Appendix of Paper "Multi-View Routing Visualization for the Identification of BGP Issues"

Stacked area charts for the cases reported in Table 2

In this document we report the stacked area chart layouts obtained for the cases in Table 2 of the Paper "Multi-View Routing Visualization for the Identification of BGP Issues". For each case we provide: a not optimized stacked area chart (large drawing), three smaller charts each produced by different heuristics (respectively: Deviation Swap, Wiggles Swap, and Near Flows), and comments on the case.

Potaroo instability 1

The following s-chart refers to an unstable prefix. Its routing changes periodically for long time (say weeks or months). These periodic changes do not correspond to outages, cable cuts or hijacks. They usually correspond to routers misconfigurations that let the routing periodically alternate among a set of states.

The observed prefix was among the top 3 instable prefixes on the Potaroo weekly instability report. This is a basic case since it has only three upstreams. The number of upstreams can easily be more than 50 for distance levels d above 3.

The three algorithms responded with three different layout orderings. The difference between them is not much, but users expressed a preference for Deviation Swap and Wiggle Swap. In this example the minimization of the standard deviation leads also to the minimization of the number of disconnections.

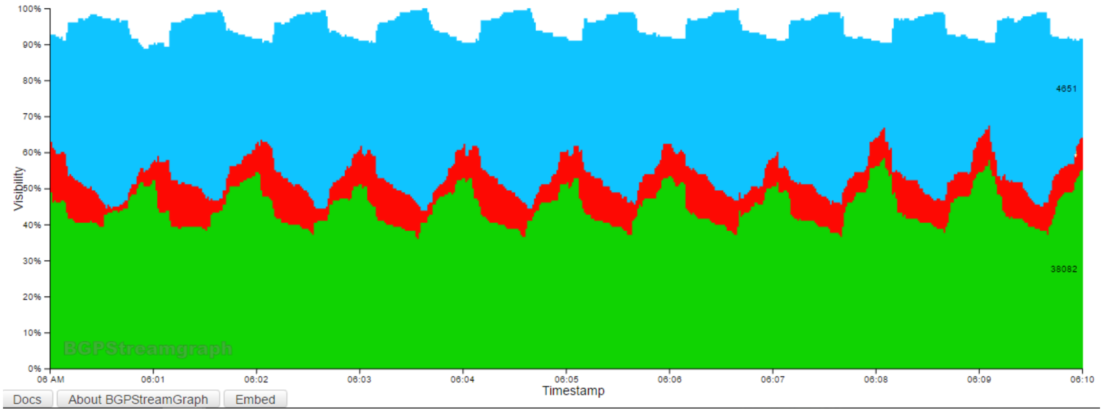
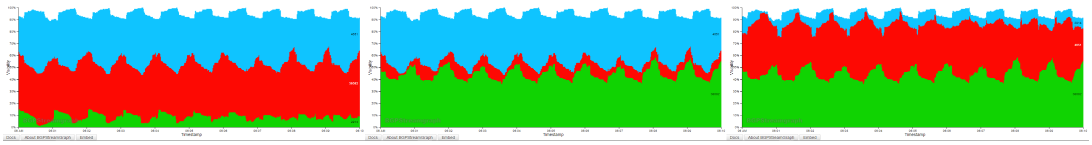


Figure 1: Potaroo instability 1



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 2: Comparison between s-chart heuristics

Potaroo instability 2

A slightly more complex scenario of an instable prefix. It was added to our list of cases due to its periodic behavior and for the presence of wide peaks. The not optimized layout contains an high number of disconnections on the blue area. The heuristics show two different results, which mainly differ only by the ordering of the green and red areas respectively. The effectiveness of the algorithms is noticeable on the blue area, which it is always stacked at the bottom in all the outputs. In this case the wiggle metric is minimized by the chosen ordering of the near flows algorithm. The disconnections are minimized by the standard deviation algorithm and wiggle algorithm.

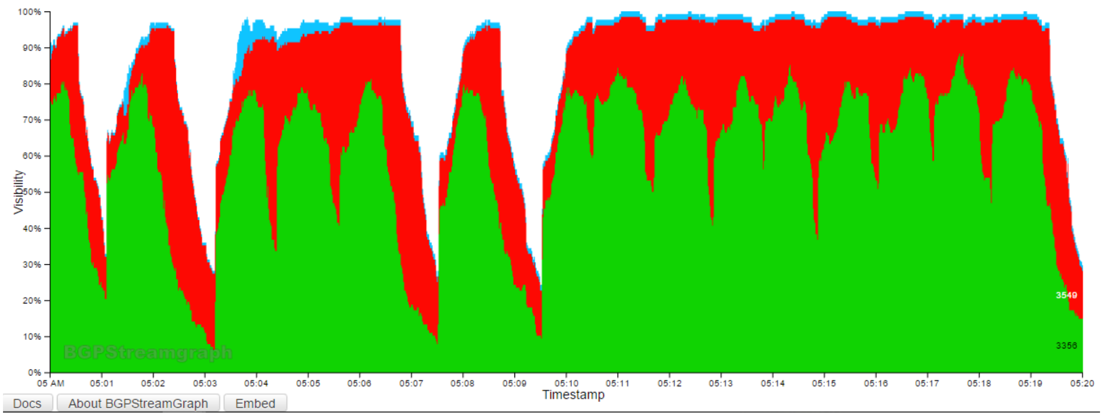
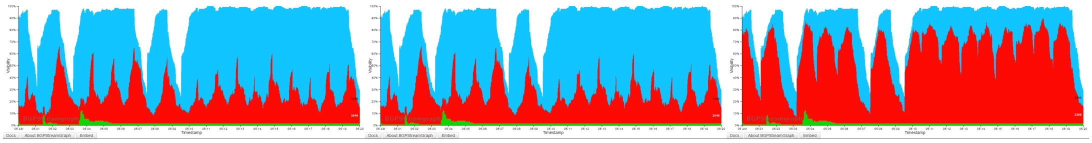


Figure 3: Potaroo instability 2



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 4: Comparison between s-chart heuristics

Potaroo instability 3

Another instable prefix scenario. This third case is one of the most useful scenario we met since it allows us to clearly observe the effectiveness of the heuristics and also the user perception of the obtained results. In terms of user preference, the deviation swap ordering has been selected as the best one, which corresponded to the minimization of the standard deviation metric too. To notice how much the optimization worths, try to accomplish this *simple* task: which area is changing most? Within the first layout you can easily tell that the brown area moves more than others.

