International Journal of Systems Science
Vol. 00, No. 00, 00 Month 20xx, 1–21

RESEARCH ARTICLE

Coordinated Road-Network Search Route Planning by a Team of UAVs

Hyondong Oh[a], Seungkeun Kim[b*], Antonios Tsourdos[a], and Brian A. White[a]

[a] *Cranfield University, Cranfield, UK*; [b] *Chungnam National University, Daejeon, South Korea*

(*Received 00 Month 20xx; final version received 00 Month 20xx*)

This paper presents a road-network search route planning algorithm by which multiple autonomous vehicles are able to efficiently visit every road identified in the map in the context of the Chinese postman problem. Since the typical Chinese postman algorithm can be applied solely to a connected road-network in which ground vehicles are involved, it is modified to be used for a general type of roadmap including unconnected roads as well as the operational and physical constraints of UAVs. For this, a multi-choice multi-dimensional Knapsack problem is formulated to find an optimal solution minimising a flight time and then solved via mixed integer linear programming. To deal with the dynamic constraints of the UAVs, the Dubins theory is used for path generation. In particular, a circular-circular-circular type of the Dubins path is exploited based on a differential geometry to guarantee that the vehicles follow the road precisely in a densely distributed road environment. Moreover, to overcome the computational burden of the multi-choice multi-dimensional Knapsack algorithm, a nearest insertion and auction based approximation algorithm is newly introduced. The properties and performance of the proposed algorithm are evaluated via numerical simulations operating on a real village map and randomly generated maps with different parameters.

Keywords: Road-Network Search Route; Dubins Path; Unmanned Aerial Vehicles; Mixed Integer Linear Programming; Auction Negotiation

1. Introduction

In the last two decades, great efforts have been made to investigate the benefits of a cooperative system composed of multiple unmanned aerial vehicles (UAVs) for both military and civil applications (Gunetti et al. 2011, Choi et al. 2011). The cooperative multiple unmanned systems can provide higher survivability in dangerous circumstances due to the inherent redundancy, and it can then contribute to the overall system to maintain or succeed in its given mission (King et al. 2006). Another merit of the cooperative unmanned system is that it can give operators better understanding of a complex environment, e.g. a battlefield or an area of natural disaster (Casbeer et al. 2006). In particular, efficient algorithms such as sensor fusion, consensus between multi-agents, and decision making need to be developed for dealing with multiple sensing sources for intelligence, surveillance and reconnaissance (ISR) mission (Yu and Wang 2012). One of the primary requirements of this sort of ISR mission using a cooperative system is a road-network search route planning which can be related to vehicle routing problem: how multiple UAVs can patrol or cover a specific region of interest having a complex road-network within time, space, dynamic, and operational constraints.

The vehicle routing problem has been mainly handled in the operational research area (Gross and Yellen (2003)) and can be generally classified by two categories: one is the Traveling Salesman Problem (TSP) which finds a shortest circular trip through a given number of cities, and the

*Corresponding author. Email: skim78@cnu.ac.kr

ISSN: 0020-7721 print/ISSN 1464-5319 online

DOI: 10.1080/00207721.20xx.CATSid
http://www.informaworld.com

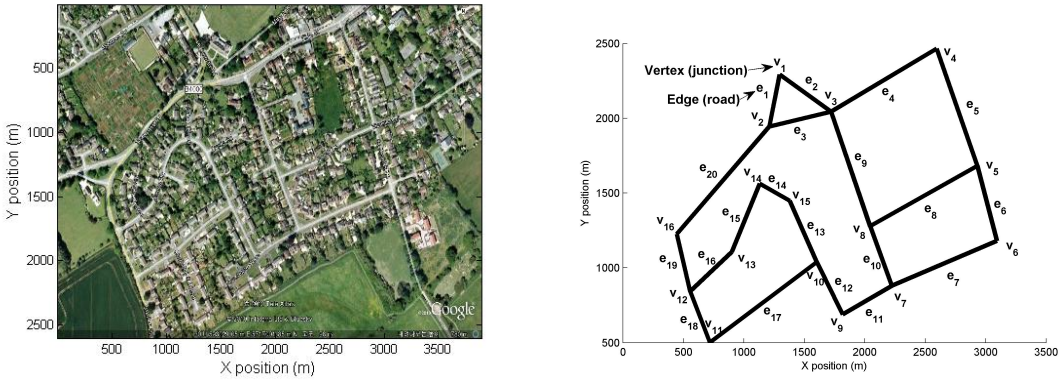
other is the Chinese Postman Problem (CPP) finding the shortest path with considering path constraints on an entire network of road. The TSP using multiple UAVs can be considered as a task assignment problem to minimise the cost of time or energy for a certain mission by assigning each target to UAVs, for which binary linear programming (Bellingham et al. (2003)), iterative network flow (Chandler et al. (2002)) and tabu search algorithm (Ryan et al. (1998)) have been proposed. Choset (2001) surveys research results in coverage path planning that determines a path for a robot to pass over all points in its free space. The CPP and its variants (Pearna and Chiub 2005) are normally used for ground vehicle applications such as road maintenance, snow disposal (Perrier et al. (2007)), boundary coverage (Easton and Burdick (2005)), and graph searching and sweeping (Alspach (2006)).

In the aforementioned works, since the general vehicle routing algorithms approximate their path to a straight line shape to reduce computational load, the physical constraints imposed on the vehicle are not to be addressed. Recently, Dubins TSP (DTSP) algorithms were developed which can accommodate the physical constraints using the so-called Dubins nonholonomic planar vehicle model constrained to move along paths of bounded curvature without reversing direction. For a single vehicle, Salva et al. (2008) proposed an alternating DTSP algorithm based on the solution to the conventional TSP and on an alternating heuristic to assign the target orientation at each target point. Ny et al. (2012) also developed a DTSP algorithm using a heading discretisation. Rathinam et al. (2007), Ahmadzadeh et al. (2006), Tang and Ozguner (2005) considered the similar problem but using multiple agents. However, these physical constraints have been rarely dealt with in the context of the CPP.

