A Comparison of Distributed and Centralized Control for Bearing Only Emitter Localization with Sensor Swarms

Hans Schily
53343 Wachtberg
GERMANY

Folker Hoffmann
53343 Wachtberg
GERMANY

Dr. Alexander Charlish
53343 Wachtberg
GERMANY

name.surname@fkie.fraunhofer.de

ABSTRACT

The pursuit of superior situational awareness and the application of sensor swarms requires sensors to adapt their behaviour automatically, if managing the sensing tasks of individual assets could overload an operator. This paper compares different approaches on path planning for the localization of multiple targets in diverse scenarios with two sensor platforms carrying bearing only sensors. The work concludes that it is key to focus on target assignments in order to minimize the time it takes sensor platforms to localize targets from bearing measurements.

1.0 INTRODUCTION

Enhanced situational awareness is an important driver of advanced sensor systems. Especially the cooperation of multiple assets in sensor swarms poses new challenges to optimally control the sensor deployment while at the same time minimizing the workload of an operator. Controlling many sensors in order for them to solve joint or multiple tasks at once leaves many possibilities to the system designer for implementing the actual control structure.

The first option is to have a central controller that evaluates the joint action space of all sensors, or platforms carrying those sensors, and computes and communicates the best actions to each individual sensor carrier. This has been used by several works in the literature, for example [1, 2, 3]. Central control requires a very capable control unit in terms of computing power, because the joint action space of many moving sensors becomes very highly dimensional, making it more cumbersome to search and find the optimal actions. In contrast, the controllers could distribute the computational load and compute a joint solution by using their on-board resources to emulate a multiprocessor machine [4]. This would have the advantage of scaling the available computing resources linearly with the number of sensors participating in solving a joint task. Nevertheless, the action space does not reduce in its dimensionality and searching for the optimal action for all sensors remains a demanding task.

An alternative to these approaches is to plan actions for each platform locally, only addressing the specific sensor alone. The coordinated behaviour on a joint task for multiple sensors is achieved either through planning on correlated information on the world state or through understanding and anticipating actions and intentions of peers. The works in [5, 6] show examples where the computation is distributed based solely on shared information. The platforms know the current estimate of the targets' states and potentially the positions of the other UAVs, however, have no knowledge or assumptions about their future behaviour. The work in [7] shows an example where the intentions of the other platforms are considered. Each platform computes its own control vector, keeping the control of the others constant. Then it sends its control vector to the other platforms. This is repeated until the solutions converge. A disadvantage of decentralized approaches is that in general they do not solve the joint problem optimally. An advantage is the reduction of the joint action space to the individual action space of each sensor participating in solving a problem, effectively simplifying the search for locally optimal actions.

A third way to implement the sensor management is to have a central assignment algorithm, which creates a central association between platforms and targets. Then each platform only needs to compute the optimal path for its associated target, drastically reducing its computational demands. Such an approach is used in [8]. Most works in the literature chose one of these approaches without comparing it to possible alternatives. This paper evaluates the advantages and disadvantages of the central or distributed control approaches with regard to a bearing only localization problem with two platforms carrying bearing sensors.

2.0 METHODS

Each sensor platform with state $\boldsymbol{x} = (x, y, \alpha)$ comprising its position and heading carries two bearing sensors that are oriented outwards to its left and right side respectively. Each of those bearing sensors have a field of view of 120°, such that a sensor platform has a blind zone of 60° towards its heading and tail. Each sensor platform moves at constant speed v with a scalar control input u and the propagation model

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \begin{pmatrix} v\ \cos(\alpha_k + u) \\ v\ \sin(\alpha_k + u) \\ u \end{pmatrix}.$$

Both sensors of each platform generate bias free readings of all targets in their respective field of view with an additional normally distributed noise term *n* with fixed standard deviation

$$h(\boldsymbol{x}_t) = \text{atan2}(y_t - y, x_t - x) + n$$

and target state $\boldsymbol{x}_t = (x_t, y_t)$. For the sake of simplicity, the data fusion is implemented as a centralized least squares batch estimator [9]. Every platform has access to all recorded measurements at all times and computes an updated estimate of each static target as soon as new measurements are available. The estimation starts from a prior that is composed of one initial bearing measurement, ignoring the sensors' field of view and a range prior. The range prior is based on the true range to the target with a standard deviation of half of the true distance to each target. In addition to the least squares point estimate, a target's uncertainty matrix $\boldsymbol{P}$ is approximated from the Cramér-Rao Bound, evaluated under the assumption that the target is truly at its estimated position. The targets are also considered threats and the platforms are required to stay a certain distance (threat radius) away from the current target estimates.

