# Graph-based algorithm for the understanding of failures in the ATLAS infrastructure

**Gustavo A Uribe[1], Ignacio Asensi Tortajada[2], Carlos Solans Sánchez[2], André Rummler[2], Kaan Yüksel Oyulmaz[3] and Haluk Denizli[3]**

[1] Universidad Antonio Nariño, Calle 58 A # 37- 94, Colombia.
[2] CERN, Espl. des Particules 1, 1211 Meyrin, Switzerland.
[3] Bolu Abant Izzet Baysal University, Baibü Gölköy Yerleşkesi, 14030 Merkez/Bolu, Turkey.

E-mail: gustavo.uribe@cern.ch

**Abstract.** The ATLAS Technical Coordination Expert System is a knowledge-based application which describes and simulates the ATLAS experiment based on its components and their relationships with differing levels of granularity but with an emphasis on general infrastructure. It facilitates the sharing of knowledge and improves the communication among experts with different backgrounds and domains of expertise. The developed software has become essential for the planning of interventions as it gives easily insight into their consequences. Furthermore, it has also proven to be useful for exploring the most effective ways to improve the ATLAS operation and reliability by identifying points of failure with significant impact. The underlying database describes more than 13,000 elements with 89,000 relationships among them. It combines information from diverse domains such as detector control and safety systems, gas and water supplies, cooling, ventilation, cryogenics, and electricity distribution. As the most recent addition, a tool to identify the most probable cause of a failure state has been developed. This paper discusses the graph-based algorithm currently implemented by that tool and shows its behaviour based on the parameters entered by the user. An example in form of a real failure event is given which demonstrates the potential of the Expert System for understanding major failures faster in urgent situations.

## 1. Introduction

The ATLAS experiment is composed of more than 13,000 monitored and remote controlled components [1]. In this complex environment, it is crucial to support the coordination of the experiment through tools that can use explicit and formally represented knowledge from various experts. The ATLAS Technical Coordination Expert System (ATCES) has been created gathering the knowledge of experts that enables to foresee the potential impact of interventions and failures, and **identify the faulty elements when a failure occurs**. Each component represented in ATCES is assigned to a subsystem. Figure 1 shows a graphical representation of the system's full data coloured by subsystem and the relationships are bidirectional ("provides" and "requires") which leads to the elliptic structures observed. The nodes in figure 1 are distributed using a Yifan Hu layout [2] where the nodes in the center are more interconnected than those in the periphery. This clearly indicates that a large fraction of the nodes is highly inter-dependant which makes it difficult to predict reliably the outcome of an intervention or estimate the cause of a failure with a high probability. Furthermore, the subsystems are described

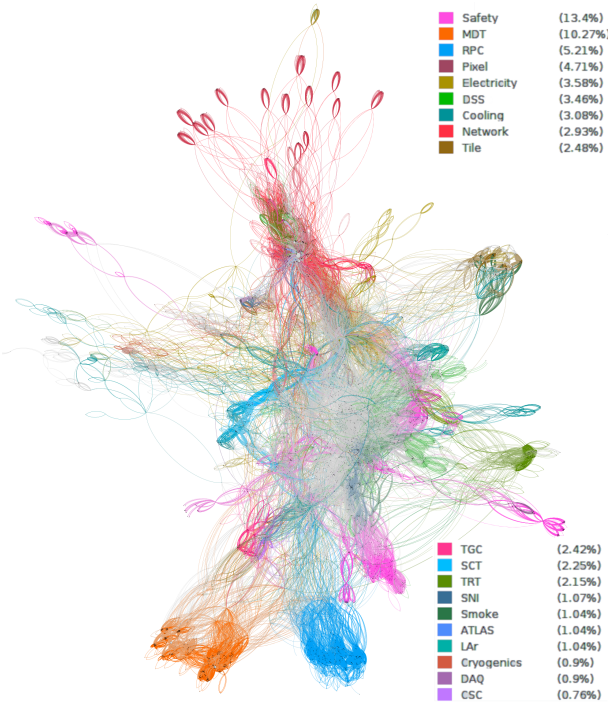


**Figure 1.** ATCES graph - components coloured by subsystem

non-homogeneously which increases the complexity of analyzing the data. One example is the misleading dominance of the MDT subsystem in the plot which is simply due to having been described with a higher granularity than the others and thus appearing larger compared to the rest of the subsystems.
This paper describes the algorithm developed for identifying the causes of a failure, the user defined parameters determining the behaviour of the algorithm, and the results obtained in a real failure scenario.

