Simulation-driven gym layout optimisation

Andrej Jočić, Matic Stare in Martin Starič

Collective behaviour course research seminar report

January 7, 2024

Iztok Lebar Bajec | izredni professor | mentor

This report describes our work on gym layout optimisation. We begin with a brief overview of the problem and related work, followed by a description of our model and simulation. We conclude with a discussion of our results and future work.

gym layout | crowd simulation | gym-goer behaviour | gym traffic

Significance statement

This report describes our work on gym layout optimisation. We begin with a brief overview of the problem and related work, followed by a description of our model and simulation. We conclude with a discussion of our results and future work.

Introduction

In this report we describe our work on gym layout optimisation. A common occurrence in gyms during peak hours is over-crowding of popular equipment and lots of wandering around searching for a free machine. Through simulating gym-goer behavior, we aimed to identify the most effective arrangement of exercise equipment to improve the flow, accessibility, and overall customer satisfaction. Our goal is to offer practical insights for gym owners, managers, and designers seeking to optimise their facility layout for the benefit of their clients and business success.

To simplify the problem we will limit ourselves to simulating the behaviour of bodybuilders (performing resistance exercises for muscle hypertrophy). In this case, a typical workout routine partitions the body into several sets of muscles, cycling through them on a per-workout basis. To further simplify the problem, we will only simulate the *push-pull-legs* (PPL) partitioning[1], which is a popular choice for beginners and intermediate lifters. We will model the gym as a rectangular grid of cells with pre-set equipment locations, distributed similarly as isles in a grocery store.

Related work. The only publication related to gym layout optimisation we could find was ref. [1], which assumed all gym clients had fixed-order workout routines (including ones with weight loss as a goal) and optimised a circular gym layout to minimise backward movement. Unfortunately, this does not give a good foundation for our work, as the assumptions diverge too far from what we are trying to model.

Thus we started from a crowd modelling review [2] for basic model design principles. We also found two articles on incremental urban layout optimisation [3, 4] useful when designing the optimisation procedure, especially the cost function.

Methods

We used Python's *Mesa* library for simulation, analysis, and visualization of agent-based models.

Gym layout model. A gym is represented as a subclass of `mesa.Model` with two discrete rectangular grids: the agent layer (where gym-goers move around) and the equipment layer (where machines are statically placed for one simulation cycle). The agents can only move around in cells that are not occupied by a piece of equipment, but there can be multiple agents in a single cell. We do not feel that agent collision resolution is necessary for realistic modelling of gym traffic. One cell on the grid's border serves as the locker area entrance, where agents enter the gym.

For generating a concrete placement of machines, we have to assign certain cells to be equipment locations (where one machine is placed) while making sure we don't create unreachable areas. We have developed some basic layout templates mimicking the arrangement of isles in a grocery store, with sensible walkways in between. In this case, layout optimisation comes down to choosing the best assignment of machines to equipment locations.

Gym-goer behaviour model. A gym-goer is represented as a subclass of `mesa.Agent` with a training checklist (multiset of muscles) and its current state. The agent's goal is to exhaust the checklist as quickly as possible and exit the gym.

A simulation cycle begins with an empty gym. Then, one agent is spawned at the entrance every t_i time steps, where t_i is a fixed inter-arrival time given as a parameter to the simulation. When an agent is created, it is assigned a workout routine (e.g.

[1] A *push* workout consists of pushing movements, mainly targeting the chest, frontal deltoids (shoulders), and triceps. Pull muscles are those involved in pulling movements, such as the back musculature, rear deltoids, and biceps. The content of a leg workout is self-evident.