Therefore, this paper presents the road-network search route planning algorithms by which multiple airborne platforms are able to efficiently patrol every road segment identified in the map of interest in the context of the CPP. The first part of this paper introduces a conventional CPP algorithm for the case that ground vehicles moves along a connected road-network. Following this, the CPP algorithm is modified to consider a general type of roadmap including unconnected roads as well as operational and physical constraints on speed and minimum turning radius of UAVs. Our previous work (Oh et al. (2011)) regarding the modified CPP (mCPP) for the road-network search route planning was to formulate Multi-choice Multi-dimensional Knapsack Problem (MMKP) so as to find an optimal solution minimising a path length or a flight time and to solve it via mixed integer linear programming (MILP).

The main contributions of this paper are threefold. Firstly, to overcome the computational burden of the MMKP algorithm and to induce a real time solution, a nearest insertion algorithm combining with an auction-based negotiation is newly proposed to be applied for the search route planning by multiple UAVs. Secondly, in order to accommodate the physical constraints of the UAV in the search pattern design, this paper uses the Dubins trajectory (Dubins (1957)) which is the shortest path connecting two configurations represented by position and pose under the constraints of a bound on curvature or turning radius. Although a CSC (circular-straight-circular) type of the Dubins path has been generally used for the path planning of autonomous systems, this paper exploits both a CCC (circular-circular-circular) and a CSC type of the Dubins path to precisely cover a densely distributed road environment. Besides, the detailed mathematical derivation of constructing the Dubins path is presented using the principle of differential geometry. Lastly, this paper systematically investigates the performance of the proposed algorithm depending on different map sizes, path planning methods (straight line and Dubins path) and the number of UAVs by using Monte Carlo simulations. Based on these results, an efficient UAV team size and path planning method is advised for the specific road-network search mission considered in this paper. Furthermore, to clarify the benefit of the proposed algorithm, this paper compares the performance of the MMKP optimisation and the approximation algorithm in terms of computational load and flight time for a specific road-network search scenario.

The remainder of this paper is organised as follows. Section 2 defines the problem of road-network search route planning introducing the TSP and the CPP. Then, Section 3 proposes the



(a) The satellite map of a village in the UK (b) Its graphic representation by straight line segments

Figure 1. The graphic representation of a road-network

road-network search route planning algorithms for multiple UAVs based on the MILP optimisation and the approximation algorithm using nearest insertion and auction negotiation. The derivation of the Dubins path is also included to consider the dynamic constraints of the fixed-wing UAVs. Lastly, the performances and the properties of the proposed algorithm are verified via numerical simulations in Section 4. Conclusions and future works are given in Section 5.

2. Road-Network Search Route Planning

For search mission, a road-network is established as a set of straight line joining waypoints. These waypoints are located either on road junctions or along the roads with sufficient separations between them to allow the accurate representation of the curved road by using a set of straight lines. This paper selects a sample road-network as shown in Fig. 1(a), which is based on the Google map of some village in the UK. This road-network can be translated to a graph composed of straight line segments as shown in Fig. 1(b). In order to search all the roads within the map of interest, there are generally two typical routing problems (Ahr (2004)).

Traveling Salesman Problem (TSP): A salesman has to visit several cities (or road junctions). Starting at a certain city, the TSP is to find a route of minimum travel distance on which the salesman traverses each of the destination cities exactly only once (and for the closed TSP, leads him back to his starting point).
Chinese Postman Problem (CPP): A postman has to deliver mails for a network of streets. Starting at a given point, e.g. the post office, the CPP is to find a route of minimum travel distance allowing the postman to stop by each street at least once (and for the closed CPP, leading him back to the post office).

This paper considers the CPP and its variants, which involve constructing a tour of all the roads with the shortest distance of the road-network. Typically, the road-network is mapped to an undirected graph $G = (V, E)$, having edge weights $w : E \to \boldsymbol{R}_0^+$, where the roads are represented by the edge set $E = \{e_1, \ldots, e_n\}$, and the road junctions are represented by the vertex set $V = \{v_1, \ldots, v_m\}$ as numbered in Fig. 1(b). Each edge $e_i = \{v_{e_{i,1}}, v_{e_{i,2}}\}$ is weighted with its length or the amount of time required to traverse it. The basic CPP algorithm involves first constructing an even graph from the road-network which has a set of vertices with an even number of edges attached to them producing a pair of entry and exit points. When the road-network graph has junctions with an odd number of edges, some roads therefore need to be selected for multiple visits as exceptions by the postman to make the even graph. The search

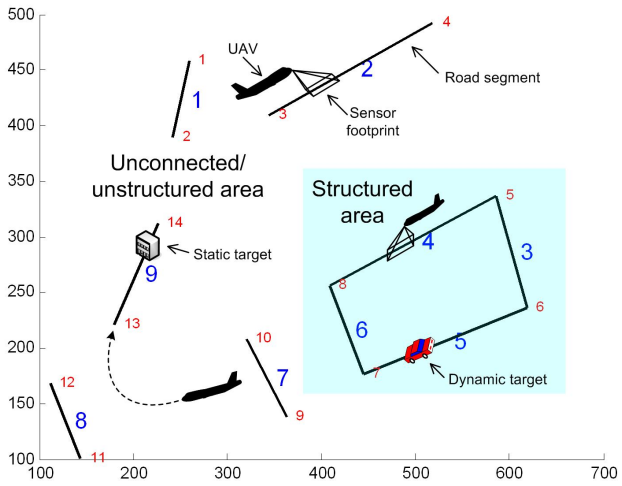


Figure 2. An illustration of the modified Chinese Postman Problem (mCPP)

pattern (tour) of the even graph is calculated by determining the Euler tour of the graph (Gross and Yellen 2003), which visits every edge of the even graph exactly once, or twice for duplicated ones.

The conventional CPP algorithm has been applied to a fully-connected road-network for use by ground vehicles. However, since UAVs do not have any restrictions such that they must only move along the roads, the CPP algorithm needs to be modified for the case that UAVs search a general roadmap having unconnected road segments. The modified CPP (mCPP) generates a tour of the road-network traveling all the roads once no matter what the type area of interest map is: an even or odd graph. Even searching the area having no road somewhere in it can be tackled by the mCPP algorithm by generating a virtual road pattern with a lawnmower (Maza and Ollero 2007) or spiral-like (Nigam and Kroo 2008) algorithm. Figure 2 exemplifies a sample road-network search problem to be solved by the mCPP.