In order to plan optimized trajectories that minimize the time until the localization of all targets the simulation implements different open loop planning algorithms. In general, the planning is executed *I* discrete time steps before execution. Therefore, any planner predicts the platform positions *I* steps into the future using the current control input and simulates ideal measurements based on the current target estimates generating a predicted world state, which is the basis for all planning. During each execution of the planning algorithm it checks, whether a target is sufficiently localized or not. It only considers non-localized targets during the planning.

Six different control strategies are evaluated: Distributed Tree Search (DTS), Distributed Optimizer (DO), Distributed Iterative Exchange of Plans (DIEP), Central Tree Search (CTS), Central Optimizer (CO) and Central Assignment (CA). In addition, the planning horizon h/I is varied, which is noted by a number as suffix to the acronym (e.g. DTS3). h denotes the total number of actions, while *I* is the number of times steps with constant control input. Not all actions, which a planner considers, are executed. Only the first action is applied I times to the sensor platforms, while the planner is executed after every *I*th action. The basic reward function is identical for all variants of the planning algorithm and is built around the Fisher Information

$$J = \frac{1}{\sigma^2 r^4}\begin{pmatrix} (y_t - y)^2 & -(x_t - x)(y_t - y) \\ -(x_t - x)(y_t - y) & (x_t - x)^2 \end{pmatrix}$$

a sensor with standard deviation σ at position $\boldsymbol{x}$ expects to gain from measuring a target at position $\boldsymbol{x}_t$ and using $r^2 = (x_t - x)^2 + (y_t - y)^2$. Based on a vector of actions $\boldsymbol{u}$ containing actions $u_{m,p}$ for each platform p and time step m the platform position can be predicted and the reward for those actions computes to

$$R(\boldsymbol{u}) = \sum_{k=i+I}^{h/I} \sum_{t \in T} \log\left(\det\left(J_{t,i} + \sum_{p \in P} \sum_{m=i+1}^{kI} w_{m,p,t} J_{t,m}\right)\right)$$

where the weight w is the probability of observing the target, ensuring that the planner prefers orientations, which cover most of the uncertainty area of the target estimate. P denotes the set of all platforms and i the current time step.

Actions, which lead to the platform being too close to the target are omitted from the search space during tree search. However, if no feasible path remains and the platform is forced to come close to a target, it becomes the goal of the platform to leave the threat radius d_{min} around the target as fast as possible. Tree search then myopically chooses the action, which maximizes the distance to the target. For the optimizers, the reward is replaced by

$$R_{collision}(\boldsymbol{u}) = \log(\det(J_{t,i})) - \sum_{p \in P} \sum_{m=i+1}^{h} \frac{d - d_{min}}{d_{min}} [\![if\ d < dmin]\!]$$

where d is the platform and time specific distance between a sensor platform and a target and $J_{t,i}$ the target prior with least information.

DTS, DIEP and CTS use Depth First Search in a search tree in order to find the best action to execute. They construct the tree by creating a branch from a root node for each feasible action out of a set of discrete actions. For each time step within the planning horizon the branching repeats from the children, therefore building a tree. The value of a leaf is judged from the sum of rewards over the path from root to leaf. In contrast, DO, CO and CA use a global optimization algorithm [10] in order to find the best action from an interval of feasible actions.

DO and DTS both compute actions for one sensor platform alone, from the shared globally available target estimates ($|P| = 1$). There could exist an individual instance of these planners on each sensor platform, hence making it a distributed planning method. On the contrary, CO and CST jointly plan actions for all sensor platforms, requiring some central node that commands all platforms.

