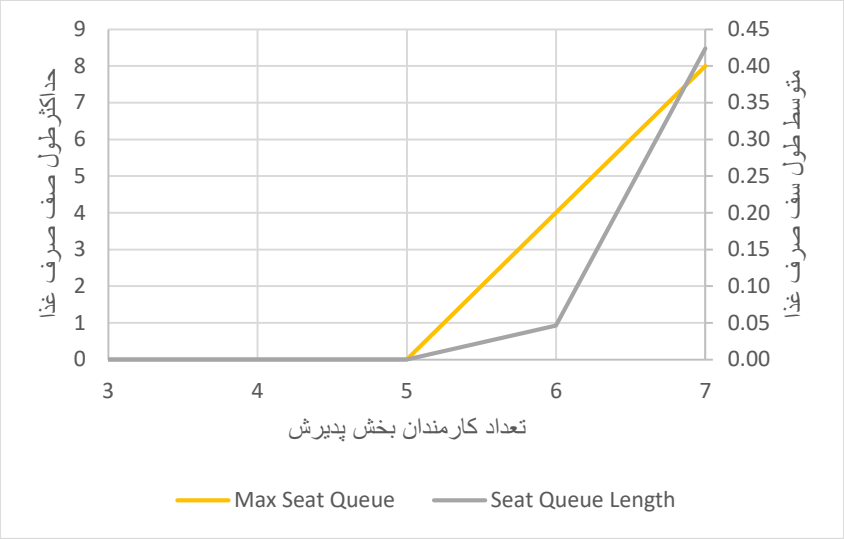
در ادامه به بررسی دقیق این تغییر می‌پردازیم.



نمودار ۱۴- تغییرات زمان حضور مشتران و دریافت غذا بر اساس تعداد کارمندان بخش پذیرش

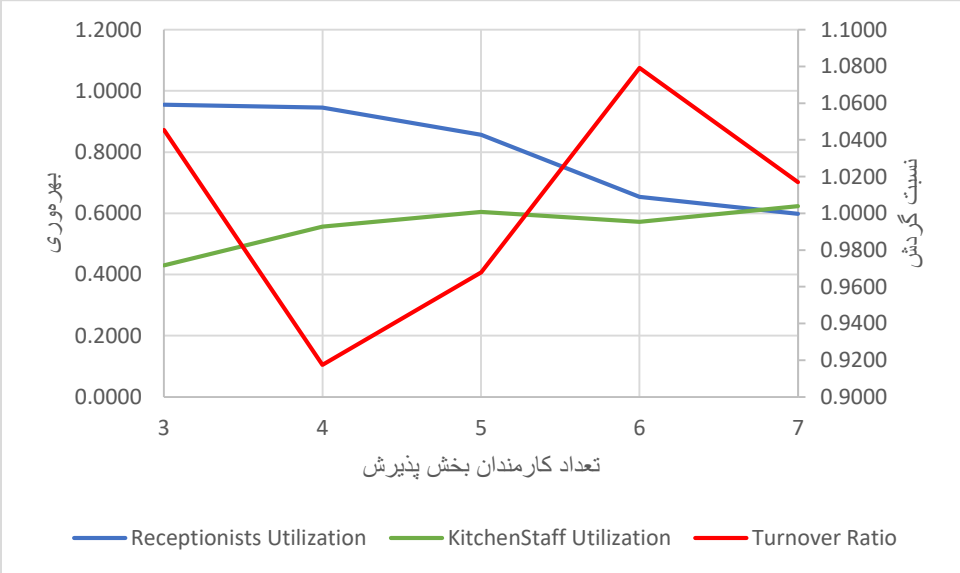
همانطور مشاهده می‌شود مطابق نمودار ۱۴ افزایش تعداد کارمندان بخش پذیرش به طور قابل ملاحظه‌ای زمان حضور مشتریان در رستوران را کاهش می‌دهد. به همین ترتیب کاهش آن، به طور فزاینده‌ای زمان حضور مشتران را افزایش می‌دهد. بنابراین سیاست صحیح این است که تا حد امکان از کاهش تعداد این افراد خودداری کنیم.

از سوی دیگر این افزایش در حد ملموسی زمان انتظار مشتریان برای دریافت غذا را نیز بیشتر می‌کند و به عبارت بهتر به نظر می‌آید که باعث ایجاد گلوگاه در بخش دریافت غذا خواهد شد. بنابراین این تغییر اثر جانبی در بخش دیگری از سیستم خواهد داشت. کاهش پارامتر مذکور اما اثر قابل توجهی بر زمان انتظار مشتران برای دریافت غذا در سیستم مورد بررسی ندارد.



نمودار ۱۵- تغییرات شرایط صف صرف غذا بر اساس تعداد کارمندان بخش پذیرش

در نمودار ۱۵ می‌توان دید که کاهش تعداد کارمندان بخش پذیرش اثری بر شرایط صف دریافت غذا ندارد. اما افزایش این افراد باعث می‌شود تا میانگین طول صف و حداکثر مقدار طول صف صرف غذا افزایش چشمگیری داشته باشد. همانطور که در تحلیل پیشین نیز ذکر شد، افزایش این افراد باعث می‌شود تا سرعت بخش پذیرش بیشتر شود و باعث ایجاد گلوگاه‌هایی در بخش‌های بعدی رستوران بشود. بنابراین به نظر می‌آید که افزایش زیاد این افراد اصلاً برای سیستم مناسب نخواهد بود و در صورت نیاز تغییرات باید با دقت انجام شود.



نمودار ۱۶ – تغییرات بهره‌وری و نسبت گردش بر اساس تغییرات تعداد کارمندان بخش پذیرش

نمودار ۱۶ به ما نشان می‌دهد که افزایش تعداد کارمندان بخش پذیرش باعث کاهش بهره‌وری و بازده آنها می‌شود و این موضوع هشداری برای ما می‌باشد که نباید تعداد این افراد را به سادگی تغییر دهیم. به این علت که

که این افزایش موجب افزایش زمان بیکاری آنها به علت تعدد نیرو می‌شود. افزایش بازده کارمندان بخش آشپرخانه با این تغییر نیز رخ می‌دهد اما به نظر می‌آید که این تغییر آنچنان چشمگیر نیست و به نظر می‌آید که باید از طرق دیگری این موضوع را بهبود بخشید.

نسبت گردش نیز تغییرات نوسانی با تغییر کارمندان بخش پذیرش دارد و به نظر می‌آید علت تغییر آن، موارد دیگری در سیستم هستند و نمی‌توان با قاطعیت در مورد این شاخص سخنی گفت. اما نکته مهم این است که در تمامی موارد شاخص حول مقدار مبنا قرار دارد و این نکته مثبتی است.

در نتیجه به صورت خلاصه می‌توان گفت تغییر کوچک در حد افزایش یک نفر به کارمندان بخش پذیرش می‌تواند اثر مثبتی بر سیستم بگذارد اما افزایش بیش از اندازه‌ی این افراد اثرات جانبی دیگری برای سیستم خواهد داشت که باید از این بازخورد جلوگیری شود.

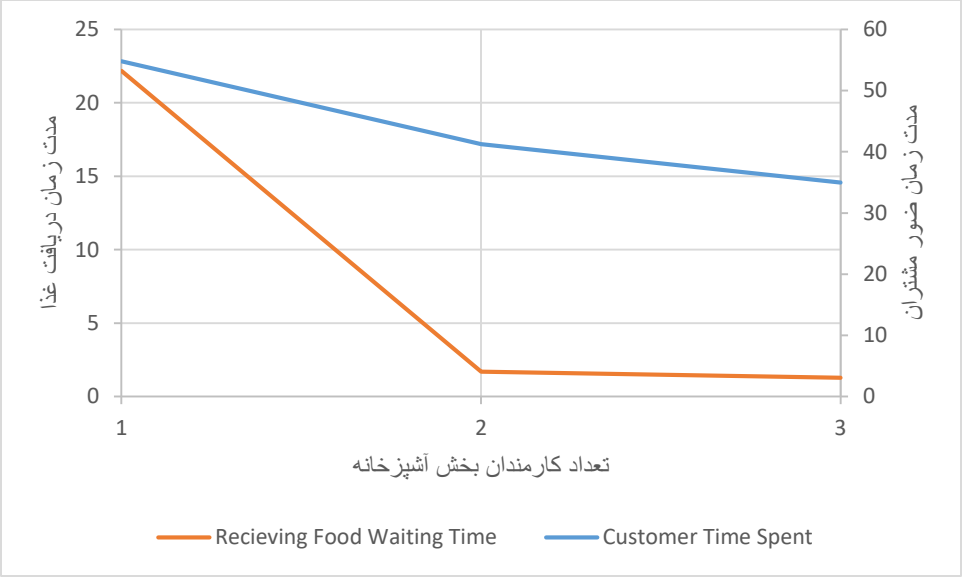
تغییر پارامتر تعداد کارمندان بخش آشپزخانه

در حالت پایه مقدار این پارامتر ۲ می‌باشد و جهت تحلیل مقدار آن را حول ۲ تغییر می‌دهیم. خروجی تحلیل حساسیت به شرح زیر است:

Kitchen Staff	Customer Time Spent	Receiving Food Waiting Time	Seat Queue Length	Max Seat Queue	Receptionists Utilization	Kitchen Staff Utilization	Turnover Ratio
1	54.7694	22.1807	0.0000	0	0.7444	0.8175	1.1111
2	***41.2585***	***1.6922***	***0.0000***	***0***	***0.8569***	***0.6048***	***0.9677***
3	34.9464	1.2780	0.0000	0	0.7157	0.3444	1.1858

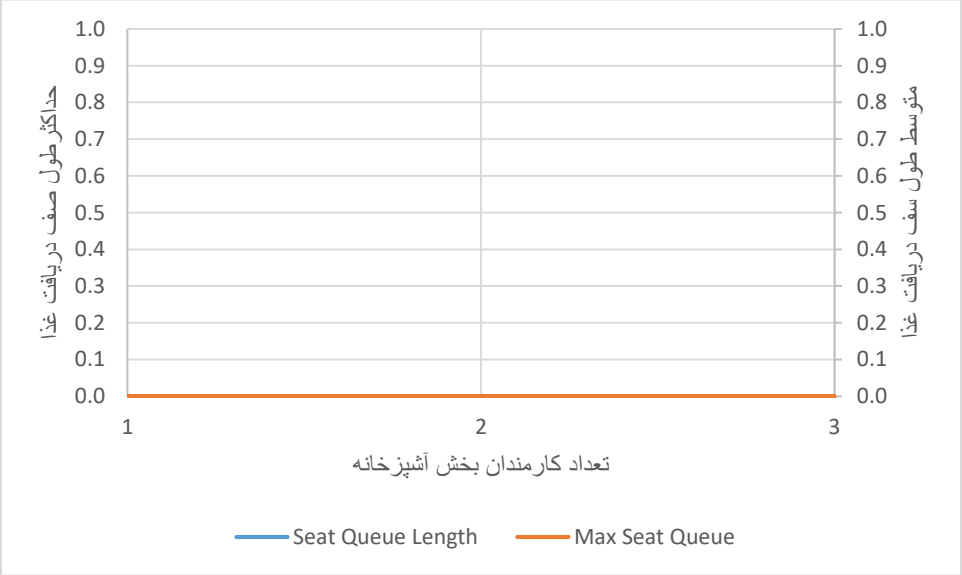
جدول ۲ - تحلیل حساسیت تغییر تعداد کارمندان بخش آشپزخانه

حال به بررسی دقیق این تغییرات می‌پردازیم.



