Theme-Park Queueing Systems

Alexander V. Frolkin
Frederick D.W. van der Wyck
Stephen Burgess
Merton College, Oxford University
Oxford, England

Advisor: Ulrike Tillmann

Summary

We determine an optimal system for allocating QuickPasses (QPs) to theme-park guests, subject to key criteria that we identify. We recommend a specific system—a way of deciding when a guest asks for a QP whether they should get one, and if so for what time. On the other hand, we warn against some plausible systems that would actually worsen the queueing situation. We also explain why some theme parks use an *unfair* way of allocating QPs, where late guests can fare better than early arrivals.

The keys to our approach are two very different simulations with the same parameters. The *Excel simulation* breaks the day into 10-min intervals, works with groups of people, and is nonrandom. It is fast and allows us to test quickly many different QP allocation systems. The *Perl simulation* breaks the day into 1-min intervals, models individual people, and includes randomness. It is more realistic and flexible. Thus, the simulations are useful in different contexts. The fact that their results are consistent provides a strong safeguard against incorrect results caused by coding errors, a risk in large simulations. In addition, we carry out extensive tests of the stability of our model and the robustness of our recommendation.

We conclude that it is best to allocate lots of QPs for slots early and late in the day, and fewer for the peak demand in the middle.

We also explore modifications to the QP concept, including charging.

The UMAP Journal 25 (3) (2004) 301–318.

Introduction

The Problem

Theme parks have introduced two basic types of new queueing systems:

- In *virtual queue* systems, guests use a pager to register in a queue; it pages when "they" are near the head of the queue and should come to the ride. For example, Lo-Q Plc. has developed such a system, used in Six Flags theme parks in the U.S. [Six flags . . . 2002].
- In *QuickPass* systems, rides can issue guests a QuickPass, allowing them to return to the ride at a specified time, when they can ride with minimal queueing. Examples of this type of system are Disney's FastPass® [Disney Tickets 2004] and Universal Studios Express Pass [Universal Studios n.d.].

Either type of system may or may not be free; Disney's FastPass® is free but guests pay for Lo-Q's pagers.

We focus on the QuickPass system and analyse how to implement it effectively. We conclude with a brief comparison with virtual queue systems.

Criteria for a Good QuickPass system

We take the following as general guidelines in choosing a QuickPass system.

- At no time should more than 50% of a ride's capacity be QP users.
- No ride should have a queue longer than 45 min.
- The average waiting time should be as short as possible.
- Waiting times should be evenly distributed. (It is better that 100 people wait 20 min than that 50 people have no wait but another 50 wait 40 min.)
- The system should seem fair. People arriving later should not get QPs when previous guests have been refused them. Similarly, people arriving later should not be allocated earlier slots than previous guests.
- QPs should not be allocated for more than 4 h in the future. (We assume, based on personal experience, that people stay for only about 5 h.)

Summary of Our Approach

- We collect data and perform calculations to obtain reasonable initial modelling assumptions.

- We construct two computer simulations and modify them with further assumptions. Eventually we find the behaviour agreeable with common sense and consistent between the two simulations.
- We use two very different simulations with the same parameters. The *Excel simulation* breaks the day into 10-min intervals, works with groups of people, and is nonrandom. The *Perl simulation* breaks the day into 1-min intervals, models individual people, and includes randomness. Each approach has its advantages. The fact that the two simulations give similar conclusions provides very powerful evidence for the validity of the conclusions.
- We list systems for allocating QPs that seem likely to work well and test them using the simulations.
- We analyse the results with graphical interpretations and summary statistics.
- We assess the stability of our model under variations in input parameters and the robustness of our recommendation under differing conditions.

The Simulation Process

Initial modelling assumptions

- We have in mind as an example a particular theme park, Thorpe Park, Surrey, UK, which we regard as a typical medium-sized park.
- We model a day running from 8 A.M.–6 P.M. The number of people in the park varies over the day and has a key impact on queue lengths.
- Thorpe Park has 2 million visitors per year, and the park is open for around 200 days per year [Thorpe Park Guide n.d.]. So we assume that 10,000 people visit the park on a typical day. Most arrive late morning or early afternoon and admissions stop well before closing time so that queues can subside.
- People who arrive early typically stay for about 5 h; later arrivals stay until a short time before closing.
- There is one overwhelmingly popular ride, the *Big Ride*, the only ride for which we issue QPs.
- We estimate from personal experience that popular rides take 40 people and leave every 5 min, so the Big Ride has a capacity of 8 people/min.
- We offer QPs for free. Since there is then no harm in taking a QP, whenever they are available, people take them.
- We do not give guests more than one QP at a time.

- For simplicity, we ignore the effect of people going around in groups; all guests behave independently of one another. The effect of grouping would be insignificant, because the size of each group is small compared to the total number of people.

The Perl Simulation

The Perl simulation breaks the day into 1-min intervals, models individual people, and includes randomness. To implement it, we need only the following further very reasonable assumptions:

- We model arrivals as a Poisson process with rate $\lambda(t)$ per minute, where $\lambda(t)$ varies with the time of day:

8 A.M.–11 A.M.:	13	11 A.M.–3 P.M.:	27
3 P.M.–4.30 P.M.:	13	4.30 P.M.–6 P.M.:	0

  We chose these numbers roughly in accord with personal experience with the aim of making the number of arrivals per day about 10,000, consistent with the modeling assumption.