As the CPP algorithms generally approximate their path to a straight line for simplicity, the mCPP algorithm using a straight line path is called the modified Euclidean CPP (mECPP) for the rest of this paper. However, in order to produce the shortest path flyable by the UAV that connects the road segment sequence selected by the search route algorithm, the flight constraints of the UAV have to be taken into account. To accommodate this, the Dubins path is incorporated into the mCPP algorithm instead of using just a straight line to connect the roads, and this is called the modified Dubins CPP (mDCPP). It is worthwhile noting that the mDCPP accommodates an approach angle constraints to a road segment by using the Dubins paths unlike the Dubins traveling salesman problem (DTSP) (Salva et al. (2008)).

3. Road-Network Search Route by Multiple Unmanned Aerial Vehicles

Since the UAV cannot change its heading angle instantaneously due to the physical constraint as shown in Fig. 3, its minimum turning radius should be taken into account explicitly to design a road-network search route pattern. Moreover, when visiting a certain edge unconnected to the main road-network or when coming back to the base, UAVs do not have to fly over the road as explained in the previous section. Therefore, this section proposes road-network search route planning algorithms for the mCPP using multiple UAVs. In the first place, the optimisation via MILP is introduced and incorporated with the Dubins path. A new approximation algorithm is then proposed as a more practical approach to reduce the complexity of the algorithm.

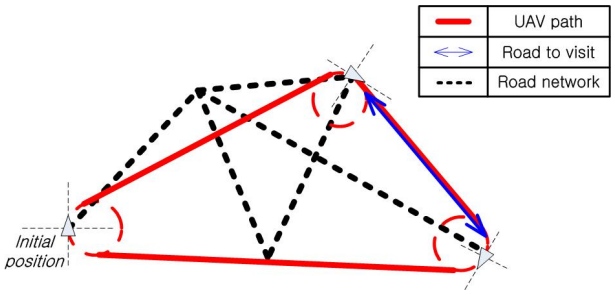


Figure 3. An illustration of the road-network search by a UAV

3.1. *Optimisation via MILP*

The mCPP is first solved by the MILP optimisation using the MMKP formulation to find an sub-optimal solution minimizing the total flight time of UAVs. A classical MMKP is to pick up the items for the knapsacks having a maximum total value so that the total resource required does not exceed the constraints of knapsacks (Hifi et al. (2006)). For applying the MMKP to the road-network search, UAVs are assumed as the knapsacks, the roads to be searched as resources, and the limited flight time or energy of each UAV as the capacity of knapsack. The MMKP formulation allows the search problem to consider the characteristics of each UAV such as flight time capacity and minimum turning radius. Moreover, since the Dubins path planning produces the shortest and flyable paths taking into consideration their dynamical constraints, it is used to calculate the cost function of the MMKP. The details of the proposed road-network search algorithm for multiple UAVs are explained as follows.

3.1.1. Generation of the shortest edge permutation

First of all, unordered feasible edge (that is, road) permutations to be visited by the UAV are generated for all possible cases with a given petal size. The petal size means the maximum number of edges that can be visited by one UAV and is determined by the amount of resources available to it. As a general description, let us assume that the edge set E is $\{e_1, e_2, e_3\}$, and the petal size is three, then all possible edge permutations can be generated as:

$$\begin{aligned} E_p = [&e_1,\ e_2,\ e_3,\ \{(e_1,e_2),\ (e_2,e_1)\},\ \{(e_1,e_3),\ (e_3,e_1)\},\\ &\{(e_2,e_3),\ (e_3,e_2)\},\ \{(e_1,e_2,e_3),\ (e_1,e_3,e_2),\ (e_2,e_1,e_3),\\ &(e_2,e_3,e_1),\ (e_3,e_1,e_2),\ (e_3,e_2,e_1)\}] \end{aligned}$$

If the end vertex of one edge $e_1 \in E_p$ and any vertex of next edge $e_2 \in E_p$ are not connected, they are connected with an additional edge which has a shorter distance. Then, the shortest order-of-visit edge permutations considering the initial position of each UAV are computed under the assumption that a path is represented as a straight line. The example of the shortest order-of-visit permutation $E_{p,s}$ among the aforementioned edge permutation set E_p are as:

$$E_{p,s} = \{e_1,\ e_2,\ e_3,\ (e_2,e_1),\ (e_1,e_3),\ (e_2,e_3),\ (e_2,e_3,e_1)\}$$

3.1.2. Dubins path planning

Once the shortest edge permutations are determined, the next step is to compute and to store the cost (path length or flight time) of them. In this step, this paper uses the Dubins path which is able to take into account the orientation and path constraints of the UAV instead of just using Euclidean distance of each edge. The Dubins path is the shortest path connecting two configuration represented by position and pose under the constraints of a bound on turning radius. A

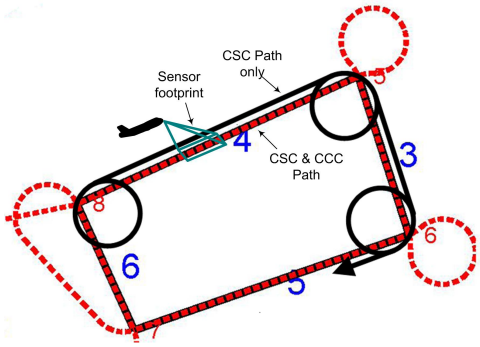
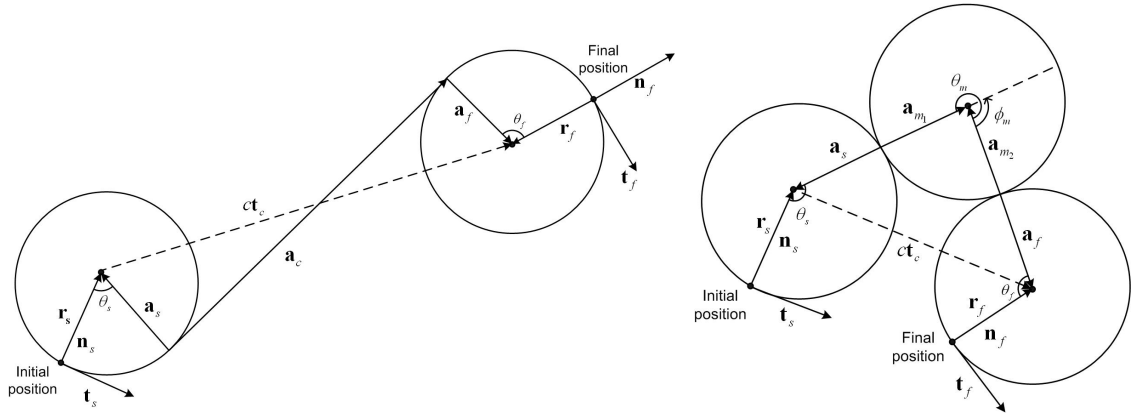