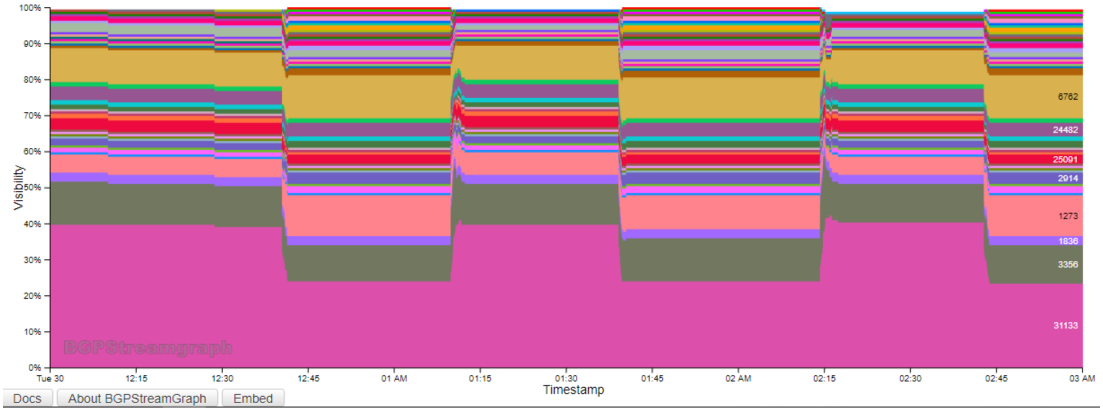
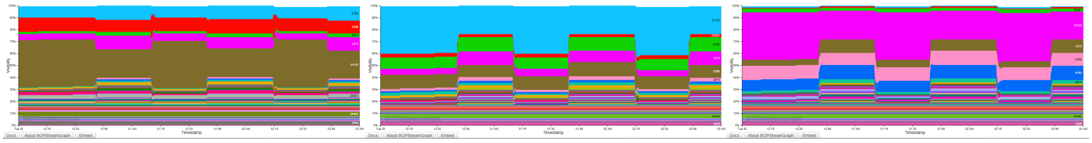


Figure 5: Potaroo instability 3



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 6: Comparison between s-chart heuristics

Potaroo instability 4

This is the last instability case of our list. In this case, the deviation swap and the near flow algorithms work almost the same. The only difference consists in a smaller area in the near flows ordering which is stacked on the top and gets disconnected many times. A totally different layout is instead drawn by the wiggle algorithm. Users did not prefer any specific layout, since all of them achieved similar results in terms of ordering.

The layout which minimizes the wiggle metric it is the near flows. The wiggle layout didn't minimize any of the knowing metrics. Also it is important to notice that the wiggle heuristic in this case, took a very high time of computation compared to the others.

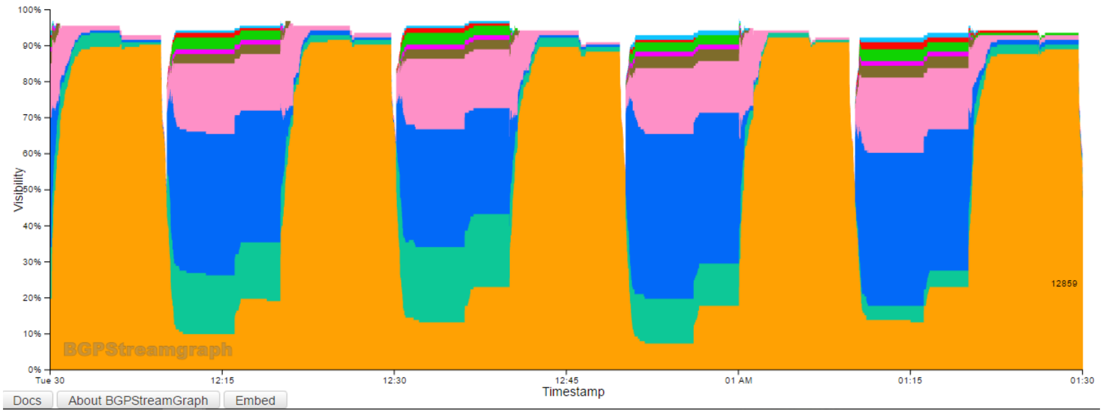
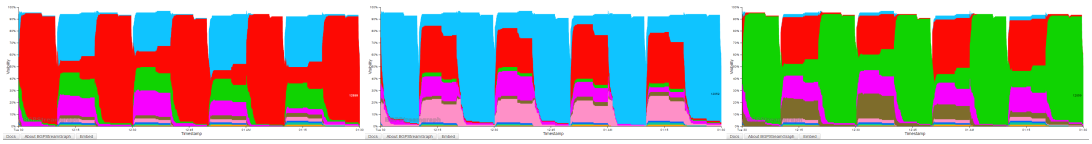


Figure 7: Potaroo instability 4



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 8: Comparison between s-chart heuristics

RIPE RIS Beacon

The following chart corresponds to a beacon, that is a router that periodically announces and withdraws a prefix. The visual pattern is quite sharp. What is remarkable is that some routes never change, independently on the fact that the prefix is announced or not. These routes are called *ghost routes*, under investigation from the networking community, are caused by routers mistreating or ignoring withdrawals.

This example looks pretty tricky for the wiggle, in particular for the presence of discontinuities on several points of the areas. The layout with a minimized wiggle is offered by the near flows algorithm. The first two layouts minimize both disconnections and standard deviations.

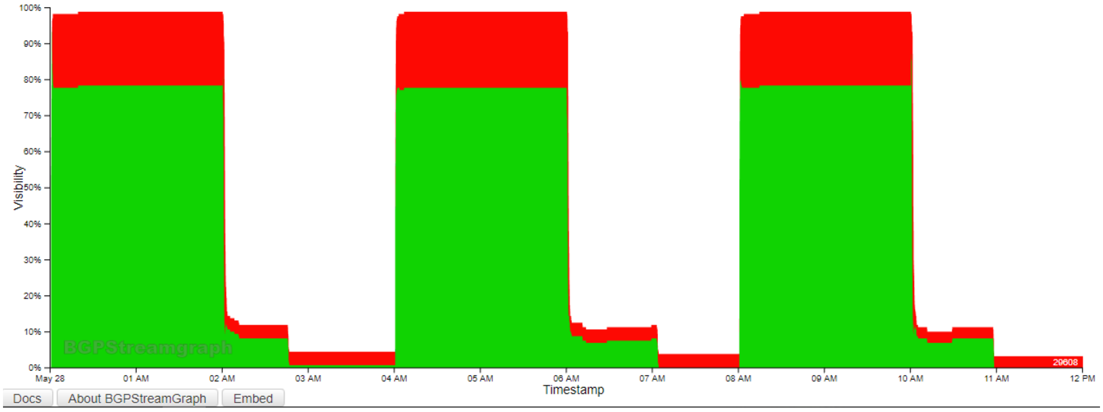
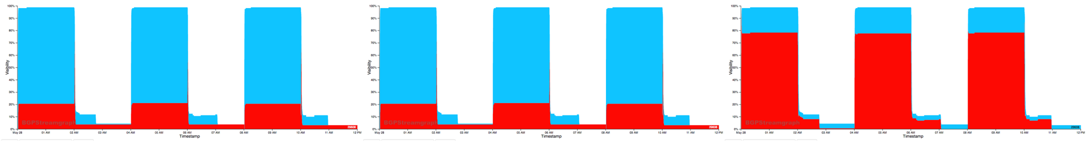


Figure 9: RIPE RIS Beacon



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 10: Comparison between s-chart heuristics

Djibouti Telekom outage

This case refers to an outage happened on a prefix of Djibuti Telekom. We can clearly observe here a common pattern for a cable cut with no backup: When the cable cut happens, the visibility drops almost to 0 (excluding possible ghost routes as described in the previous case). This happens mostly in case of submarine cable cuts, where it is more likely that most of the providers share the same cable and so the same fate. Two things can happen next: (1) if another cable/connection is available another colored area (another provider with a different cable) will take over and recover almost the entire visibility (i.e. the light blue area) as long as the original connection is not re-established. (2) if no other routing paths are available, the prefix will remain not visible as long as the cable is not repaired. This would be represented by a longer white gap in the chart.