- We model departures as follows:
  - If someone arrives before 12.30 P.M., their length of stay (in hours) is a normally distributed $\mathrm{N}(5, 0.5^2)$ random variable.
  - If someone arrives after 12.30 P.M., their length of stay is a $\mathrm{N}(k, 0.25^2)$ random variable, where k is the time between arrival and 5.30 P.M.. (So late arrivals typically leave at 5.30 P.M. and 97.5% of them leave by 6 P.M.).

The Excel Simulation

The Excel simulation breaks the day into 10-min intervals, works with groups, and uses expected values and no randomness. So it requires more-significant further assumptions:

- We model guests as either on the Big Ride, queueing for the Big Ride, or elsewhere. So we do not worry about the effect of other rides.

- We adopt the following distribution of arrivals every 10 min to implement the arrival behaviour described above:

8 A.M.–11 A.M.:	130	11 A.M.–3 P.M.:	270
3 P.M.–4.30 P.M.:	130	4.30 P.M.–6 P.M.:	0

- We adopt the following departure distribution, in departures per 10 min, to implement the departure behaviour described above:

8 A.M.–12.30 P.M.:	0	12.30 P.M.–3 P.M.:	130
3 P.M.–4.30 P.M.:	270	4.30 P.M.–6 P.M.:	270–750 (incr. linearly)

  The spreadsheet medium was not suited to modelling departures in the more realistic manner of the Perl simulation.

At the end of the day, queues shut at 6 P.M. and people then in the queues get a last ride after 6 P.M.

Baulking

The initial simulations generated implausibly large queues, some more than a day long! We realised that we needed to introduce *baulking*—people being put off by long queues. Also, people with QPs for a ride should be much more easily discouraged from queueing for it than people without. (But if there is a small queue, QP holders *should* want to join it for an early extra ride.) We experimented with linear, exponential, and polynomial baulking models in both simulations, and found that the most realistic behaviour is given by using:

- for non-QP-holders: the inverse quartic model,

$$\mathrm{P}(\text{choosing to queue}) = \left(1 + \frac{qd}{40(1+p)c}\right)^{-4},$$

  where

  q is the queue length (people),

  d is the ride duration (minutes),

  c is the ride capacity (people taken each time the ride runs), and

  p is the relative popularity of the ride (the probability of choosing it).

  Figure 1 shows a graph from this function family. Any guest will join an empty queue, but only one-sixteenth are prepared to wait 40 min. The adjustment factor $(1 + p)$ makes people more likely to persevere for more popular rides.

- for QP-holders: the linear baulking model,

$$\mathrm{P}(\text{choosing to queue}) = \max\left(0,\ 1 - \frac{qd}{15c}\right).$$

  An example graph is shown in **Figure 2**. Here, any guest will join an empty queue but none are prepared to wait for more than 15 min, irrespective of the popularity of the ride. (After all, they have a QP to come back later.)

Finally, we found a problem in the Perl simulation. All rides in the simulation ran for 5 min and were "in-sync." The result was an implausible 5-min-periodic behaviour in the Big Ride queue. Removing the synchronicity by staggering the ride departures (some rides at 8:01, 8:06, 8:11, . . . ; others at 8:02, 8:07, 8:12, . . .) eliminated this problem.

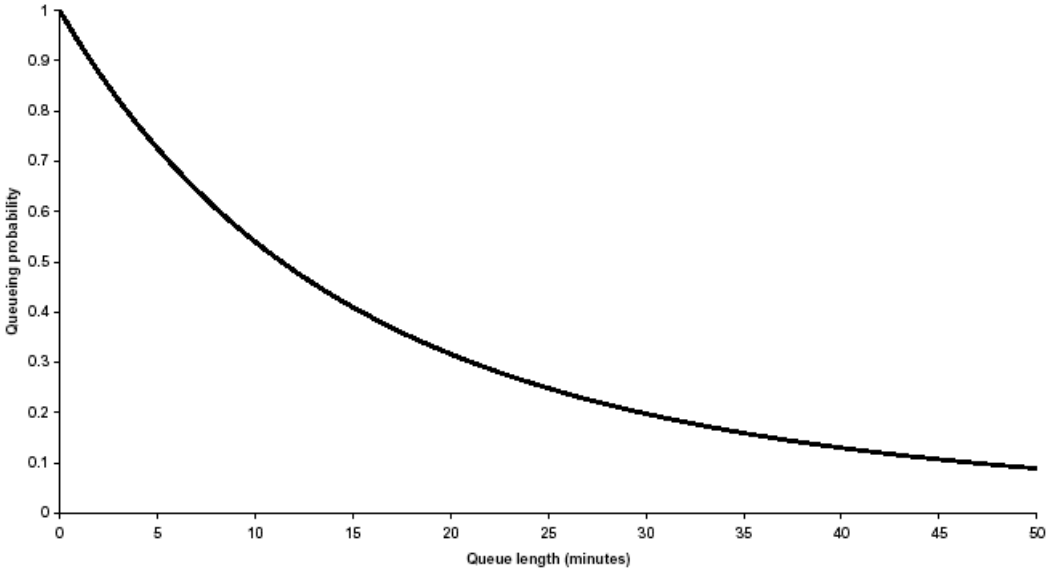

Figure 1. An inverse quartic baulking function.