Figure 4. An example of CCC and CSC paths following the road



(a) CSC path (b) CCC path

Figure 5. Dubins path geometry

mathematical proof is provided by Dubins and further dealt with by others (Boissonnat et al. (1994), Shkel and Lumelsky (2001), Shanmugavel et al. (2007)), and it is commonly accepted as a reasonably accurate kinematic model for aircraft motion planning problem (LaValle (2006), Tomlin et al. (2001)). It is formed either by concatenation of two circular arcs with their common tangent or by three consecutive tangential circular arcs. Although the CSC path is being used in general case, this study explores the CCC path as well as the CSC path for the search route planning within a densely distributed road environment because CSC path cannot be applied to a pair of positions having a distance less than the minimum turning radius. Moreover, to follow the road precisely taking into account the sensor footprint coverage, the path should consist of both CCC and CSC forms of Dubins path. Figure 4 shows an example of a road search path using CSC and CCC path for a small sensor footprint, which results in a detour at the road intersection. Note that using only CSC path cannot generate the path exactly following the road as black solid line due to turning constraints.

Geometry for both CSC and CCC type of the Dubins path is shown in Fig. 5. This paper first derives the details of the construction of CCC paths using the principle of differential geometry to simplify the generation of the paths in a vector form. The vector sum for the position vector $\boldsymbol{p}$ which is a position of the final point $\boldsymbol{p}_f$ relative to the start point $\boldsymbol{p}_s$ in start axes is given by:

$$\begin{aligned} \boldsymbol{p} = \boldsymbol{p}_f - \boldsymbol{p}_s &= \boldsymbol{r}_s - \boldsymbol{a}_s + \boldsymbol{a}_{m_1} - \boldsymbol{a}_{m_2} + \boldsymbol{a}_f - \boldsymbol{r}_f \\ \boldsymbol{p} - \boldsymbol{r}_s + \boldsymbol{r}_f &= -\boldsymbol{a}_s + \boldsymbol{a}_{m_1} - \boldsymbol{a}_{m_2} + \boldsymbol{a}_f. \end{aligned} \tag{1}$$

The left hand side of this equation represents the vector $\boldsymbol{t}_c$ connecting the centres of the initial

and the final circles. Hence,

$$c\boldsymbol{t}_c = -\boldsymbol{a}_s + \boldsymbol{a}_{m_1} - \boldsymbol{a}_{m_2} + \boldsymbol{a}_f \tag{2}$$

where c is the length of the centre vector. The remaining connecting vectors $\boldsymbol{a}_s$, $\boldsymbol{a}_{m_1}$, $\boldsymbol{a}_{m_2}$, and $\boldsymbol{a}_f$ can be expressed in terms of start basis vectors as:

$$\begin{aligned}
\boldsymbol{a}_s &= \mathbf{R}(\theta_s)' \begin{pmatrix} 0 \\ \frac{\pm 1}{\kappa_s} \end{pmatrix} \\
\boldsymbol{a}_{m_1} &= \mathbf{R}(\theta_s)' \begin{pmatrix} 0 \\ \frac{\pm 1}{\kappa_m} \end{pmatrix} \\
\boldsymbol{a}_{m_2} &= \mathbf{R}(\theta_s)' \mathbf{R}(\phi_m) \begin{pmatrix} 0 \\ \frac{\pm 1}{\kappa_m} \end{pmatrix} \\
\boldsymbol{a}_f &= \mathbf{R}(\theta_s)' \mathbf{R}(\phi_m) \begin{pmatrix} 0 \\ \frac{\pm 1}{\kappa_f} \end{pmatrix}
\end{aligned} \tag{3}$$

where

$$\phi_m = \pi - \cos^{-1}\left(\frac{c^2 - (\frac{1}{\kappa_s} + \frac{1}{\kappa_m})^2 - (\frac{1}{\kappa_m} + \frac{1}{\kappa_f})^2}{2(\frac{1}{\kappa_s} + \frac{1}{\kappa_m})(\frac{1}{\kappa_m} + \frac{1}{\kappa_f})} \right) \tag{4}$$

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{5}$$

and κ_s, κ_m and κ_f represent the curvature of the initial, the middle and final manoeuvre, respectively. Since a CCC type of the Dubins path consists LRL and RLR, the sign of k_s and k_f are the same, and that of k_s and k_m are different. Then, the center vector equation (2) now becomes

$$\boldsymbol{t}_c = \mathbf{R}(\theta_s)' \frac{1}{c} \begin{pmatrix} \pm(\frac{1}{\kappa_m} + \frac{1}{\kappa_f}) \sin(\phi_m) \\ \pm((\frac{1}{\kappa_s} + \frac{1}{\kappa_m}) + (\frac{1}{\kappa_m} + \frac{1}{\kappa_f}) \cos(\phi_m)) \end{pmatrix}. \tag{6}$$

This is a rotation equation; hence, the right hand vector should have the unit magnitude, to give

$$\left| \frac{1}{c} \begin{pmatrix} \pm(1/\kappa_m + 1/\kappa_f) \sin(\phi_m) \\ \pm((1/\kappa_s + 1/\kappa_m) + (1/\kappa_m + 1/\kappa_f) \cos(\phi_m)) \end{pmatrix} \right| = 1. \tag{7}$$

This can be used to test for a feasible solution of the CCC path by:

$$1 - \frac{((1/\kappa_s + 1/\kappa_m) + (1/\kappa_m + 1/\kappa_f) \cos(\phi_m))^2}{c^2} > 0. \tag{8}$$