نمودار ۱۷- تغییرات مدت زمان حضور و دریافت غذای مشتریان بر اساس تعداد کارمندان بخش آشپزخانه

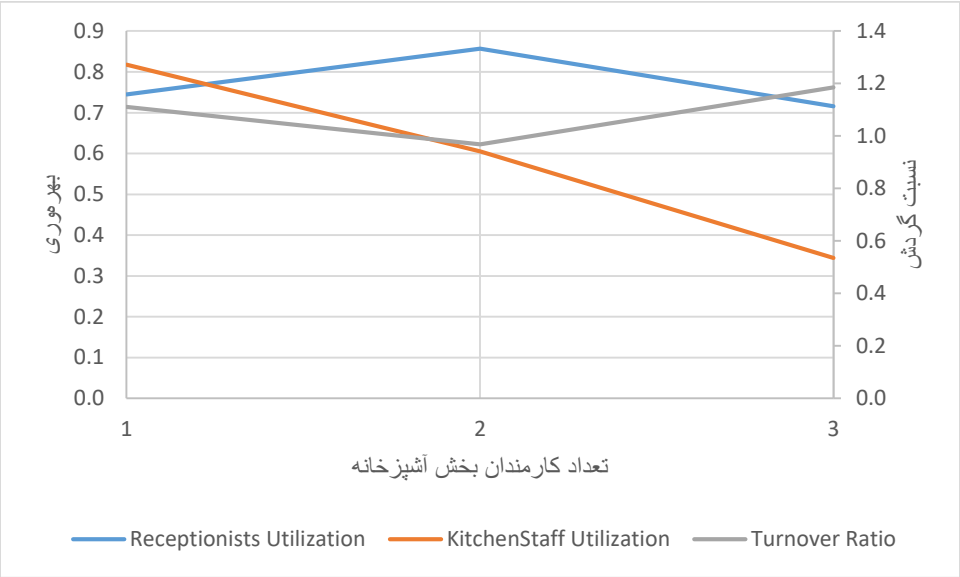
در نمودار ۱۷ می‌توان دید که افزایش تعداد کارمندان بخش پذیرش به طور قابل توجهی می‌تواند زمان حضور مشتران در سیستم را کاهش دهد و در کنار این موضوع شاید بتوان علت اصلی را کاهش زمان دریافت غذا دانست. با مشاهده‌ی نمودار به دست آمده می‌توان گفت که افزایش افراد مورد بررسی سرعت سیستم را افزایش می‌دهد و در حال حاضر می‌توان این تغییر را مثبت ارزیابی کرد. البته قطعاً این موضوع نیازمند بررسی دقیق‌تر می‌باشد.



نمودار ۱۸- تغییرات شرایط صف صرف غذا بر اساس تعداد کارمندان بخش آشپزخانه

بررسی‌های انجام شده نشان می‌دهد که افزایش و یا کاهش کارمندان آشپزخانه اثری بر شرایط صف صرف غذا ندارد و گلوگاه اصلی سیستم پیش از این قسمت قرار دارد. به همین دلیل آن بخشی که باعث تغییر عمده در

صف صرف غذا و شرایط ورود به سالن می‌باشد بخش آشپزخانه نیست و باید بخش‌های قبلی را مورد بازبینی قرار داد.



نمودار ۱۹- تغییرات بهره‌وری و نسبت گردش بر اساس تعداد کارمندان بخش آشپزخانه

همانطور که قابل پیش‌بینی نیز بود نمودار ۱۹ به ما نشان می‌دهد افزایش تعداد کارمندان بخش آشپزخانه اثری بر بازده و بهره‌وری کارمندان بخش پذیرش ندارد. اما به طور قابل توجهی می‌توان دید که افزایش کارمندان بخش آشپزخانه بازده آنها را کاهش می‌دهد و این موضوع به طور چشمگیری نیز رخ می‌دهد. بنابراین افزایش کارمندان این بخش اصلاً توصیه نمی‌شود و در صورت نیاز باید بر کاهش آنان تحلیلی انجام شود. نکته جالب اما تغییرات جزئی نسبت گردش است که شاید بتوان از آن به این عنوان برداشت کرد که در این حالت تعداد افراد خدمت گرفته در رستوان به صورت کلی تغییرات زیادی نداشته است. البته این موضوع با توجه به نوسان موجود در نمودار تنها یک گمان است و نیازمند بررسی بیشتر می‌باشد.

بنابراین می‌توان این‌گونه گفت سیستم در حال حاضر با توجه به سایر پارامترها نیازی به افزایش کارمندان بخش آشپزخانه ندارد و در صورت کاهش این افراد نیز آسیب زیادی به سیستم وارد نمی‌شود. اما عنصر اصلی تعیین‌کننده شرایط این پارامتر، بخش‌های پیشین این سیستم می‌باشد.

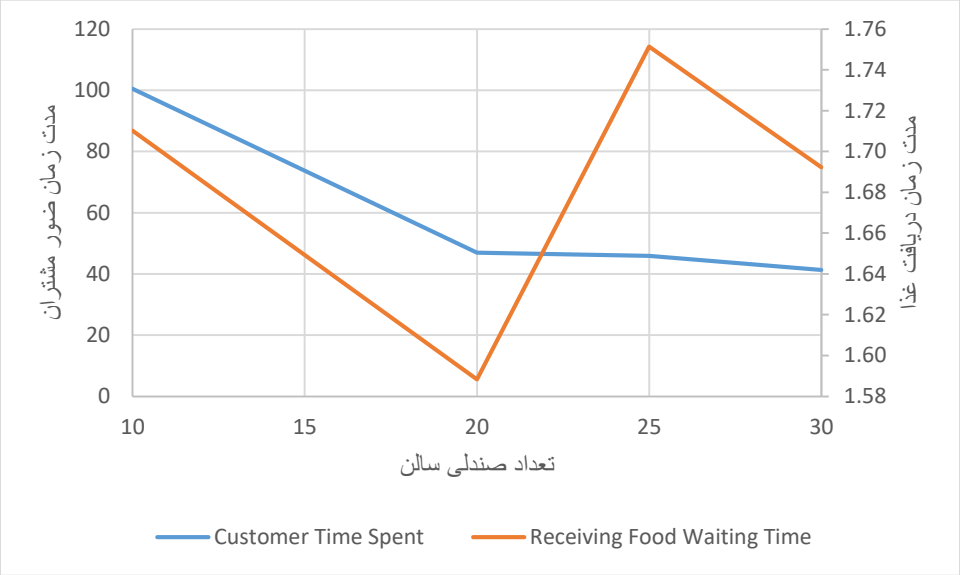
تغییر پارامتر تعداد صندلی‌های سالن غذاخوری

در حالت پایه مقدار این پارامتر ۳۰ می‌باشد و جهت تحلیل مقدار آن را حول ۳۰ تغییر می‌دهیم. لازم به ذکر است با توجه به شرایط سیستم و عدم ایجاد صف برای این بخش در تکرارهای متعد، تنها مقادیر کوچک‌تر از ۳۰ را بررسی می‌کنیم. خروجی تحلیل حساسیت به شرح زیر است:

Seats	Customer Time Spent	Receiving Food Waiting Time	Seat Queue Length	Max Seat Queue	Receptionists Utilization	Kitchen Staff Utilization	Turnover Ratio
30	***41.2585***	***1.6922***	***0.0000***	***0***	***0.8569***	***0.6048***	***0.9677***
25	45.8955	1.7515	0.1302	4	0.8105	0.5954	0.9146
20	46.9131	1.5882	7.8814	21	0.8460	0.6124	0.9901
10	100.4526	1.7101	65.0930	127	0.8050	0.5801	1.0417

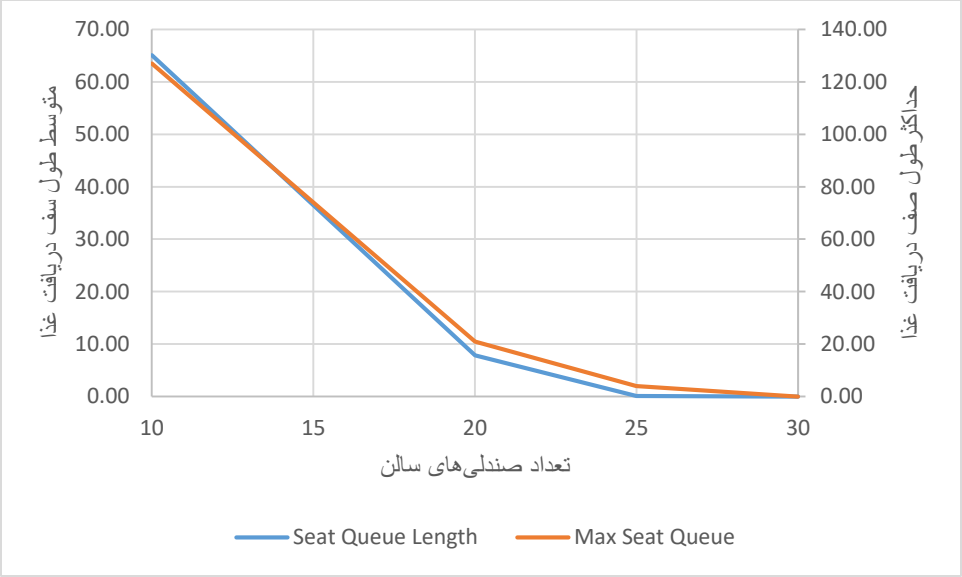
جدول ۳ - تحلیل حساسیت تعداد صندلی

در این تغییر پارامتر چون در حالت پایه میانگین طول صف سالن و حداکثر مقدار آن صفر می‌باشد، تنها حالت کاهش آن را به دقت مورد بررسی قرار می‌دهیم. حال به بررسی این تغییرات می‌پردازیم.



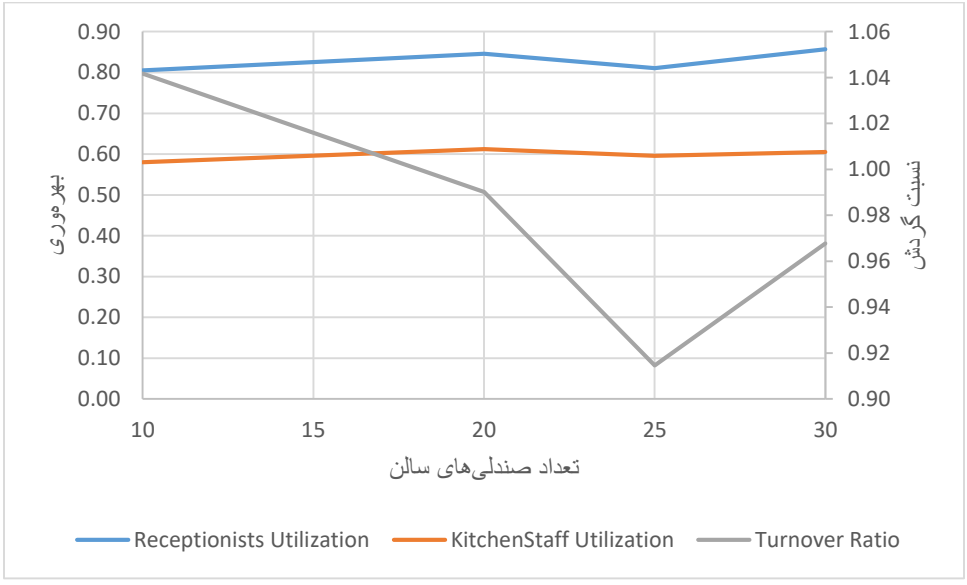
نمودار ۲۰ – تغییرات مدت زمان حضور و زمان دریافت غذای مشتریان بر اساس تعداد صندلی‌های مشتریان

نتایج حاصل از نمودار ۲۰ نشان می‌دهد که کاهش تعداد صندلی‌های تأثیر بسیار زیادی در مدت زمان حضور مشتران در رستوان دارد و باعث می‌شود این مدت زمان به طور ملموسی افزایش پیدا کند. بنابراین نباید به طور قابل توجهی تعداد صندلی‌ها را کاهش دهیم. اما همانطور که انتظار می‌رفت میانگین مدت زمان دریافت غذای مشتریان حرکت نوسانی دارد و نمی‌توان رابطه‌ای بین تعداد صندلی‌های سالن غذاخوری و زمان دریافت غذا در نظر گرفت.



