Problem definition

This simulation tries to solve an optimalization problem in a local restaurant during lunchtime (peak hours). As guests come for the lunch they need to be served as fast as possible because they have just a limited time for their lunch.

Our goal is to find an optimal number of tables, seats, kitchen performance and number of waiters to satisfy all incoming guests in the local restaurant and recommend changes if necessary.

The local (existing) restaurant parameters:

- 20 tables in square layout
- each table has 4 seats
- 1 entrance
- 2 waiters
- at peak hours (11:00 AM – 1:00 PM) every 1 minute 1 guest comes (in average, personally measured). Some guests come in groups but the average is similar.
- typical guest has 60 minutes to lunch
- 45 minutes of this time could be spent in the restaurant
- average time for a meal preparation is about 1 minute (in average, personally measured)

Method

To simulate and model the issue, Netlogo 5.0.3. software is used because it's easy to use, describe and visualize the whole restaurant including distances in place. Also, objects and their properties are close to reality so Netlogo is the best tool to set it up.

The analysis was about running the simulation with the same time-frame and different parameters. Our goal is to keep the number of waiters as small as possible, kitchen as slow as possible, keep the average visit time low, increase number of incoming guests and keep them satisfied (nobody likes waiting) and decide if we can to optimize the flow in the restaurant somehow.

Detailed description of the method

Quick Start

If you're impatient or you don't waste your time reading the manual, just press „reset to default values“ button and then „go“. The following is the detailed description of the model.

Guests

- Green / orange / red arrow shape (set by time remaining, see bottom)
- One guest comes every n-th tick set by ***guest-every-nth-tick*** input
- A maximum time for the lunch is 60 minutes.
- When guests spend (0; 75> % of the maximum time in the restaurant they're OK and they're green
- When they spend (75; 90> % they're in rush, orange.
- When they spend (90; 100> % they are unsatisfied because they won't be back at their work

on time, red.

- A guest's flow is following:
  - **coming** – the first state when guests come to the restaurant
  - **seating** – they're looking for a free place at any table but they prefer to sit alone
  - **ordering** – when they get a seat they're waiting for their waiter (every waiter has his table where he's serving to)
  - **waiting** – whey they put an order to the waiter they're waiting for ordered meal
  - **eating** – when the waiter brings the meal they're eating
  - **wanna pay** – when they're finished with the meal they're waiting for the waiter again as they want to pay
  - **leaving** – done with pay, they're looking for the nearest exit and leaving the restaurant
  - **left** – the last and „fake“ state, just for statistical reasons. Guests in this state doesn't exist anymore but we need to know how many guests visited our restaurant since we opened.

Waiters

- blue arrow shape
- waiters serve to guests through tables
- at the setup phase, every waiter will draw the tables where they're serving at. The drawing process is random and „fair“ as every waiter has almost same number of tables. For example if we have 4 tables and 2 waiters, each of them has 2 tables. If we have 5 tables and 2 waiters, first waiter will have 2 tables and the second one has 3 tables because 1 table can be served just by 1 waiter, not more.
- If the number of waiters is greater than number of tables, the remaining waiters don't serve and they're stucked and useless.
- Waiters circulate between tables and kitchen.
- They skip empty tables.
- They skip kitchen if there is nothing to put or pick up from it.
- When they receive an order from a customer they put it to the „***orders-to-kitchen***“ queue and this queue is passed to the kitchen on the next visit.
- Whey they pick up finished meal from the kitchen they put the meal to the „**orders-to-table**“ queue and the meal (order) is served to the table where it belongs to.
- The number of waiters can be set via „**waiters-count**“ slider.
- The optimum is to have as many waiters as tables but in reality it's not possible.

Kitchen

- white square shape
- there is always one kitchen in the simulation and its position is always in the middle of world.
- kitchen receives orders from waiters and prepares meals. The „speed“ of the preparation can be set by „***max-ticks-needed-for-preparing-meal***“. Using this value, one meal is prepared every n-th tick.
- kitchen has two queues: „**orders-to-cook**“ (queue to be done) and „**orders-cooked**“ (finished queue). Orders-to-cook is filled by waiters (they empty their „orders-to-kitchen“ here) and orders-cooked is filled by the kitchen itself.

- If we want to scale up the kitchen to be as fast as possible we need to set up the „***max-ticks-needed-for-preparing-meal“*** slider to the smallest value (1). Then every tick one meal can be prepared.

Tables

- brown square shape
- tables are served by waiters
- number of tables can be set by „**tables-count**“ slider
- tables are positioned randomly except the case when „**fixed with 20 tables and 1 entrance**“ layout is used. In this case, fixed layout and only 20 tables are set.

