A comparative study of task assignment and path planning methods for multi-UGV missions

Johan Thunberg and Petter Ögren
Department of Autonomous Systems
Swedish Defence Research Agency (FOI), SE-164 90, Stockholm, Sweden

David A. Anisi
Division of Optimization and Systems Theory
Royal Institute of Technology (KTH), SE-100 44, Stockholm, Sweden

Many important problems involving a group of unmanned ground vehicles (UGVs) are closely related to the multi traviling salesman problem (m-TSP). This paper comprises a comparative study of a number of algorithms proposed in the litterature to solve m-TSPs occuring in robotics.

The investigated algoritms include two mixed integer linear programming (MILP) formulations, a market based approach (MA), a Voronoi partition step (VP) combined with the local search used in MA, and a deterministic and a stocastic version of the granular tabu search (GTS).

To evaluate the algoritms, an m-TSP is derived from a planar environment with polygonal obstacles and uniformly distributed targets and vehicle positions.

The results of the comparison indicate that out of the decentralized approaches, the MA yield good solutions but requires long computation times, while VP is fast but not as good. The two MILP approaches suffer from long computation times, and poor results due to the decomposition of the assignment and path planning steps. Finally, the two GTS algorithms yield good results in short times with inputs from MA as well as the much faster VP. Thus the best performing centralized approach is the GTS in combination with the VP.

I. Introduction

Often missions with systems of unmanned ground vehicles (UGVs) translate into $\mathcal{NP}$-hard problems. A way to simplify the problem is to decompose the mission into a set of subproblems. One such decompositioning is to design a task assignment problem which allocates tasks to the vehicles, and a path planning box which solves the path planing problem for each vehicle. Beneath the path planning box low level boxes dealing with trajectory planing and vehicle actuators can be used.

If task assignment and path planning occurs in a cooperative manner, *i.e.* there is no clear dividement between task assignment and path planning, it will be referred to as cooperative task assignment and path planning (CTAPP) or a one-box approach. On the contrary if task assignment and path planning occurs sequentially, *i.e.* there is no feedback from path planning to task assignment, it will be referred to as sequential task assignment and path planning (STAPP). In[8] a taxonomy is presented over many multivehicle task assignment problems.

A multivehicle system have many axis of freedom[10] such as centralized system *vs.* decentralized system and explicit comunication *vs.* implicit communication. A good algorithm should be able to handle these different aspects. Many algorithms have been suggested over the years, but comparative studies of the their performance are still lacking, presenting such a comparative study is the main contribution of the paper in hand. In[12] a comparative study over UAV path planning, can be found.

Some of the main research directions include meta heuristic approaches such as tabu search[3,7,14,15] simulated annealing or ant colony optimization.[7] Market based approaches where the system of vehicles shall resemble a market economy, have also drawn much attention lately.[4,13,18,21,22] Often the missions

can be formulated as mixed integer linear programming (MILP) problems.[1,5,17] Some MILP approaches are introduced in[11] that solve task assignment and path planning sequentially (STAPP) for a problem similar to the multi traveling salesman problem (m-TSP). In missions including threats, voronoi diagrams are commonly used.[2,9] Voronoi diagrams can also be used in task assignment[6] or in obstacle avoidance.[16]

The outline of this paper is as follows. In Section II the problem will be defined on which the algorithms are compared and in Section III, the algorithms will be presented. Then, in Section IV it will be described how they are implemented and results will be presented. Finally, conclusions are drawn in Section V.

II. Comparative study

There are assumed to be a group of N vehicles that live on a connected compact subset of $\mathbb{R}^2$, for simplicitly a square. This set, which will be referred to as Q, will have polygonal holes representing obstacles. Targets or tasks are defined as points in Q that must be visited to fulfill an objective.

In this setting the algorithms will be compared on the following problem.

Problem 1 (Multi-vehicle multi-task problem)**.** *Consider N vehicles and M targets uniformly distributed on Q. Each vehicle has the capacity to visit all targets and each target must be visited by one vehicle and one vehicle only. Find the optimal paths for the vehicles so that all targets are visited and either the mission completion time (a) or the fuel consumption (b) is minimized.*

Constant vehicle speed is assumed and minimizing completion time is hence equivalent to minimizing the length of the longest path for any vehicle. Therfore, Problem 1a is refered to as the *min-max* objective. Minimizing fuel consumption for the vehicle group is further more assumed to be the same as minimizing the total distance traveled for the vehicle group. Problem 1b is thus referred to as the *min-sum* objective.