نمودار ۲۱ – تغییرات شرایط صف صرف غذا بر اساس تعداد صندلی‌های سالن

نمودار فوق نشان می‌دهد که شاخص‌های مذکور مهمترین شاخص‌های مربوط به تعداد صندلی‌ها می‌باشد. کاهش تعداد صندلی‌ها به طور چشمگیری شرایط صف سالن را تضعیف می‌کند و باعث می‌شود که میانگین طول صف حتی تا ۶۵ واحد افزایش پیدا می‌کند. شرایط برای شاخص حداکثر طول صف نیز به همین صورت می‌باشد. بنابراین به نظر می‌آید که باید از کاهش صندلی‌ها خودداری کنیم و حتی اگر نیاز به چنین اقدامی داریم نباید این کاهش بیش از ۵ صندلی باشد.



نمودار ۲۲- تغییرات بهره‌وری و نسبت گردش بر اساس تعداد صندلی‌های سالن

همانطور که انتظار می‌رفت نسبت‌های مربوط به بازده کارمندان نباید رابطه‌ی معنی‌داری با تعداد صندلی‌های سالن داشته باشد و نتایج نیز به همین صورت است. اما نسبت گردش با کاهش صندلی‌ها افزایش چشمگیری دارد و حتی از مقدار مبنی یعنی ۱ نیز بیشتر می‌شود که بیانگر نارضایتی مشتریان است. بنابراین کاهش تعداد صندلی‌ها اصلاً اقدام مناسبی نیست و باعث می‌شود سرعت خدمت‌رسانی به مشتریان تا حد زیادی کاهش پیدا کند.

بررسی‌ها به صورت کلی نشان می‌دهد کاهش تعداد صندلی‌ها در صورت نیاز تا ۵ عدد قابل اجرا می‌باشد و آسیب چندانی به سیستم وارد نمی‌شود. اما همانطور که در ابتدا نیز ذکر شد افزایش تعداد صندلی‌ها سودی برای رستوران نخواهد داشت.

در پایان این بخش ذکر این نکته ضروری است که بررسی‌های انجام شده به صورت ایزوله و در شرایط شبیه‌سازی انجام شده‌اند و همواره باید به این نکته توجه کرد که این تغییرات در سیستم اثرات جانبی به همراه دارد و به همین دلیل نمی‌تواند در مورد تغییرات مذکور به صورت قاطعانه نظر داد اما خروجی‌های مذکور می‌تواند دید مناسبی از شرایط سیستم را برای ما ایجاد کند. همچنین اثرگذاری تغییر برخی پارامترها بر سایر اجزای سیستم که به نظر می‌رسد نباید به این تغییرات عکس‌العمل نشان دهند ، به علت ماهیت تصادفی بودن داده‌های مورد استفاده رخ می‌دهد و این اتفاق قابل پیش‌بینی بوده است.

تعیین براورد فاصله‌ای و نقطه‌ای

خروجی‌های انتخاب شده در ایبن بخش، میانگین مدت ماندن مشتری در سیستم، میانگین مدت انتظار مشتری جهت دریافت غذا و میانگین بهره‌وری کارکنان قسمت پذیرش که به ترتیب با TimeSpent، ReceivingTime، RUtil و KUtil نشان داده شده اند هستند. محاسبات این بخش در اکسل انجام شده است.

برای تعیین براورد نقطه‌ای، خروجی‌های مذکور برای ۵ تکرار مجزا محاسبه شدند که نتایج آن در جدول زیر قابل مشاهده است. سپس میانگین نتایج، هر کدام به صورت مجزا تحت عنوان $\overline{Y}_{..}$ به دست آمده است و این مقدار برآورد نقطه‌ای خروجی مد نظر را به ما نشان می‌دهد.

	Across-Rep Data			
R	**TimeSpent**	**ReceivingTime**	**RUtil**	**Kutil**
1	40.173208	1.616067	0.750841	0.525252
2	38.615792	1.576181	0.780569	0.52979
3	35.981477	1.584905	0.799434	0.53979
4	32.067689	1.716872	0.717635	0.478415
5	39.160109	1.641003	0.880636	0.554858

جدول ۴ - براورد نقطه‌ای و فاصله‌ای

برآورد نقطه‌ای در نهایت به صورت زیر خواهد بود:

	TimeSpent	ReceivingTime	RUtil	Kutil
$\overline{Y}..$	37.1997	1.6270	0.7858	0.5256

جدول ۵ – براورد نقطه‌ای

برای محاسبه‌ی برآورد فاصله‌ای از روابط زیر استفاده شده است:

$$\overline{Y}.. = \frac{1}{5}\sum_{i=1}^{5} \overline{Y}_i$$

$$S^2 = \frac{1}{4}\sum_{i=1}^{4} (\overline{Y}_i - \overline{Y}..)^2$$

$$H = \frac{S}{\sqrt{5}} \times t_{\alpha/2,4}$$

نتایج به دست آمده به شرح زیر هستند و Left ابتدای بازه و Right انتهای بازه را نشان می‌دهد:

	TimeSpent	ReceivingTime	RUtil	Kutil
S^2	10.6280	0.0032	0.0038	0.0008
S	3.2601	0.0565	0.0614	0.0287
H	4.0479	0.0701	0.0762	0.0357
Left	33.1518	1.5569	0.7096	0.4899
Right	41.2475	1.6971	0.8621	0.5613

جدول ۶ – براودر فاصله‌ای

دوباره‌سازی‌های لازم برای نصف کردن طول بازه اطمینان یک خروجی انتخابی

در مورد خروجی متوسط زمان صرف شده‌ی مشتریان در سیستم، برای سنجش طول فاصله‌ی اطمینان از H استفاده می‌کنیم که معرف نصف طول بازه است و در ادامه آن را با ε مقایسه می‌کنیم تا بتوانیم مقدار مورد نیاز برای تعداد دوباره‌سازی در حالتی که طول بازه نسبت به نتایج ۵ بار دوباره‌سازی نصف شده است محاسبه کنیم. روابط به کار رفته در این بخش به صورت زیر هستند:

$$H = \frac{S_0}{\sqrt{R}} \times t_{\frac{\alpha}{2},R-1} \leq \varepsilon$$

$$R \geq \left(\frac{S_0 \times t_{\frac{\alpha}{2},R-1}}{\varepsilon}\right)^2$$

$$R \geq \left(\frac{S_0 \times z_{\frac{\alpha}{2}}}{\varepsilon} \right)^2$$

رابطه ۱

لازم به ذکر است مقدار S_0 با استفاده از $R_0 = 5$ دوباره‌سازی به دست آمده است.

محاسبات این بخش در اکسل پیوست شده "Half_CI_95104182_95104114" آورده شده است. نتایج نهایی به شرح زیر هستند:

ε	R_0	$z_{\alpha/2}$	S_0
2.023947	5	1.96	3.2601

جدول ۷ – پارامترهای مورد نیاز جهت تعیین فاصله اطمینان هدف

با جایگذاری این مقادیر در رابطه ۱، مقدار ۱۰ برای R به دست می‌آید. با توجه به این که در گذشته ۵ دوباره‌سازی انجام شده است، نتیجه‌ی به دست آمده نشان می‌دهد که ۵ دوباره‌سازی دیگر لازم است تا به دقت مورد نیاز دست پیدا کنیم و طول بازه‌ی فاصله‌ی اطمینان نصف شود.

R	Extra R
10	5

جدول ۸ – تعداد دوباره‌سازی مورد نیاز

بنابراین بازه‌ی اطمینان ۹۵ درصدی θ برای به صورت زیر خواهد بود:

$$35.176 \leq \theta \leq 39.224$$

بررسی وجود و شناسایی دوره‌ی سرد و گرم سیستم

در این حالت فرض بر این است که ورود اتوبوس و مشتریان آن وجود ندارد و همچنین فست فود شبانه‌روزی است. برای این حالت نیز تعداد دوباره‌سازی‌ها[۱] را ۵ در نظر می‌گیریم.

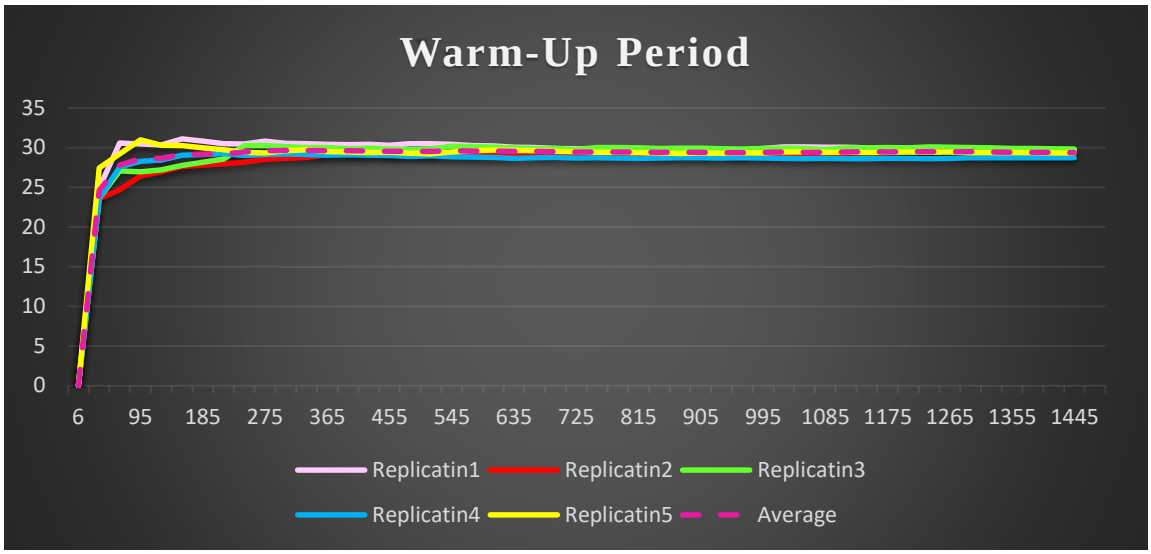
برای شناسایی این مورد نیز از اکسل استفاده شده است و جزئیات محاسبات در فایل‌های پیوست آمده است.

در ابتدا با توجه به این که توزیع‌های ورود به سیستم، زمان‌های خدمت‌رسانی، زمان صرف غذا و سایر پارامترها مانند تعداد کل سرورها، فاصله‌ها و رفتار کلی مشتری‌ها در طول زمان بررسی ثابت است و از لحاظ قواعد، قانون

[۱] Replications

و فرایند، تغییری در سیستم رخ نمی‌دهد به نظر می‌رسد که سیستم بعد از گذراندن دوره‌ی سرد، به دوره‌ی گرم خود برسد. در ادامه این فرضیه را به کمک نمودارهای رسم شده مورد بررسی قرار می‌دهیم.

