Bolletott's balanced, multiple item handling, order receiving train station

Capable of handling multiple items, keeping each of the items in the buffer chests balanced, while avoiding that the chests fill up and start blocking other items. It also manages input signals, allowing the Factory itself to signal when it needs more items, giving a lot of flexibility inside the factory. It also avoids a deadlock when all chests are perfectly balanced.

This will be my attempt to explain how my design for a train station that unloads trains carrying multiple items while keeping the buffer chests balanced works.

I will be explaining this very thoroughly. Too thoroughly, many will probably say, and not read this at all. I still choose to do that, thinking that in the end, it's more efficient. To me, it's better to have people read for twenty minutes, but then (hopefully) understand everything when they're done reading, rather than give a very short description that takes two minutes to read, but then have them sit for an hour trying to keep track of all the moving pieces themselves.
I will not explain in detail every single challenge I had to solve, though. That would take forever. I'll try to limit myself to the aspects of the story that are relevant to the final(?) design.

A few disclaimers: I'm on my first game. I may be wrong about some aspects. Some things probably could have been done in an easier way. Please let me know if you spot any such case, or if, when you've read the whole thing, you see a way to get the same result with a some adjustments that makes a neater, more compact design, for instance.
Also, English is a second language, so while I think my vocabulary is sufficiently good, there are probably a lot of cultural norms that escape me, that will make me sound "odd". Sorry about that.

The basic idea

So, the idea at first was that I wanted to have a train station that could unload my trains in an efficient way. Naturally, when I made my first train station, it would quickly back up and become horribly uneven. At first, the stations that unloaded weren't even balanced, but I'll simply skip that part, to try to keep this document as short as I feel is possible while still keeping it easy to follow.

At this point in the game, I had researched the circuit network in the tech tree, but hadn't used it yet. My "problem" was that because I realised the possibilities using these simple (if a bit confusing) tools are practically endless, I couldn't settle for something I thought less efficient, and probable that I would phase out eventually anyway, so I decided to see if I could use them to make the station balance itself.

Ultimately, I wanted to design a station that could handle multiple items, and would "take orders" from the Factory and deliver the items only when needed, while keeping the buffer chests balanced, and not filling them up completely, blocking some items from arriving at the station.

But to make this as easy to understand as possible, I'll explain it all to the best of my ability, starting (almost) at the beginning.

Getting the buffer chests balanced

After some experimenting with the circuit network and the combinators for myself, I looked online and quickly found Madzuri's design for balancing chests. To me, it was painfully confusing at first, even though the design is simple enough, but eventually I got it working.

Check out the brilliant design here:
https://www.reddit.com/r/factorio/comments/4e03g2/madzuris_smart_loading_train_station_guide/

However, as I wanted to send multiple items to the Factory, but didn't want to make one train station for each item (to me, that's just horribly space inefficient), I had to come up with a way to keep multiple items balanced in the same chest.

Balancing multiple items in the same chest

Madzuri's design relies on the fact that positive and negative numbers are summed inside the filter inserters. Now, I don't remember why, but as far as I can understand now, several hours later, at this point I kinda make a mistake, because I decide that I have to add one decider combinator for each line of inserters in my station to handle the items in the chests, because the deciders can use the "each" signal.

This probably happened in my confusion at first, but in the end, that helped me, if I understand all this correctly myself.

So, here's a simple test/demonstration of how it works so far:

Picture 1

As you can see, the contents of the chests are sent to the decider, which sends the signals to the filter inserters. Utterly unnecessary, I know, but this is what I had at the time.

This was great, but now I had to figure out a way to make sure that I didn't get too many of one item in the buffer chest, blocking others from getting to the factory at all.

Adding an average item cap

The solution was fairly simple. I added one arithmetic that would output the average of *each* item in the chests, and send it to a decider that would say "if there are more than *n* items on average, stop adding that item to the chests".