To compute the initial turning angle θ_s first, the equation can be written in the form

$$\boldsymbol{t}_c = \mathbf{R}(\theta_s)' \frac{1}{c} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \frac{1}{c} \begin{pmatrix} \beta & \gamma \\ \gamma & -\beta \end{pmatrix} \begin{pmatrix} \cos(\theta_s) \\ \sin(\theta_s \end{pmatrix} \tag{9}$$

where $\beta = \pm(1/\kappa_m + 1/\kappa_f)\sin(\phi_m)$ and $\gamma = \pm((1/\kappa_s + 1/\kappa_m) + (1/\kappa_m + 1/\kappa_f)\cos(\phi_m))$. Solving this for θ_s with $\beta^2 + \gamma^2 = c^2$ from Eq. (7) gives:

$$\begin{pmatrix} \cos(\theta_s) \\ \sin(\theta_s) \end{pmatrix} = \frac{1}{c} \begin{pmatrix} \beta & \gamma \\ \gamma & -\beta \end{pmatrix} \boldsymbol{t}_c. \tag{10}$$

The final turning angle θ_f can be determined by using

$$\begin{aligned} \theta_d &= \theta_s - \theta_m + \theta_f \\ \theta_f &= \theta_d - \theta s + \theta_m \end{aligned} \tag{11}$$

where θ_d represents the difference between the initial and final heading angle and $\theta_m = \pm(\pi + |\phi_m|)$. Finally, the path length is computed by summation of the arc lengths as:

$$L_{CCC} = \frac{\theta_s}{\kappa_s} + \frac{\theta_m}{\kappa_m} + \frac{\theta_f}{\kappa_f}. \tag{12}$$

Similarly, construction of CSC paths is explained following from Shanmugavel et al. (2007). First of all, the vector sum for the position vector $\boldsymbol{p}$ is given by:

$$\boldsymbol{p} = \boldsymbol{r}_s - \boldsymbol{a}_s + \boldsymbol{a}_c + \boldsymbol{a}_f - \boldsymbol{r}_f. \tag{13}$$

Then, the vector $\boldsymbol{t}_c$ becomes:

$$c\boldsymbol{t}_c = \boldsymbol{p} - \boldsymbol{r}_s + \boldsymbol{r}_f = -\boldsymbol{a}_s + \boldsymbol{a}_{m_1} - \boldsymbol{a}_{m_2} + \boldsymbol{a}_f. \tag{14}$$

The remaining connecting vectors $\boldsymbol{a}_s$, $\boldsymbol{a}_c$, and $\boldsymbol{a}_f$ can be expressed in terms of start basis vectors as:

$$\begin{aligned} \boldsymbol{a}_s &= \mathbf{R}(\theta_s)' \begin{pmatrix} 0 \\ \frac{\pm 1}{\kappa_s} \end{pmatrix} \\ \boldsymbol{a}_c &= \mathbf{R}(\theta_s)' \begin{pmatrix} a \\ 0 \end{pmatrix} \\ \boldsymbol{a}_f &= \mathbf{R}(\theta_s)' \begin{pmatrix} 0 \\ \frac{\pm 1}{\kappa_f} \end{pmatrix}. \end{aligned} \tag{15}$$

Using above equations, the centre vector equation (14) can be expressed as:

$$\boldsymbol{t}_c = \mathbf{R}(\theta_s)' \frac{1}{c} \begin{pmatrix} a \\ \pm\frac{1}{\kappa_f} - \pm\frac{1}{\kappa_s}) \end{pmatrix}. \tag{16}$$

To compute the initial turning angle θ_s, the equation can be written as:

$$\boldsymbol{t}_c = \mathbf{R}(\theta_s)' \frac{1}{c} \begin{pmatrix} \sqrt{c^2 - [(\pm 1/\kappa_f) - (\pm 1/\kappa_s)]^2} \\ (\pm 1/\kappa_f) - (\pm 1/\kappa_s) \end{pmatrix}. \tag{17}$$

Solving this with the same way as CCC path case gives the initial turning angle θ_s, and the final angle θ_f can then be determined by using

$$\begin{aligned} \theta_d &= \theta_s + \theta_f \\ \theta_f &= \theta_d - \theta s. \end{aligned} \tag{18}$$

The path length for this CSC case is computed by summation of the arc lengths and connecting tangent length as:

$$L_{CSC} = \frac{\theta_s}{\kappa_s} + a + \frac{\theta_f}{\kappa_f}. \tag{19}$$

If both of the initial and final heading angle are given, a set of six paths including CSC and CCC path $\{LSL, LSR, RSL, RSR, RLR, LRL\}$ can be produced, and the shortest path is selected from these set of available paths.

Following Shanmugavel et al. (2010), now let us briefly look at safe paths. The Dubins paths satisfy the maximum bound on curvature. However, safety of UAVs needs additional constraints to be met to produce safe flyable paths. In this respect, the Dubins paths have to meet the safety conditions: (i) minimum separation distance and (ii) non-intersection of paths at equal lengths. The minimum separation distance d_{sep} between any two UAVs should at least be equal to the summation of corresponding radii of the safety circles as given:

$$d_{sep} \geq R_{s,1} + R_{s,2} \tag{20}$$

where $R_{s,i}$ represents the radii of the safety circle of the UAV i. Although two Dubins paths failed to meet the minimum separation distance, this does not necessarily imply a collision. To verify a collision, the path length L_{int} of each path from its starting point to the point of failure on minimum separation should be calculated. The difference between the path lengths $d_{int} = |L_{int,1} - L_{int,2}|$ must be less than the summation of corresponding radii of safety circles to confirm a potential collision as given:

$$d_{int} \geq R_{s,1} + R_{s,2} \tag{21}$$

In the event of failure of both conditions, the replanning should be done by increasing the radius of the turning circles or choosing the next shortest path from the Dubins set. For a group of N UAVs, taking r UAVs at a time, the safety conditions have to be tested for n_u times, where

$$n_u = 2\frac{N!}{r!(N-r)!} \tag{22}$$

3.1.3. MMKP formulation and MILP optimisation

The final step of the proposed algorithm is allocating the edge permutations to each UAV so as to cover all the roads with a minimum flying time. This can be expressed by a MMKP

formulation as:

$$\min\ J = \sum_{i=1}^{N_{UAV}} \sum_{j=1}^{N_{p_i}} T_{ij} x_{ij} \tag{23}$$

$$\text{s.t} \sum_{i=1}^{N_{UAV}} \sum_{j=1}^{N_{p_i}} E_{kj} x_{ij} \geq 1,\ \text{for } k = 1, \ldots, N_{edge} \tag{24}$$