DIEP and CA implement different behaviours. CA performs one-to-one platform to target assignments before optimizing the path of each platform individually, only regarding its associated target. The assignment is done globally, giving CA the character of a centralized planning algorithm. The planner does not compute a novel assignment in each execution, but keeps the previous assignment and only computes a new assignment, if a sensor platform has solved its task (localized its assigned target) and hence becomes available. The assignment is determined using a standard linear sum assignment algorithm. DIEP follows a different strategy. Instead of letting each platform plan independent actions in parallel (or centralized), the planning process is serialized. One platform starts optimizing its actions assuming default actions for each other participating sensor platform and publishes its plan to all other platforms. Next, another platform optimizes its actions based on the known actions of previously published plans and default actions. A fixed amount of planning iterations in a fixed order is performed in this way. Therefore, the global plan of all sensor platforms is updated iteratively, while the action space for the actual optimization is kept smaller compared to the centralized planning, because each sensor platform only optimizes over its own actions.

3.0 EVALUATION

For evaluation, the planning methods given above run on three different types of scenarios (cp. Figure 3-1), all exposing two sensor platforms to a set of static targets: *Angle*, *Horizontal* / *Vertical* and *Circle*. The *Angle* scenarios comprise two targets at a distance of 1300 meters to the origin, which are separated by a given angle symmetrically around the direction of north. The *Horizontal* and *Vertical* scenarios also comprise two targets, but the two targets are located 1000 meters from the origin and are collinear positioned on the x or y direction, respectively. In contrast, the number of targets in the *Circle* scenarios amounts to either four or eight and the targets form a circle centred at the origin and with a radius of 1300 meters.

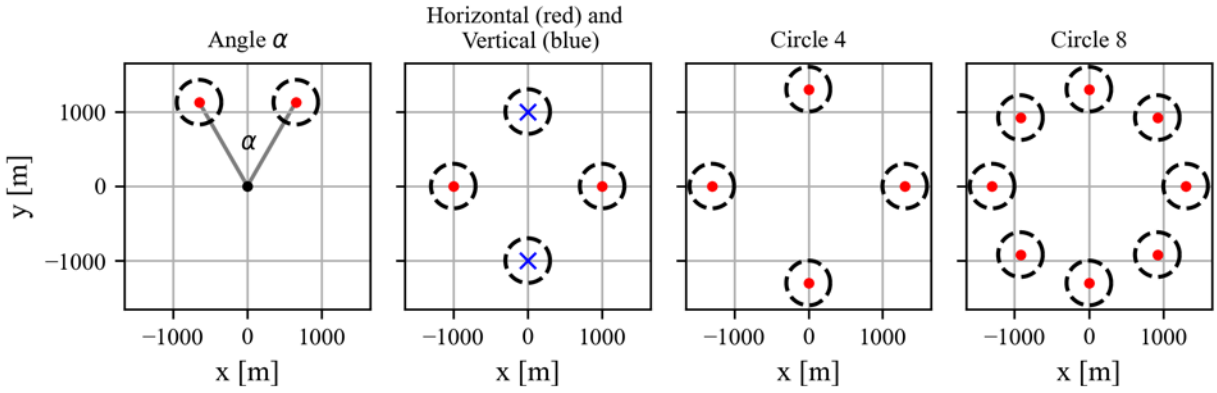

Figure 3-1: Evaluation scenarios. *Angle* scenarios comprise multiple scenarios with targets at a distance of 1300 meters, but with differing separation angles of two targets in the north of the map.

In addition to the different scenario types, two distinct starting configurations for two sensor platforms are tested. Either the platforms' initial positions are (0, 0) and (100, 0) with both having an initial heading of 0°, or the platforms start from (-200, 0) and (200, 0) with initial headings of -180° and 0°, respectively. The first start configuration of the platforms is referred to as *Same*, while the second is called *Opposite* referring to the initial headings of the platforms. The sensor platforms move at a constant speed of 100 meters per second and the tree search algorithms are limited to choose an action from five discrete angles: -20°, -10°, 0°, 10° and 20°. The planners using optimization methods constrain their actions to the interval (-20°, 20°). Each planning step has a duration of two seconds. A sensor platform takes measurements of all visible targets every second. Therefore, each action is executed in two consecutive time steps ($I = 2$).