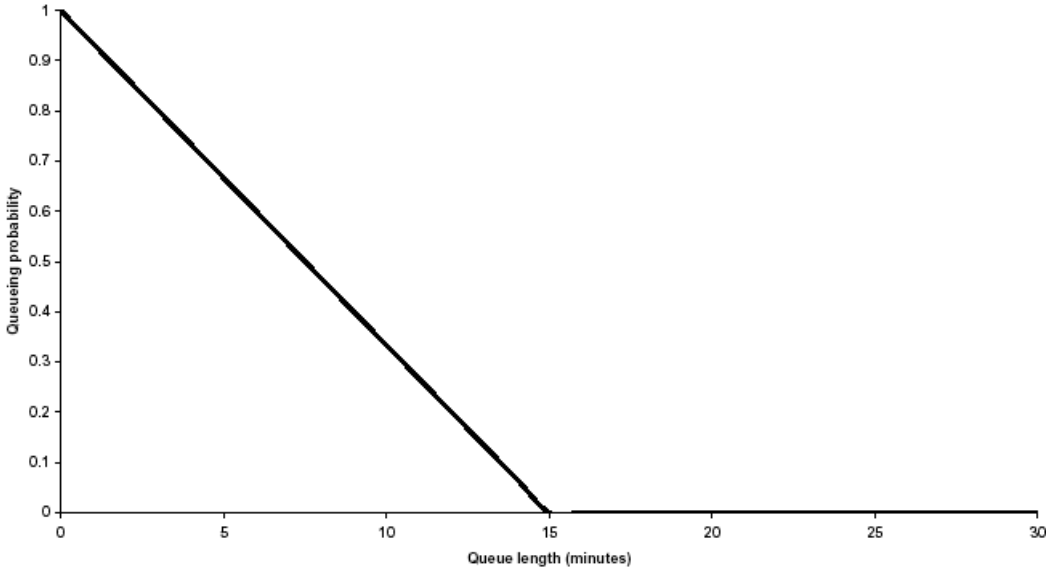

Figure 2. A linear baulking function.

Allocation Systems

A QP system can be represented by an *allocation matrix*, a 10×10 matrix indicating the maximum number of QPs that can be issued by the end of a given hour for a particular future hour. (At the detail level of our simulations, a finer-grained system would make no difference.)

For example, the matrix in **Figure 3** indicates that by the end of the 08:00–09:00 hour, *at most* 100 QPs are issued for slots starting during the 11:00–12:00 hour and at most 50 for slots starting during the 12:00–13:00 hour. We always give out the earliest slots available at a given time, so that the system is seen to be fair.

	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00
08:00	0	0	100	50	0	0	0	0	0	
09:00		0	100	50	0	0	0	0	0	
10:00			100	50	0	0	0	0	0	
11:00				50	0	0	0	0	0	
12:00					0	0	0	0	0	
13:00						0	0	0	0	
14:00							0	0	0	
15:00								0	0	
16:00									0	
17:00										

Figure 3. Sample allocation matrix.

For a different example, the matrix in **Figure 4** indicates that by the end of the 08:00–09:00 hour, at most 100 QPs are issued for 13:00 slots, at most 150 for 14:00 slots, and so on. The values in a column are cumulative; so if 100 people take QPs between 08:00 and 09:00, only $150 - 100 = 50$ remain for people arriving between 09:00 and 10:00.

	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00
08:00	0	0	0	0	100	0	0	0	0	
09:00		0	0	0	150	0	0	0	0	
10:00			0	0	200	0	0	0	0	
11:00				0	250	0	0	0	0	
12:00					300	0	0	0	0	
13:00						0	0	0	0	
14:00							0	0	0	
15:00								0	0	
16:00									0	
17:00										

Figure 4. Another example of an allocation matrix.

Notice some features of the matrix:

- The lower triangular entries are irrelevant—we cannot issue QPs for the past, and we allocate them only for at least an hour in the future.
- The columns are nondecreasing because the entries are cumulative.

- If a column is strictly-increasing, the system is not *fair*. In **Figure 4**, if 120 people want QPs between 08:00 and 09:00, the last 20 will be refused them. But there will still be 50 available for people arriving between 09:00 and 10:00—later. Similarly, consider the matrix in **Figure 5**. Now if 120 people want QPs between 08:00 and 09:00, the last 20 get 14:00 slots, while the first 50 people arriving during 09:00–10:00— later–get 13:00 slots.

	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00
08:00		0	0	0	0	100	20	0	0	0
09:00			0	0	0	150	20	0	0	0
10:00				0	0	200	20	0	0	0
11:00					0	250	20	0	0	0
12:00						300	20	0	0	0
13:00							20	0	0	0
14:00								0	0	0
15:00									0	0
16:00										0
17:00										

Figure 5. A third allocation matrix.

How the Perl Simulation Works

The Perl simulation works in discrete time steps (usually 1-min long). Each person is assigned one of the *states*: Walking around, Queueing, Queueing with QuickPass, Riding, or Gone home.