## 2. Methods

ATCES is based on the OKS (Object Kernel Support) persistency system that stores and manages the data following principles of the object oriented programming paradigma [3]. In order to run graph algorithms over the OKS data, the NetworkX python library has been used [4, 5]. A directed multigraph was generated replacing the OKS objects by nodes and the OKS relations by edges. NetworkX has been used for detecting circular dependencies (cycles), to propagate the changes in the simulations, and to list the most probable causes (MPC) of a faulty state. In the MPC context the Algorithm 1 is used. The current implementation of the algorithm uses the eigenvector centrality to sort the affected nodes (line 2) [6]. This algorithm measures the influence of a node in a network considering its neighborhood, the score of a node is correlated with the scores of the neighbors. Ordering the affected nodes using the eigenvector centrality instead of following the breadth-first order, improves the precision of the results by 11% and the recall by 9.5%. Understanding precision as the ratio of correct results retrieved (true positives) over the complete amount of results retrieved (true postives + false positives) and recall as the ratio of true positives over the complete amount of possible correct results (true positives + false negatives). In our case, there is only one correct result therefore the recall is 1 when the correct result is included and otherwise 0. The improvement in the precision

and recall is due to the fact that the eigenvector centrality algorithm ranks nodes according to their influence on other nodes. That influence is assumed to correlate with the probability of being the responsible entity for an effect while being independent of the distance between the influenced nodes. Despite in ATCES the relationships are bidirectional, the MPC calculation only consider the direction from provider to consumer.

**Algorithm 1** Most Probable Cause (MPC)

**Require:** $out\_centrality \leftarrow$ EIGENVECTOR_CENTRALITY($graph$)
1: **function** FINDMPC($affected\_nodes$,$elapsed\_time$,$max\_faulty\_nodes$,$max\_results$, $max\_trys$,$is\_exhaustive$)
2: SORT($affected\_nodes, out\_centrality$)
3: $predecessors \leftarrow$ COMMON_PREDECESSORS($affected\_nodes$)
4: $MPC \leftarrow []$
5: **for** $combination\_size \leftarrow 1, max\_faulty\_nodes + 1$ **do**
6: $combinations \leftarrow$ COMBINATIONS($predecessors, combination\_size$)
7: $iteration \leftarrow 0$
8: **for all** $combination$ **do**
9: ▷ While $|MPC| < max\_results \& iteration < max\_trys \times combination\_size$
10: $iteration \leftarrow iteration + 1$
11: **for all** $predecessor$ **do**
12: SWITCH_OFF($predecessor$)
13: $affected\_nodes\_prime \leftarrow$ PROPAGATE_CHANGE($predecessor, elapsed\_time$)
14: ▷ Uses breadth first search recursively
15: **end for**
16: **if** $is\_exhaustive$ **then**
17: **if** $affected\_nodes = affected\_nodes\_prime$ **then**
18: ▷ only nodes with the classes of the affected_nodes are evaluated
19: $MPC \leftarrow MPC$.ADD($combination$)
20: **end if**
21: **else** ▷ When the affected_nodes is a partial list of the total impacted by a fault
22: **if** $affected\_nodes \subseteq affected\_nodes\_prime$ **then**
23: $MPC \leftarrow MPC$.ADD($combination$)
24: **end if**
25: **end if**
26: **end for**
27: **end for**
28: **return** $MPC$
29: **end function**