BGP has a convergence time of around 5 minutes, which can appear sometimes as a reduction of visibility (white area). The white gap in this case can be clearly identified as a manual intervention. A new autonomous system (the light blue), which was not available before, appears and takes over the entire visibility. Probably, a peering established with the purpose to temporarily address the outage.

Each heuristic succeeded on minimizing its own metric. Three different layouts are produced and the most preferred by the users was the near flows layout. One of the most appreciated outcome by the users was the continuity of the blue area in the main picture (non optimized s-chart). In the near flows layout, the blue area is now represented in red for the color scheme reasons explained in the paper.

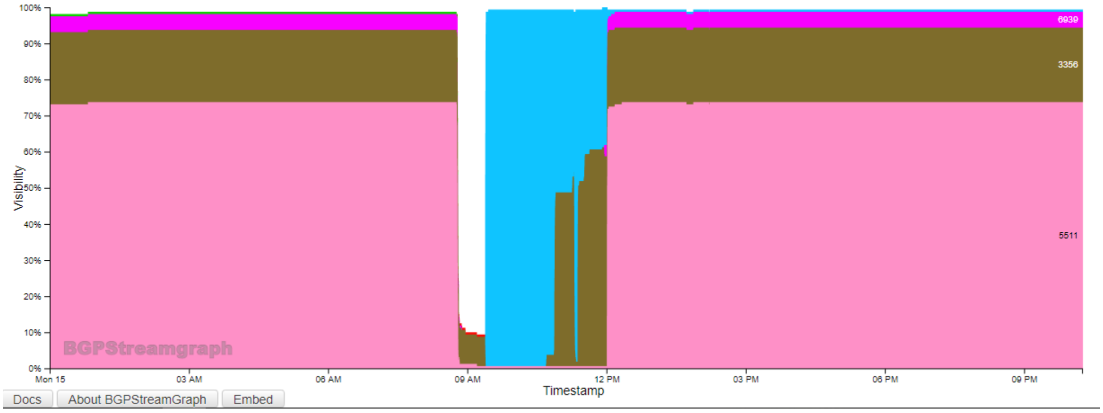
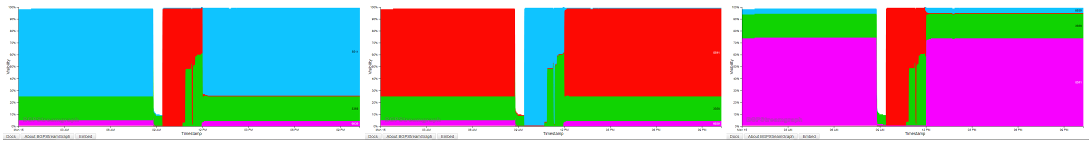


Figure 11: Djibouti Telecom outage



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 12: Comparison between s-chart heuristics

Suriname AS27775 disappears

This case is a cable outage. The first two heuristics give similar results, but the standard deviation performs better, because it puts the small areas at the right-bottom corner. Near flows algorithm also performs well but fails at positioning the two upper areas which seems to be exchanging flows. Watching the metrics results, the winning layout is computed by the standard deviation algorithm which minimizes both std dev and disconnections. Also in this case, the wiggle algorithm doesn't minimize it is natural metric.

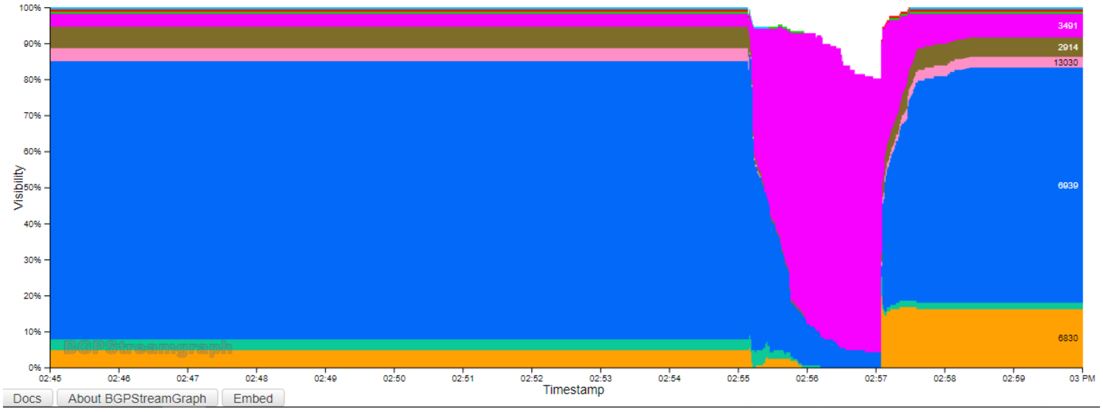
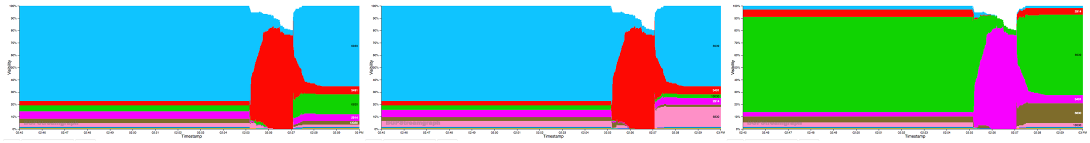


Figure 13: Suriname AS27775 disappears



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 14: Comparison between s-chart heuristics

Visa russian hijack

This is a hijack case. An hijack is characterized by an Autonomous System (a colored area) annuncing an IP prefix owned by someone else. A common visual pattern for an hijack is the appearance of new colored areas taking part to most of the visibility for a period of time. With this case we wanted to understand the impact of the heuristics on sharp and quick time series variations. In fact the variances of the areas are high in the hijack interval but average or void during the previous and afterward periods. This situation seems to penalize near flows algorithm which, working on the minimization of the number of disconnections, prefers a configuration of several little disconnections rather than a single wide one. All of the layouts present almost the same output, they differentiate only on the bottom part. The users seems to prefer the near flows layout even if it doesn't minimize any of the adopted metrics. On this case, both the wiggle and the standard deviation metrics are minimized by the corresponding algorithms. The disconnections metrics is minimized by the deviation swap algorithm.