نتایج به دست آمده نشان می‌دهند که در ابتدا در نمودارها یک روند صعودی داریم که بعد از مدتی به ثبات می‌رسد. بنابراین این سیستم دارای دوره‌ی سرد و گرم می‌باشد که در ادامه بیشتر به آن می‌پردازیم.



نمودار ۲۳ – بررسی زمان سرد و گرم سیستم

مبنای سنجش دوره‌ی سرد و گرم در این بخش، میانگین مدت ماندن مشتری در سیستم در نظر گرفته شده است و برای زمان‌های ۳۰ دقیقه‌ای این مقدار را ثبت می‌کنیم. این کار را ۵ بار تکرار می‌کنیم که نمودار هر تکرار در شکل بالا دیده می‌شود. در نهایت از داده‌های به دست آمده میانگین می‌گیریم که نمودار آن با خطچین نشان داده شده است. با توجه به این که داده‌ها نویز زیادی ندارند استفاده از میانگین پاسخ مناسبی در اختیار ما خواهد گذاشت. در نهایت می‌توان گفت که به صورت حدودی تا زمان ۱۰۰ دقیقه، دوره‌ی سرد سیستم است و پس از آن دوره‌ی گرم آغاز می‌شود. در واقع نقطه‌ی برش برای این سیستم را ۱۰۰ در نظر می‌گیریم.

بررسی و مقایسه سیاست جایگزین برای سیستم

در این بخش قصد داریم تا با ارائه‌ی یک سیاست و جایگزین کردن پارامترهای آن بررسی کنیم که آیا می‌توانیم شرایط و خروجی‌های سیستم رستوران مورد بررسی را بهبود بخشیم و یا خیر.

در اعمال این سیاست ابتدا فرض می‌کنیم کارمندان بخش پذیرش و آشپزخانه در حالت پایه به ترتیب برابر با ۴ و ۲ نفر هستند. همانطور که پیش‌تر نیز اشاره شد این تعداد در حالت اصلی مورد بررسی ۵ و ۲ نفر بود. حال با توجه به اینکه می‌دانیم ورود مشتریانی که به وسیله‌ی اتوبوس می‌آیند حدفاصل ساعت ۱۱ الی ۱۳ بار ترافیکی

قابل توجهی را برای سیستم ما ایجاد می‌کند، در این زمان با اضافه کردن کارمندان اضافه، افراد حاضر در بخش پذیرش و آشپزخانه را به ترتیب تبدیل به ۶ و ۳ نفر می‌کنیم. فرض مهم در این بخش این است که بعد از ساعت ۱۳ کارمندان اضافه شده پس از اتمام خدمت‌رسانی بخش خود را ترک می‌کنند.

حال می‌خواهیم با استفاده از روش نمونه‌گیری مستقل با اجرای ۱۰ دوباره‌سازی[۱۲]سیستم اصلی را با سیاست ارائه شده مقایسه کنیم. علت استفاده از ۱۰ تکرار نیز به علت محدودیت‌ها و ضعف روش نمونه‌گیری مستقل در تعداد تکرار کم است. جهت مقایسه از شاخص «میانگین مدت زمان حضور مشتری در سیستم» استفاده می‌کنیم. نتایج ۱۰ تکرار به صورت زیر گزارش می‌شود (واحد اعداد ذکر شده دقیقه می‌باشد):

Replication	Current	Alternative
1	41.4396	39.8353
2	35.0807	34.2307
3	42.4534	32.7134
4	37.8760	36.5212
5	30.3371	36.5212
6	38.4766	33.8618
7	39.0399	38.2309
8	33.0514	34.4755
9	36.1967	44.9021
10	36.6465	30.5475
Average	37.0598	36.1840

جدول ۹ – زمان انتظار مشتریان در سیستم موجود و پیشنهادی به ازای ۱۰ تکرار

حال باید با استفاده از داده‌های جدول ۹ و روابط آماری متناسب با روش نمونه‌گیری مستقل به مقایسه دقیق این دو سیستم و ارائه فاصله اطمینان[۱۳]جهت تصمیم‌گیری بپردازیم. این روابط به شرح زیر می‌باشد:

- براوردگر نقطه‌ای

$$\text{Point Estimator} = \bar{Y}_{\cdot 1} - \bar{Y}_{\cdot 2}$$

- واریانس نمونه‌ی سیستم

$$\text{Sample variance for the system} = S_i = \frac{1}{R_i - 1} \times \left(\sum_{r=1}^{R_i} Y_{ri}^2 - R_i \bar{Y}_{\cdot i}^2\right)$$

- خطای استاندارد

[۱۲] Replication

[۱۳] Confidence Interval

Standard error = s.e.$(\bar{Y}_{.1} - \bar{Y}_{.2}) = \sqrt{\frac{S_1^2}{R_1} + \frac{S_2^2}{R_2}}$

حال با با توجه به تعریف پارامترهای مورد نیاز برای محاسبه‌ی فاصله اطمینان، این فاصله از طریق رابطه‌ی زیر محاسبه می‌شود:

$$\text{C.I.} = \bar{Y}_{.1} - \bar{Y}_{.2} \pm t_{\frac{\alpha}{2},\nu} \times \text{s.e.}(\bar{Y}_{.1} - \bar{Y}_{.2})$$

رابطه ۲

همانطور که مشاهده می‌شود برای محاسبه‌ی رابطه ۲ نیاز داریم تا مقدار خطای نوع اول و درجه‌ی آزادی را محاسبه کنیم. مقدار α را برای ارزیابی خود برابر با ۵٪ در نظر می‌گیریم. جهت محاسبه‌ی درجه‌ی آزادی نیز از رابطه‌ی زیر استفاده می‌کنیم:

$$\nu = \frac{\left(\frac{S_1}{R_1} + \frac{S_2}{R_2}\right)^2}{\left[\frac{\left(\frac{S_1^2}{R_1}\right)^2}{R_1 - 1} + \frac{\left(\frac{S_2^2}{R_2}\right)^2}{R_2 - 1}\right]}$$

حال که روابط مورد نیاز را تعریف کرده‌ایم، به کمک اطلاعات جدول ۹ به محاسبه‌ی مقادیر مورد نیاز و تکمیل جدول مذکور می‌پردازیم. خروجی به شرح زیر است:

Replications	Current	Alternative
1	41.4396	39.8353
2	35.0807	34.2307
3	42.4534	32.7134
4	37.8760	36.5212
5	30.3371	36.5212
6	38.4766	33.8618
7	39.0399	38.2309
8	33.0514	34.4755
9	36.1967	44.9021
10	36.6465	30.5475
Mean	37.0598	36.1840
Variance	13.4403	16.6328
S.E.	1.7342	
Degrees of Freedom	17.7994	
Ta/2	2.1098	
Point Estimator	0.8758	
H	3.659	
Upper Bound	***4.5346***	

Lower Bound	***-2.7829***

جدول ۱۰ – ارائه فاصله اطمینان جهت مقایسه سیستم کنونی و سیاست پیشنهادی

همانطور که مشاهده می‌شود بازه‌ی اطمینان به دست آمده به ازای تعداد دوباره‌ی‌سازی انتخاب شده و خطای نوع اول مشخص شده عدد صفر را در بردارد. با توجه به اینکه که عدد صفر در داخل بازه قرار گرفته‌است، نمی‌توانیم با استناد به شواهد موجود بگوییم کدام یک از سیستم‌ها عملکرد بهتری دارند و از لحاظ آماری دلیلی بر ارجحیت یک سیستم بر دیگری وجود ندارد. البته لازم به ذکر است که نمی‌توانیم این دو سیستم را برابر در نظر بگیریم، تنها نمی‌توانیم برتری را از مقایسه خود نتیجه بگیریم. البته جهت اینکه بتوانیم دقت مقایسه خود را تا حدی بالا ببریم تا از لحاظ آماری شواهدی پیدا کنیم که کدام یک از سیستم‌ها ارجحیت دارد می‌توانیم تعداد دوباره‌سازی‌های مستقل را افزایش دهیم اما به نظر می‌آید، حتی اگر بتوانیم از لحاظ آماری برتری یک سیستم را بر دیگری اثبات کنیم، تفاوت این سیستم‌ها از لحاظ عملی و اجرایی[۴]قابل توجه نیست و نمی‌توان با این تغییر، بهبود عمده‌ای در عملکرد رستوران ایجاد کرد.

تغییرات میانگین سایر پارامترهای مهم سیستم نیز به شرح زیر است:

	Receiving Time		*Kutil*		*RUtil*		*Turnover*	
	Alternative	Current	Alternative	Current	Alternative	Current	Alternative	Current
1	1.3390	1.7334	0.4655	0.6324	0.8154	0.9083	0.9494	0.9317
2	1.4238	1.6412	0.4808	0.5723	0.8226	0.7873	1.0000	1.0870
3	1.3912	1.7596	0.4757	0.6171	0.8283	0.8577	1.0601	0.9524
4	1.4427	1.4935	0.5322	0.5611	0.8992	0.7781	0.9868	1.1111
5	1.4427	1.6320	0.5322	0.4525	0.8992	0.6603	0.9868	1.2931
6	1.4017	1.8274	0.4703	0.6409	0.8261	0.8693	1.0676	0.9772
7	1.3882	1.7564	0.4248	0.6083	0.7512	0.8240	1.0239	1.0135
8	1.4112	1.5497	0.4472	0.4962	0.7924	0.6959	1.0791	1.2245
9	1.3906	1.6102	0.5155	0.4803	0.9032	0.6724	0.9494	1.2552
10	1.3447	1.6629	0.4240	0.5325	0.7470	0.7291	1.1628	1.1321
Average	***1.3976***	***1.6666***	***0.4768***	***0.5594***	***0.8284***	***0.7782***	***1.0266***	***1.0978***

جدول ۱۱ - بررسی سایر پارامترهای مهم سیستم در مقایسه با سیاست پیشنهادی

همانطور که در جدول بالا مشاهده می‌شود تغییرات پارامترهای مهم سیستم، به طور مشهود رخ نداده‌است. مدت زمان دریافت غذا و بهره‌وری کارمندان بخش پذیرش کمی بهبود یافته و بهره‌وری کارمندان بخش آشپزخانه افت کرده‌است. اما با وجود اینکه محاسبات دقیق آماری انجام نشده‌است می‌توان پیش‌بینی کرد که این تغییرات از لحاظ عملی و اجرایی مشهود نیست. نسبت گردش نیز به طور مختصری پیشرفت کرده‌است و نشان

[۱] Practically

می‌دهد مشتریان با سرعت مناسب‌تری خدمت دریافت کرده‌اند اما در این شاخص نیز تفاوت بسیار کم و غیرقابل اتکا است. بنابراین با بررسی‌های انجام شده همچنان نمی‌توانیم ارجحیت سیاست پیشنهادی به سیستم فعلی و حالت برعکس را گزارش کنیم.