All simulations use 100 Monte Carlo runs, making the evaluation robust to the noise induced through the simulated measurements. In addition, although some planning algorithms are suitable for parallelization they all run in a single thread, simplifying the implementation of the methods. As performance measure the number of steps until the localization of all targets is measured. A target is considered to be localized, if its expected localization error from its error covariance matrix $\boldsymbol{P}$

$$\sqrt{trace(\boldsymbol{P})} < 10,$$

is below 10 meters. All sensors produce noisy bearing readings with a standard deviation of two degrees. The minimum distance all sensor platforms shall keep from any target is set to 300 meters.

3.1 *Angle* Scenarios

Figure 3-2 summarizes the results of the simulations on the *Angle* scenario for angles of 0°, 15°, 30°, …, 165° and 180° and both starting configurations. It illustrates the median time until localization over the given range

of scenarios and with both starting configurations of the two platforms. For the configuration *Same* the 95% confidence regions are plotted additionally.

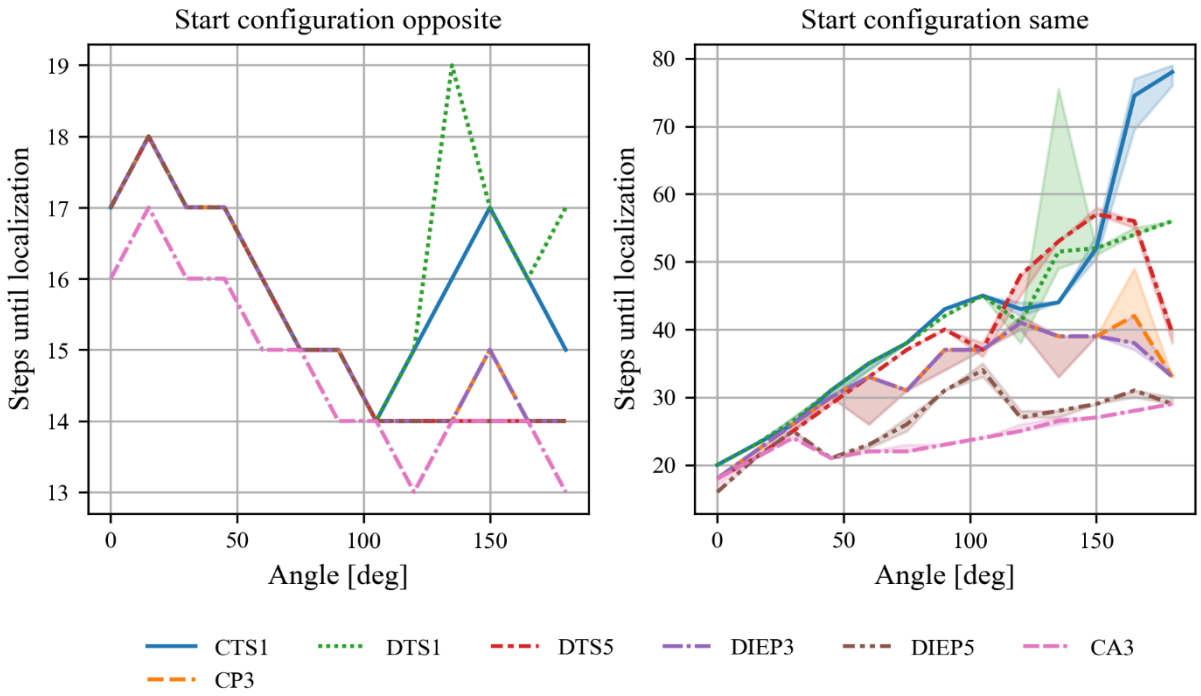

Figure 3-2: Comparison of the median time until localization of the planning methods and start configurations for the *Angle* scenario. Mind the different scales of left and right plot. Error tubes are omitted for start configuration *Opposite*, because the confidence interval is small. DTS5, DIEP3, DIEP5 and CP3 show very similar behaviour for the *Opposite* start configuration, while CA3 beats the other methods in both start configurations.

In both configurations, the CA3 method shows the best results while the myopic planners (CTS1 and DTS1) perform worst. CP3 and DIEP3 show almost indistinguishable behavior. Only for start configuration *Same* and the *Angle* scenario with targets being 165° or 180° apart these planners show distinct results. Furthermore, it is overall faster to let the sensor platforms start in the configuration *Opposite* compared to the starting configuration *Same*. For larger angular separations of the targets, the difference in median time until localization also becomes larger in between the control methods. Overall, a longer planning horizon is advantageous for each method individually, while centralized planning beats distributed planning and the central assignment of targets to the platforms performs best.