It looked like this:

Picture 2

I forgot to highlight the decider, but the arithmetic divides the sum of *each* item by the number of chests, sends it to the decider, which says [*each* > 500] -> [1*each*], if that's understandable.

Because the chests are balanced, this means that when all chests have gotten 500 of *each* item, the decider outputs 1 of that item, which is sent to the filter inserters, blacklisting/blocking that item.

This served me perfectly for a long time, but then came the point where the wagons contained more than five different items because I had several stations on the same track. Some were loading, some were unloading, but not all of them would completely empty the wagon every time, so to keep the inserters from taking items that were headed for other destinations, I added a constant to the station that blocked/blacklisted the unwanted items. You can see two of them in picture 2. This was a good idea, but because the filter inserters (FIs from now on) can only blacklist five items at a time, that system was quickly overloaded, meaning the FIs couldn't blacklist any more items, causing them to take the items when I didn't want them to.

At first the solution was to move some production to another train line, as it was slow on this one anyway, but this wasn't satisfying, as something was always happening after items were unloaded into the factory. Items were backing up all the time. I had to think of something new.

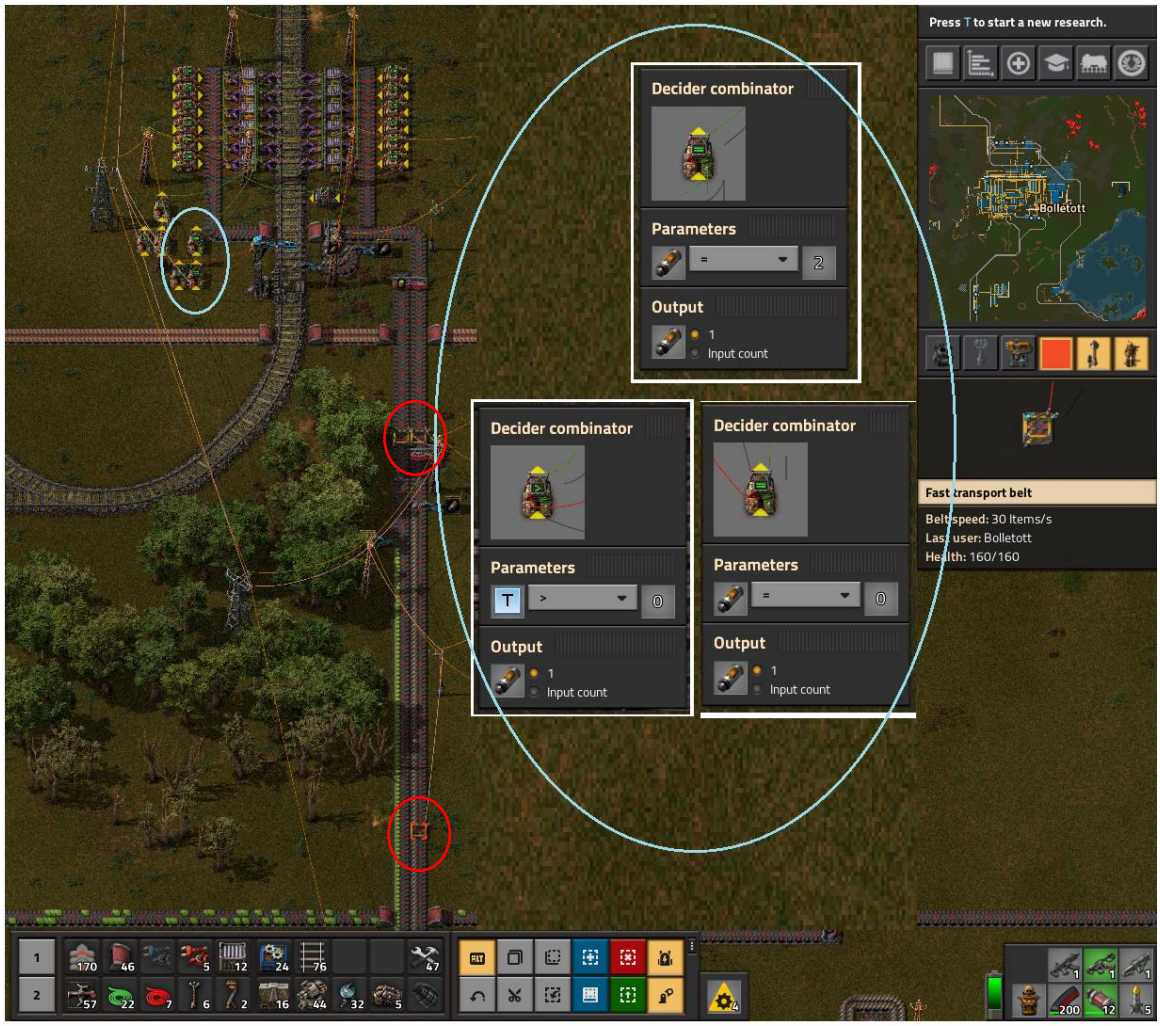
The “order” system

This is the point where I won’t bore you with all the details on this process. I probably wouldn’t remember half of it anyway, and besides, this document is too long already.

So, I came up with a system that would send a signal to block items from being sent into the factory until there was enough room on the belts to accommodate them. Or, as I like to think of it: When the Factory realises that it needs more of an item to keep production going, it will order more from the train station, by not blocking the items anymore.

However, this “double denial”: “not blocking” was confusing to me, so eventually, I redesigned it so that it would actually send a signal (place an order) to the FIs when it needed more.

Here’s how it looks:



Picture 3

In the two red circles, you’ll see belts being monitored. If there are items there, they’ll send the number to the red network, which is connected to the 2x3 combinators in the top left of the picture.

The belt is being monitored in two places to make sure there is enough room on the belt to take all the items when they are unloaded from the chests. I'll not explain that any further unless anyone asks.