The algorithm is parameterized in order to return results before considering all the possible combinations reducing the execution time (see line 8). The user selects the average *waiting time*, then the *max_results* and *max_tries* parameters are set based on the performance evaluation done using the current infrastructure which is running the system [7] (see table 1). If the *is_exhaustive* parameter is *true* (exhaustive mode), then the list of *affected_nodes* will be considered as the complete list of nodes that cannot work normally due to a failure. If it is false it will be considered only as a subset of the complete list. The *affected_nodes* list is used to find the common predecessors, which are the elements that can cause an effect in all those nodes. The combinations evaluated are the mathematical k-combinations of the common predecessors starting with k=1 up to k=max_faulty_nodes, and while *max_results* and *max_trys* are not reached. In order to evaluate the combinations, the common predecessors in a combination

are turned off in a simulation (line 12). The nodes affected after the simulation are named *affected_nodes_prime* (line 13). The combinations added to the list of results (MPC) are the once with the *affected_nodes* contained in or equal to the *affected_nodes_prime* (line 17).
The user interface offers the user the possibility to inhibit actions on objects according to the current state extracted from the detector safety system (DSS). Thus they will be excluded from the MPC calculation as inhibited actions will not be executed in the real system under any circumstances.

**Table 1.** Parameters per average waiting time

| Average Time [min] | Algorithm Parameters |
|---|---|
| 5 | max_results=16<br>max_tries=64 |
| 10 | max_results=128<br>max_tries=128 |
| $\infty$<br>(Unlimited) | max_results=2048<br>max_tries=2048 |

## 3. Results

The MPC tool of the ATCES has been used successfully for scenarios like the annual maintenance of the water circuit chillers [8]. But on 5th July 2021 an incident occurred for which the tool did not provide the expected answers. During the incident, which will be discussed in the following, the Uninterruptible Power Supply (UPS) system supplying power to the switchboard *EXD1_1X* was cut due to a trigger of its emergency circuit. That UPS system consists of the static-bypass switch *EXS102_1X* and the UPS units *EXS103_1X – EXS106_1X* which are connected in parallel. This emergency stop circuit should usually occur only when one of the emergency buttons located on each of the five units is pushed. A post-mortem in-depth investigation by the manufacturer showed that indeed none of the buttons had been pressed but ultimately the underlying reason could not be discovered and had to be classified as a glitch.
The list of affected nodes generated by all simulations of this failure corresponds with the information extracted from logs after the event. However, the exhaustive mode (line 16) was not able to produce results because some alarms that should have been triggered due to the failure, were not triggered in reality on this day. For example, the alarm *AL_GAS_MUN_TGC_GasFailure* had been already triggered by another intervention and thus could not be triggered again. If an experienced expert user notices this and provides the alarms already triggered manually to the tool, then the *MPC* results become correct in the 10 minutes mode (*EXD1_1X* and *EMT208_1X*). In case the user is not able to recognise that alarms are missing, the MPC tool still provides good answers in the non-exhaustive mode requiring an average waiting time greater than 10 minutes, see figure 2. The MPC results show combinations of 1 or 2 faulty nodes. The switchboard *EXD1_1X* is listed on position 97 whereas the first part of the list is filled with a long and thus potentially confusing series of emergency buttons. Emergency buttons are a valid cause of the introduced affected systems , however, those can be easily grouped and also potentially discarded as unlikely by an expert. The non-exhaustive mode using an average waiting time of 5 minutes does not deliver any result because the algorithm used for sorting the common parents (eigenvector_centrality) ranks the faulty switchboard poorly. This is also the cause for the low ranking in the 10 minutes MPC results.