ارائه روشی بهتر جهت مقایسه سیستم پیشنهادی و موجود

مقایسه ذکر شده در بخش قبلی با استفاده از روش نمونه‌گیری مستقل[۱۵]انجام شد که ایراداتی به آن وارد است. ایراد اصلی این روش این است که اعداد تصادفی استفاده شده برای شبیه‌سازی فعالیت‌ها و زمان‌های ورود برای هر دو سیستم در هر تکرار متفاوت هستند و بنابراین در این حالت نمی‌توانیم سیستم‌ها را در یک حالت برابر و عادلانه مقایسه بکنیم. بنابراین اگر بتوانیم اعداد تصادفی مورد استفاده در هر تکرار و در هر بخش را یکسان کنیم می‌توانیم این ایراد را به طور کامل برطرف و به طور عادلانه و صحیح دو سیستم را با یک‌دیگر مقایسه کنیم. بنابراین باید از روش نمونه‌گیری CRN[۱۶] استفاده نماییم.

جهت استفاده صحیح از این روش باید اعداد تصادفی مورد استفاده برای هر فعالیت مشخص را در یک آرایه ذخیره و سپس از آن برای سیستم پیشنهادی استفاده کنیم. به عبارت بهتر، در هر تکرار می‌توانیم سیستم اصلی را اجرا و اعداد تصادفی تولید شده برای هر فعالیت مانند فعالیت سفارش غذا و پرداخت پول را در یک آرایه به‌خصوص ذخیره کنیم. این مورد را برای زمان‌های بین ورود مشتریان نیز باید انجام دهیم. سپس هنگام اجرای سیستم پیشنهادی، به ازای تولید اعداد تصادفی جدید، باید از اعدادی که در آرایه‌ها ذخیره کرده‌ایم استفاده کنیم. علت ذخیره‌ی اعداد هر فعالیت در آرایه‌های جداگانه این است که بتوانیم به درستی همگام‌سازی هر دو سیستم را انجام دهیم و از اعداد در موقعیت مناسب خود استفاده کنیم.

به همین جهت باید در قسمت‌هایی از کد شبیه‌سازی که اعداد تصادفی تولید می‌شوند، به صورت در لحظه و برخط این اعداد را ذخیره کنیم تا بتوانیم از آنها، پس از اتمام شبیه‌سازی، برای اجرای سیستم پیشنهادی استفاده نماییم. در سیستم پیشنهادی نیز این اعداد باید به ترتیب ورود آنها به آرایه‌ها استفاده شوند تا مقایسه زوجی به درستی انجام شود. بنابراین ترتیب ذخیره‌سازی و استفاده از اعداد نیز بسیار حائز اهمیت است. به صورت خلاصه نتیجه می‌گیریم که ساختار دو کد کمی متفاوت خواهد بود. در یک سیستم اعداد تولید می‌شوند و در سیستم دیگر این از اعداد استفاده خواهد شد.

---

۱۵ Independent Sampling

۱۶ Common Random Numbers

سیاست‌های بهبود سیستم

سیاست پیشنهادی اول

برای بهبود شرایط موجود، تمرکز ما در ابتدا باید بر روی شناسایی گلوگاه‌هایی باشد که وجود آن در کلیه‌ی قسمت‌های رستوران محتمل است . با بررسی بیشتر، به این نتیجه می‌رسیم که بخش پذیرش یکی از گلوگاه‌های اصلی است که در آن هم متوسط طول صف (۸,۴۷ نفر) و مدت زمان انتظار مشتریان (۸,۸ دقیقه) بالاست و هم فرایند خدمت‌رسانی به مشتریان (سفارش‌دهی و پرداخت) نسبتاً زمان‌بر و طولانی است. با توجه به اهمیت بسیار بالای زمان، همزمان انجام شدن برخی از فرایندها در صرفه‌جویی منابع موجود بسیار کمک‌کننده خواهد بود. به این منظور، از منوی آنلاین استفاده می‌کنیم و مشتریان سفارش خود را هنگامی که در صف پذیرش قرار دارند ثبت می‌کنند و این سفارش به صورت همزمان به سیستم خدمت‌دهندگان بخش پذیرش منتقل می‌شود. این کار باعث می‌شود مسئولیت بخش پذیرش، تنها بررسی نهایی سفارشات و عملیات پرداخت باشد و در واقع هم مشتریان هنگامی که در صف منتظر هستند تا حدودی مشغول خواهند بود و هم مدت زمان سفارش‌دهی کاهش پیدا خواهد کرد. در بخش پرداخت نیز می‌توان تمهیداتی در نظر گرفت تا مدت زمان کمتری از متصدیان گرفته شود. برای مثال استفاده از کیف پول الکترونیک و باشگاه مشتریان، انتخاب‌های مناسبی خواهد بود؛ به این صورت که پرداخت از طریق QR-Code و RFID در مدت زمان بسیار کوتاه‌تری انجام شود. همچنین اعضای باشگاه مشتریان، حساب خود را در مجموعه‌ی ما شارژ و از اعتبار موجود در آن برای خرید استفاده کنند. به این ترتیب زمان کل سفارش‌دهی کاهش پیدا خواهد کرد که باعث امکان رسیدگی خدمت‌دهندگان به مشتریان بیشتر، کاهش طول صف و کاهش زمان صرف شده در آن و افزایش رضایت مشتریان خواهد شد.

سیاست پیشنهادی دوم

مورد دومی که قابلیت اصلاح و تغییر دارد، سیاست تحویل غذا به مشتریان است که در حال حاضر به صورت FIFO پیاده‌سازی شده‌است؛ به این معنا که فردی که در صف جلوتر از سایرین قرار دارد، صرف نظر از مدت زمان آماده‌سازی غذایش، سفارش خود را زودتر دریافت می‌کند. ایراد وارد شده به این روش، این است که فردی که سفارشش زودتر آماده می‌شود همچنان در صف باقی می‌ماند تا کار نفرات جلویی صف به اتمام برسد. سیاست جدید به این صورت است که افراد به محض آماده شدن غذای خود می‌توانند آن را دریافت کنند و به بخش سالن منتقل شوند. در پی این اتفاق و انتقال سریع‌تر افراد به بخش سالن، از ظرفیت صندلی‌های خالی سالن نیز بهتر استفاده می‌شود. لازم به ذکر است که در واقعیت نیز این روش برای دریافت سفارش‌ها پیاده‌سازی می‌شود.

تنظیم نتیجه‌گیری

به صورت خلاصه، در این گزارش به شبیه‌سازی یک رستوران فست فود، از ساعت ۱۰ صبح تا ۳ بعد از ظهر به همراه ۵ کارمند پذیرش و ۲ کارمند در آشپزخانه پرداختیم. مشتریان بعد از ورود به صورت پیاده، با ماشین یا با اتوبوس، در بخش‌های مختلف پذیرش، آشپزخانه و سالن خدمت دریافت می‌کنند و یا در صورت نیاز در صف قرار می‌گیرند. حال به بررسی کلی هر بخش می‌پردازیم.

۱) پذیرش

با وجود این که این بخش به صورت نسبی کار خود را به خوبی انجام می‌دهد و با تعداد کارمندان ۵ نفر در ابتدا، ضریب بهره‌وری ۷۹٪ دارد، اما نکات منفی نیز در این بخش نیز بیشتر به چشم می‌آید و به نوعی گلوگاه محسوب می‌شود. صف این بخش با میانگین طول ۸,۴۷ نفر در سیستم ما یک صف طولانی محسوب می‌شود. همچنین افراد نیز در این بخش به طور متوسط ۸,۸ دقیقه در صف انتظار قرار می‌گیرند. طبق بررسی‌های انجام شده، به این نتیجه می‌رسیم که افزایش تعداد کارمندان در این بخش با وجود این که تا حدی مدت زمان ماندن مشتریان در سیستم را کاهش می‌دهد، اما به صورت معناداری باعث کاهش نرخ بهره‌وری کارمندان پذیرش خواهد شد که در کل مطلوب نیست و با تحمیل هزینه‌ی بیشتر، نتیجه‌ی چندان مناسبی ندارد. برای بهبود این وضعیت، پیشنهاد می‌شود که از سیستم‌های پرداخت جدیدتر و سریع‌تر استفاده شود و همچنین مشتریان هنگامی که در صف پذیرش قرار دارند، امکان ثبت سفارشات خود را داشته باشند تا در زمان خدمت‌دهی آن‌ها صرفه جویی شود و در کل بازدهی این بخش بالاتر رود. در ادامه نیز با مقایسه‌ی سیستمی با ۴ کارمند پذیرش با سیستمی با ۶ کارمند پذیرش در ساعت ۱۱ تا ۱۳، به این نتیجه رسیدیم که با تکیه بر ۱۰ دوباره‌سازی انجام شده، امکان ایجاد تمایز بین این دو سیستم وجود ندارد.

۲) آشپزخانه (دریافت غذا)

این بخش در ابتدا کار خود را با ۲ کارمند آغاز می‌کند. نقطه ضعف اصلی این بخش، مقدار نرخ بهره‌وری بسیار پایین کارمندان آن (تقریبا ۵۲٪) است. افراد به طور متوسط ۰,۵۷ دقیقه در صف آشپزخانه منتظرند و متوسط تعداد افراد موجود در صف این قسمت تقریبا ۰,۵۴ و هر دو اعداد مناسبی هستند.

با مقایسه‌ی این سیستم با سیستمی که در ساعت ۱۱ تا ۱۳ تعداد کارمندان آشپزخانه آن ۳ نفر است به این نتیجه می‌رسیم که با تعداد ۱۰ دوباره‌سازی، امکان ایجاد تمایز بین این دو سیستم وجود ندارد. همچنین با کاهش تعداد این افراد، با وجود این که نرخ بهره‌وری کارمند آشپزخانه افزایش می‌یابد، مدت زمان سپری شده‌ی مشتری در سیستم نیز افزایش زیادی خواهد داشت و مشتریان در این بخش زمان زیادی را از دست خواهند داد، بنابراین

گزینه‌ی مناسبی نیست. برای بهبود این بخش پیشنهاد داده شد که سیاست سیستم از FIFO تغییر کند و مشتریانی که زودتر سفارششان آماده می‌شود بتوانند زودتر نیز آن را دریافت کنند.

۳) سالن

این بخش در سیستم با تعداد ۳۰ صندلی و شرایط موجود، صفی به خود نمی‌بیند و افراد بعد از دریافت سفارش خود مستقیما به سالن می‌روند و غذای خود را صرف می‌کنند. با کاهش تعداد صندلی‌ها، برای این بخش نیز صف ایجاد می‌شود و به دلیل افزایش مدت زمانی که مشتریان در سیستم سپری می‌کنند، این کار توصیه نمی‌شود و بهتر است از ظرفیت سالن به صورت کامل بهره برده شود.

با توجه به این که برای محاسبه‌ی دوره‌ی سرد و گرم سیستم فرضیات تا حدی با موارد ذکر شده در بالا تفاوت دارند از آوردن آن‌ها پرهیز شد اما به طور کلی با فرض شبانه‌روزی بودن رستوران و نبودن اتوبوس، به این نتیجه رسیدیم که این سیستم دوره‌ی سرد و گرم دارد و در فاز جمع‌آوری داده باید این مسئله را مدنظر داشته باشیم و در تحلیل‌های بیشتر، داده‌های بخش سرد سیستم را در نظر نگیریم.