Entrances / Exits

- red square shape
- number of entrances can be set through „***entrances-count***“ slider
- guests come and leave througt entrances / exits
- coming through particular entrance is random
- when guests leave they're looking for the nearest entrance from the table which they just left.
- entrances and exists are positioned randomly except the case when „**fixed with 20 tables and 1 entrance**“ layout is used. In this case, fixed layout and only 1 entrance is set.

Layouts

- layout is a positioning of tables and entrances for simulation use-cases
- 2 layouts (via chooser) can be set:
  - ***fixed with 20 tables and 1 entrance*** – this layout is just for purposes of „Simulation of Systems, 4IT496“ Course at University of Economics, Prague.
  - ***random using set tables and entrances*** **–** this layout puts the tables and entrance randomly in the world

Interface elements and derived variables

Controls

- **layout** – table and entrance positioning, see above „layout“
- **tables-count** – number of tables used in the simulation. This variable will take into account only if we use „***random using set tables and entrances“*** layout. Default is 20.
- **entrances-count** – number of entrances/exits. This variable will take into account only if we use „***random using set tables and entrances“*** layout. Default is 1.
- **waiters-count** – number of waiters. Typical value is 2-5 waiters for one restaurant. Default is 2.
- **table-seats** – number of seats at one table. More seats we have, more guest we can serve. Default is 4 so 1 table can hosts 4 guests.
- **max-ticks-needed-for-preparing-meal** - time for preparing one meal in the kitchen.

Default is 60 (60 ticks=60 seconds). It uses modulo: When ***ticks mod max-ticks-needed-for-preparing-meal*** *= 0*, one meal (lunch) is prepared for pull out by a waiter for particular table.

- **max-ticks-for-lunch** - guest's time for lunch. Default is 2700 ticks which stands for 2700 seconds (45 minutes).
- **max-ticks-needed-for-eating** - time for eating lunch. Default is 1800 which stands for 1800 seconds (30 minutes). It uses modulo: When *ticks mod max-ticks-needed-for-eating = 0*, a guest is done with his lunch.
- **max-ticks** - maximum ticks when the simulation will stop. Just for convenience, doesn't have any impact to the model but we need to stop running simulation at the same time for make an analysis.
- **guest-every-nth-tick -** every n-th tick a new guest will come. Default is 60 so 1 guest will come each minute. Recommended value is between 20-100.
- **show-labels?** - would we like to see guests states and orders on the fly? Default on.

Buttons

- **setup** – this button set up the simulation, place objects and make drawing of waiters for their tables.
- **go** – run the simulation up to „***max-ticks***" variable, then stops.
- **go once** – run one step of simulation.
- **reset to default values** – set up the simulation for 4IT496 purposes (a local restaurant problem). Simulation conclusion will follow values set up by this button.

Plots

- **guest's satisfaction when left** – this plot will show % graph of guests mood. Green – they left restaurant on time, Orange – they're in rush (time is almost gone) and Red – they exceeded the time dedicated to their lunch.

Monitors

- **guests-here** - number of guests in restaurant now
- **left-ok** - number of guests who left the restaurant with good feeling (time for lunch hasn't been exceeded)
- **left-in-rush** - number of guests who left the restaurant in rush (time for lunch has been almost exceeded)
- **left-unsatisfied** - number of guests who left the restaurant unsatisfied (time for lunch has been exceeded)
- **% left-ok** - rate of guests who left the restaurant with good mood (time for lunch hasn't been exceeded) related to total guests
- **% left-in-rush** - rate of guests who left the restaurant in rush (time for lunch has been almost exceeded) related to total guests
- **% left-unsatisfied** - rate of guests who left the restaurant unsatisfied (time for lunch has been exceeded) related to total guests

- **avg visit time –** how many secods (ticks) guests spent in restaurant in average. The lower is the better.
- **coming, seating, ordering, waiting, eating, wanna-pay, leaving, left** - number of guests with these states
- **kitchen orders-to-cook** - number of orders in kitchen waiting to be cooked
- **kitchen orders-cooked** - number of orders in kitchen waiting to be pull-out by its waiter

Model limitations

- Objects don't avoid. Even I planned to solve avoiding to the very first version the issue was too complicated because the code grew rapidly, has huge side effects and even got stucked. Almost 1/3 of the code should have been dedicated just to avoiding so it was skipped in the version 1.0.
- Waiters don't help each other. Each waiter has his tables and will not go to another table at all. In reality they could help themselves. For example if my tables are empty I can serve another tables.
- Technically, waiters don't serve to guests but to tables and we don't register order for particular guest. So each order to particular table is dedicated just to the table. Then, when one guest will make an order and second guest at the same table will make another order, the second one can eat meal for the first one and vice versa. You can imagine that we have just one kind of meal, for example. In reality, we have more kinds of meals and the earlier guest will typically eat earlier than the later one.
- Waiters can bring unlimited orders and meals at once. In reality, their capacity is limited.
- The simulation doesn't recognize „peak hours“ and „weak hours“. Simulation is strictly dedicated to peak hours only so the flow of the guests is the same all the time (through „***guest-every-nth-tick***“ input).