At each time step:

- New arrivals are added to the people in the park; departures have their state set permanently to "Gone home."
- For each person in the park:
  - If the person is walking or has just entered, they check if they have overstayed their staying time, and if so, leave. If they have a QP for the current time, they move to the front of the queue for that ride; if not, they carry on walking with probability 0.8. (This is based on a geometric distribution with mean 5 min.) Otherwise, they choose a ride based on the rides' popularities and proceed as follows.
    - If the ride does not have QPs, they decide whether to queue or to carry on walking, based on the queue length, using the inverse quartic baulking model.
    - If the ride does have QPs, they try to get a QP if possible (i.e., if available and they do not already have a QP).
      - If they obtain a QP, they decide whether to queue for the ride or carry on walking, using the linear baulking model.

  - If they do not obtain a QP, they decide whether to queue or carry on walking, using the inverse quartic baulking model.
- If the person is queueing for a ride, they check if they have a QP for the current time, in which case they will leave their current queue and move to the front of the queue for their QP ride.

- For each ride, carry out the following:
  - If it is time for the ride to take more people, all people on the ride are taken off (and put into the walking state), and the maximum number of people from the front of the queue (i.e., the ride's capacity, or the number of people queueing, if that is less) are put on the ride.

The Perl simulation allows the QP system to be easily modified, by simply changing the allocation matrix entries. In addition, all the following parameters can easily be adjusted:

- length of a time slot in the QP allocation matrix;
- length of a day;
- the function $\lambda(t)$ giving the mean number of people entering at time t;
- the function $\mu(t)$ giving the mean duration of a stay for a person entering at time t;
- the function $\sigma(t)$ giving the standard deviation of stay durations for people entering at time t;
- the probability that a walking person carries on walking at the next time step;
- the number of rides,
- for each ride, the capacity, the duration, the popularity, and the start-time delay (e.g., whether the ride runs at $0, 5, 10, \ldots$ or at $2, 7, 12, \ldots$ time units).

How the Excel Simulation Works

The Excel simulation tabulates the number of people in different places at 10-min intervals. At each time, it knows:

- the number of people entering, the number of people leaving, and the total number of people in the park;
- the number of people in the queue for the Big Ride and hence its length;
- the number of people on the Big Ride;
- the number of people elsewhere without a QP for the Big Ride;

- the number of people elsewhere with a QP for the Big Ride; and
- the number of QPs issued for this slot and hence the remaining capacity on the Big Ride for this slot.

The number of people entering and leaving at each step is specified in advance. The main calculation at each step involves the following:

- Add people coming off the Big Ride from the previous step to "elsewhere."
- Check how many QPs can be issued now for slots later in the day and give them out to anybody elsewhere without a QP who is interested in the Big Ride (fixed interest level, no baulking); change their status to having a QP.
- Check how many people are in the queue for the Big Ride.
- Deduce via the appropriate baulking models what fraction of people elsewhere without a QP are willing to join the queue for the Big Ride, and what fraction with a QP are nonetheless willing to queue for an early go on the Big Ride. Add these people to the queue.
- Remove from the queue people taking the Big Ride without QPs.

The QP system can be easily modified (by changing the allocation matrix entries) and other parameters can be adjusted, but there is less flexibility than in the Perl simulation.

Results and Interpretation

Comparing the Two Simulations

We first ran the two simulations with no QP allocation (system QP1), to calibrate the simulations; in particular, this involved setting the Big Ride popularity parameter appropriately. The Perl simulation gave generally longer wait times, but the overall qualitative behaviour of the simulations was the same.

Figure 6 shows the queue profile from one run of the Excel simulation, and **Figure 7** shows a waiting frequency barchart, averaged over 3 runs of the simulation. The Perl simulation generates similar results, but people tend to wait a little longer. In both simulations, the queue builds up over the day. It is long over several hours, so many people get a long wait.

After the initial calibration, we did *not* modify the parameters; nonetheless, the simulations continued to give similar results, so we felt justified in using only the Excel simulation for testing allocation strategies.

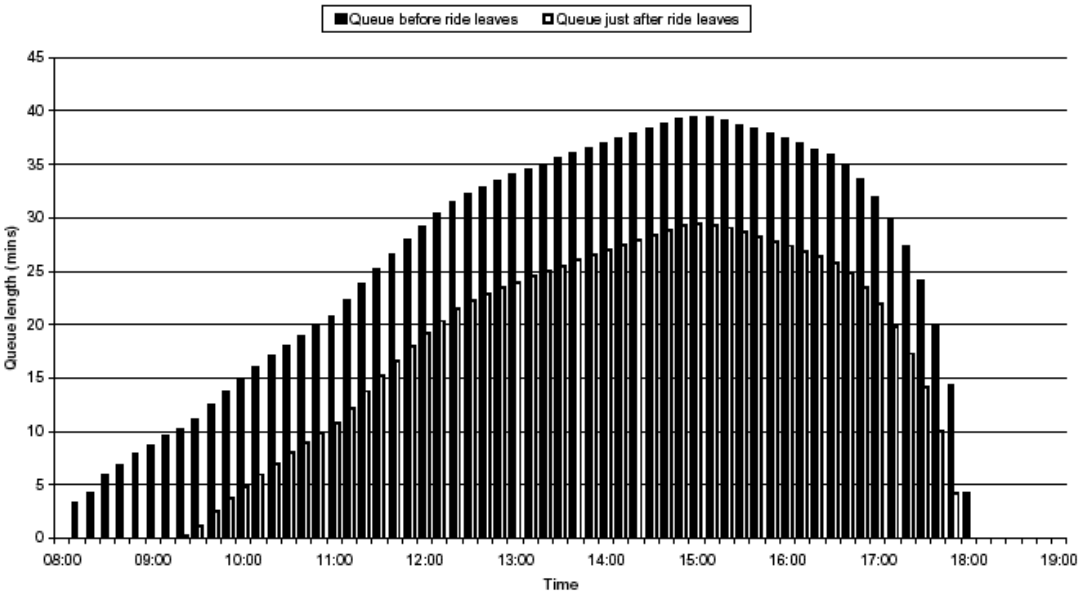

Figure 6. Results of run of Excel simulation for no QuickPasses: queue profile.