| | |
|---|---|
| **Name** | Calculation_20210924__18_59_06 |
| **Type** | MPC-non-exhaustive |
| **Expected Waiting Time** | 10 min |
| **Status** | Processed |
| **Date** | 2021-Sep-24 |
| **Affected** | AL_COL_BeamPipe_VJA_CoolingNotRunning, AL_COL_BeamPipe_VJC_CoolingNotRunning, AL_COL_Cables_CoolingFailure, AL_COL_MUN_CoolingFailure_SideA, AL_COL_MUN_CoolingFailure_SideC, AL_COL_MUN_StationA_Loop1_Stopped, AL_COL_MUN_StationA_Loop2_Stopped, AL_COL_MUN_StationA_Loop4_Stopped, AL_COL_MUN_StationA_Loop5_Stopped, AL_COL_MUN_StationA_Loop6_Stopped, AL_COL_MUN_StationA_Loop7_Stopped, AL_COL_MUN_StationA_Loop8_Stopped, AL_COL_MUN_StationA_Loop9_Stopped, AL_COL_MUN_StationC_Loop1_Stopped, AL_COL_MUN_StationC_Loop2_Stopped, AL_COL_MUN_StationC_Loop3_Stopped, AL_COL_MUN_StationC_Loop4_Stopped, AL_COL_MUN_StationC_Loop5_Stopped, AL_COL_MUN_StationC_Loop6_Stopped, AL_COL_MUN_StationC_Loop7_Stopped, AL_COL_MUN_StationC_Loop8_Stopped, AL_COL_MUN_StationC_Loop9_Stopped, AL_GAS_MUN_MDT_GasFailure, AL_INF_Power_SX1_EXS103_UPSFailure, AL_INF_Power_SX1_EXS104_UPSFailure, AL_INF_Power_SX1_EXS105_UPSFailure, AL_INF_Power_SX1_EXS106_UPSonBattery, AL_INF_Power_UX15_EXD13_HS_US_L4_Failure, AL_INF_Power_UX15_EXD15_BW_A_Failure, AL_INF_Power_UX15_EXD25_BW_C_Failure, |
| **Inhibited Objects** | AL_COL_MAG_CoolingFailure, AL_COL_MUN_StationC_Loop10_Stopped, AL_COL_MUN_StationC_Loop11_Stopped, AL_COL_MUN_StationC_Loop13_Stopped, AL_INF_WaterLeak_MUN_CSC_Y5823X0, AL_MAG_Solenoid_FastDump, AL_MAG_Solenoid_SlowDump, AL_MAG_Toroid_FastDump, AL_MAG_Toroid_FastDump_atHighCurrent, AL_MAG_Toroid_SlowDump, O_INF_MUN_Power_Y1207S2, O_MUN_NSW_SideA_PWR_US15, O_MUN_NSW_SideA_PWR_USA15, O_MUN_NSW_SideA_SRV_US15, O_MUN_NSW_SideA_SRV_USA15, O_MUN_NSW_SideC_PWR_US15, O_MUN_NSW_SideC_PWR_USA15, O_MUN_NSW_SideC_SRV_US15, O_MUN_NSW_SideC_SRV_USA15, O_MUN_RPC_Power_Y3019A1, PT_LUC_Temp_Probe13_SideC, PT_LUC_Temp_Probe1_SideC, PT_LUC_Temp_Probe5_SideC, |
| **MPC** | ['EUB110_15A', 'EUB120_15A', 'EUB121_15A', 'EUB122_15A', 'EUB123_15A', 'EUB124_15A', 'EUB125_15A', 'EUB130_15A', 'EUB131_15A', 'EUB140_15A', 'EUB141_15A', 'EUB142_15A', 'EUB143_15A', 'EUB144_15A', 'EUB150_15A', 'EUB151_15A', 'EUB152_15A', 'EUB160_15A', 'EUB161_15A', 'EUB210_15A', 'EUB211_15A', 'EUB212_15A', 'EUB220_15A', 'EUB221_15A', 'EUB222_15A', 'EUB223_15A', 'EUB230_15A', 'EUB231_15A', 'EUB232_15A', 'EUB233_15A', 'EUB234_15A', 'EUB310_15A', 'EUB311_15A', 'EUB312_15A', 'EUB313_15A', 'EUB314_15A', 'EUB315_15A', 'EUB1101_15X', 'EUB1102_15X', 'EUB1103_15X', 'EUB1104_15X', 'EUB1105_15X', 'EUB1106_15X', 'EUB2101_15X', 'EUB2102_15X', 'EUB2103_15X', 'EUB3101_15X', 'EUB3102_15X', 'EUB3103_15X', 'EUB3104_15X', 'EUB3105_15X', 'EUB1201_15X', 'EUB1202_15X', 'EUB1203_15X', 'EUB1204_15X', 'EUB1205_15X', 'EUB1206_15X', 'EUB1207_15X', 'EUB1208_15X', 'EUB2201_15X', 'EUB2202_15X', 'EUB3201_15X', 'EUB3202_15X', 'EUB3203_15X', 'EUB3204_15X', 'EUB1301_15X', 'EUB1302_15X', 'EUB1303_15X', 'EUB2301_15X', 'EUB2302_15X', 'EUB3301_15X', 'EUB3302_15X', 'EUB3303_15X', 'EUB3304_15X', 'EUB3305_15X', 'EUB3306_15X', 'EUB1401_15X', 'EUB1402_15X', 'EUB1403_15X', 'EUB1404_15X', 'EUB2401_15X', 'EUB3401_15X', 'EUB3402_15X', 'EUB3403_15X', 'EUB3404_15X', 'EUB3405_15X', 'EUB3406_15X',**'EXD1_1X'**, 'EMT208_1X', 'EMD2_1E', 'EMD8_1E', 'EHT102_1E', 'EHD21_9E', ('EUB110_15A', 'EUB120_15A'), ('EUB110_15A', 'EUB121_15A'), ('EUB120_15A', 'EUB121_15A'), ('EUB110_15A', 'EUB122_15A'), ('EUB120_15A', 'EUB122_15A'), ('EUB121_15A', 'EUB122_15A'), ('EUB110_15A', 'EUB123_15A'), ('EUB120_15A', 'EUB123_15A'), ('EUB121_15A', 'EUB123_15A'), ('EUB122_15A', 'EUB123_15A'), ('EUB110_15A', 'EUB124_15A'), ('EUB120_15A', 'EUB124_15A'), ('EUB121_15A', 'EUB124_15A'), ('EUB122_15A', 'EUB124_15A'), ('EUB123_15A', 'EUB124_15A'), ('EUB110_15A', 'EUB125_15A'), ('EUB120_15A', 'EUB125_15A'), ('EUB121_15A', 'EUB125_15A'), ('EUB122_15A', 'EUB125_15A'), ('EUB123_15A', 'EUB125_15A'), ('EUB124_15A', 'EUB125_15A'), ('EUB110_15A', 'EUB130_15A'), ('EUB120_15A', 'EUB130_15A'), ('EUB121_15A', 'EUB130_15A'), ('EUB122_15A', 'EUB130_15A'), ('EUB123_15A', 'EUB130_15A'), |