The exception from this pattern is the DTS5 planner for the *Same* starting configurations and scenarios with and angular separation between 120° and 165° of the targets. This is, because the myopic planner encounters a situation where it is advantageous so split the paths of both platforms for immediate reward. It then either visits both targets with one platform, or directs the platforms to observe both targets simultaneously. The latter tactic leads to longer times until both targets are localized. In contrast, the longer planning horizon of the DTS5 planner finds more reward in directing both sensors to the closest target and afterwards to the other farther away target, which takes more time than directly visiting both targets with one platform, but less than trying to observe both targets with each platform simultaneously.

3.2 *Horizontal*, *Vertical* and *Circle* Scenarios

The evaluation on the *Horizontal*, *Vertical*, *Circle* 4 and *Circle* 8 scenarios is performed with both starting configurations, *Same* and *Opposite*. The median time until localization of all targets, together with the 95% confidence interval of all simulation runs is depicted in Figure 3-3. The large spread in the results for a planning algorithm and scenario is due to multiple characteristic paths an algorithm finds depending on the exact bearing measurements it receives. The *Horizontal* and *Vertical* scenarios together with the *Same* starting configuration show that, as expected, the centralized planning algorithms localize the targets faster and a longer planning horizon also helps localizing targets, regardless of implementing the path planning in a centralized or distributed manner. The wide spread in outcomes for the DTS3 planner for the *Vertical* scenario is due to two different types of paths. If the planning algorithms for both sensor platforms decide to pursue the same target, they take more time compared to the case when each platform localizes a different target. The *Horizontal* scenario in the *Opposite* start configuration is the easiest, since both sensor platforms start by approaching the targets. The *Vertical* scenario in the *Opposite* initial configuration shows similar qualitative results compared to the *Same* starting configuration, with the exception of the DTS5 planner. The DTS5 planner directs both sensor platforms towards the same target north or south after localizing the targets east and west in the *Opposite* configuration, depending on whichever target is estimated to be closer to the platforms. This is suboptimal compared to splitting the platforms for localizing the north and south targets, which is preferred by the other planning algorithms.