III. Algorithms

In this section we first list the algorithms being compared, and then give a more detailed description of them. The first algorithm is the market based approach, presented in.[21] The next two methods are two of the mixed integer linear programming (MILP) methods presented in[11] and the fourth method is a voronoi partitioning algorithm inspired by.[6] Also included in this study is the granular tabu search[20] and a stochastic modification of it. The algorithms are slightly modified to fit Problem 1 and compared in Monte carlo simulations. The methods are not only chosen on the basis of their closeness to optimality, characteristics like computational time and possibilities to work in different systems such as decentralized or centralized ones are also important.

A. Market based approach

This market based algorithm is presented in[21] and is a CTAPP algorithm. The algorithm will be referred to as MA. The mission is to allocate a group of targets to a group of vehicles. For that purpose bidding rules are introduced. Assume there are N vehicles $r_1, ..., r_N$ and M currently unallocated targets $t_1, ..., t_M$. Assume further that a set of targets T_i is assigned to each vehicle r_i, *i.e.* there is a set $\{T_1, ..., T_N\}$. Let PC(r_i,T_i) denote the minimum path cost of vehicle r_i when visiting all targets in T_i. The values of PC(r_i,T_i) are calculated locally using a meta-heuristic procedure. The two team objectives are

- Problem 1a (min-sum): $\min_T \sum_j$ PC(r_i,T_i), and
- Problem 1b (min-max): $\min_T \max_j$ PC(r_i,T_i)

Each unallocated target is bidden upon by vehicles in a so-called bidding round according to a bidding rule. The vehicle with the lowest bid will win, and the target will be allocated to that vehicle.

Bidding Rule Robot r bids on target t the difference in performance for the given team objective between the current allocation of targets to vehicles and the allocation that results from the current one if robot r is allocated target t. (Unallocated targets are ignored.)

The advantage of having this auction based system, is that each vehicle is decoupled from the others in the bidding process, which makes the system robust. The process will continue if a vehicle suddenly stops

responding, due to a software or hardware failure. The targets that were allocated to that vehicle will be unallocated once again. Since an auctioneer has to be present, the system is not fully decentralized. If the auctions are administrated by a single unit, there will be a single point of failure. However the process can be made more robust if the auction is administrated by a group of vehicles.

Each vehicle maintains a path visiting the targets that are allocated to it. A bid on a target is performed in the following steps. First the target is inserted into all possible positions of the path associated with the vehicle, and new temporary paths are created.

Each temporary path, will now be improved by two sequentially performed local search heuristics, or hill climbing algorithms. The first algorithm uses the 2-opt improvement rule, which takes the path and reverses the order of each of its continuous subpaths, takes the best path and repeats the process until no improvement is possible. The second algorithm is performed on the path that is generated by the first algorithm and uses the 1-target 3-opt improvement rule, which takes a target and puts it between two other targets.

B. Two MILP approaches

The mixed integer linear programming (MILP) algorithms described below can be found in.[11] The algorithms will be referred to as MILP1 and MILP2 and are STAPP algorithms. These algorithms are only applied to Problem 1a, i.e. the min-max objective, and we use the notation of [12] below.

The objective is to calculate T^*, where

$$T^* \equiv \min_j \max_{A(j)_i} T(j)_i^*,$$

is the least maximum traveling cost among all vehicles finishing all its subassignments. The assignment $A(j) \in A$ is one of the feasible assignments, $A(j)_i$ is the the subassignment given to the i^{th} vehicle, and $T(j)_i^*$ is the traveling cost for the i^{th} vehicle finishing all its subassignments.

Two problems are described in ,[12] one in which the vehicles need to return to their start positions and one in which they do not. Only the latter one will be considered, since it coincide with Problem 1 for the min-max objective. The cost $C_0(i,k)$ is the cost of traveling from the start position of the i^{th} vehicle to the position of the k^{th} target, and C_0 is a matrix. The function $\eta(\upsilon,\omega)$ is defined as $\eta(\upsilon,\omega) = (\upsilon-1)N - \upsilon(\upsilon-1)/2 + \omega - \upsilon$, where $\eta \leq \upsilon$. The cost vector $c(\eta(\upsilon,\omega))$ is the cost of traveling between the target υ and ω.

An exact formulation E and two non-exact algorithms H_1 and H_2 are proposed for the problem. The two non-exact methods are STAPP algorithms. The exact MILP formulation E is:

$$F_E\text{: minimize } r$$

$$\sum_{k=1}^{M} a_{ij}^k \leq 1 \quad \forall i,j, \tag{1}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{q} a_{ij}^k = 1 \quad \forall k, \tag{2}$$

$$a_{ij}^k \in \{0,1\} \quad \forall i,j,k, \tag{3}$$

$$\sum_{k=1}^{M} a_{i(j+1)}^k \leq \sum_{k=1}^{M} a_{ij}^k \quad \forall i,j, \tag{4}$$