In the blue circles are the same combinators. The bottom one on the left is there to avoid the FIs unloading the train for one tick, before it realises that it is not getting an actual order. **This happens because to make it so that the FIs work when they *get* an order (as opposed to them working when the signal isn't there), I switched the innermost FIs to whitelist instead.** That is key. The decider on the bottom left does this by taking a signal from the train station itself via a red cable. Make sure the train station is set to "Read stopped train". It then says "If I'm getting a signal on the "T" channel, I'll order more Jet Fuel", or: [T > 0] -> [1jetfuel], as you can see.

The one on the bottom right gets the signal from the red network and says "if there's no jet fuel in this network, (being sent from the belts) I'll order more, by outputting one jet fuel.

The one on the top says "If I'm getting an order, that's good, but I also need a train to be here before I'll put the order through". So, when the train arrives, the bottom left AND the bottom right both output jet fuel. That makes the one on top happy, and it actually places the order, putting the inner FIs to work.

In this design, the balancing was supposed to be done by the outer FIs in the same way as explained above, using the useless deciders outside the belts, but although I'm happy because everything seems to be working as intended (after quite a lot of trial and error, of course), I realise that this design doesn't balance the buffer chests. Instead, it keeps all the buffer chests empty. I decide that it's better to keep the buffer chests full so that they can start delivering immediately when the order comes from the factory, rather than wait for the train.

To fix that, I simply switched it around so that the outermost FIs are the ones whitelisting, and the innermost blacklist items if the average is bigger than what the decider allows (ref. the part on "Adding an average item cap).

When I did that, there was no need for the bottom left decider in the blue circles in picture 3.

(I don't have a picture of this sadly. This version was the one that has been developed into this final(?) version.)

For a little while, I'm perfectly happy, but then another problem came up. A problem that had actually bothered me so many times during all this, but I've not mentioned until now: All the chests are perfectly balanced, leading to a deadlock.

Preventing a deadlock

I did a lot of research into this, but couldn't find a satisfactory answer, so I posted my first question in the forums. I'll try to write in a way that makes it possible to follow this without reading my post in the forums, but I'd recommend that you do.