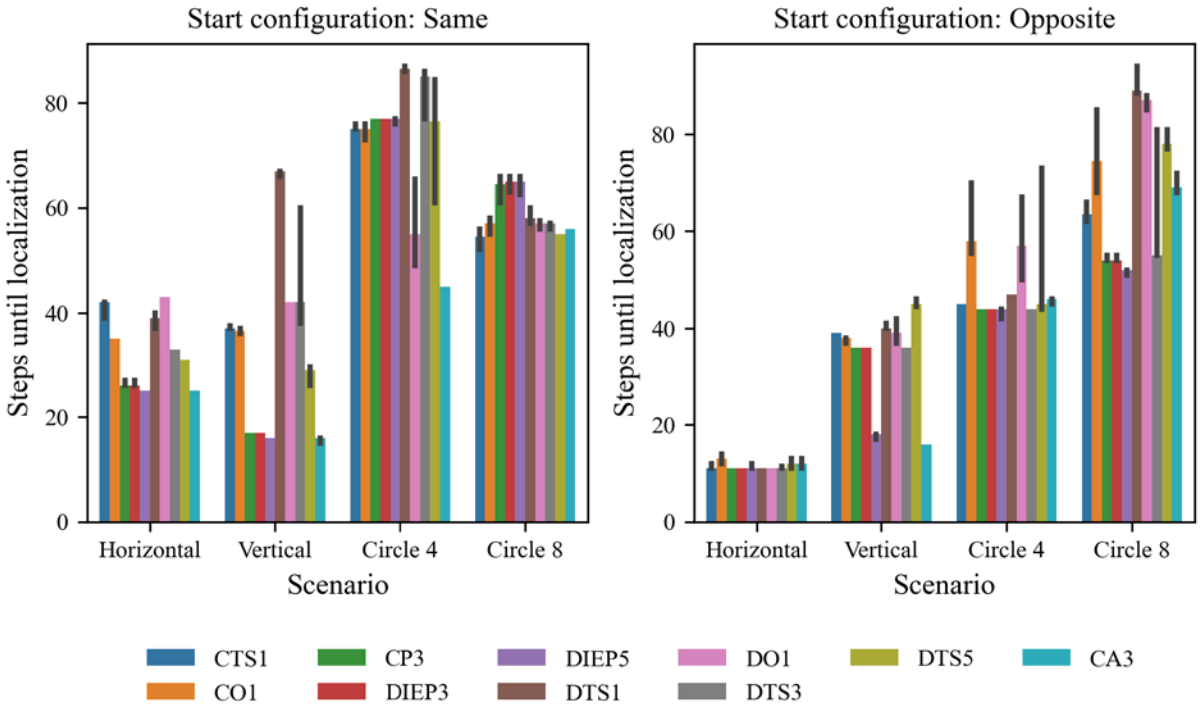

Figure 3-3: Comparison of different planning strategies for *Horizontal*, *Vertical*, *Circle* 4 and *Circle* 8 scenarios. The error bars show the 95% confidence interval.

The good results of the CA planner for the *Circle* 4 scenario show, that the quickest strategy for path planning is to pick targets successively and localize them one after the other. However, while in the *Opposite* configuration most planners manage to behave similarly to CA, in the *Same* configuration the sensor platforms are incentivised by the reward to each try to observe two targets at a time, which turns out to be slower in localizing all targets. A similar effect is observed for the *Circle* 4 scenario in the *Opposite* starting configuration. The constraint to discretized actions helps the CTS1 planner, because it is not able to optimize

its steering to observe multiple targets at once as well compared to the continuous action space of other planners. Therefore, it tends to approach a specific target faster, which is the better strategy for this scenario. In contrast, the CO1 planner behaves worse though having the less constrained action space. A similar logic applies to results of the distributed versions of planning algorithms in this scenario. In the *Circle* 8 scenario, there are eight targets, which need to be localized by two sensor platforms. For the initial configuration *Same* the CA performs very well, while is does not perform as well for the *Opposite* configuration. During the *Same* scenario the platforms split into clock and anti clockwise directions, because they are close to each other when that decision is made. While it seems again to be the best strategy to visit the targets sequentially, the CA algorithm struggles to find the optimal assignment for the *Opposite* configuration. This is, because the platforms are not next to each other when deciding, if they visit the targets clock or anti clockwise. Hence, this decision is less often successful compared to the *Same* starting configuration for the CA algorithm. A similar observation is true for the other planning algorithms. While in the *Same* scenario the splitting of the platforms is the optimal decision, the distance between the sensor platforms often leads to suboptimal decisions on the visiting order of the targets in the *Opposite* case. An exception to this are CTS3 and the DIEP planning algorithms. Here the separation of the platforms in the *Same* configuration actually happens too early. While gaining additional reward through observing multiple targets during the early phase of the scenario, the sensor platforms often have to revisit targets, making this strategy less effective.

3.3 Runtime Comparison

Finally, Figure 3-4 highlights the runtime of the planning algorithms for four different scenarios with two, four or eight targets present. The time measured is the time it took an algorithm to optimize the action of the next time step for all platforms. Therefore, one could argue, that the measured time for DTS1, DO1, DTS3, DTS5 and DO3 must be divided by the number of platforms in order to achieve a good prediction of a more performant implementation, because they would run in parallel and on different devices instead of sequentially in one thread.