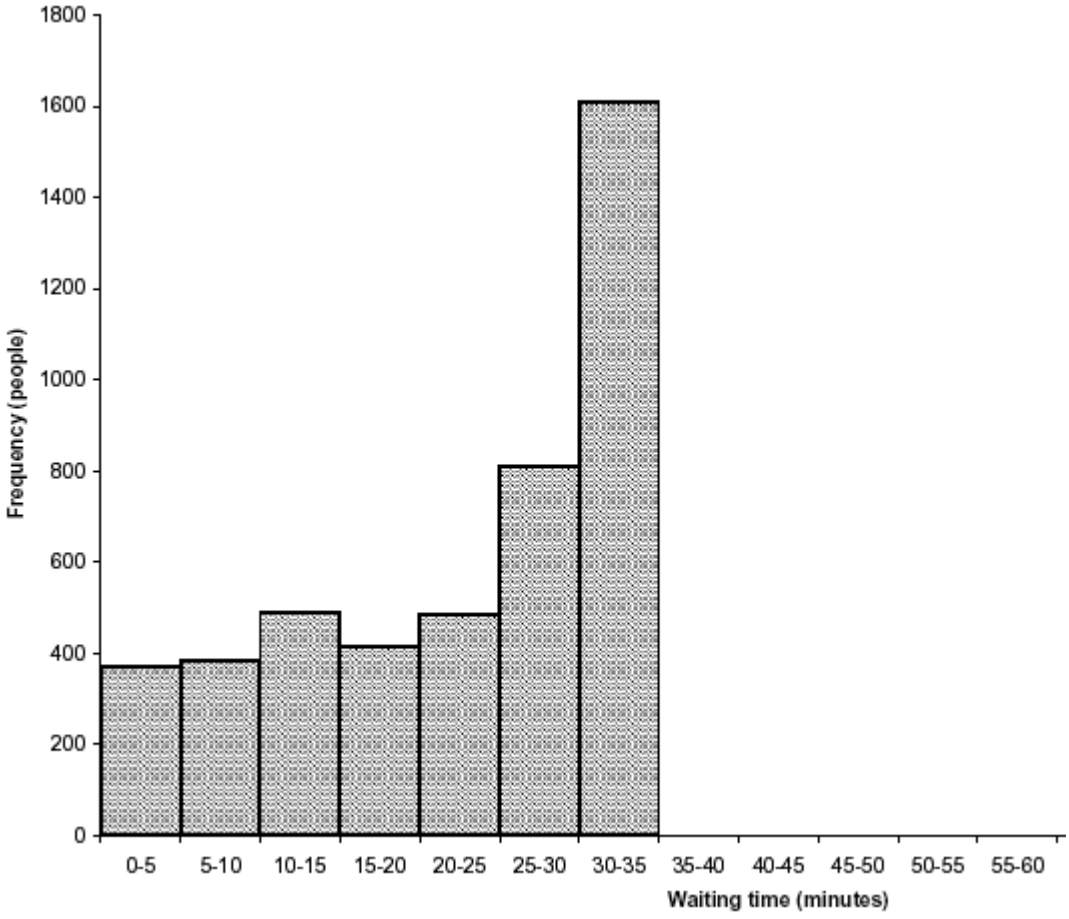

Figure 7. Results of run of Excel simulation for no QuickPasses: waiting frequency.

Key Results

We tried several models:

- System QP2 allocates the full capacity of the ride to QPs, giving them out as soon as people want them. The queue for the Big Ride becomes huge, and many people have to wait a *very* long time, because no capacity remains for non QP-holders. This is a very bad way of allocating QPs.
- System QP3 allocates only half the capacity of the ride in QPs but still gives them out as soon as people want them. Now the queue stays short while QPs are issued (because people take a QP and tend to go away); but as soon as the QPs run out, the queue grows large, because the ride's capacity has been halved. Many people get a long wait. Things are worse than with no QPs, because the system shortens queues early (when they are short anyway), then reduces capacity at the busy time.
- System QP4 allocates QPs only up to 3 h in the future. This is better. A lot of people have a short wait and not many have an excessive one.
- System QP5 uses the allocation matrix in **Figure 8** to issue QPs. It gives out many QPs for slots early in the morning and late afternoon, in a way carefully adapted to the arrival distribution. This system gives the best results, shown in **Figures 9** and **10**. The queue is roughly constant for much of the day, many people have a short wait (the QP users!), and nobody waits more than 40 min.

	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00
08:00		240	120	60	30	0	0	0	0	0
09:00			240	120	60	30	0	0	0	0
10:00				240	90	60	0	0	0	0
11:00					120	90	0	0	0	0
12:00						120	0	0	0	0
13:00							0	0	30	0
14:00								0	60	60
15:00									120	120
16:00										240
17:00										

Figure 8. Allocation matrix for scheme QP5.

