**LISTED RESPONSE TO COMMENT/SUGGESTIONS OF**

***REVIEWER 1***

|  |  |  |  |
| --- | --- | --- | --- |
| ***No*** | ***Reviewer’s Comment/Suggestion*** | ***Author’s Comment*** | ***Action Taken\*\*\**** |
| 1 | Please explain the difference between reference [9] and this paper. They are both presented the MOD\* Lite methods. | Thank you for your comments and suggestions. Citation [9] is the prior version of this study. However, we extend the MOD\* Lite method in this paper with adding a soft computing genetic algorithm (MOGPP) and executing the tests from scratch. We also explained MOD\* Lite more deeply as well. | No actions taken. |
| 2 | Please check the type errors in the paper, such as "For the experimental study, MOD\* Lite is is realized," in Page 5 (Line 4-5). I find several errors. In Section IV, the word "multi objective" does not have the '-'. But, in the previous sections, the word is "multi-objective." Please correct them. | All “multi objective” phrases are replaced with “multi\*objective”, to keep consistency. Also other errors are typos are corrected. | Corrections are done in accordance with author reply. |
| 3 | In this paper, the MOD\* Lite method is applied to 2D UAV problem whose environment is represented by the square cells. Please explain how to transform the cell-base environment to the real space in real application. | For the sake of simplicity, we applied MOD\*Lite to a cell-based grid environment. As D\*Lite naturally does not restrict to any environment or domain (firmly environment independent), we could argue that MOD\*Lite is environment and domain independent as well. For real world applications, cell-based grid environment can be transformed into hexagonal form by adding diagonals. | No actions taken. |
| 4 | In Algorithm1, I suggest to put the PLAN() procedure to the top and indicate that it is the main procedure to avoid confusion. | Thanks for your suggestion. PLAN() procedure in Algorithm 1 is put at the top of other procedures. Also it is stated that PLAN() procedure is the main one in Section III-D. | Corrections are done in accordance with author reply. |
| 5 | In Section III, it seems not easy to be understand about the algorithms presented in this section. Can you illustrate the important concept of the algorithm by using another figure? |  |  |
| 6 | In Fig. 6, why MOA\* requires the most computing time? | In Figure 6, actually MOA\* does not have the most computing time, MOGPP takes the utmost time to find a path in all sensor range percents. | No actions taken. |
| 7 | For MOGPP method, please explain that how to judge the chromosome is non-dominated and how to update or replace the old chromosome. | In MOGPP, each chromosome encodes a valid path from initial location to target. Each generated path represents an individual in population. These individuals are kept in a directed acyclic graph to cope with multi objectivity. The vertices and edges represent individuals and domination of multi objective path costs of these individuals, respectively. If a path cost of an individual dominates to other's, an edge is established between these individuals' vertices. Non domination and equality do not come up with an edge. When an individual is generated and desired to be added to population, the cost function of this individual is compared with existing individuals' costs and required edge connections are established. The directed acyclic graph structure has the same essence and representation with MOD\* Lite' s priority structure.  At each evolution phase, predefined number of best individuals are chosen from topologically sorted directed acylic graph and transferred to new population. After selecting two mates by roulette-wheel selection method, crossover and mutation operations are applied to update old chromosomes.  As a single chromosome stands for a valid path in MOGPP, new generated children should also obey this rule. Thus, genetic operations must guarantee that new generated children are consistent and have valid paths. For crossover operation after choosing parents; one of the random intersecting cells of those parents are selected. Both parents' paths are splitted from that cell, switched with each other and merged to other parent's path starting from that cell. On the other hand, mutation is done as follows; a cell from corresponding individual' s path is selected randomly first. This cell is the reference point to split up the path into two sub-paths. Then, the sub-path which contains target location is thrown away and a random path to the target is generated instead.\* | No actions taken. |
| 8 | In Section V-B, page 10, the agent has found three paths with costs (15, 200), (18, 230) and (23, 260). The agent tends to choose the path with cost (18, 230). Please explain why it does not choose the path with (15, 200) since (15,200) dominates (18,230). | Thanks for your correction. The path costs are updated as (15, 260), (18, 230) and (23, 200) where none of the path costs could dominate each other. In such cases the middle one, (18, 230), is selected by agent. | Corrections are done in accordance with author reply. |
| 9 | In Section V, both MOD\* Lite and MOA\* method require re-plan while the environment changes. Please state that why MOA\* requires much time compared with MOD\* Lite. | As MOA\* is an off-line path-planning algorithm and need to re-plan from scratch when environment or agent state has changed. On the other hand, MOD\* Lite is an on-line incremental path planning algorithm just like D\* Lite and capable of considering only updated parts of environment during re-planning phase. That's why MOA\* takes much more time on finding a path compared with MOD\* Lite. | No actions taken. |

*\* Visualized crossover and mutation operations for MOGPP can be found on appendix.*

**Appendix**

**Crossover and Mutation Visualization for MOGPP**

A description...