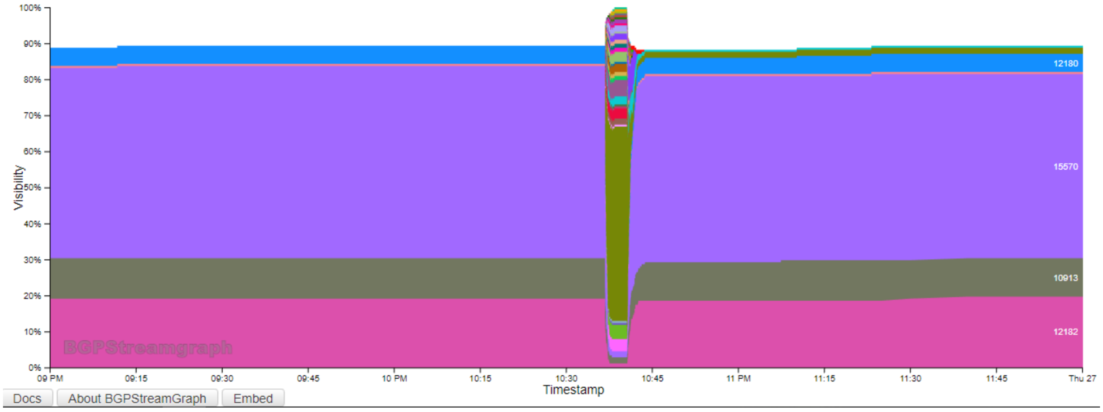
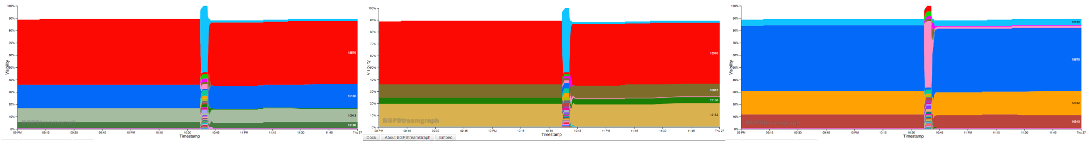


Figure 15: Visa russian hijack



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 16: Comparison between s-chart heuristics

Google acquires an IPv4 prefix

This case is about the acquisition and the propagation of the updates, for a newly acquired IPv4 prefix. The starting period is blank since no routes were propagated for the specified prefix before the acquisition date (the prefix was not used).

The heuristics generally worked well on this case, with the exception of the wiggles swap which gained the worst scores. This is the best case for the deviation swap, which finds the ordering that minimizes all the proposed metrics. Near flows is the preferred one for the users experience.

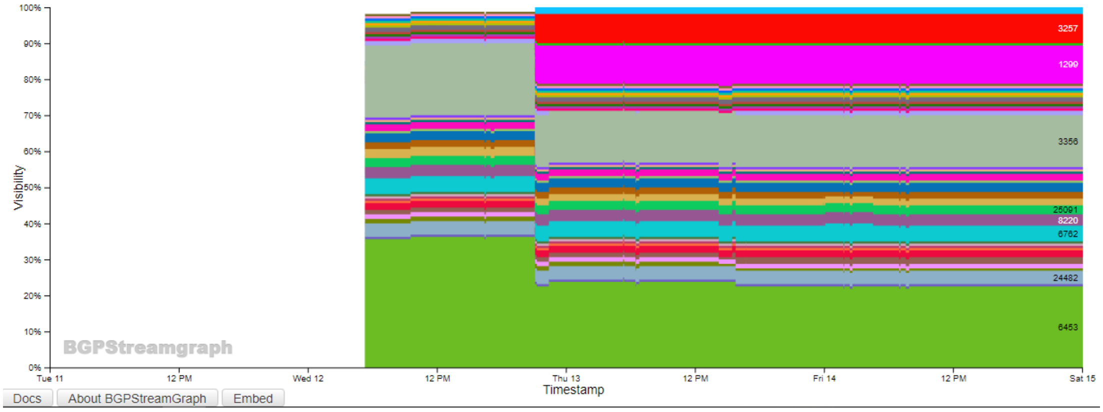
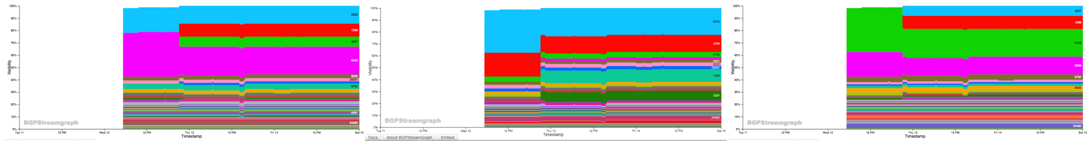


Figure 17: Google acquires an IPv4 prefix



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 18: Comparison between s-chart heuristics

Iran Hijack

This case is an Hijack of an Iranian provider. The initial white space is due do the fact that we are monitoring only the most specific prefixes announced for executing the hijack. The attacker can be identified by the first two ASes represented in brown and green. The users preferred standard deviation with respect to the others layouts. Notice the two little squares formed in the conjunction of the two routing configurations around 3.00 PM. The deviation swap algorithm is the only one to keep them coupled for the entire time range.

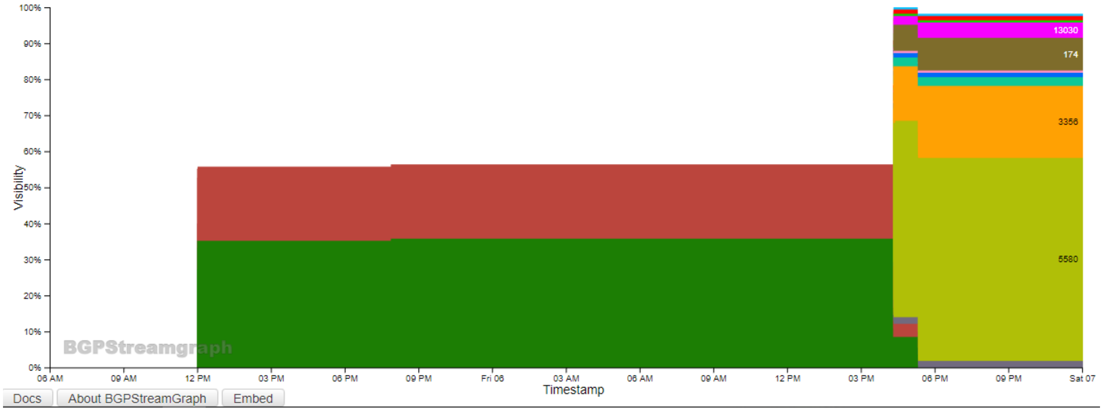
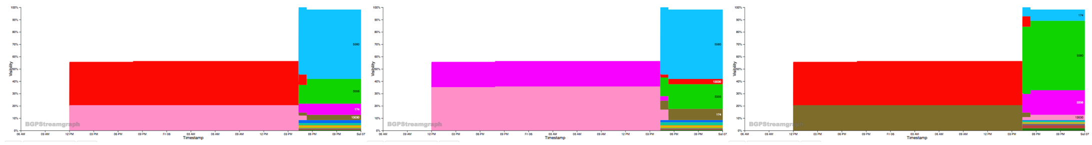