Results

Following results are for fixed layout (20 tables and 1 entrance) like at local restaurant we're trying to simulate.

Each test (row in following table) has been run 3 times and results were averaged. Because number of ticks were huge the differencies between individual runs are minimal and there is no need to run tests more times.

Due to performance reasons and simulation time running (in hours) some tests were run with less ticks than the others but their result can be approximated easily.

Legend

Reality (like at the restaurant) has yellow background.

The best 3 results have green background.

The worst 3 results have red backgroud.

The plus and minus signs after numbers means a change compared to the previous row.

Columns with bold heading are those we have changed during tests. These values are under our control and we can change or influence it somehow. We have already tested a possibility that customer incoming rate (***guest-every-nth-tick*** variable) would increase.

The number of ticks ran is in the last column „number of ticks running“.

tables-count	entrances-count	**waiters-count**	table-seats	**max-ticks-needed-for-preparing-meal**	max-ticks-for-lunch	max-ticks-needed-for-eating	**guest-every-nth-tick**	avg visit time (ticks)	% ok	% in rush	% unsatisfied	total guests left	number of ticks running
20	1	1	4	60	2700	1800	60	1639	71,51	22,1	6,39	16640	1 mil.
20	1	2 (+)	4	60	2700	1800	60	1419 (-)	83,93 (+)	15,78 (-)	0,29 (-)	16643	1 mil.
20	1	3 (+)	4	60	2700	1800	60	1291 (-)	90,85 (+)	9,15 (-)	0 (-)	16645 (+)	1 mil.
20	1	2 (-)	4	55 (-)	2700	1800	60	1209 (-)	95,15 (+)	4,85 (-)	0	16646 (+)	1 mil.
20	1	2	4	55	2700	1800	55 (-)	1341 (+)	87,91 (-)	12,07 (+)	0,02 (+)	18157 (+)	1 mil.
20	1	2	4	55	2700	1800	50 (-)	21449 (+)	3,45 (-)	1,37 (-)	95,18 (+)	9063 (+)	0,5 mil. (-)
20	1	2	4	50 (-)	2700	1800	50	1318 (-)	89,14 (+)	10,86 (+)	0 (-)	19972 (+)	1 mil. (+)
20	1	2	4	45 (-)	2700	1800	45 (-)	1333 (+)	88,33 (-)	11,24 (+)	0,42 (+)	22193 (+)	1 mil.
20	1	2	4	40 (-)	2700	1800	40 (-)	1368 (+)	86,56 (-)	12,81 (+)	0,63 (+)	24967 (+)	1 mil.
20	1	1 (-)	4	35 (-)	2700	1800	35 (-)	1778 (+)	64,52 (-)	18,72 (+)	16,76 (+)	28523 (+)	1 mil.
20	1	1	4	30 (-)	2700	1800	30 (-)	2597 (+)	20,59 (-)	22,25 (+)	57,16 (+)	33244 (+)	1 mil.
20	1	2 (+)	4	30	2700	1800	30	1313 (-)	89,22 (+)	10,5 (-)	0,29 (-)	33288 (+)	1 mil.

Our goal was keep the rate of unsatisfied guest at minimum and not spend too much money on employees.

As it's shown, the biggest influence on the smooth flow in the restaurant has kitchen performance. The number of waiters is important too but kitchen is much more important. In case of guest's increasing rate we should start with kitchen optimalization. Our goal should be to have as fast kitchen as possible which can be achieved by better positioning of desks, sinks and tools in the kitchen. Also, number of chefs can be increased but this option is quite expensive.

Optimal and affordable number of waiter is 2. As said, when the number of guests will increase we should reduce the time for preparing meal (kitchen performance) at first. If we have more waiters but kitchen performance is low nothing other can help.

Conclusion

Back to reality. I suggest keeping the number of waiters as current (2). Also we should try to make meal preparation faster. If we speed up the kitchen from 60 seconds to 55 all the guests would be satisfied. And when we speed up the kitchen to 50 seconds we would handle number of incoming customers rate from 1 guest every 60 seconds to 1 guest every 50 seconds without any complaints.

Model source code

(xls, spm, nlogo, mdl, etc. file)