Before you do, though: a few comments:

- Skip my first question and the two kind answers. As it turned out, I didn't really understand what I wanted to know before I started typing out the second post I did, further down.
- The order of some things I did may be different from what I've said here. Honestly, I can't remember it all perfectly, but it doesn't matter.

- (Yes, I was missing something, as I write at one point, regarding the unnecessary deciders. I didn't understand it just yesterday. This is not important, just a note.)

Here's the post: https://forums.factorio.com/viewtopic.php?f=18&t=83549

As mentioned, I realised I was wondering something else as I was typing out my second post. Therefore, a new heading is required:

Making a balanced, multiple item handling, order receiving train station

This is it, guys!

So, here's a key point from my post: [The reason I need to unload the buffer chests evenly is] because the FIs between the wagon and the buffer chests are always unloading evenly because they get access to the items at exactly the same time (as mmmPI wrote), but when the items are unloaded from the buffer chests, the belt is filling up from further down the line, eventually catching up with the inserters furthest to the back of the line. This means that a few more items are unloaded from the chests in the back (green circle), than from the chests at the front (red circle) every time the train arrives.
(A small note: The items from the inserters at the back also partially block the inserters further down the line, making them unload even slower, thus making the chests even more unbalanced).

As you might have read in my forum post, this turned out to be a huge challenge for me. As I write:

To combat this, I wanted the FIs to unload from the chest that has more items than average [first].
...
So, the decider with an orange circle, marked "1" is currently outputting 1 sulfur, because the factory needs more sulfur, as described above. However, no sulfur is sent to the factory, and the reason for that is that the arithmetic in the shameful, yellow circle is doing its job, outputting the negative average of sulfur, in an attempt to balance these chests, and that is my undoing. No sulfur is sent because the FIs are getting a much bigger negative number from our friend the yellow arithmetic decider, (I meant arithmetic combinator, of course) making this a perfect example of a "catch 22", if I'm not mistaken.
...
So I guess my question ultimately is this: How can I balance the unloading of the chests, while at the same time having the factory "order" items by whitelisting what it needs? Is it possible without sorting everything first (taking up a lot of space), or do I have to abandon this challenge?

As I was waiting, hoping for an answer from my kind new friend mmmPI, I made a new, separate save, and went to work, trying to figure it out for myself, and to my joy, I did it!

This is what it looks like (This is currently only handling one item, but could handle more):

Picture 4

I have used the same colour circles as the ones in my forum post.

As you can see, I've added another row of deciders, and this is where the redundant deciders I added by mistake earlier, finally come into their own.

The combinators in the blue circle are the ones managing the average item cap. They send their blocking signal to the inner FIs as before. I have also added a constant (pink circle) that blocks items not wanted in this station, but those are the only similarities to the earlier versions.

(By the way, I should mention that I have come to the conclusion that no station can handle more than five items, because that's how many a FI can blacklist at the same time. Add more, and items will be sent into the buffer chests that weren't supposed to be there, because the avg.cap-handlers in the blue circle can't block any more.)

Now is a good time to answer the question I asked in the second post in my forum post:

"How can I balance the unloading of the chests…" etc.

It *is* possible. The way I did it was that I added that second row of deciders (effectively working as AND gates) inside the first row, which has been moved out, that said "I need two things before I'll place the orders to my FI. First, I need the actual order. Second, I need to know that the chest my

inserter is pulling from has more than the average number of items in it. If both these conditions are true, I'll put my inserter to work."

The negative average is sent to the outermost deciders, that I kept from the last version. They are asking "is the sum of the negative average of *each* item and the number of *each* item in my chest a positive number or 0 (meaning the chest has more than or the same amount as the average is)? If so, go to work, inserter (it outputs a single of *each*) . Or: [*each* >= 0] -> [1*each*].