Figure 19: Iran Hijack



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 20: Comparison between s-chart heuristics

Iraq submarine cable cut

This case is another cable cut. It shares the same visual pattern described in the Djibuti Telekom one. In this case, each layout leads to some kind of area disconnections. The wiggle and the near flows heuristics produce the same layout, the standard deviation a barely different one. The difference in terms of metrics is instead more consistent. The first layout minimizes both standard deviation and wiggle, the other layouts minimize the disconnections. The user preference is not conclusive in this case.

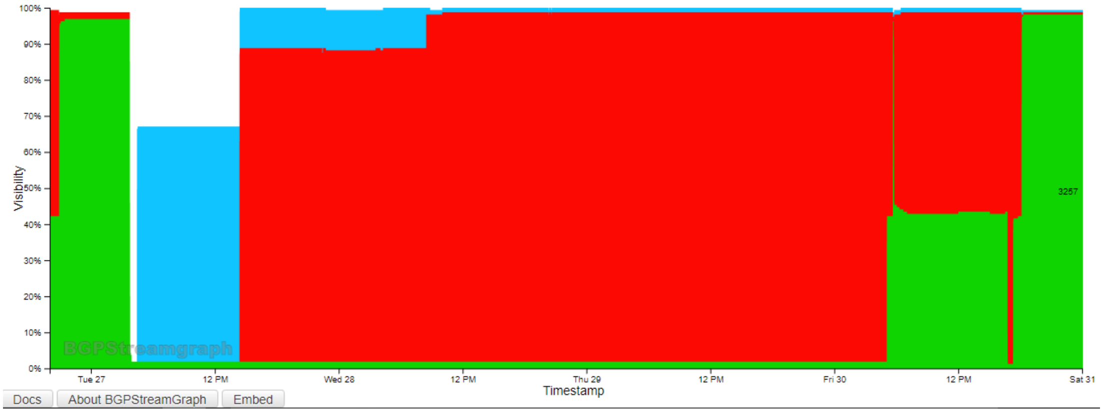
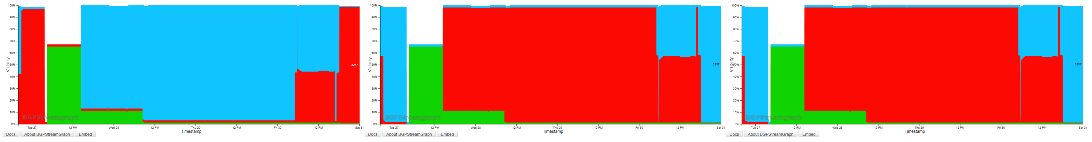


Figure 21: Iraq submarine cable cut



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 22: Comparison between s-chart heuristics

DDoS attack on Dyn DNS

The reference case presented here is one of the flaw-cases against the standard deviation algorithm. In fact the identified profiles are quite complex, irregular and sharp edged. The winning algorithm here is the wiggle one, confirmed also by the user preference. The main reason is the strange visual artifact that takes place in the middle of the chart. The vertical bands of colors, overlapping each other, result very disturbing, considered by many as a visualization fault. This behavior is not noticeable on the wiggle layout which, even if it presents strong disconnections of the blue area, results more readable and easy to read.

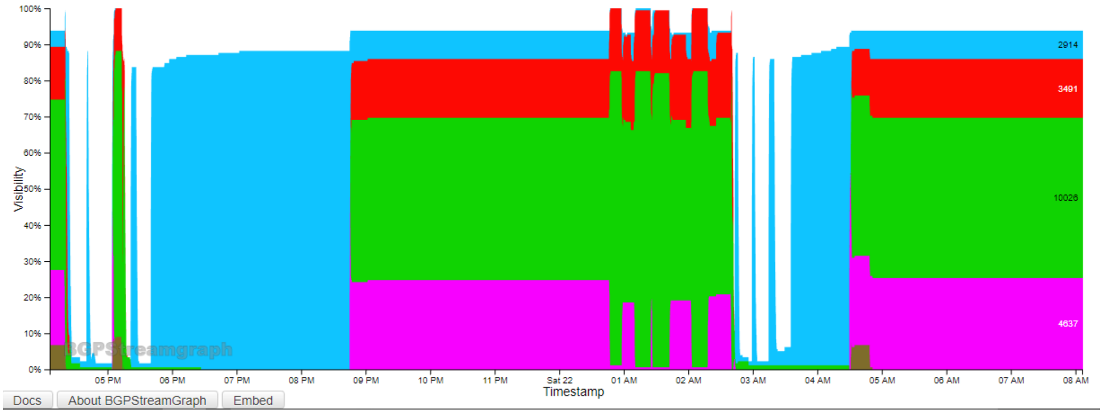
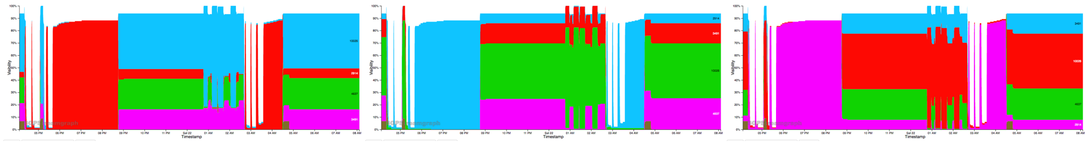


Figure 23: DDoS attack on Dyn DNS



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 24: Comparison between s-chart heuristics

Telekom Malaysia route leak

In this case the heuristics produced similar results. There are few small areas on the top of the non optimized layout, those areas are brutally disconnected and shifted by the lower ones. All the heuristics placed those small areas on the bottom of the s-chart. The only choice left is to find the ordering for the two bigger areas, which is a trivial task with no impact. This case shows how some principles that are commonly followed from all the heuristics.