در نهایت می‌توان گفت برای اعمال تغییرات در سیستم و مقایسه با حالت فعلی، به نظر می‌رسد تعداد دوباره‌سازی‌های بیشتر از ۱۰ دور نیاز باشد تا به نتایج دقیق‌تر برسیم و با توجه به دقتی که مسئول این پروژه در نظر دارد، تعداد این دوباره‌سازی‌ها را برای هر حالت پیشنهادی، می‌توانیم با استفاده از روابط زیر محاسبه کنیم.

$$H = \frac{S_0}{\sqrt{R}} \times t_{\frac{\alpha}{2},R-1} \leq \varepsilon$$

$$R \geq \left(\frac{S_0 \times t_{\frac{\alpha}{2},R-1}}{\varepsilon}\right)^2$$

$$R \geq \left(\frac{S_0 \times z_{\frac{\alpha}{2}}}{\varepsilon}\right)^2$$

پیوست اول: فایل اکسل

در این پیوست بخش کوچکی از خروجی فایل اسکل جهت نمایش آورده شده‌است:

STEP	CURRENT EVENT	CLOCK	ORDER QUEUE LENGTH	RECEIVE QUEUE LENGTH	SEAT QUEUE LENGTH	EMPTY SEATS
1	Start of Simulation	0	0	0	0	30
2	Entrance On foot	0.285707246	0	0	0	30
3	Entrance On foot	0.403944281	0	0	0	30
4	Entrance On foot	0.680715076	0	0	0	30
5	Entrance On foot	0.713896938	0	0	0	30

6	End of Reception Service	4.070524305	0	0	0	30
7	End of Reception Service	4.830813207	0	0	0	30
8	End of Reception Service	4.878766831	0	0	0	30
9	Start of Recieving Food	4.887381588	0	0	0	30
10	End of Reception Service	5.038435164	0	0	0	30
11	Start of Recieving Food	5.16040473	0	0	0	30
12	Start of Recieving Food	5.238792558	0	1	0	30
13	Entrance On foot	5.375203042	0	1	0	30
14	Start of Recieving Food	5.550398516	0	2	0	30
15	Recieving Food	5.886232579	0	1	0	30
16	Start of Eating Food	6.025882042	0	1	0	29
17	Entrance By Car	6.469724013	0	1	0	29
18	Entrance On foot	6.571903859	0	1	0	29
19	Recieving Food	6.664628105	0	0	0	29
20	Start of Eating Food	6.778499856	0	0	0	28