$$\sum_{j=1}^{N_{p_i}} x_{ij} = 1,\ x_{ij} \in \{0, 1\}, \tag{25}$$

$$\text{for } i = 1, \ldots, N_{UAV},\ j = 1, \ldots, N_{p_i}$$

where N_{UAV}, N_{edge}, and N_{p_i} denote the number of UAVs, edges to be visited and permutations generated by the i-th UAV, respectively. T_{ij} represents mission cost (flight time) of j-th permutation of i-th UAV and E_{kj} represents the matrix whose k-th element of j-th permutation is 1 if edge k visited, otherwise 0 and x_{ij} is either 0, implying permutation j of the i-th vehicle is not picked, or 1 implying permutation j of the i-th UAV is picked. First constraint represents that UAVs should visit every edge once or more and second one represents the allocation of the exact one edge permutation to the each UAV. This MMKP problem is solved by SYMPHONY MILP solver (SYM (2010)). It should be noted that depending on the petal size, the computational burden of the mission cost T_j of all edge permutations would be increased significantly. Moreover, obtaining a solution to any MILP formulation is NP-hard and it is claimed that problems in which the sum of the number of vehicles and targets is less than 12 are solvable in less than a minute on a modern desktop computer (Shima and Rasmussen (2010), Kingston and Schumacher (2005)). Therefore, large problems with complex road-network will require a reformulation of the problem such as restricting the petal size and using receding horizon concept (Alighanbari (2004)) or partitioning of the road-network in a timely manner to use the MILP optimisation approach.

3.2. *Approximation Algorithm*

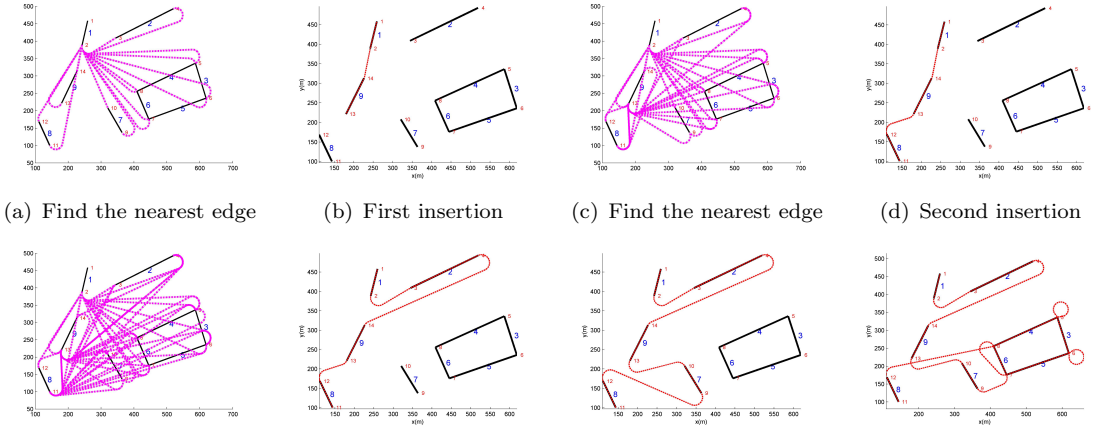
3.2.1. Nearest insertion based mDCPP

Due to the complexity of the problem, instead of using the optimisation method explained above, an approximation algorithm is developed as a more practical way for the mCPP. To develop the approximation algorithm, the TSP algorithm was first studied. Although there are a lot of algorithms for the TSP (Rosenkrantz et al. (2009)), one heuristic approach, a nearest insertion method is adopted since it is fast and easy to implement. The basic idea of it is to construct the approximation tour by a sequence of steps in which tours are constructed for progressively larger subsets of the nodes. It produces a tour no longer than twice the optimal regardless of the number of nodes in the problem and runs in a time proportional to the square of the nodes (Rosenkrantz et al. (2009)). Having these in mind, this paper proposes the nearest insertion based mDCPP (NI-mDCPP) algorithm. Firstly, the NI-mDCPP algorithm for the single UAV is described as follows.

Algorithm description

(1) Start from a certain point or road junction and select the nearest road to it using the Dubins path length

(2) Make and grow a tour by finding the nearest road to any of the selected tour roads
(3) Compute the cost of insertion to decide whether to insert before or stack after the closest road to the tour
(4) Insert the selected road in the decided position
(5) 2) $\sim$ 4) are repeated until all roads are included in the tour



(a) Find the nearest edge (b) First insertion (c) Find the nearest edge (d) Second insertion

(e) Find the nearest edge (f) Third insertion (g) Fourth insertion (h) Eighth insertion

Figure 6. An example simulation of the NI-mDCPP road search algorithm

3.2.2. Euclidean distance order approximation

To reduce the computation burden further for the dynamic environment, an additional approximation algorithm which uses the Euclidean distance order is incorporated into the NI-mDCPP algorithm. This algorithm is described as follows. First of all, make the ascending order of road list for both the Euclidean distance and the Dubins path length between all pairs of end points of the road-network, and find the maximum number, $n_{order,max}$ which guarantees that road list of Euclidean distance within that number contains the shortest Dubins path in advance. Note that although $n_{order,max}$ is determined before running the algorithm with given information of the road-network, a size (tendency) of $n_{order,max}$ would remain almost the same against minor changes of road information for an uncertain dynamic environment. Then, when finding the nearest roads, make the ascending order list of distance between edge of interest and all the other roads with Euclidean distance first, and find the nearest road whose Dubins path length is the shortest among roads in the $n_{order,max}$ Euclidean distance order. In other words, this method computes only $n_{order,max}$ Dubins path distances instead of computing all Dubins lengths between one road to the others. For example, if $n_{order,max}$ is three as shown in table 1, among ascending road list of Euclidean distance from a certain road to the others, only first three roads (3, 6 and 4) are tested by the Dubins path, and the shortest one (6) is selected as the nearest road. Note that as the distance between the roads increases, since the Euclidean distance and Dubins path length gets closer, $n_{order,max}$ decreases resulting in less computation time.