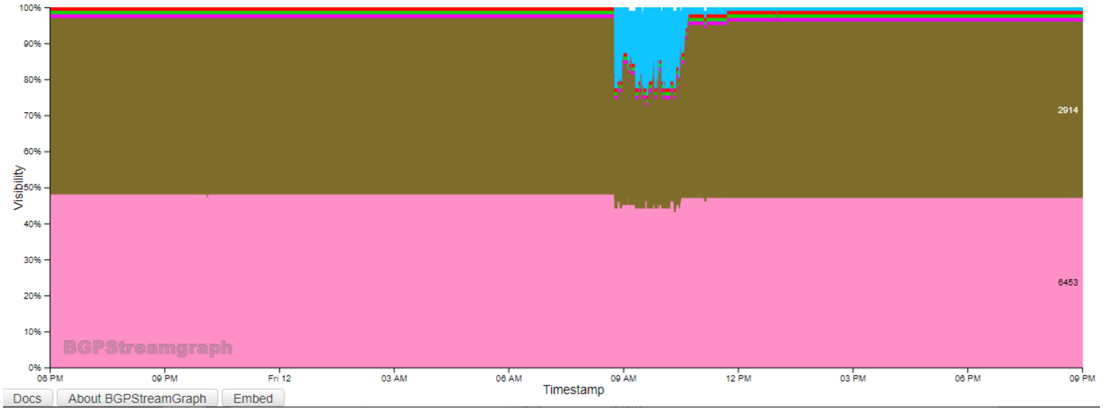
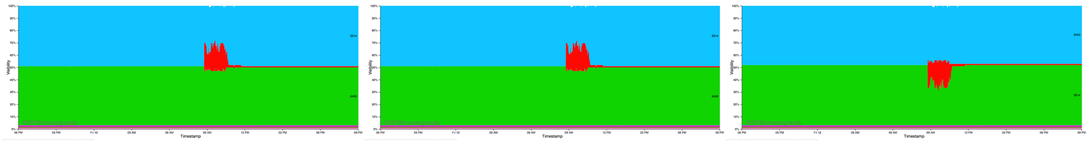


Figure 25: Telekom Malaysia route leak



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 26: Comparison between s-chart heuristics

Facebook down

The effectiveness of the heuristics, from the visual point of view, is not quite noticeable. The deviation swap algorithm and the near flows algorithm succeeded on minimizing their metrics. The same is not true for the wiggles swap algorithm. User preference is distributed among the three layouts, confirming anyway the preference compared to the not optimized one. The sharp profiles of the white areas penalize the wiggle algorithm which, as stated before, dislikes sharp edges.

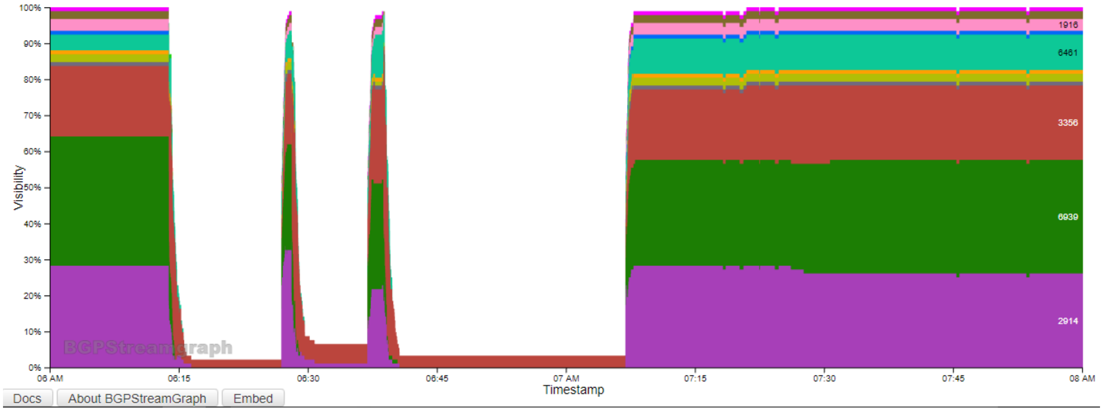
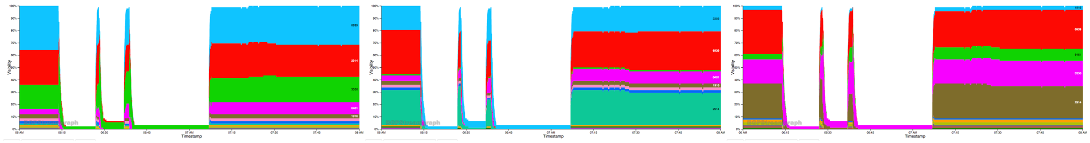


Figure 27: Facebook down



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 28: Comparison between s-chart heuristics

Ireland cable cut

This cable cut scenario is one of the most complex and satisfying scenarios for testing the visual quality of the heuristics. Unfortunately, the near flows algorithm exposes its weaknesses on this case. The wiggles and the standard deviation succeeded on the ordering, obtaining a clear picture of the outage, drastically reducing the disconnections. The near flows works on areas which exchange flows. When many areas exchange flows with the same area there are less decision factors. The users preferred the first two layouts and heavily penalized the one produced by near flows.

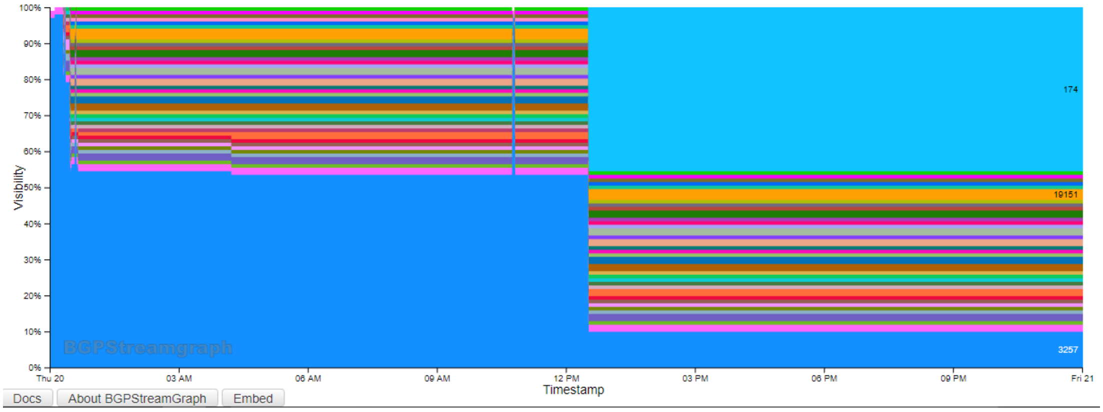
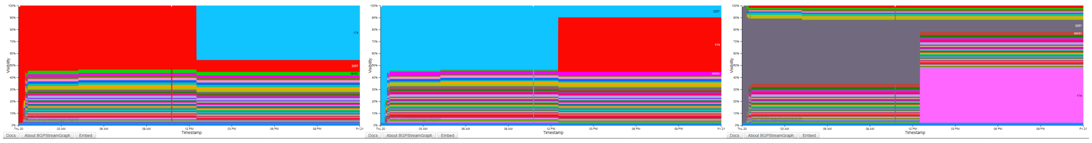


Figure 29: Ireland cable cut



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 30: Comparison between s-chart heuristics

Time Warner cable outage

The Time Warner outage is another cable outage. By comparing the result of the wiggle to the others heuristics, it is clear how the wiggle algorithm totally lacks the idea of disconnections and flows pairing. Looking at the light blue area on the wiggle layout, its condition compared to they gray area on the original it is barely changed. That area is one of the most stable areas and also the biggest one, it should be considered more carefully. Compare it with the results of the deviation swap (dark blue), and to the near flows (pink) layouts. Also this example resulted well pictured by the deviation swap algorithm.