Recommendation

We recommend System QP5; it gives a lot of people a queue-free ride, while otherwise leaving the queue situation not much worse than without QPs. This is really the best we can hope for, because QPs cut normal capacity and so tend to worsen the normal queueing situation. Here is how QP5 performs on our criteria:

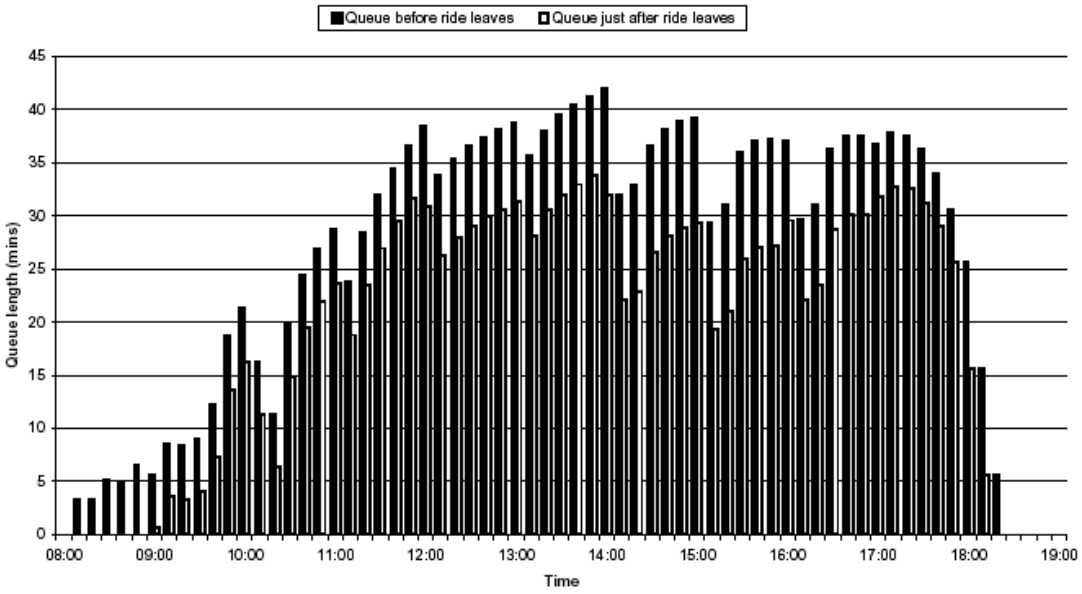

Figure 9. QP5: queue profile.

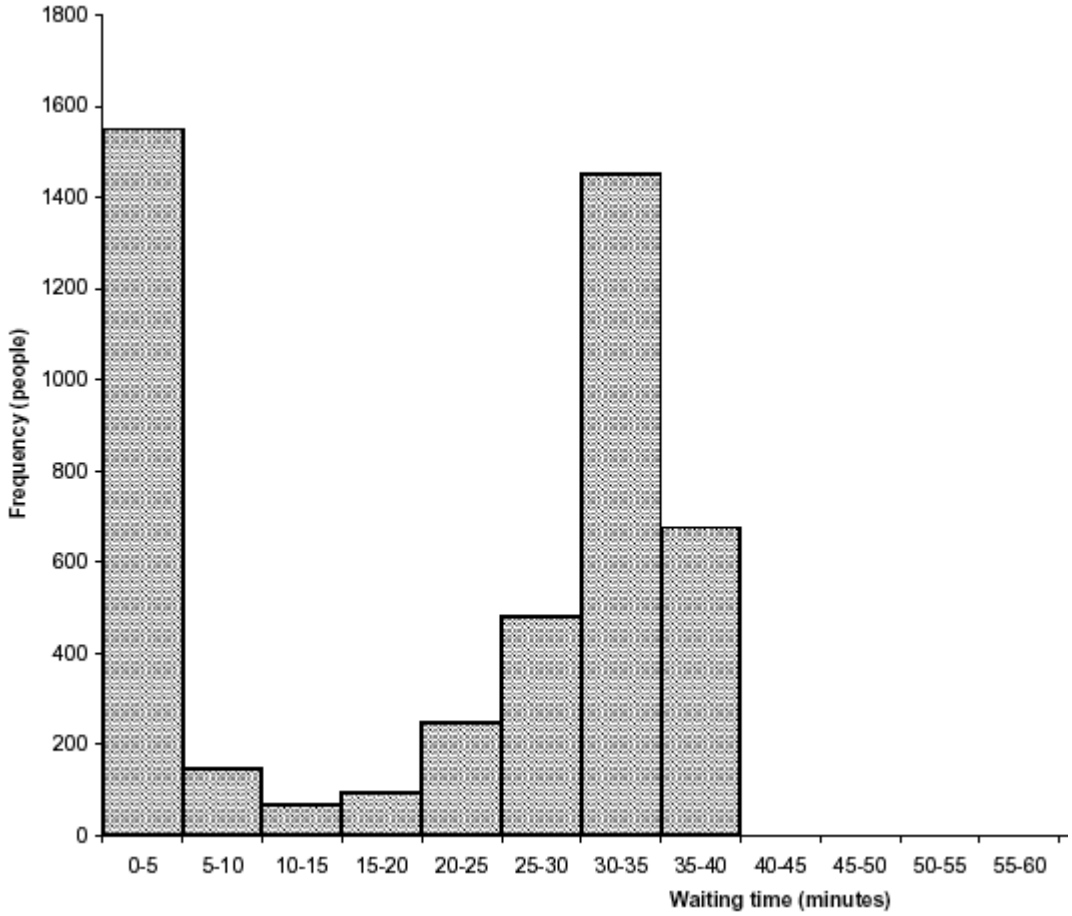

Figure 10. QP5: waiting frequency.