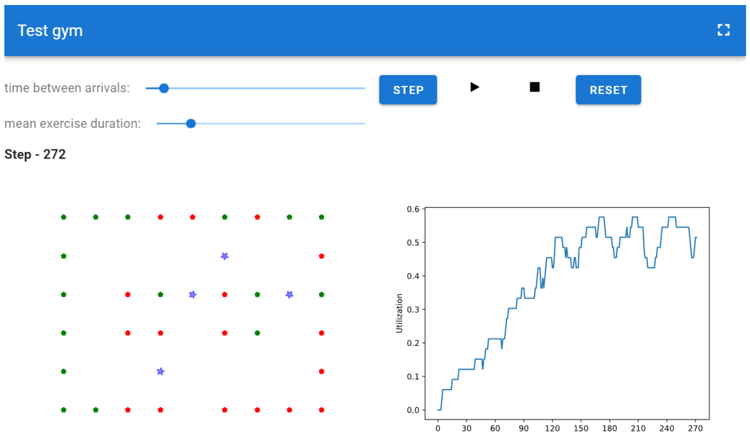


Figure 1. A gym traffic simulation cycle frozen at 272 time steps. On the left we can see the gym environment, generated from a layout template with the entrance at the bottom. Red pentagons are occupied machines, while green ones are free. Agents that are currently walking around in search of a machine are shown with blue stars. On the right, we can see the time progression of one of our gym evaluation metrics (described in a later section).

push) and a corresponding workout plan (e.g. chest, chest, triceps, frontal deltoids). In our experiments we sampled the routine selection from a uniform probability distribution. The workout plans are constructed ad hoc, in keeping with general workout programming practice. The agent also keeps track of the machines it has already used, so it can avoid using the same machine twice in one workout.

Upon entering the gym, an agent starts in state *searching*, where it looks for a free piece of equipment it can use to check any muscle off its training checklist. A path to the nearest free machine is found using the A* algorithm, which the agent begins to follow. If the target machine is no longer free upon arrival, the agent continues in state *searching*. When a free machine is found, the agent enters state *exercising* and occupies the machine for a certain amount of time. In our experiments, we used a deterministic exercise duration of 30 time steps for all exercises. After the time has elapsed, the agent re-enters state *searching* or exits the gym if it has completed its workout.

Layout optimisation. We formalized our problem as a constrained multi-objective optimisation problem, where a layout is constrained by a given layout template. This constraint makes the optimisation problem more tractable, and we can also ensure that no unreachable areas are ever created. We explored the space of possible layouts using genetic algorithms. We used Python's *PyGAD* library, since it supports multi-objective optimisation with the NSGA-II algorithm [5]. This converges to a set of Pareto-optimal solutions, meaning that no other solution can improve on any of the objectives without worsening at least one of the others. This way, we didn't have to combine optimisation objectives into one number with a weighted sum and thus decide the relative importance of each objective.

A chromosome is a list of workout machines, which will be sequentially assigned to equipment locations in the layout template. We took care to "unfold" layout templates in a spatial-structure preserving way, so that k-point crossover preserves as much spatial structure as possible. optimisation begins with a set of random layouts. Then the fitness of each solution is evaluated by running a simulation cycle with the given layout and extracting some metrics (described in the next section). These metrics constitute the compound objective function.

Note that random mutations and crossover can easily lead to layouts on which some of our hand-crafted workout routines cannot be completed; we simply discarded these layouts from the population by setting their fitness to negative infinity. In this way, roughly 15% of solutions were discarded from each generation on average.

We used population sizes around 50. Since this is a complex optimisation problem, we would ideally use even larger ones, but we were limited by the performance of our simulation.

Objective functions. The layout quality metrics which we maximized were:

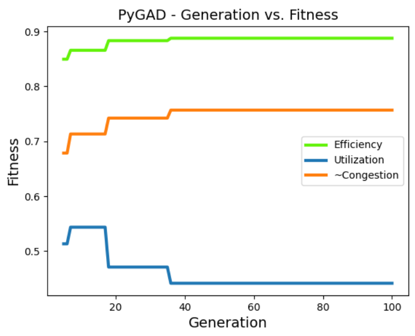
- agent *efficiency* (proportion of time spent exercising) averaged over all agents in the simulation,
- gym *utilization* (proportion of machines in use) averaged over time steps, and

- gym *crowdedness* (proportion of pairs of searching agents that are within 2 cells of each other) averaged over time steps. This value had to be inverted, since PyGAD only supports maximization of the objective function.

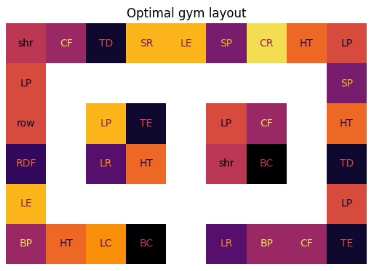
Results

We tested our optimiser on a few small layout templates. First we used a template which has machines around the walls and two 2-by-2 islands of machines in the middle (Figure 2b). With a random mutation rate of 15% and two-point crossover, the optimiser converged at around 40 generations. We used a relatively small iter-arrival time $t_i = 2$ to make our optimisation objectives converge faster (we stopped at 250 time steps).