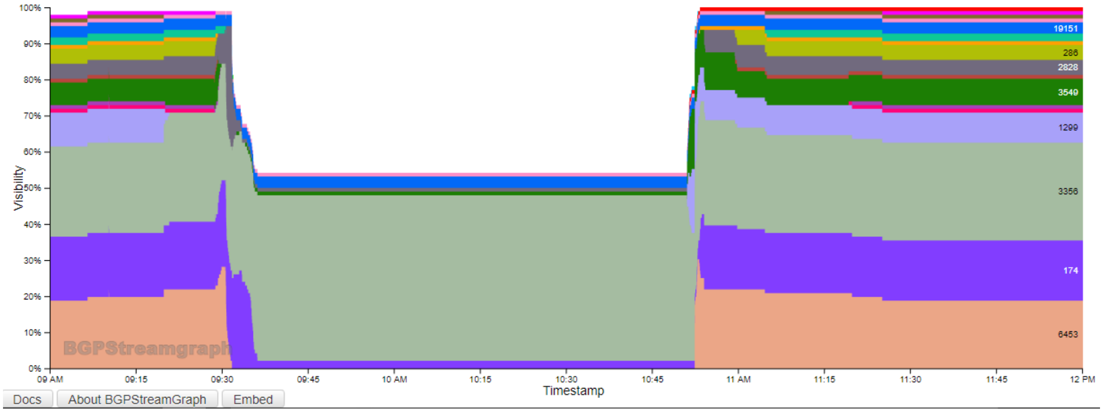
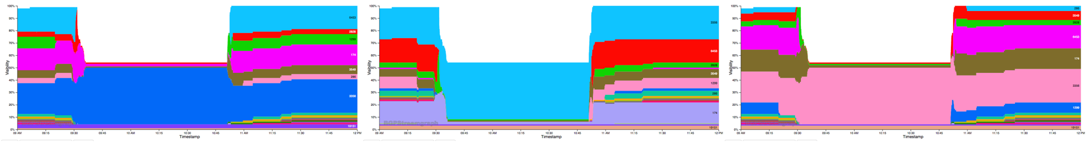


Figure 31: Time Warner cable outage



(a) Std Dev Swap

(b) Wiggle Sum Swap

(c) Near Flows

Figure 32: Comparison between s-chart heuristics

Aruba HackingTeam hijack

This case seems to be a simple one, but the truth beyond is a bit different. The edges around the "hole" are not as sharp as they seem and if we zoom-in and watch it closer we can spot the existence of thousands tight and tiny stepped events. They aren't even noticeable by the users, at this zoom level, but they result useful for us to study the correlation between heuristics, metrics and user readability of the layouts. The results are simple to be interpreted. All the heuristcs converge to a single (and desirable) solution, the hole is filled and no disconnections or other unwanted shiftings are generated. The metrics results are all quite the same, and so does the user perception of the readability.

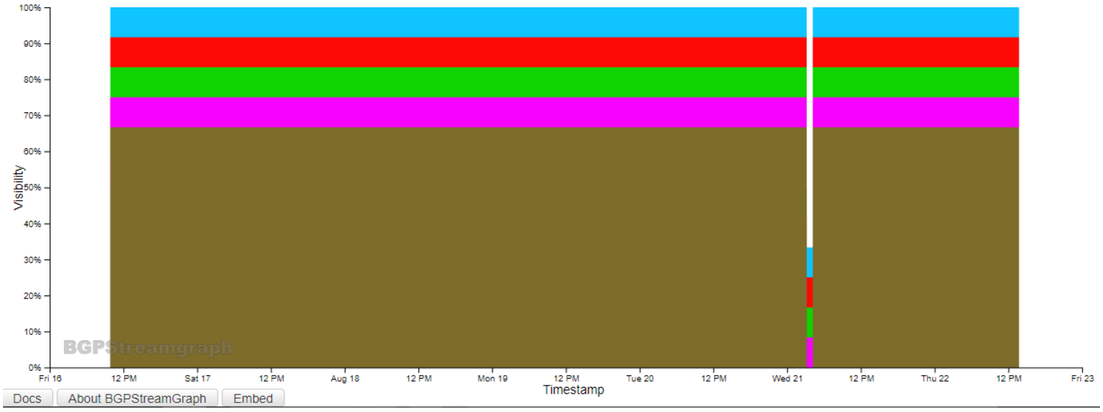
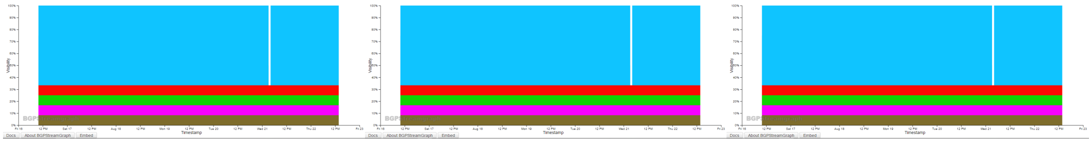


Figure 33: Aruba HackingTeam hijack



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 34: Comparison between s-chart heuristics

Cloudflare outage

This case was selected in order to test the ability of the heuristics to deal with the small areas in blue of the main picture. You can see how in the original layout how the light blue area is continuously broken by the presence of the yellow area below it. All algorithms performed well, compared to the non optimized layout, but some differences can be spotted, both from the visualization and from the values of the metrics. The near flows is the winner on this scenario both in terms of the user preferences and in terms of the metrics values. It minimizes wiggle and disconnections metrics. Wiggle swap is very slow. Also, deviation swap result as the minimizer for its own metric like in most of the cases.