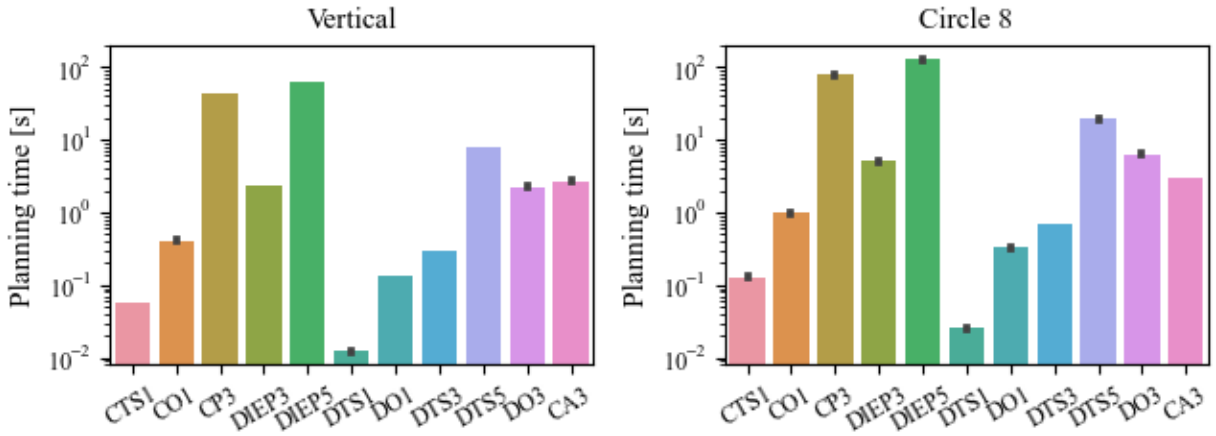

Figure 3-4: Cumulative time each planner requires on one processor for scenarios with a different amount of targets.

Furthermore, in comparison to the centralized algorithms, the distributed path planning needs fewer computational resources, which is also expected due to the reduced action space of the distributed approach. The DIEP algorithm is an interesting alternative to the central planning of paths from the perspective of necessary computational resources, since it achieves comparable results at lower computational costs.

4.0 CONCLUSION

This work evaluates different path planning algorithms for two sensor platforms equipped with two bearing sensors on a set of scenarios with a varying number of targets. The planning algorithms are compared with regard to the time until all targets in a scenario are localized and with regard to their necessary computation time. Overall, the CA algorithm shows very promising results and appears to implement a good compromise between centralized and distributed planning methods. Its only weakness, in the experiments conducted, is the simultaneous localization of many targets that are distributed uniformly in all directions with respect to the starting point of the sensor platforms (*Circle* 8). Solving the path planning problem through target assignment algorithms is especially interesting since there exist approaches on solving linear assignment problems on distributed systems, only connected through a dynamic communication graph [11].

5.0 REFERENCES

[1] Dogancay, K. (2012). UAV Path Planning for Passive Emitter Localization. IEEE Transactions on Aerospace and Electronic Systems, 48(2), 1150–1166.

[2] Leung, C., Huang, S., Kwok, N., & Dissanayake, G. (2006). Planning under uncertainty using model predictive control for information gathering. Robotics and Autonomous Systems, 54(11), 898–910.

[3] Hoffmann, F., Charlish, A., & Koch, W. (2016). Trajectory Optimization for Multi-Platform Bearing-Only Tracking with Ghosts. Proceedings of the 19th International Conference on Information Fusion (FUSION), 39–44.

[4] Grocholsky, B. (2002). Information-Theoretic Control of Multiple Sensor Platforms. University of Sydney.

[5] Drake, S., Brown, K., Fazackerley, J., & Finn, A. (2005). Autonomous Control of Multiple UAVs for the Passive Location of Radars. *2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, *17*, 403–409.

[6] Grocholsky, B., Makarenko, A., & Durrant-Whyte, H. (2003). Information-Theoretic Coordinated Control of Multiple Sensor Platforms. *IEEE International Conference on Robotics and Automation (ICRA)*, 1521–1526.

[7] Hoffmann, G. M., & Tomlin, C. J. (2010). Mobile Sensor Network Control Using Mutual Information Methods and Particle Filters. *IEEE Transactions on Automatic Control*, *55*(1), 32–47.

[8] Sarunic, P., & Evans, R. (2014). Hierarchical Model Predictive Control of UAVs Performing Multitarget-Multisensor Tracking. IEEE Transactions on Aerospace and Electronic Systems, 50(3), 2253–2268.

[9] Bar-Shalom, Yaakov, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[10] Storn, R and Price, K, *Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*, Journal of Global Optimization, 1997, 11, 341 - 359.

[11] Chopra, S., Notarstefano, G., Rice, M., & Egerstedt, M. (2017). *A Distributed Version of the Hungarian Method for Multirobot Assignment*. IEEE Transactions on Robotics, *33*(4), 932–947.