3.2.3. Negotiation for multiple UAVs

Having the characteristic of the NI-mDCPP for the single UAV, the algorithm can be extended to the case of multiple UAVs using auction-based negotiation. The auction algorithm is a method of solving the assignment problem using elements from both the primal and dual formulations based on approximate optimality, called ϵ-*complementary slackness (CS)* condition (Bertsekas 1992, 1998). Suppose that there is a benefit a_{ij} for matching an agent i and an object j, object

Table 1. Procedure of Euclidean distance order approximation algorithm ($n_{order,max} = 3$)

Euclidean				Dubins		Nearest Road
Order	Length	Road		Length	Order	
1	1.2	**3**	→	3.1	3	
2	1.3	**6**	→	2.8	1	**6**
3	1.4	**4**	→	2.9	2	
4	1.5	2				
5	1.6	5				
6	1.7	7				
⋮	⋮	⋮				

j has a price p_j, and the person who receives the object must pay the price. Let us then set a positive scalar ϵ and state that an agent i is almost happy with an assignment and a set of prices if the value of its assigned object j_i is within ϵ of being maximal as:

$$a_{ij_i} - p_{j_i} \geq \max_{j=1,\ldots,n} \{a_{ij} - p_j\} - \epsilon \tag{26}$$

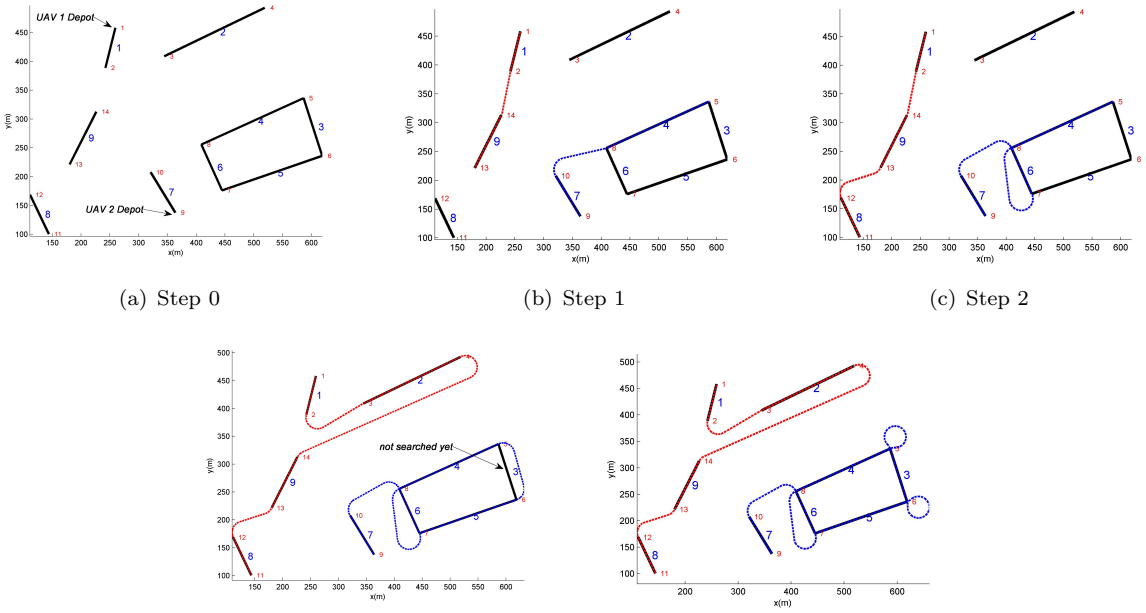
Above ϵ-CS condition is interpreted as stating if an agent i is assigned to object j, then its profit is equal to the benefit it receives from that object minus its price, and that object j provides an agent i more profit (within some tolerance ϵ) than any other object. The auction algorithm starts with an empty assignment (which trivially satisfies ϵ-CS), and iteratively proceed by modifying it through bidding (decision is based on values, which are bidder's benefits plus bidding increment ϵ at each time) and assignment phase until all agents are assigned satisfying ϵ-CS. This auction process is proved to be terminated in finite number of rounds, and the performance in terms of computation and optimality depends strongly on the size of ϵ (Bertsekas 1992). In this study, the auction benefit a_{ij} of UAV i for road segment j is its corresponding flight time to that road segment. Now, the NI-mDCPP algorithm for multiple UAVs is described as follows.

Algorithm description

(1) Start with N initial positions or roads of N UAVs
(2) Make and grow a tour by using single NI-mDCPP algorithm while storing the cost (path length or flight time) between selected tour roads and remaining edges (which was needed for finding the nearest vertex)
(3) When conflict occurs, i.e. more than one UAV wants the same road for the next tour, the auction algorithm using stored cost is used to match UAVs with the task (road) to minimise the cost

Figure 7 illustrates the procedure of the algorithm. Since road 3 is not searched yet in Fig. 7(d), each UAV sends its cost for the given task (in this case, visiting road 3), then, auction or bipartite (linear programming) is used to match UAVs with the task to minimise the cost. Although overlapping road segments is avoided using the auction algorithm, collision between UAVs might occur during the transition from one to another road. In other words, the distance between any two UAVs can be less than a threshold value, called as minimum separation distance. In that case, if necessary, the path can be replanned either by visiting new road or varying the curvature of the arc of the Dubins path as explained in section 3. For simplicity, this paper assumes that the collision avoidance is done by local flight controller or operating UAVs in different altitudes. The proposed algorithm is rather simple but straightforward and can be run in real time. Moreover, by including additional factors such as different minimum turning radius

and total path length (or the number of roads) assigned to each UAV so far into the cost, the auction-based negotiation can be varied flexibly, even for heterogeneous UAVs.



(a) Step 0 (b) Step 1 (c) Step 2

(d) Step 3 (e) Step 4

Figure 7. Negotiation procedure for multiple UAVs

4. Numerical Simulations

4.1. *Monte Carlo Simulation of the NI-mDCPP Algorithm*

To evaluate the properties and performance of the proposed approximation algorithm, Monte Carlo simulations are performed using a random map with different parameters about the map size and the number of UAVs. A map environment is composed of 10 by 10 vertices as numbered in Fig. 8, and road edges are generated by connecting two vertex randomly selected. To check the impact of the map size on Dubins path planning, the distance d_{map} between the adjacent vertex is set to be in proportion to the minimum turning radius ρ_{min} of the UAV as:

$$d_{map} = K_s \times \rho_{min} \tag{27}$$

where K_s is the scale factor. Moreover, some of the selected edges whose lengths are longer than three times of d_{map} are discarded to get a well-distributed road-network and to distribute the roads to each UAV with a similar length. Figure 8 shows the sample road-network with 20 randomly chosen edges. By Monte Carlo simulations, this paper tries to find out the effect of three major factors for the road-network search route planning as below. These aspects would provide users with information on priorities for the efficient use of the UAV group in the planning phase of the autonomous search mission.