**Figure 2.** MPC results for normal power failure (correct result in bold)

## 4. Conclusion

The MPC tool continues showing its usefulness for the understanding of failures in the ATLAS infrastructure. The analysis of the discussed event demonstrated the clear need for pruning in the centrality algorithm. In order to achieve a more useful ranking of probable causes, the MPC results can be further sorted using the number of common elements between *affected_nodes* and *affected_nodes_prime*, and the probability of failure of the elements.

## 5. References

[1] I. Asensi Tortajada, A. Rummler, C. Solans Sánchez, and J. Torres País, "Planning of Interventions With the Atlas Expert System," *Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems*, vol. ICALEPCS2019, p. 4, 2020.

[2] Y. Hu, "Efficient, high-quality force-directed graph drawing," *Mathematica journal*, vol. 10, no. 1, pp. 37–71, 2005, publisher: Redwood City, Ca.: Advanced Book Program, Addison-Wesley Pub. Co., c1990-.

[3] R. Jones, L. Mapelli, Y. Ryabov, and I. Soloviev, "The OKS persistent in-memory object manager," *IEEE Transactions on Nuclear Science*, vol. 45, no. 4, pp. 1958–1964, 1998, publisher: IEEE.

[4] *NetworkX*. [Online]. Available: https://networkx.org/

[5] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[6] B. Ruhnau, "Eigenvector-centrality — a node-centrality?" *Social Networks*, vol. 22, no. 4, pp. 357–365, Oct. 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873300000319

[7] I. Asensi Tortajada, A. Rummler, G. Salukvadze, C. Solans Sánchez, and K. Reeves, "ATLAS Technical Coordination Expert System," *EPJ Web Conf.*, vol. 214, 2019.

[8] I. Asensi Tortajada, C. Solans Sánchez, A. Rummler, G. A. Uribe, and J. Torres País, "Understanding ATLAS infrastructure behaviour with an Expert System," *European Physical Journal (EPJ) Web of Conferences*, vol. 251, Aug. 2021.