پیوست دوم: کد پایتون

```
# Simulation of fast food restaurant:

# There exists 3 different forms of entrance:
# 1.Entering on foot dist.~ Negative exponential with  3 min as mean
#First entrance after 10AM

# 2.Entering by car dist.~ Negative exponential with mean = 5 min #First
entrance after 10AM
# Number of passengers in a car    Probability
#          1                          0.2
#          2                          0.3
#          3                          0.3
#          4                          0.2
# 3.Entering by bus dist.~Uniform dist.[11AM,13PM] #There is only one bus
# Number of passengers in the bus dist.~ Poisson with 30 person per hour
as mean
# Ordering food dist.~triangular[1,2,4]
# Payment dist.~[1,2,3]
# Receiving food dist.~uniform dist. [0.5,2]
# Eating food dist.~triangular[10,20,30]
# Travel time dist between parts except "Exit and salon".~Negative
exponential with 0.5min as mean
# Travel time dist "Exit and salon" parts.~Negative exponential with 1min
as mean
# People get service in a FIFO system
# No limit on any Queue's length
# Number of receptionists = 5 , Number of kitchen staff = 2
```

```
# Determined rest times:10:50, 11:50, 13:50, 14:50 #Due to circumstances
these times might change
# Rest duration = 10 min

# Outputs : 1- Customer Time Spent in System's Mean
#           2- Time waiting to Receive food Mean
#           3- Max of Seat queue's length
#           4- Seat queue's length Mean
#           5- Receptionist's efficiency mean
#           6- Kitchen staff's efficiency mean
#           7-our output
# Starting State = System is empty and servers are idle


# importing the libraries
import numpy as np
import xlsxwriter as xs
import pandas as pd
import matplotlib.pyplot as plt

# Generating exponential random number
ExpRandom = lambda y: -(1 / y) * (np.log(np.random.random()))

# Generating uniform random number
UniRandom = lambda a, b: a + (b - a) * (np.random.random())

# Generating triangular random number
TriRandom = lambda a, c, b: a + np.sqrt((b - a) * (c - a) *
np.random.random()) \
    if np.random.random() < (c - a) / (b - a) else b - np.sqrt((b - a) *
(b - c) * (np.random.random()))


# Generating poisson random number
def PosRandom(alpha):
  Z = np.sqrt(-2*np.log(np.random.random()))*np.sin(2*math.pi*np.random.random())
  X = int(np.round(alpha+np.sqrt(alpha)*Z,0))
  return X


# Calculating the number of people in a car
def CarRandom():
    Rnd = np.random.random()
    if Rnd < 0.2:
        i = 1
    elif Rnd < 0.5 and Rnd >= 0.2:
        i = 2
```

```
    elif Rnd < 0.8 and Rnd >= 0.5:
        i = 3
    else:
        i = 4
    return i


# This function shows our initial state
def starting_state():
    state = dict()
    # These are the state variables of this simulation
    state['Order Queue Length'] = dict()
    state['Receive Queue Length'] = dict()
    state['Empty Seats'] = 30
    state['Seat Queue Length'] = dict()
    state['Idle Receptionists'] = 5
    state['Idle Kitchen Staff'] = 2
    state['Resting Receptionists'] = 0
    state['Resting Kitchen Staff'] = 0
    state['WRr'] = 0
    # number of receptionists waiting to rest which is a binary variable
    state['WRk'] = 0
    # number of kitchen staff waiting to rest which is a binary variable
    state['Number of Current Customers'] = 0

    # Data Collecting Dict: saves the times below for each customer
    #   t0- Entrance, t1- Start of reception process, t2- End of reception
process
    #   t3- Entering kitchen queue, t4-Start of Receiving food, t5- End of
receiving food
    #   t6- Entering seat queue, t7- Start of Eating food, t8- End of
eating food, t9- Exit
    data = dict()
    data['Event Clock'] = 0  # The event clock
    data['Customers'] = dict()
    # The customer {'Ci':[t0,t1,t2,t3,t4,t5,t6,t7,t8,t9]}

    # Cumulative statistics
    cum_stat = dict()
    # cum_stat records history cumulatively from the beginning till a
given time
    cum_stat['Order Queue Length'] = 0
    cum_stat['Receive Queue Length'] = 0
    cum_stat['Seat Queue Length'] = 0
    cum_stat['Order Queue Waiting Time'] = 0
    cum_stat['Receive Queue Waiting Time'] = 0
    cum_stat['Seat Queue Waiting Time'] = 0
    cum_stat['Receiving Food Waiting Time'] = 0
    # It consists of the time spent in the Receive queue
    # and the time it takes for the food to be prepared by the kitchen
staff

    cum_stat['Total Number of Customers'] = 0
    # Total number of customers from the beginning till a specific time
```

```
    cum_stat['Receptionists Busy Time'] = 0
    # For all 5 servers
    cum_stat['Kitchen staff Busy Time'] = 0
    # For both servers

    cum_stat['Customer Time Spent in System'] = 0
    # For all the customers till a specific time

    # At the beginning, the first entrance on foot,by car and by bus
should be generated as C1,Car1 and B1
    future_event_list = list()
    FEL_maker(future_event_list, 'Entrance On foot', 0, 'C1')
    # Difference: make an entrance of specific customer (C1)
    FEL_maker(future_event_list, 'Entrance By Car', 0, 'Car1')
    # Difference: make an entrance of specific Car (Car1)
    FEL_maker(future_event_list, 'Entrance By Bus', 0, 'B1')

    # All predefined rest times are converted into minutes and appended in
the FEL at the beginning.
    # The 'RR" and the "RK" show that we are talking about server's rest.
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 50, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 110, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 230, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Receptionist
Resting', 'Event Time': 290, 'Customer': 'RR'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 50, 'Customer': 'RK'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 110, 'Customer': 'RK'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 230, 'Customer': 'RK'})
    future_event_list.append({'Event Type': 'Start of Kitchen staff
Resting', 'Event Time': 290, 'Customer': 'RK'})

    return state, data, future_event_list, cum_stat


# This function makes the FEL.
def FEL_maker(future_event_list, event_type, clock, customer):
    global simulation_time
    event_time = 0
    if event_type == "Entrance On foot":
        event_time = clock + ExpRandom(1 / 3)
        if event_time > simulation_time:
            # It prevents another entrance on foot after 15:00 which is
300 min
            return
    elif event_type == "Entrance By Car":
        event_time = clock + ExpRandom(1 / 5)
        if event_time > simulation_time:
            # It prevents another entrance by car after 15:00 which is 300
```

```
min
            return
    elif event_type == "Entrance By Bus":
        event_time = clock + UniRandom(60, 180)
        if event_time > simulation_time:
            # It prevents another entrance by bus after 15:00 which is 300
min
            return
    elif event_type == "End of Reception Service":
        event_time = clock + TriRandom(1, 2, 4) + TriRandom(1, 2, 3)
    elif event_type == "Start of Receiving Food":
        event_time = clock + ExpRandom(2)
    elif event_type == "Receiving Food":
        event_time = clock + UniRandom(0.5, 2)
    elif event_type == "Start of Eating Food":
        event_time = clock + ExpRandom(2)
    elif event_type == "End of Eating Food":
        event_time = clock + TriRandom(10, 20, 30)
    elif event_type == "Exit":
        event_time = clock + ExpRandom(1)
    elif event_type == 'End of Receptionist Resting':
        event_time = clock + 10
    elif event_type == "End of Kitchen staff Resting":
        event_time = clock + 10
    new_event = {'Event Type': event_type,
                 'Event Time': event_time, 'Customer': customer}
    # As mentioned above, for resting events, customer component is equal
to 'RR'.
    # additional element in event notices (Customer No.)
    future_event_list.append(new_event)


def Entrance_On_Foot(future_event_list, state, data, clock, customer,
cum_stat):
    state['Number of Current Customers'] += 1
    # Number of customers who are still in the system
    cum_stat['Total Number of Customers'] += 1
    data['Customers'][customer] = []
    # Add a place for the new customer

    if state['Idle Receptionists'] > 0:
        data['Customers'][customer].extend([clock, clock])
        # In this case, the time in which the customer enters the queue is
equal to
        # the time he/she leaves the queue. t0 = t1 = clock
        state['Idle Receptionists'] -= 1
        # Make server busy
        FEL_maker(future_event_list, "End of Reception Service", clock,
customer)
        # determine when  this customer's service ends.
    else:
        state['Order Queue Length'][customer] = clock
        data['Customers'][customer].append(clock)
        # t0
```

```
    data['Event Clock'] = clock
    # put the current clock on the last event clock  for the next event

    # Extracting the customer number
    customer_num = cum_stat['Total Number of Customers'] + 1
    # Every customer needs a label to be individually recognized
    FEL_maker(future_event_list, 'Entrance On foot', clock, 'C' +
str(customer_num))
    # predict the next customer's Arrival


def Entrance_By_Car(future_event_list, state, data, clock, car, cum_stat):
    Car_Customers = CarRandom()
    # The number of customers in the car
    customer = []
    for i in np.arange(cum_stat['Total Number of Customers'] + 2,
                       cum_stat['Total Number of Customers'] +
Car_Customers + 2, 1):
        customer.append('C' + str(i))
        # This loop labels the customers of the car. +2 is to prevent
repeating customer labels

    cum_stat['Total Number of Customers'] += Car_Customers
    state['Number of Current Customers'] += Car_Customers

    for i in np.arange(0, Car_Customers, 1):
        data['Customers'][customer[i]] = []
        # Checking whether the server is busy or not
        if state['Idle Receptionists'] > 0:
            data['Customers'][customer[i]].extend([clock, clock])  # t0 =
t1 = clock
            state['Idle Receptionists'] -= 1  # Make a server busy
            FEL_maker(future_event_list, "End of Reception Service",
clock,
                      customer[i])  # Determine when this customer's
service ends.
        else:
            state['Order Queue Length'][customer[i]] = clock
            data['Customers'][customer[i]].append(clock)  # t0

    data['Event Clock'] = clock
    # put the current clock on the last event clock  for the next event

    # Extracting the car number
    car_num = int(car[3:])
    car_num += 1
    FEL_maker(future_event_list, 'Entrance By Car', clock, 'Car' +
str(car_num))
    # predict the next car's Arrival


def Entrance_By_Bus(future_event_list, state, data, clock, bus, cum_stat):
    Bus_Customers = PosRandom(30)
    # The number of customers in the bus
```

```
    customer = []
    for i in np.arange(cum_stat['Total Number of Customers'] + 2,
                       cum_stat['Total Number of Customers'] +
Bus_Customers + 2, 1):
        customer.append('C' + str(i))
        # This loop labels the customers of the bus. +2 is to prevent
repeating customer labels

    cum_stat['Total Number of Customers'] += Bus_Customers
    state['Number of Current Customers'] += Bus_Customers

    for i in np.arange(0, Bus_Customers, 1):
        data['Customers'][customer[i]] = []
        # Checking whether the server is busy or not
        if state['Idle Receptionists'] > 0:
            data['Customers'][customer[i]].extend([clock, clock])
            # t0 = t1 = clock
            state['Idle Receptionists'] -= 1
            # Make server busy
            FEL_maker(future_event_list, "End of Reception Service",
clock,
                      customer[i])  # Determine when this customer's
service ends.
        else:
            state['Order Queue Length'][customer[i]] = clock
            data['Customers'][customer[i]].append(clock)  # t0

    data['Event Clock'] = clock
    # put the current clock on the last event clock  for the next event


def End_of_Reception_Service(future_event_list, state, data, clock,
customer, cum_stat):
    FEL_maker(future_event_list, "Start of Receiving Food", clock,
customer)
    # Forward look to generate the time to start receiving food
    data['Customers'][customer].append(clock)
    # t2 for the customer
    if state['WRr'] > 0:
        state['WRr'] -= 1
        state['Resting Receptionists'] += 1
        FEL_maker(future_event_list, "End of Receptionist Resting", clock,
"RR")
    else:
        if len(state['Order Queue Length']) > 0:
            # Accessing the first person in the queue
            first_in_queue = min(state['Order Queue Length'], key=lambda
k: state['Order Queue Length'][k])
            data['Customers'][first_in_queue].append(clock)  # t1

            cum_stat["Order Queue Waiting Time"] += (
                        data['Customers'][first_in_queue][1] -
data['Customers'][first_in_queue][0])
            FEL_maker(future_event_list, "End of Reception Service",
```

```
clock, first_in_queue)
            del state['Order Queue Length'][first_in_queue]
        else:
            state['Idle Receptionists'] += 1
    data['Event Clock'] = clock


def Start_of_Receiving_Food(future_event_list, state, data, clock,
customer, cum_stat):
    if state['Idle Kitchen Staff'] > 0:
        state['Idle Kitchen Staff'] -= 1
        data['Customers'][customer].extend([clock, clock])  # t3 = t4
        FEL_maker(future_event_list, "Receiving Food", clock, customer)
    else:
        state['Receive Queue Length'][customer] = clock
        data['Customers'][customer].append(clock)  # Queue entering time
or t3
    data['Event Clock'] = clock


def Receiving_Food(future_event_list, state, data, clock, customer,
cum_stat):
    cum_stat['Receiving Food Waiting Time'] += clock -
data['Customers'][customer][3]  # clock - t3
    FEL_maker(future_event_list, "Start of Eating Food", clock, customer)
    data['Customers'][customer].append(clock)
    # Appending t5 to customer's times
    if state['WRk'] > 0:
        state['WRk'] -= 1
        state['Resting Kitchen Staff'] += 1
        FEL_maker(future_event_list, "End of Kitchen staff Resting",
clock, "RK")
    else:
        if len(state['Receive Queue Length']) > 0:
            first_in_queue = min(state['Receive Queue Length'], key=lambda
k: state['Receive Queue Length'][k])
            data['Customers'][first_in_queue].append(clock)
            cum_stat["Receive Queue Waiting Time"] += (
                    data['Customers'][first_in_queue][4] -
data['Customers'][first_in_queue][3])
            FEL_maker(future_event_list, "Receiving Food", clock,
first_in_queue)
            del state['Receive Queue Length'][first_in_queue]
            # Bringing out the first person in the queue
        else:
            state['Idle Kitchen Staff'] += 1
    data['Event Clock'] = clock


def Eating_Start(future_event_list, state, data, clock, customer,
cum_stat):
    if state['Empty Seats'] > 0:
        state['Empty Seats'] -= 1
        data['Customers'][customer].extend([clock, clock])  # t6 = t7
```

```
        FEL_maker(future_event_list, "End of Eating Food", clock,
customer)
    else:
        state['Seat Queue Length'][customer] = clock
        data['Customers'][customer].append(clock)  # t6
    data['Event Clock'] = clock


Seat_Queue_Length = []
def Eating_End(future_event_list, state, data, clock, customer, cum_stat):
    FEL_maker(future_event_list, "Exit", clock, customer)
    data['Customers'][customer].append(clock)  # t8
    global Seat_Queue_Length
    Seat_Queue_Length.append(len(state['Seat Queue Length']))
    if len(state['Seat Queue Length']) > 0:
        first_in_queue = min(state['Seat Queue Length'], key=lambda k:
state['Seat Queue Length'][k])
        data['Customers'][first_in_queue].append(clock)

        cum_stat["Seat Queue Waiting Time"] += (
                data['Customers'][first_in_queue][7] -
data['Customers'][first_in_queue][6])
        FEL_maker(future_event_list, "End of Eating Food", clock,
first_in_queue)
        del state['Seat Queue Length'][first_in_queue]
        # Bringing out the first person in the queue
    else:
        state['Empty Seats'] += 1
    data['Event Clock'] = clock


def Exit(future_event_list, state, data, clock, customer, cum_stat):
    data['Customers'][customer].append(clock)  # t9
    state['Number of Current Customers'] -= 1
    # A customer gets out of the system
    cum_stat["Customer Time Spent in System"] +=
(data['Customers'][customer][9] - data['Customers'][customer][0])