$$a_{ij}^{\upsilon} + a_{i(j+1)}^{\omega} + a_{ij}^{\omega} + a_{i(j+1)}^{\upsilon} = 2y_{i\eta(\upsilon,\omega)}^j + \tilde{y}_{i\eta(\upsilon,\omega)}^j \quad \forall i,j,\upsilon,\omega, \tag{5}$$

$$y_{i\eta(\upsilon,\omega)} = \sum_j y_{i\eta(\upsilon,\omega)}^j \quad \forall i,\upsilon,\omega, \tag{6}$$

$$y_{i\eta(\upsilon,\omega)}^j \in \{0,1\}, \quad \tilde{y}_{i\eta(\upsilon,\omega)}^j \in [0,1] \quad \forall i,j,\upsilon,\omega, \tag{7}$$

$$\sum_{k=1}^{M} C_0(i,k)a_{i1}^k + \sum_{\upsilon=1}^{n-1}\sum_{\omega=\upsilon+1}^{M} c(\eta(\upsilon,\omega))y_{i\eta(\upsilon,\omega)} \leq r \quad \forall i, \tag{8}$$

where $i \in \{1, ..., M\}$, $j \in \{1, ..., q\}$ for (1), (2) and (3) or $j \in \{1, ..., q-1\}$ for (4), (5), (6) and (7). Each vehicle has a capacity limit of handling q targets. The binary variable a_{ij}^k only equals 1 when the k^{th} target is contained in the j^{th} so-called *room* of the i^{th} vehicle. Each room can at most contain one vehicle, and every vehicles must be contained in some room, (1) and (2), respectively. The rooms of a vehicle must be filled in ascending order, (4). In (5) the binary variable $y^j_{i\eta(\upsilon,\omega)}$ equals 1 if target υ and ω are contained in the j^{th} and the $(j+1)^{\text{th}}$ room of the i^{th} vehicle. If only one room is filled, $\tilde{y}^j_{i\eta(\upsilon,\omega)}$ equals 1. In (6) the room index is reduced and the binary variable $y_{i\eta(\upsilon,\omega)}$ is used in the total cost of each vehicle, (8). Equation (8) is the total cost of each vehicle when finishing all its tasks. The amount of binary variables needed for E is of order MN^2q.

The only modifications that need to be done to adapt this formulation to the formulation in Problem 1, is that the room size of each vehicle, q, equals M. This algorithm is only implemented for the min-max objective.

The two nonexact MILP formulations H_1 and H_2 are two step procedures. In the first step a partitioning MILP will run and allocate targets to vehicles, resulting in a target set for each vehicle. In the second step F_E will run on each vehicle-target set, *i.e.* F_E will run N times in total. The two different partitioning methods are presented below.

Partition H_1

$$F_1: \quad \text{minimize} \quad r$$

subject to:

$$\sum_{j=1}^{M} x_{ij} \leq q \quad \forall i, \tag{9}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j, \tag{10}$$

$$C_0(i,j)x_{ij} \leq r \quad \forall i, \tag{11}$$

$$x_{i,j} \in \{0,1\} \quad \forall i, j, \tag{12}$$

where vehicle index $i \in \{1, ..., N\}$ and target index j $\in \{1, ..., M\}$. This MILP requires only MN binary variables and reduces the computation time significantly. However in this partitioning the maximum distance between a target and a vehicle is minimized, which means that an optimal solution not necessarily is the voronoi partitioning. The voronoi partitioning is however always guaranteed to be an optimal solution to the problem above. The binary variable x_{ij} equals one if target j is allocated to vehicle j.

Partition H_2

$$F_2: \quad \text{minimize} \quad r$$

subject to:

$$\sum_{j=1}^{M} x_{ij} \leq q \quad \forall i, \tag{13}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j, \tag{14}$$

$$y_{i\eta(j,k)} \leq \frac{x_{ij} + x_{ik}}{2} \leq y_{i\eta(j,k)} + \frac{1}{2} \quad \forall i, j, k \ (j < k) \tag{15}$$

$$c(\eta(j,k))y_{i\eta(j,k)} \leq r \quad \forall i, j, k \ (j < k), \tag{16}$$

$$C_0(i,j)x_{ij} \leq r \quad \forall i, \tag{17}$$

$$x_{i,j}, y_{i\eta(j,k)} \in \{0,1\} \quad \forall i, j, k \ (j < k), \tag{18}$$

where $i \in \{1, ..., N\}$ and $j \in \{1, ..., M\}$. This program also minimizes the distance between the two targets that are assigned to the same vehicle and are furthest away from each other. This MILP requires M^2N binary variables. The two methods of solving a partitioning problem H_1 and then E, and H_2 and then E will from now on be referred to as MILP1 and MILP2 respectively.

C. Voronoi partitioning

This algorithm is a STAPP algorithm. First a task assignment process is performed using a so-called Voronoi partitioning,[6] then the local search described in (MA) above is applied.