- No more than 50% of the ride is ever filled with QP users. (GOOD)
- The Big Ride never has a queue longer than 45 min. (GOOD)
- The average waiting time is similar to without QPs. (OK)
- Waiting times are not very evenly distributed, although better than in some systems. (POOR)
- The system is not fair. (POOR)
- QPs are not allocated for slots far in the future. (GOOD)

Assessment

Stability

Overview

A good model should be stable: A small variation in the input parameters should cause a small change in the results. In addition, the direction of the change should usually be consistent with common sense.

We varied the following parameters to see how much they changed results and whether the change matched what we expected.

- relative popularity of the rides: We increased the popularity of the less popular rides (but kept them less popular than the Big Ride). This had the expected effect of decreasing the queue lengths.
- total number of rides: We removed 10 of the less popular rides (out of 20 rides). This had the expected effect of increasing the queue lengths by 50%.
- probability of continuing to walk around: We increased it from 0.8 to 0.95. This had the effect of decreasing the queue length, because people were now more likely to carry on walking, so fewer people queued for rides.
- baulking models: We changed the fourth power in the inverse quartic model to the sixth power and changed the 15-min cutoff in the linear model to 20-min. The maximum queue length decreased from 50 min to 35 min.
- arrival rate: We changed the arrival rate for the first hour from 13 people/min to 50 people/min. This made the queue length grow much more steeply at the start of the day, as expected.

Conclusion

The results of the tests above are all favourable. The model is stable and responds in the expected way to changes.

As a further check, we calculated the standard deviation of the mean waiting times from 40 runs of the Perl simulation of systems QP1 and QP5. These were 0.46 min and 0.39 min, respectively, which is again favourable, as it indicates little change between the runs.

Robustness

We checked the impact of big changes in the distribution of arrivals at the park (caused, for example, by weather conditions).

We modified the arrival distribution to reflect a typical weekend day, or a day with very good weather. Queues lengths increased significantly in both cases, but were only slightly worse with QPs, which is good.

Next, we modified the arrival distribution to reflect a day with bad weather in the morning—most arrivals after lunch. Again, the QP system coped with the change.

So our recommendation is quite robust.

Improvements and Extensions

In this section we discuss possible improvements to our model and extensions to the QP system.

Guests with Memory

Our model assumes that a guest's behaviour at a given time is independent of their previous behaviour. In fact, people don't go on the same ride time and again. We could modify the Perl simulation to account for this: the more times a person takes a ride, the less likely they are to take it again (except perhaps after their first go, which could actually encourage them). The Excel simulation's structure makes it impossible to implement this change.

Finer-Grained QuickPass Allocation

A larger QP allocation matrix could be used; this would be easy to implement, but such a level of detail would be incompatible with the limited accuracy of the simulations.

Charging for QuickPasses

Charging for QPs would increase revenue to the park and (by reducing QP uptake) leave more capacity on the Big Ride for the normal queue.

We model the extent to which cost deters guests. If we make the probability of a guest paying for a QP constant, the net effect is the same as issuing fewer QPs.

A more plausible model has long queues making people more willing to pay; **Figure 11** gives an example graph for a function from the family

$$\mathrm{P}(\text{pay for QuickPass}) = \min\left[1,\ \frac{1}{k}\log_{10}\left(1+\frac{qd}{2c}\right)\right],$$

where

q is the queue length (people),

d is the ride duration (minutes),

c is the ride capacity (people taken each time the ride runs), and

k is an arbitrary constant, initially set to 2.

In the morning, fewer QPs sell; later, when there are long queues, they all sell.