The innermost decider isn't happy yet. However, the Factory has sent an order to the decider in the orange circle (the one crossed out isn't currently in use, but will be in the future), and *it* has forwarded that request to <u>all</u> the inner deciders.

This means that all of them are getting an order, but only some of them have gotten the go ahead from their outer decider (because, as stated before, "their" chest has more items than the average). The ones that do, though, leap into life.

The FIs that unload the wagons aren't balanced, but they all get access to the items at the same time, so indirectly, they are. If for some reason it gets a little bit off, now the combination of all these combinators keep them balanced, while not filling up the chests.

If you add items that have smaller number stacks, you just have to reduce the number in the decider in the blue circle, to allow room in the chests for all the different items.

This is actually working, and I couldn't be happier!

This only leaves one final challenge! The most persistent one: The deadlock.

<u>Preventing a deadlock, again</u>

This is the part that I find most confusing about all this, but I think this works.

To avoid a deadlock, I want one inserter, preferably the one furthest away from the factory (didn't happen in this version, though), to only work when none of the others are, because that means that all the others have less than, or the same amount of items as the average, <u>but</u> only when the Factory has actually ordered said item.

To achieve that, I made an XOR gate. If the Factory orders the item, but none of the other deciders put their inserters to work, that can only be because of a deadlock.

This is set up using the two combinators in the green circle. The one on the right gets the sum of all items in the top left chest and says "If there are more than (or equal in Picture 4. That has now been changed, so it says '>', not '>=') 0 items in this chest, output one of *each*, or: [each > 0] -> [1each].

That is sent to the decider on the left, which says "if I only get <u>one</u> of *each*, then I'll activate my inserter, or [*each* = 1] -> [1*each*]. This is sent to the FI in the green circle via a red wire.

So far, that condition is true. However, the decider to the left also gets a signal from all the outermost deciders except for the one connected to the FI in the green circle. That would be all the teepee-shaped, red wires in picture 4.

This means that the deciders in the green circle will activate the FI in a green circle when none of the others get a go-signal from their decider pair even though the Factory sends an order. Or, in other

words: When an order is placed by the Factory AND none of the outer deciders send a signal to activate their FI, activate this FI. But there's more (of course).

That same FI is also getting the same go signal as the others via a green wire from its own "regular" combinators. This means that it'll work just as the others most of the time, but if a deadlock situation occurs, the left decider in the green circle gets only one signal, from the decider immediately to it's right, (which is always outputting a signal, as long as there's something in the chest to the top left), and that makes the FI in the green circle swing into life, breaking the deadlock.

As this version of this document is being finalized, days later, I haven't actually observed the deadlock breaking system operate, as it happens very fast, so it's hard to catch. However, I've now had this station set up several times in several places, transporting multiple items, and have yet to experience a deadlock, which happened with earlier versions, so there's reason to believe that this actually works.

The locomotive schedule

Because everything is handled by the station itself, the locomotive schedule is really simple. Here's one example:

Picture 5

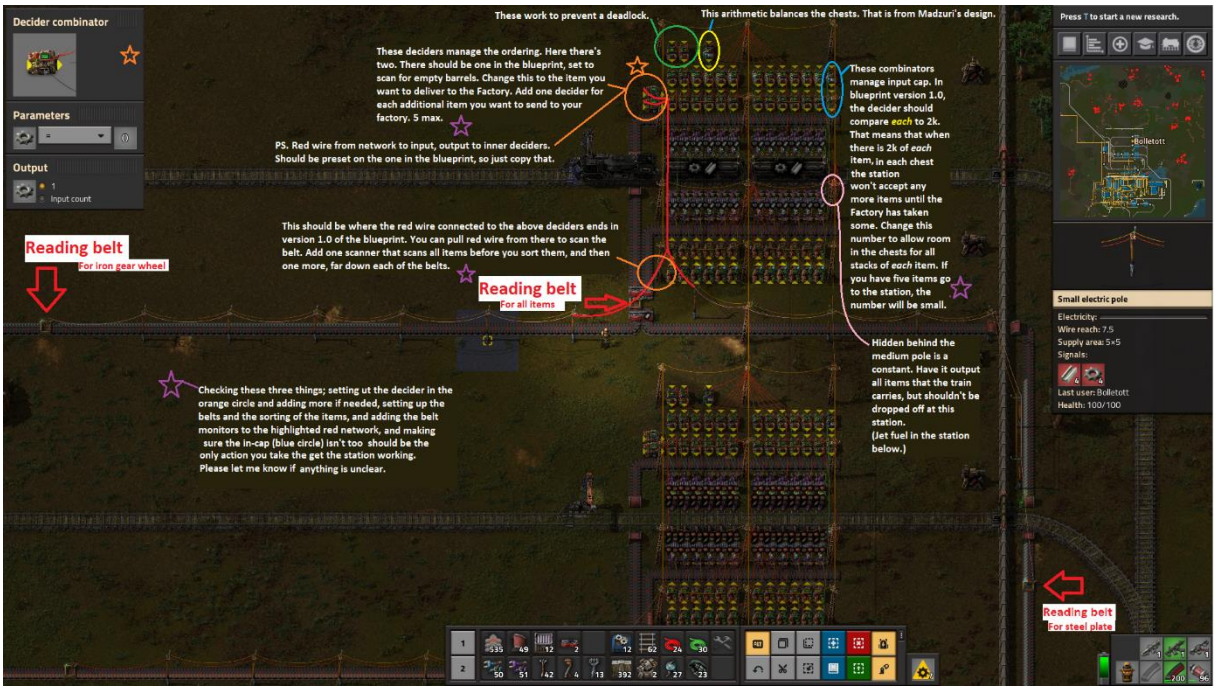
Generally, I keep them all at 2 sec inactivity, and that's all that is needed, really.

I chose this example, in which I've added the 30sec time limit as well. I did that because in the beginning, (or if there is a very high demand of one item in the Factory), items will be coming to your load station and be loaded straight into the train. This means that the train could be stuck for a very long time. But anyway, this is about the load station, and as such is not within the scope of this document.

The station itself is not connected to any network.

Set up manual

Here's an example of a fully functional station in my game. Everything should be explained in the picture itself. If you need to understand how the individual operators work, they are all explained further up in this document, and I think reading them again while looking at the following picture should be enough. Please let me know if I'm wrong.



Picture 6

Concluding

So, what do we have here?

Well, it's my dream station, I guess. It took *so* many hours of trial and error to get it right, but I'm very happy with the result.

It's very space efficient, because you don't need one station or one wagon for every item you want to send. You can send up to five items with the same train, but only have one loading* and one unloading station. You can also send all the items to and away from the stations using only one belt, and then sort them elsewhere, saving space around the train station.

It'll never get backed up, because it keeps the buffer chests balanced *and* manages the different items so than none of them fill up the chests, blocking others.

It also operates in a way that lets the Factory itself "tell" the station when it needs more of any one item. Using splitters to filter items right outside the station (or further away, if you want) will allow you to clear the station very quickly, and send the items wherever you want.

It also has a system to avoid deadlocks.

*If enough people want me to, I can make a similar explanation for my Balanced, multiple item handling, order receiving loading station, that pairs perfectly with this.

Well, that was it, friends.
It took me several hours to put all this together, and frankly I'm questioning all my life choices right now.

Hopefully this can be interesting, and maybe even helpful to someone. At the very least I hope someone will read this.
That way this wasn't a complete waste of time. Heh.

If you actually read all this: Cheers!
If you understood it: Cudos!
If you see something that doesn't make sense, or something that can be improved, be it making it more neat and compact, more flexible or whatever, please let me know. That would be so great!
Also, don't hesitate to ask if anything is unclear.

Peace out, everybody!
Bolletott

Version log

1.10 Added two sections: The locomotive schedule and the set up manual. Also added the version log.

1.01 Made some minor edits to wording.