This algorithm is inspired by,[6] but is not an exact implimentation. The same partitioning technique is used. The idea is to first allocate each target to its nearest vehicle. This is the same as constructing a so-called Voronoi diagram. After this partitioning has been performed, each vehicle will solve a traveling salesman problem (TSP) on its subset of targets, using the local search of (MA). The Voronoi partitioning is polynomial in time.

D. Granular tabu search

This is a slightly adjusted version of the granular tabu search (GTS), described in.[20] The algorithm is applicable to a broad class of combinatorial optimization problems, but is implemented for Problem 1. Some new features are added to make the algorithm a bit more effective, and the algorithm is adjusted to fit both objectives; 1a (min-max) and 1b (min-sum). A stochastic variant of this algorithm was also evaluated. These algorithms are CTAPP algorithms, and will be referred to as GTS and SGTS respectively.

This is a meta heuristic method that is applied to a feasible starting guess, the result of another algorithm. First one of the other algorithms are applied to Problem 1, and a solution is generated. This solution is the input to GTS, which aims at improving the solution further. The only constraint on the input solution is that it is feasible. However it will work better on a good solution because of the granularity threshold explained below.

The solution paths of all vehicles are represented by two vectors of size $N + M$, and are denoted IN and OUT. The first N elements of the vectors correspond to the vehicle positons and the last M elements correspond to the target positions. Each vehicle and target has a unique id-number in the interval (0,...,$N + M - 1$). The elements $IN(i)$ and $OUT(i)$ are the positions that are visited before and after target i on the vehicle path that visits target i.

A virtual depot position is created, which has id-number $N + M$. The elements $IN(i) = N + M \quad \forall i \in \{0, ..., N - 1\}$. The output, *i.e.* OUT, of the targets that are the last targets on a vehicle path also equals $N + M$. The virtual depot is an easy way to see where a path starts and ends. Of course, the inputs of the vehicles must always be the virtual depot.

The main idea behind tabu search heuristics are that local search or hill climbing algorithms are used, but to avoid local minima tabu lists are introduced. In these lists recent moves are stored and regarded as tabu, *i.e.* they cannot be used. Also when a local minimum has been reached, the algorithm is forced to make a *move* even though it will not improve the solution, see Figure 1. When the solution has not been improved for n iterations the algorithm stops and the best solution achieved during the algorithm is collected. In this algorithm an overriding criteria is used, where moves in the tabu lists can be used if it leads to a new global minimum.

The term *move* needs to be explained further. Moves or exchange operators operate on arcs, where an arc in this context is the directed closest path from one target or vehicle to another target. In a k-exchange or k-opt move, k arcs are removed and replaced by k other arcs, hence the cardinality of the neighborhod of all possible k-exchanges is $\mathcal{O}((N + M)^k)$. Special exchange operators that are 2-opt, 3-opt and 4-opt will be used that only has the neighborhod cardinality $\mathcal{O}((N + M)^2)$.

The following exchange/move operators will be used: The *two exchange*, the *target insertion*, the *two target insertion* and the *swap*. These are illustrated in Figure 2.

There are $(N + M)^2$ arcs, however the arcs between the vehicles are given a very large cost. Not all arcs are used in the algorithm, which is the core aspect of this algorithm. Only arcs that have an associated cost that is smaller than a certain granularity threshold υ. The threshold υ is calculated as

$$\upsilon = \beta \frac{z}{N + M},$$

where z is the cost of the solution of the input algorithm to GTS, *e.g.* the market based algorithm. Running GTS with MA as input algorithm will be referred to as GTS-MA. The threshold is a factor β of the average arc cost in the input solution to GTS. Various values of β can be used. If the solution is not improved for a certain number of steps, β is increased to a higher value during some consecutive iterations, and then decreased to a lower value again.

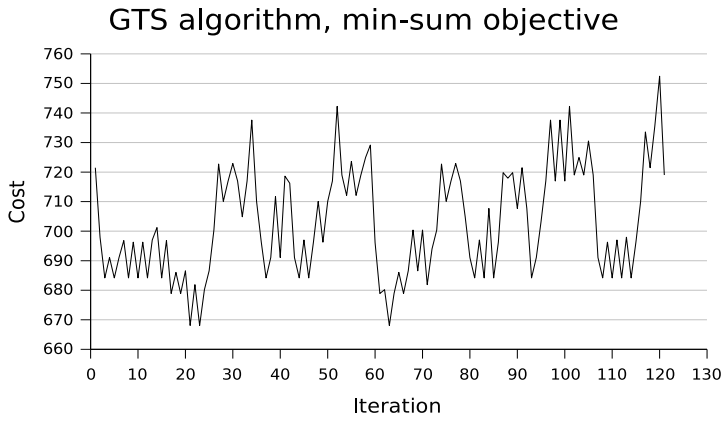

Figure 1. The cost at each iteration is shown for the tabu search algorithms (GTS) in a scenario with 20 vehicles and 40 targets.