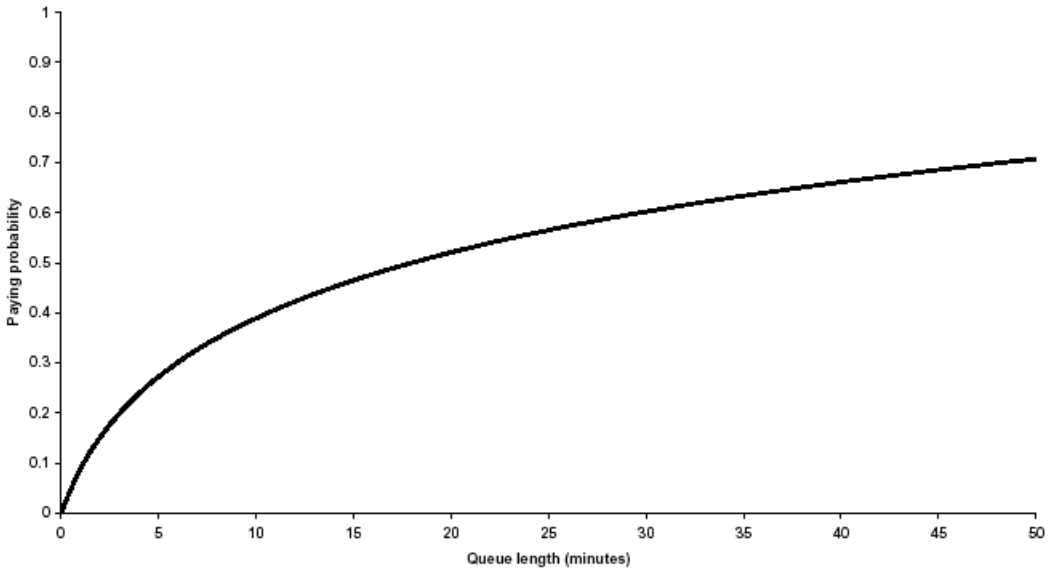

Figure 11. Logarithmic cost-reaction model: probability of paying for a QuickPass vs. queue length.

A bigger effect could be achieved by charging more, corresponding to increasing k. In principle, we could determine how k varies with the cost of a QP; then running the simulation with different values of k and recording the number of QPs sold, one could maximise revenue. **Figure 12** shows the response of sales to price, *assuming that a price of £p per QP gives $k = p$.*

The graph's slope is sufficiently shallow that the highest price gives the highest revenue. However, the assumption that a price of £p gives $k = p$ is unrealistic; a subtler model is needed.

At any rate, charging certainly improves the queueing situation as well as raising money. On the other hand, it reduces guest satisfaction, a major drawback.

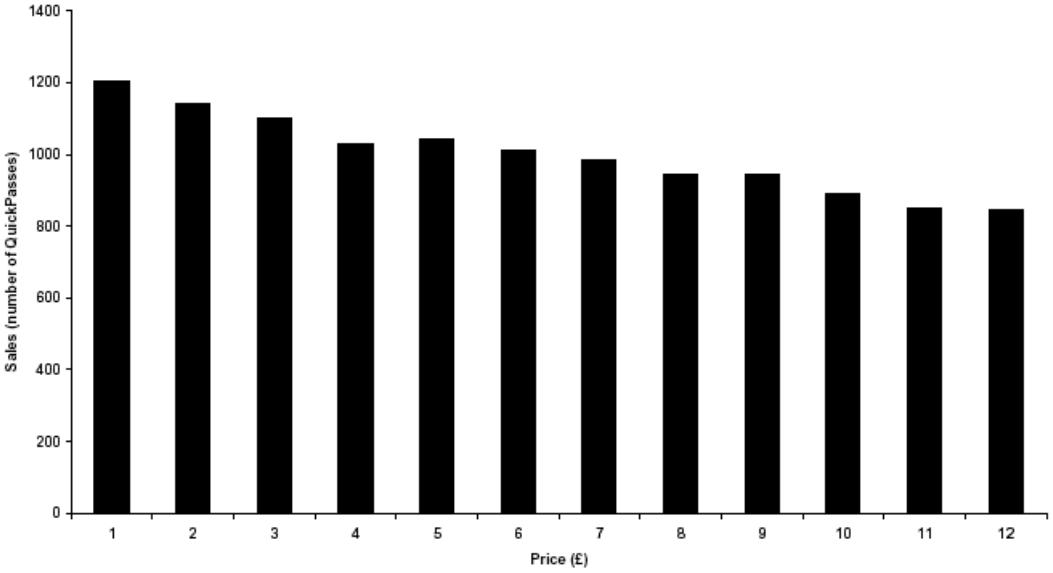

Figure 12. Sales of QuickPasses vs. price.

More than One Ride Offers QuickPasses

We modified the Perl simulation to test what happens if QPs are issued for more than one ride (but never more than one at a time per guest); the behaviour is much the same as for QPs for only one ride. So we recommend the use of QPs for all popular rides.

More than One QuickPass at a Time

Issuing people more than one QP at a time would require a sophisticated allocation system to avoid clashes, but the potential benefit is to move the system toward full scheduling and efficiency. We did not have time to test this idea.

Conclusion

Review of Our Approach

Our two different simulations provide a strong safeguard against incorrect results. Extensive testing of the stability of the model and the robustness of our recommendation provides further support for our conclusions.

On the other hand, our model has limitations. Treating guests as having a memory and using a finer-grained QP allocation matrix are important possible improvements.

Key Conclusions

We recommend a QP allocation system that allows many people a queueing-free ride, without causing the normal queue to grow too much.

However, early guests may be given QPs for a late slot, or even refused a QP, while later arrivals fare better. Our investigations show that unfairness is crucial to a good QP system.

References

Disney Tickets. 2004. `http://www.disney-ticketshop.com/` .

Grimmett, G., and D. Stirzaker. 2001. *Probability and Random Processes*. Oxford, UK: Oxford University Press.

Grimmett, G., and D. Welsh. 2001. *Probability: An Introduction*. Oxford, UK: Oxford University Press.

Six Flags brings Lo-Q technology to nine parks for 2002. 2002. `http://www.ultimaterollercoaster.com/news/stories/021402_03.shtml` .

Thorpe Park Guide. `http://www.thorpeparkguide.com/` .

Universal Studios. n.d. Universal Studios discount tickets. `http://universalstudios-orlandoflorida.com/universalexpress.htm`.