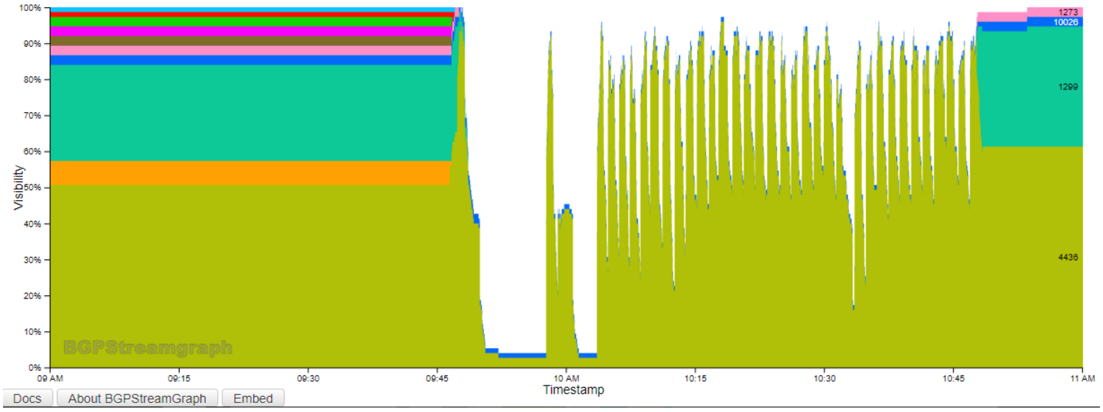
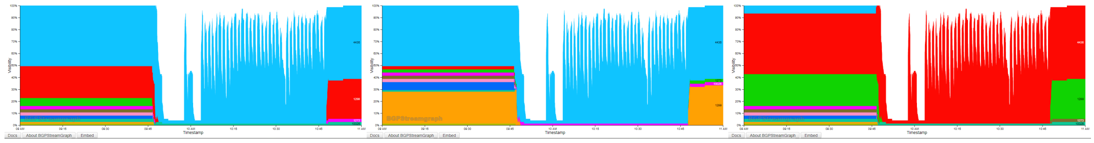


Figure 35: Cloudflare outage



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 36: Comparison between s-chart heuristics

Syria outage

This case is very similar to case number 10 "Iraq submarine cable cut". Also, in this case is not possible to get a layout without the interruption of some area. The metrics values beyond this case prove how the disconnection metric and standard deviation metrics are related. In fact, analyzing the previous results we can argue that in most cases the minimization of the standard deviation can positively affect the disconnection metric. However, this case exposes a rare and different behavior where this rule it is not valid anymore. The number of disconnections are heavily increased, order of thousands, by the minimization of the standard deviation. Even the wiggle algorithm obtained a much better disconnection score, the same as the near flows. In this kind of situation the values of the metrics can't us tell much more of the user preference itself. Sometimes the area ordering is a matter of subjective preferences even if the metrics tell us the opposite.

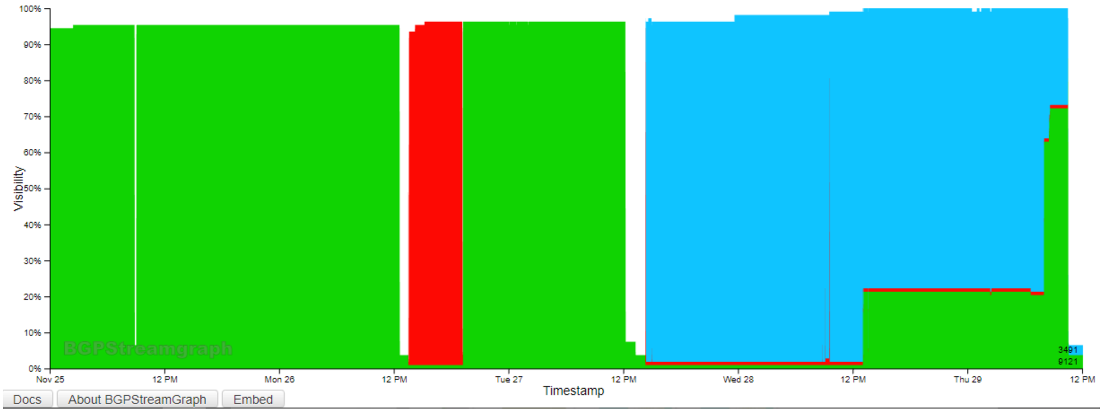
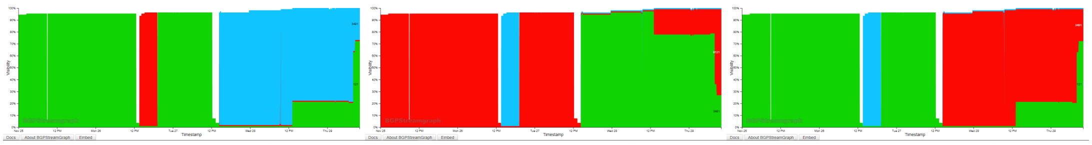


Figure 37: Syria outage



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 38: Comparison between s-chart heuristics

Youtube Pakistan Telecom

Finally, our last case, which contains a noticeable feature. The whole layout is flat, except at the beginning. The winner is the near flow algorithm, which is the only one capable of putting together the two shifted areas, minimizing the disconnections and the wiggle score.

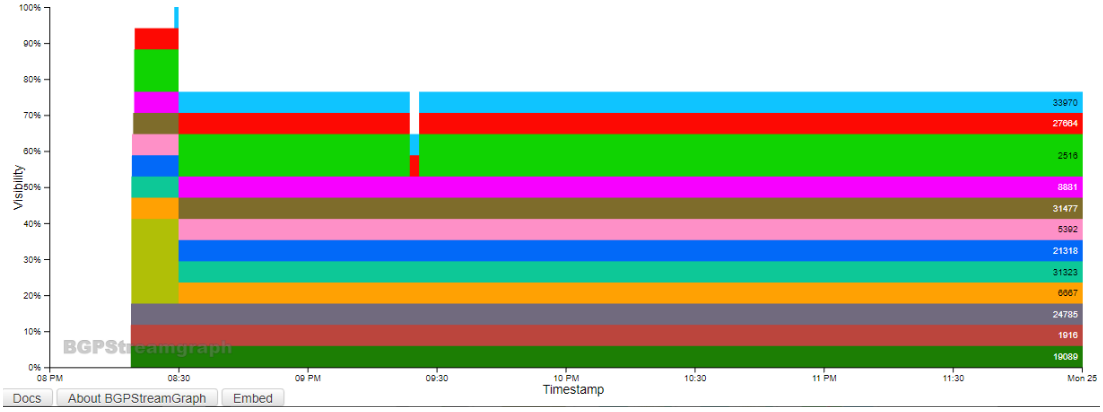
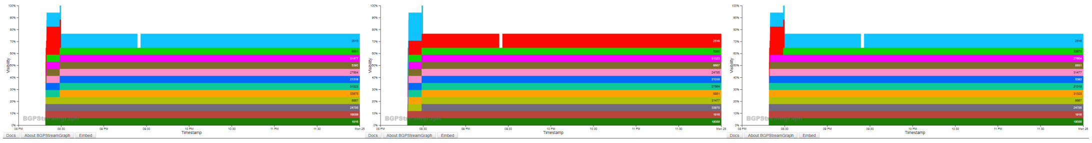


Figure 39: Youtube Pakistan Telecom



(a) Std Dev Swap (b) Wiggle Sum Swap (c) Near Flows

Figure 40: Comparison between s-chart heuristics