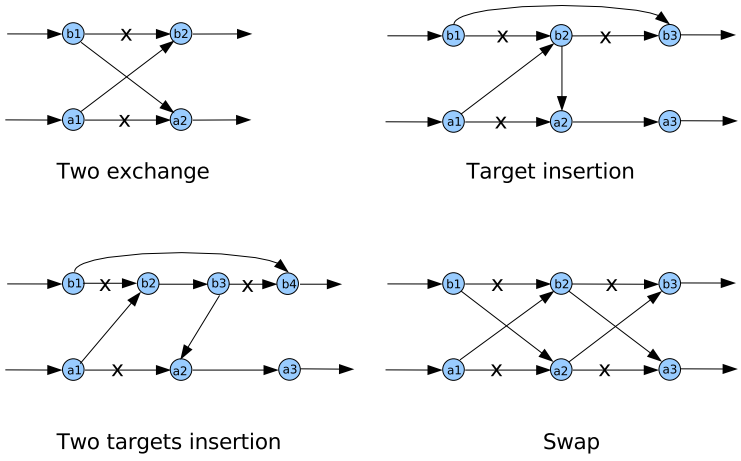

Figure 2. The four exchange operators. Here a and b are two different vehicle paths.

When using the vectors IN and OUT the change of the cost of the total solution when applying a move is calculated in constant time. Just to demonstrate this fact, the exchange operator with the min-sum objective will be looked at. The exchange is applied on arc $(a1, b2)$ in Figure 2. The change in total cost will be

$$-cost(a1, OUT(a1)) - cost(IN(b2), b2) +$$

$$cost(a1, b2) + cost(IN(b2), OUT(a1)),$$

where $OUT(a1) = a2$ and $IN(b2) = b1$. In the stochastic version of this algorithm, the best moves are not always chosen in the local search process. With some probability, other moves can also be chosen.

This concludes the algorithm descriptions.

IV. Implementation and Results

In this section we will describe the simulation environment and the results of running the algoritms.

A. Simulation environment

The simulation environment consists of two parts. In the first part a problem instance in terms of an environment is generated, and in the second part the algorithms are applied to this problem instance.

In the environmental generation part, first a scenario is generated according to Problem 1. From the scenario a visibility graph is generated. The third and last step of the environmental generation process is to calculate a cost matrix. This is a matrix where the (i,j):th entry is the optimal cost of traveling between vehicle/target i and vehicle/target j, where element (i,j) is equal to element (j,i). The minimum cost path between two nodes is calculated using the A*-graph search algorithm.

In the second part the algorithms are compared on the same cost matrix. All the algorithms except for the MILP methods are implemented in C++. The MILP methods are implemented in the matlab interface of CPLEX 8.0.

B. Results

The results are presented in Figures 3-8. First we describe the technical details of the data set and then we discuss the results one figure at a time.

In all scenarios there are 12 rectilinear possibly nonconvex polygonal buildings. For each combination of vehicles and targets there are 50 generated scenarios in figure 3, and 25 generated scenarios in Figures 5-8. The MILP algorithms are not included in Figures 5-8, because of the high computational times on large problems. The computational times presented in Figure 4 are found using CPLEX 8.0. on an Intel Pentium 4 2.40GHz with 512 MB ram, while the times in Figures 6 and 8 are from simulations on an Intel Pentium M 1.6 GHz with 1280 MB ram.

Running the algoritms on a small problem of 4 vehicles and up to 11 targets we get the costs depicted in Figure 3, where it can be seen that the GTS and SGTS find lower costs than the rest. The corresponding computation times are an order of magnitude higher for the two MILP approaches. These are therefore shown separately in Figure 4, where the median computation times are shown. In fact the worst computation times on these problems were as large as 200 seconds for the MILP1 algorithm on a problem with 4 vehicles and 10 targets.