    # t9-t0
    data['Event Clock'] = clock


def Start_of_Receptionists_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if state['Idle Receptionists'] > 0:
        state['Resting Receptionists'] += 1
        FEL_maker(future_event_list, "End of Receptionist Resting", clock,
customer)
        state['Idle Receptionists'] -= 1
        # Sending a receptionist to get some rest
    else:
        state['WRr'] += 1
        # If no server is idle to go to rest, the number of servers
waiting for a rest increases.
    data['Event Clock'] = clock
```

```
def Start_of_kitchen_staff_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if state['Idle Kitchen Staff'] > 0:
        state['Resting Kitchen Staff'] += 1
        FEL_maker(future_event_list, "End of Kitchen staff Resting",
clock, customer)
        state['Idle Kitchen Staff'] -= 1
    else:
        state['WRk'] += 1
        # If no server is idle to go to rest, the number of servers
waiting for a rest increases.
    data['Event Clock'] = clock


def End_of_Receptionists_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if len(state['Order Queue Length']) > 0:
        first_in_queue = min(state['Order Queue Length'], key=lambda k:
state['Order Queue Length'][k])
        data['Customers'][first_in_queue].append(clock)
        FEL_maker(future_event_list, "End of Reception Service", clock,
first_in_queue)
        del state['Order Queue Length'][first_in_queue]
        state['Resting Receptionists'] -= 1
    else:
        state['Idle Receptionists'] += 1
        state['Resting Receptionists'] -= 1
    data['Event Clock'] = clock


def End_of_kitchen_staff_Resting(future_event_list, state, data, clock,
customer, cum_stat):
    if len(state['Receive Queue Length']) > 0:
        first_in_queue = min(state['Receive Queue Length'], key=lambda k:
state['Receive Queue Length'][k])
        data['Customers'][first_in_queue].append(clock)
        FEL_maker(future_event_list, "Receiving Food", clock,
first_in_queue)
        del state['Receive Queue Length'][first_in_queue]
        state['Resting Kitchen Staff'] -= 1
    else:
        state['Idle Kitchen Staff'] += 1
        state['Resting Kitchen Staff'] -= 1
    data['Event Clock'] = clock

def End_of_Simulation(future_event_list, state, data, clock, customer,
cum_stat, max_Seat_Queue_Length):
    Order_Queue_Waiting_Time = 0
    Receive_Queue_Waiting_Time = 0
    Number_Of_Total_Customers = 0
    Order_Queue_Waiting_Time += cum_stat["Order Queue Waiting Time"]
    Receive_Queue_Waiting_Time += cum_stat['Receive Queue Waiting Time']
```

```
    Number_Of_Total_Customers += cum_stat['Total Number of Customers']
    #In the lines above, the values of the cum stats are stored at the end
of the simulation time

    Time_Spent = 0
    for i in data['Customers'].keys():
        # data['Customers'].keys() provides the times allocated to each
customer
        if len(data['Customers'][i]) == 10:  # checking whether the
customer is still in the system or not by checking whether "Exit" time has
been generated or not
            Time_Spent += data['Customers'][i][9] -
data['Customers'][i][0]

    #Calculating the number of customers who are not in the system anymore
    Number_Of_Completely_Served_Customers = 0
    Number_Of_Completely_Served_Customers += cum_stat['Total Number of
Customers'] - state['Number of Current Customers']

    #Calculating the number of customers which the process of receiving
food has been completed for them with their waiting time.
    Number_Of_Customers_Received_Food = 0
    Receiving_Food_Waiting_Time = 0
    for i in data['Customers'].keys():
        # data['Customers'].keys() provides the times allocated to each
customer
        if len(data['Customers'][i]) >= 6: # checking whether the customer
has received his/her food or not
            Receiving_Food_Waiting_Time += data['Customers'][i][5] -
data['Customers'][i][3]
            Number_Of_Customers_Received_Food += 1

    #Calculating the length and the max length of the seat queue.
    max_Seat_Queue = 0
    max_Seat_Queue += max_Seat_Queue_Length
    Seat_Queue_Length = cum_stat['Seat Queue Length']
    if len(state['Seat Queue Length']) > 0:
        max_Seat_Queue = max(max_Seat_Queue, len(state['Seat Queue
Length']))

    Receptionists_Busy_Time = 0
    Receptionists_Busy_Time += cum_stat['Receptionists Busy Time']
    Kitchen_staff_Busy_Time = 0
    Kitchen_staff_Busy_Time += cum_stat['Kitchen staff Busy Time']

    return Number_Of_Total_Customers,
Number_Of_Completely_Served_Customers, Number_Of_Customers_Received_Food,
Order_Queue_Waiting_Time, Receive_Queue_Waiting_Time, Time_Spent,
Receiving_Food_Waiting_Time, Seat_Queue_Length, max_Seat_Queue,
Receptionists_Busy_Time, Kitchen_staff_Busy_Time


def output_excel(worksheet, future_event_list, state, row_num):
    global max_fel
```

```
    global header_list
    # we update the header list in this function
    future_event_list = sorted(future_event_list, key=lambda x: x['Event
Time'])
    # print(future_event_list)
    new_row = [row_num, future_event_list[0]['Event Type'],
future_event_list[0]['Event Time'],
               len(state['Order Queue Length']), len(state['Receive Queue
Length']),
               len(state['Seat Queue Length']), state['Empty Seats'],
               state['Idle Receptionists'], state['Idle Kitchen Staff'],
state['Resting Receptionists'],
               state['Resting Kitchen Staff'], state['WRr'], state['WRk'],
state['Number of Current Customers']]
    # Creating new row

    # Update the header list and max_fel
    if len(future_event_list) - 1 > max_fel:
        for fel_counter in range(max_fel, len(future_event_list) - 1):
            header_list.extend(
                ("Future Event Type " + str(fel_counter + 1), "Future
Event Time " + str(fel_counter + 1)))
        max_fel = len(future_event_list) - 1

    else:
        for add_number in range(max_fel - len(future_event_list) + 1):
            future_event_list.append({"Event Type": "", "Event Time": ""})

    for fel in future_event_list[1:]:
        new_row.extend((fel['Event Type'], fel['Event Time']))
    for col in range(len(header_list)):
        worksheet.write(0, col, header_list[col])
        worksheet.write(row_num, col, new_row[col])

    return worksheet

#Setting the excel format
def excel_formatting(workbook, worksheet, row_num):
    cell_format_header = workbook.add_format()
    cell_format_header.set_align('center')
    cell_format_header.set_align('vcenter')
    cell_format_header.set_font('Times New Roman')
    cell_format_header.set_bold(True)
    worksheet.set_row(0, None, cell_format_header)

    worksheet.set_column(0, 0, 5)
    worksheet.set_column(1, 1, 13)
    worksheet.set_column(2, 2, 9)
    worksheet.set_column(3, 4, 8)
    worksheet.set_column(5, 4 + 2 * max_fel, 19)
    cell_format = workbook.add_format()
    cell_format.set_align('center')
    cell_format.set_font('Times New Roman')
    for row in range(row_num):
```

```
        worksheet.set_row(row + 1, None, cell_format)

    return workbook


def simulation(simulation_time):
    state, data, future_event_list, cum_stat = starting_state()
    # Initial state
    max_Seat_Queue_Length = 0
    clock = 0
    check = 6 #This variable is for the warm-up period detection part to
prevent the denominator to be 0
    mem = [] # A list for the warm-up period detection part which holds
the avg of customer spent time.
    memc = [] # A list which stores the times in the warm-up period
detection part
    future_event_list.append({'Event Type': 'End of Simulation', 'Event
Time': simulation_time, 'Customer': ''})
    future_event_list.append({'Event Type': 'Start of Simulation', 'Event
Time': 0, 'Customer': ''})

    workbook = xs.Workbook('Simulation_Project.xlsx')
    worksheet = workbook.add_worksheet('Restaurant')
    row_num = 1

    # The simulation continues til the last customer in the restaurant
exits.
    # The point to make here is that the restaurant will not accept any
new customers after 15:00
    while clock < simulation_time or state['Number of Current Customers']
> 0:
        sorted_fel = sorted(future_event_list, key=lambda x: x['Event
Time'])
        #print("sorted fel=")
        #print(sorted_fel)
        current_event = sorted_fel[0]
        # The first element is the thing that's happening now
        clock = current_event['Event Time']
        # Move the time forward
        # print(current_event['Event Type'])
        cum_stat['Order Queue Length'] += len(state['Order Queue Length'])
* (clock - data['Event Clock'])
        cum_stat['Receive Queue Length'] += len(state['Receive Queue
Length']) * (clock - data['Event Clock'])
        cum_stat['Seat Queue Length'] += len(state['Seat Queue Length']) *
(clock - data['Event Clock'])

        temp = len(state['Seat Queue Length'])
        max_Seat_Queue_Length = max(temp, max_Seat_Queue_Length)

        cum_stat["Receptionists Busy Time"] += (5 - state['Idle
Receptionists'] - state['Resting Receptionists']) * (
                clock - data['Event Clock'])
```

```
        cum_stat["Kitchen staff Busy Time"] += (2 - state['Idle Kitchen
Staff'] - state['Resting Kitchen Staff']) * (
                clock - data['Event Clock'])

        if current_event['Event Type'] == 'End of Simulation':
            Number_Of_Total_Customers,
Number_Of_Completely_Served_Customers, Number_Of_Customers_Received_Food,
Order_Queue_Waiting_Time, Receive_Queue_Waiting_Time, Time_Spent,
Receiving_Food_Waiting_Time, Seat_Queue_Length, max_Seat_Queue,
Receptionists_Busy_Time, Kitchen_staff_Busy_Time =
End_of_Simulation(future_event_list, state, data, clock,
current_customer,cum_stat, max_Seat_Queue_Length)

        # print("state=")
        # print(state)
        # print("cum_stat=")
        # print(cum_stat)
        # print("data=")
        # print(data)

        if clock < simulation_time or state['Number of Current Customers']
> 0:
            current_customer = current_event['Customer']
            if current_event['Event Type'] == 'Entrance On foot':
                Entrance_On_Foot(future_event_list, state, data, clock,
current_customer, cum_stat)
            elif current_event['Event Type'] == 'Entrance By Car':
                Entrance_By_Car(future_event_list, state, data, clock,
current_customer, cum_stat)
            elif current_event['Event Type'] == 'Entrance By Bus':
                Entrance_By_Bus(future_event_list, state, data, clock,
current_customer, cum_stat)
            elif current_event['Event Type'] == 'End of Reception
Service':
                End_of_Reception_Service(future_event_list, state, data,
clock, current_customer, cum_stat)
            elif current_event['Event Type'] == 'Start of Receiving Food':
                Start_of_Receiving_Food(future_event_list, state, data,
clock, current_customer, cum_stat)
            elif current_event['Event Type'] == 'Receiving Food':
                Receiving_Food(future_event_list, state, data, clock,
current_customer, cum_stat)
            elif current_event['Event Type'] == 'Start of Eating Food':
                Eating_Start(future_event_list, state, data, clock,
current_customer, cum_stat)
            elif current_event['Event Type'] == 'End of Eating Food':
                Eating_End(future_event_list, state, data, clock,
current_customer, cum_stat)

            elif current_event['Event Type'] == 'Start of Receptionist
Resting':
                Start_of_Receptionists_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)
            elif current_event['Event Type'] == 'Start of Kitchen staff
```

```
Resting':
                Start_of_kitchen_staff_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)

            elif current_event['Event Type'] == 'End of Receptionist
Resting':
                End_of_Receptionists_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)
            elif current_event['Event Type'] == 'End of Kitchen staff
Resting':
                End_of_kitchen_staff_Resting(future_event_list, state,
data, clock, current_customer, cum_stat)

            elif current_event['Event Type'] == 'Exit':
                Exit(future_event_list, state, data, clock,
current_customer, cum_stat)

            output_excel(worksheet, future_event_list, state, row_num)
            row_num += 1
            future_event_list.remove(current_event)
            if clock >= check:
                if cum_stat['Total Number of Customers'] - state['Number
of Current Customers'] == 0:
                    mem.append(cum_stat['Customer Time Spent in System'] /
state['Number of Current Customers'])
                    memc.append(clock)
                    check += 30 #This part is for detecting warm-up period
and checks the times each 30 min
                else:
                    mem.append(cum_stat['Customer Time Spent in System'] /
(cum_stat['Total Number of Customers'] - state['Number of Current
Customers']))
                    memc.append(clock)
                    check += 30

    # outputs
    #This part is more detailed in the excel file provided for the warm-up
period detection
    plt.plot(memc, mem, c='b')
    plt.xlabel('Clock')
    plt.ylabel('Length')
    plt.show()

    # simulation time outputs
    # Output1
    Customer_Time_Spent_1 = Time_Spent /
Number_Of_Completely_Served_Customers
    print("Customer_Time_Spent_1")
    print(Customer_Time_Spent_1)
    # Output2
    Receiving_Food_Waiting_Time_1 = Receiving_Food_Waiting_Time /
Number_Of_Customers_Received_Food
    print("Receiving_Food_Waiting_Time_1")
    print(Receiving_Food_Waiting_Time_1)
```

```
    # Output3
    Seat_Queue_Length_1 = Seat_Queue_Length / simulation_time
    print("Seat_Queue_Length_1")
    print(Seat_Queue_Length_1)
    max_Seat_Queue_1 = max_Seat_Queue
    print("max_Seat_Queue_1")
    print(max_Seat_Queue_1)
    # Output4
    Receptionists_util_1 = Receptionists_Busy_Time / (simulation_time * 5)
    print("Receptionists_util_1")
    print(Receptionists_util_1)
    KitchenStaff_util_1 = Kitchen_staff_Busy_Time / (simulation_time * 2)
    print("KitchenStaff_util_1")
    print(KitchenStaff_util_1)
    # Output5
    Turnover_Ratio_1 = simulation_time / Number_Of_Total_Customers
    print("Turnover_Ratio_1")
    print(Turnover_Ratio_1)


    # outputs till the restaurant gets empty
    #Note that since the results were almost the same as the outputs
above, these parts do not have a "print" section.
    # Output1
    Customer_Time_Spent_2 = cum_stat['Customer Time Spent in System'] /
len(data['Customers'])
    # Output2
    Receiving_Food_Waiting_Time_2 = cum_stat['Receiving Food Waiting
Time'] / len(data['Customers'])
    # Output3
    Seat_Queue_Length_2 = cum_stat['Seat Queue Length'] / clock
    max_Seat_Queue_2 = max_Seat_Queue_Length
    # Output4
    Receptionists_util_2 = cum_stat['Receptionists Busy Time'] / (clock *
5)
    KitchenStaff_util_2 = cum_stat['Kitchen staff Busy Time'] / (clock *
2)
    #print("final clock")
    #print(clock)

    workbook = excel_formatting(workbook, worksheet, row_num)
    workbook.close()
    print("End!")
    #return Customer_Time_Spent_1, Receiving_Food_Waiting_Time_1,
Receptionists_util_1, KitchenStaff_util_1
    return mem, memc

max_fel = 0
# Maximum length that FEL gets (the current event does not count in)
header_list = ['Step', 'Current Event', 'Clock', 'Order Queue Length',
'Receive Queue Length', 'Seat Queue Length',
               'Empty Seats', 'Idle Receptionists', 'Idle Kitchen Staff',
'Resting Receptionists',
               'Resting Kitchen Staff', 'WRr', 'WRk', 'Number of Current
```

```
Customers']
simulation_time = int(input("Enter the Simulation Time: "))
simulation(simulation_time)
"""""
#Performing 5 replications
Result = []
for i in np.arange(0, 5, 1):
    Result.append(simulation(simulation_time))

Result = pd.DataFrame(Result)
#print(np.mean(Result), axis=0)
#Result.columns = ['TimeSpent', 'ReceivingTime', 'RUtil', 'KUtil']
print(Result)
"""
```

منابع و مآخذ

- Crason. B, (2013), Discrete event system Simulation. Pearson.
- Dunne. S, 2018, *5 KPIs Every Savvy Multi-Location Restaurant Manager Should Track*, <https://www.bizimply.com/blog/5-kpis-every-savvy-multi-location-restaurant-manager-should-track/>.
- Sigurdson. S, 2018, *15 Important Restaurant KPIs*, <https://www.franchiseblast.com/restaurant-kpis/>
- صدقی. ن، (۱۳۹۹-۱۳۹۸)، اسلایدهای تدریسی درس شبیه‌سازی در دانشگاه صنعتی شریف: آموزش شبیه‌سازی برگرفته از کتاب Discrete event system simulation.