The Lifelong planning A* algorithm is an approach to combine the popular A* pathfinding algorithm with artificial intelligence. Heuristic search methods such as A* approximate the distance from the final point to focus on the search and work better than un-informed search methods. On the other hand, incremental search methods such as DynamicSWSF-FP reuse information from previous searches to find the shortest path faster than solving the shortest path from scratch at each new point. The research paper proposes a method that combines both the A* algorithm and incremental search, and is faster at finding the shortest path as compared to the above methods individually. This new algorithm is known as lifelong planning A*.

There have been attempts to build an AI system that can find the shortest path however these systems replan from scratch and solve the path planning problems independently. This is a very impractical method in environments with frequent changes. These changes are usually very small, and therefore solving these path planning problems from scratch can be wasteful. This led to the idea of making use of incremental search in combination with an informed, heuristic-based method.

The first example of LPA* in action makes use of a 15x20 grid where some of the cells are blocked off. Some of the cells are changed from blocked cells to open cells, but the number of blocked cells remains constant. A modified version of DynamicSWSF-FP, which terminates after its sure that it has found the shortest path, has been used to compare against the performance of LPA*. The modification has been made to increase its efficiency and so that the test isn't skewed in favor of LPA*. The authors have graphed each outcome, and the cells whose start distances have changed after the change in the grid have been shaded in grey. As we can see LPA* has the least grey shaded squares suggesting that it is able to replan faster and more efficiently than DynamicSWSF-FP and A* can individually.

The authors of this paper go on to prove their algorithm mathematically with concise mathematical proofs as well as analytically and logically working out how a combination of the two algorithms performs better. They have also provided the pseudocode and the approach to programming the algorithm, in case one wants to implement the algorithm themselves. The experimental evaluation section of the paper explores and compares the performance between A*, breadth-first search, Dynamic-SWSF-FP and LPA*. The same modified version of DynamicSWSF-FP was used as the one in the first example. Two versions of the A* algorithm was used as well, one which breaks ties between same f-values in favor of smaller g-value and another which breaks the tie in favor of larger g-values. Since the performance (or rather the runtime) of these algorithms is system dependent, they also considered machine independent variables to measure such as the total number of vertex expansions, ve and the total number of heap percolates, hp. They even considered multiple vector expansions if LPA* expands the same vertex more than one time, in order to not skew the results of this experiment in favor of LPA*.

During the above experiment, the authors noted an interesting observation. After every set of changes in the grid, LPA* relied less and less upon tie-breaking f-values, as it was reusing information from the previous searches. This means that it was able to overcome one of the limitations of the A* algorithms. Based on the final result of the experiment, LPA* was the best performer.

One of the biggest strengths of this research paper is that the authors made sure to take measures in relation to any biases in their experiments and analysis of the results. This makes their results much more reliable. They even provide us with a practical example of a changing environment, and how LPA* is able to take these changes into consideration in real time instead of having to run the algorithm all over again.

The paper also has a section based on ways to optimize and extend the algorithm as well as the pseudocode for these optimizations. This gives other researchers a good starting point if they want to attempt developing similar but more efficient algorithm, or even expand upon LPA* itself. It also shows that the authors have not just developed a new and better algorithm but have also put in the effort to maximize its efficiency and performance.

Another interesting section of the paper is the 'Related Research' section. It covers similar research of combining two different search algorithms, like the D* algorithm. The section gives us more insight into the inspiration behind this research paper and the initial thought process of the authors. This gives us readers other resources and research papers to refer to.

Unfortunately, we do not get to see the performance of the unmodified version of DynamicSWSF-FP, and are only shown the modified and supposedly better performing version in comparison to LPA*. This is one of the downsides of the paper, and the authors should have either done a test of the regular version of DynamicSWSF-FP against the modified version or also included the regular version of the algorithm in the series of experiments that they conducted in comparison to LPA*. Furthermore, while the authors have run experiments in grid-worlds and given theoretical examples of real world applications, it would have been interesting to actually see the results of the algorithm in a real world, dynamic environment.