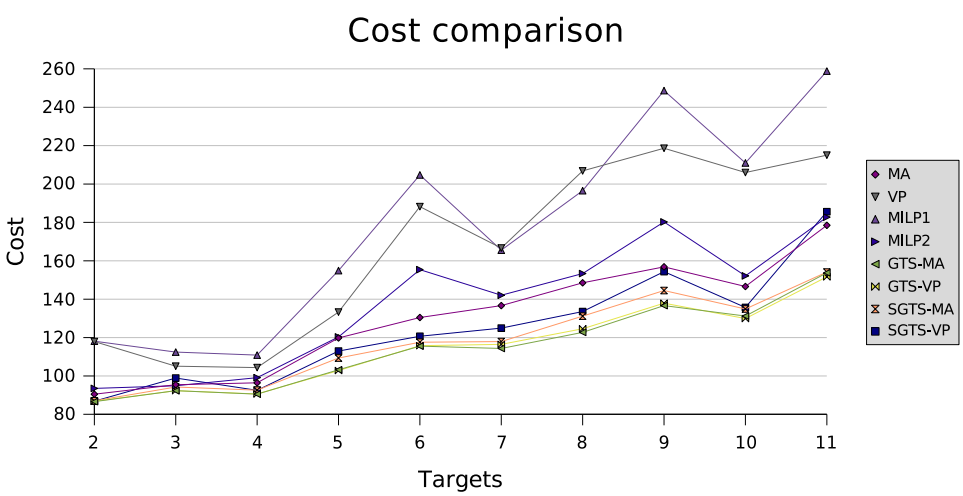

Figure 3. Cost comparison for 4-vehicle problems using the min-max objective. Note that VP and MILP1 find the highest costs, followed by MILP2 and MA while most of the time, all (S)GTS find lower costs.

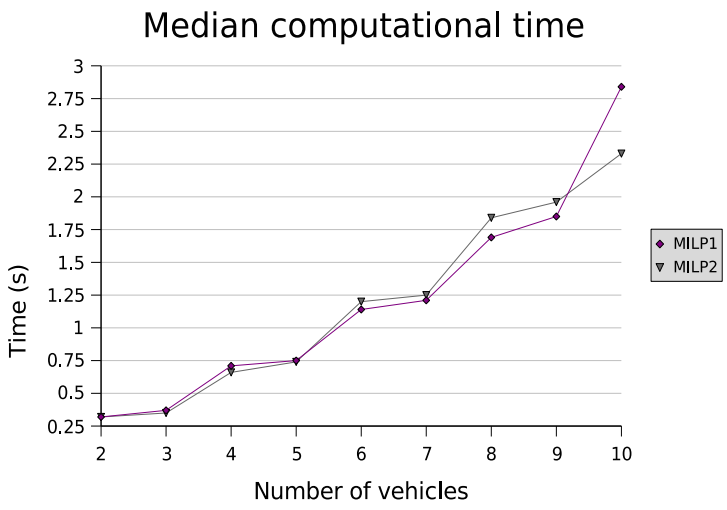

Figure 4. Time comparison for MILP algorithms on the 4-vehicle problems using the min-max objective. Computation times for the other algorithms on a larger problem can be found in Figure 6.

The results from a larger problem with 20 vehicles and up to 70 targets can be found in Figures 5-6. Here it can be seen that while the VP is faster than the MA, the MA finds lower costs than VP. Both of MA and VP are easily to implement in a decentralized manner to improve overall system robustness.

If, on the other hand, centralized computations can be made, the GTS and SGTS algorithms can improve the results of the VP considerably. Using MA or VP as inputs to GTS or SGTS the resulting costs differ only slightly, while the combined computation times are still much longer for the algorithms using MA as starting solution.

The differences between the GTS and the SGTS are small. Most of the time the GTS find better solutions than the SGTS, but sometimes the opposite holds.

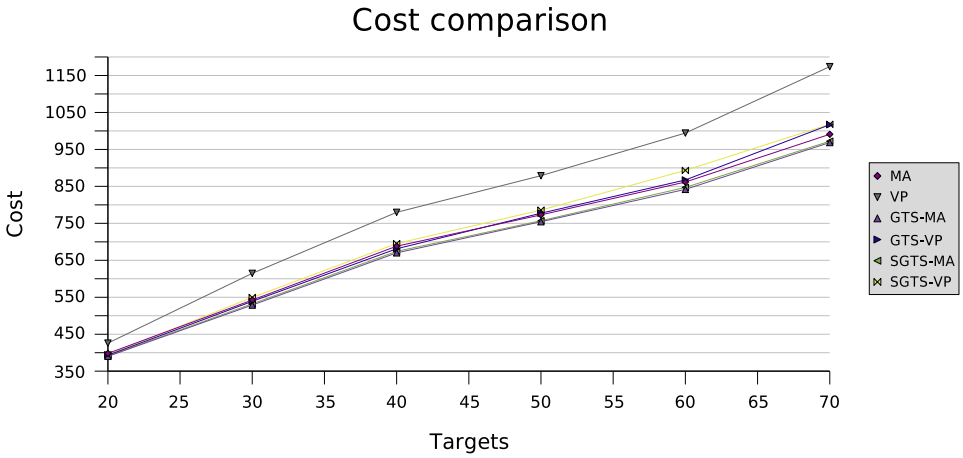

Figure 5. Cost comparisons for 20-vehicle problems using the min-sum objective. Note that the MA is consistently better than the VP, both as stand alone algorithms and as inputs to the (S)GTS. The best costs are found using MA-GTS and MA-SGTS.