Note that the utilization criterion (2a, blue) decreases as the other two increase. This represents the trade-off between customer satisfaction and gym revenue, as an efficiently utilized gym is cheaper to set up and maintain, but ends up being more crowded.



(a) Evolution of fitness functions over generations.



(b) A Pareto-optimal gym layout.

Figure 2. optimisation constrained to a 2-island layout template. Machines (on the left) are labeled with name abbreviations, see the `Equipment` class in `gym_model.py` on our GitHub repository [6] for the list of machines we used. See Figure 3 for interpretation of colors.

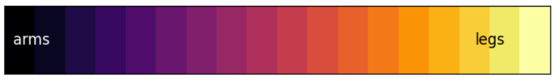
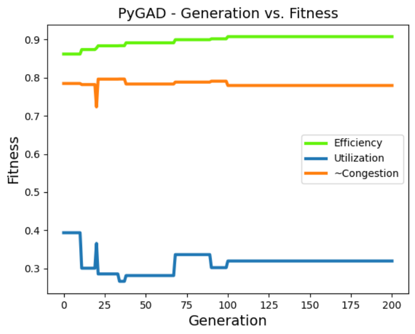
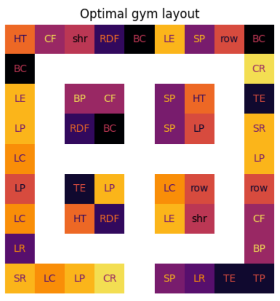


Figure 3. The colormap we used for visualizing a gym layout. Each machine is colored by the muscle it trains, where muscles are mapped onto the color spectrum in (roughly) the same order as they appear in the human body (from fingertips to heels).

Next, we ran the optimiser on a template with 4 islands of machines (Figure 4b). Using the same parameters as before, the optimiser converged after around 110 generations. Due to the increased size of the gym and more complex search space, this experiment took quite a long time to run. Since we ran out of time for code optimisation, we were unable to try larger layout templates.



(a) Evolution of fitness functions over generations.



(b) A Pareto-optimal gym layout.

Figure 4. optimisation constrained to a 4-island layout template.

Discussion

We successfully implemented a gym layout optimisation framework, but our results stand only as a proof of concept. We made many simplifying assumptions which likely render our model too unrealistic to be of practical use. We didn't invest much time into code optimisation, so we couldn't perform a comprehensive analysis of the parameter space.

Future work. Our gym layout model could be made more realistic by dropping the constraints on layout topology and adding support for shared resources (e.g. free weights) and multipurpose equipment. We could also implement more fine-grained agent behaviour, such as parallel use of gym equipment (where one agent is working while the other is resting), random inter-arrival times (e.g. according to a Poisson process) and exercise durations, as well as more realistic routine selection (perhaps with parameters estimated from real-world data).

It would be interesting to let our optimiser come up with a layout from scratch (without hand-crafted templates). Perhaps this could be done with a Metropolis-Hastings state search variation, which has been successfully applied to urban layout optimisation [3, 4]. We could also account the cost of equipment (and its installation) in the optimisation criteria, which would be given by a gym designer as input to the optimiser.

CONTRIBUTIONS. Everyone worked on the report and gym-goer behaviour model. Andrej Jočić worked on model implementation and the layout optimiser. Martin Starič implemented the pathfinding algorithm. Matic Stare made the project presentation.

Bibliography

1. Küçükdeniz T (2022) Gym layout modeling and optimization. *Management Science Letters* 12(3):185–192.
2. Yang S, Li T, Gong X, Peng B, Hu J (2020) A review on crowd simulation and modeling. *Graphical Models* 111:101081.
3. Feng T, Yu LF, Yeung SK, Yin K, Zhou K (2016) Crowd-driven mid-scale layout design. *ACM Trans. Graph.* 35(4):132–1.
4. Mathew CT, Knob PR, Musse SR, Aliaga DG (2019) Urban walkability design using virtual population simulation in *Computer Graphics Forum.* (Wiley Online Library), Vol. 38, pp. 455–469.
5. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 6(2):182–197.
6. (2024) Github repository: gymulator (https://github.com/andrejjocic/gymulator).