Fixing the number of targets at 20 and varying the number of vehicles from two to ten you get the results depicted in Figures 7-8. The overall trends are the same as in Figures 5-6.

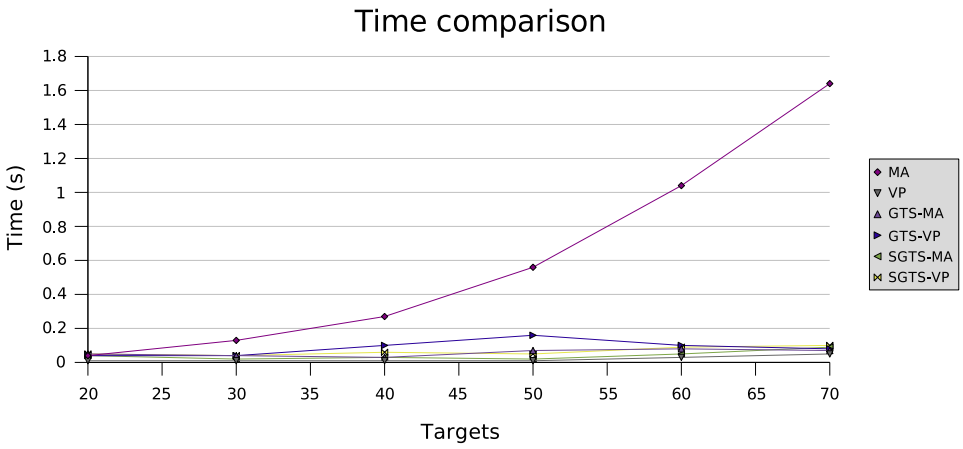

Figure 6. Time comparison for 20-vehicle problems using the min-sum objective. All VP algorithms finish well under 0.2 s, while the MA ones need up to 1.8 s. Note that the total running time of e.g.the GTS-MA is the sum of the MA and the GTS-MA curves.

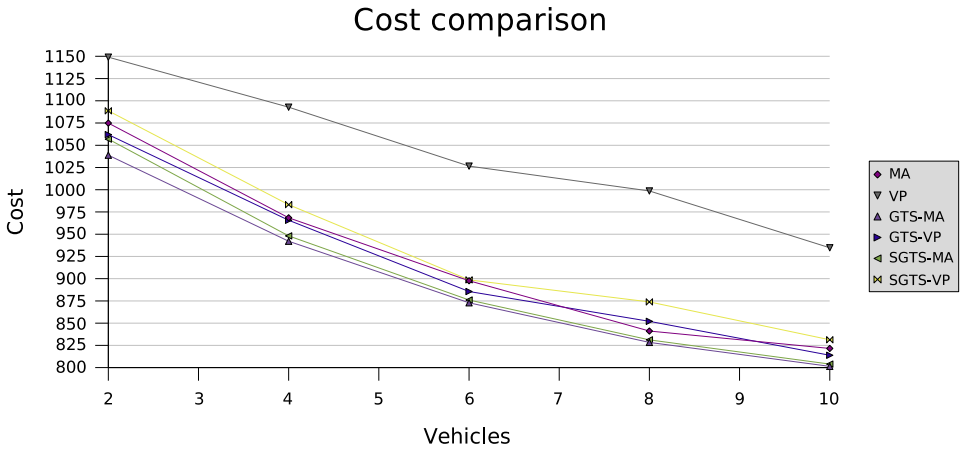

Figure 7. Cost comparison for 20-target problems using the min-sum objective. As in Figure 5 above, the best costs are found using MA-GTS and MA-SGTS.

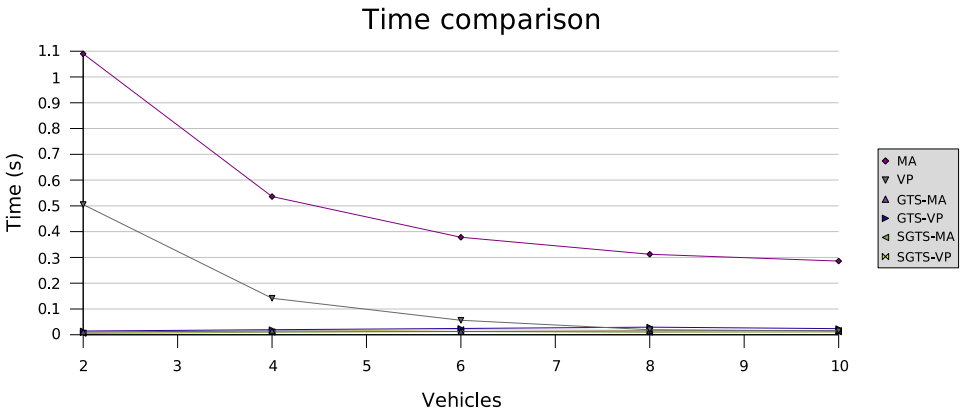

Figure 8. Time comparison for 20-target problems using the min-sum objective. As in Figure 6, the total computational times of the MA algorithms are larger then the VP ones.

As seen above, the division of the combined task assignment and path planning problem (CTAPP) into two separate problems (STAPP), as in VP and MILP1,2 can give lower computation times, but at the price of solution quality. The solutions can however be improved upon using heuristics, such as the (S)GTS, if centralized computations can be performed.

We were surprised by the performance of the algorithm proposed in [20] and we think that heuristics such as the GTS should be considered, before trading solution quality for computation time by partitioning hard problems into subproblems with different optimal solutions. For more results see.[19]

V. Conclusion

From the results it can be deduced that the STAPP algorithms VP, MILP1 and MILP2 are outperformed by the CTAPP algorithms MA, GTS and SGTS. The best algorithm was the granular tabu search, also when considering computational time. The market based algorithm (MA) is robust and easy to implement in a decentralized system, if it is possible to run the algorithms centralized, GTS is preferably used. On most problems, the computational times are low enough for the best algorithms to run on-line to respond to changes in the environmental information.

Acknowledgments

The first two authors would like to gratefully acknowledge the financial support by the Swedish Defence Materiel Administration (FMV) throuth the TAIS program, 297316-LB726396.

References

[1]M. Alighanbari, L.F. Bertuccelli, and J.P. How. Filter-Embedded UAV Task Assignment Algorithms for Dynamic Environments. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–15, 2004.

[2]R.W. Beard, T.W. McLain, MA Goodrich, and EP Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.

[3]R. Bent and P. Van Hentenryck. A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 38(4):515–530, 2004.

[4]M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2, 2003.

[5]L.F. Bertuccelli, M. Alighanbari, and J.P. How. Robust planning for coupled cooperative UAV missions. In *Proc. of the* 43^{rd} *IEEE Conference on Decision and Control, 2004. CDC.*, volume 3, 2004.

[6]E. Frazzoli and F. Bullo. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. In *Proc. of the* 43^{rd} *IEEE Conference on Decision and Control, CDC*, 2004.

[7]L.M. Gambardella, É. Taillard, and G. Agazzi. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. *Mcgraw-Hill'S Advanced Topics In Computer Science Series*, pages 63–76, 1999.

[8]B.P. Gerkey and M.J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939, 2004.

[9]Y. Guo, L.E. Parker, and R. Madhavan. Towards collaborative robots for infrastructure security applications. In *Proceedings of International Symposium on Collaborative Technologies and Systems*, pages 235–240, 2004.

[10]C. Jones, D. Shell, M.J. Mataric, and B. Gerkey. Principled Approaches to the Design of Multi-Robot Systems. *Proc. of the Workshop on Networked Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, 2004.

[11]Yoonsoo Kim, Da-Wei Gu, and Ian Postlethwaite. Real-time optimal mission scheduling and flight path selection. Submitted to IEEE Transactions on Automatic Control, accepted Jan. 12, 2007.

[12]Yoonsoo Kim, Da-Wei Gu, and Ian Postlethwaite. A comprehensive study on flight path selection algorithms. In *the IEE Seminar on Target Tracking: Algorithms and Applications*, pages 77–90, 2006.

[13]M.G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. *Robotics: Science and Systems*, 2005.

[14]G. Laporte et al. *Classical and Modern Heuristics for the Vehicle Routing Problem*. Blackwell Synergy, 1999.

[15]A. Le Bouthillier and T.G. Crainic. A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, 32(7):1685–1708, 2005.

[16]M. Lindhe, P. Ögren, and KH. Johansson. Flocking with Obstacle Avoidance: A New Distributed Coordination Algorithm Based on Voronoi Partitions. *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1785–1790, 2005.

[17]D. M. Panton and A. W. Elbers. Mission planning for synthetic aperture radar surveillance. *Interfaces*, 29(2):73–88, 1999.

[18]A. Pikovsky, P. Shabalin, and M. Bichler. Iterative Combinatorial Auctions with Linear Prices: Results of Numerical Experiments. *E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on*, pages 39–39, 2006.

[19]J. Thunberg. Task assignment and path planning for systems of surveillance unmanned ground vehicles. Master's thesis, The Royal Institute of Technology, KTH, 2007.

[20]P. Toth and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4):333–346, 2003.

[21]C. Tovey, M.G. Lagoudakis, S. Jain, and S. Koenig. The generation of bidding rules for auction-based robot coordination. *Proceedings of the 3rd International Multi-Robot Systems Workshop*, pages 14–16, 2005.

[22]R. Zlot and A. Stentz. Market-based Multirobot Coordination for Complex Tasks. *The International Journal of Robotics Research*, 25(1):73, 2006.