The distribution density of road-network: Densely or sparsely distributed road relative to the minimum turning radius of UAVs
The type of path planning: Straight line or Dubins path
The number of UAVs: Computation time, path length, and the longest path length of a

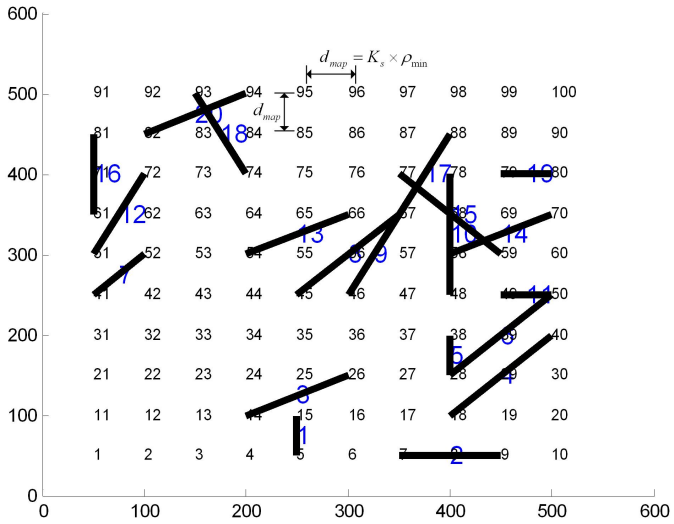


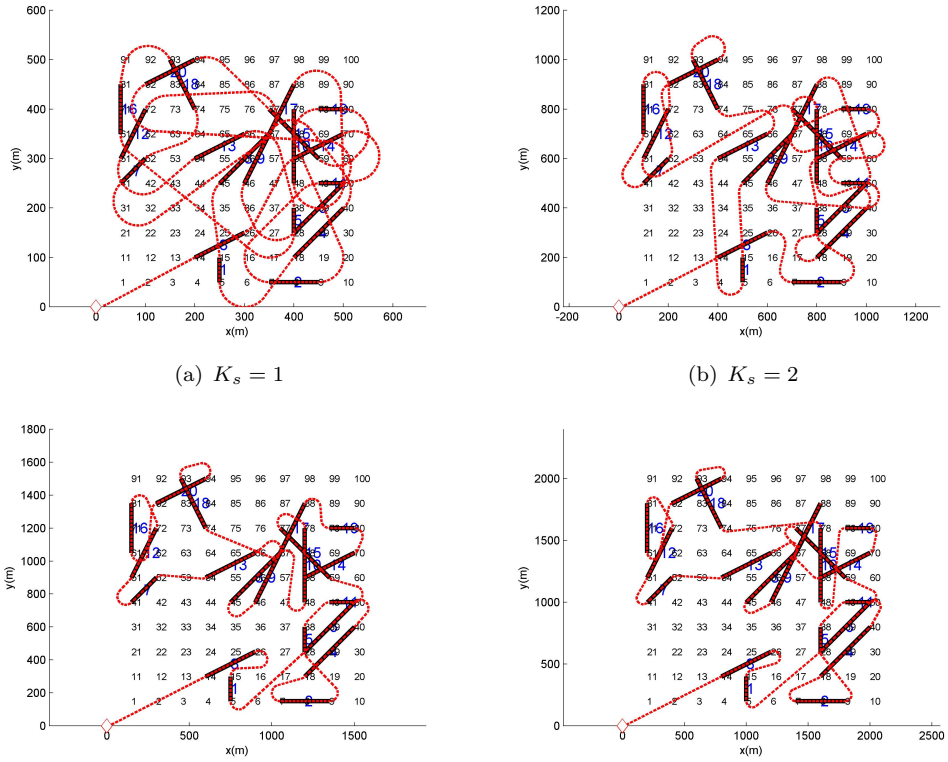
Figure 8. A sample road-network with 20 randomly chosen edges ($K_s = 1,\ \rho_{min} = 50m$)

single UAV

In the simulations, UAVs are assumed to have a constant velocity and minimum turning radius $\rho_{min} = 50\ m$. The simulation results are the average of 50 runs with the MATLAB algorithm.

4.1.1. Single UAV case

The first set of the simulation is performed by using a single UAV with different road map scales. For the rest of this section, the terms of mDCPP and the mECPP are used for the NI-mDCPP and the NI-mECPP, respectively. One of search route planning results using the mDCPP for the random map is shown in Fig. 9, which covers all the roads satisfying turning constraints of UAVs. Figure 10(a) displays the computation time ratio between the Dubins path (mDCPP) and the straight line (mECPP). Regardless of the map scale, the mDCPP algorithm is around 40 times slower than the mECPP constantly. Meanwhile, the computation time of the mDCPP along with the Euclidean distance order approximation as explained in section 4 decreases as the map scale increases resulting from decrease of the maximum order $n_{order,max}$ as shown in Fig. 10(b). Figure 10(c) compares total path length to cover the entire roadmap using the mDCPP and the mECPP. For fair comparison, the length of the mECPP (denoted by L^*_{mECPP}) is computed by road search route from the mECPP algorithm but connecting roads using Dubins path. This is because although road search route planning is performed by the mECPP, a real trajectory of the UAV connecting roads should be the Dubins type of path restricted by its maximum curvature. When the minimum turning radius is relatively small compared to the distance between roads, that is, when the map scale is small, the path length of the mDCPP is shorter than that of the mECPP. However, as the map scale gets bigger, the path length ratio gets closer to nearly one (or even over one) since the road search order of the Dubins path would be almost the same as the one from the straight line as one can expect from Fig. 9. Based on these analysis on the computation time and the path length advantage, it is concluded that using the Dubins path algorithm is preferable for the road-network searching by a single UAV in case that the map scale is smaller than two.



(a) $K_s = 1$ (b) $K_s = 2$

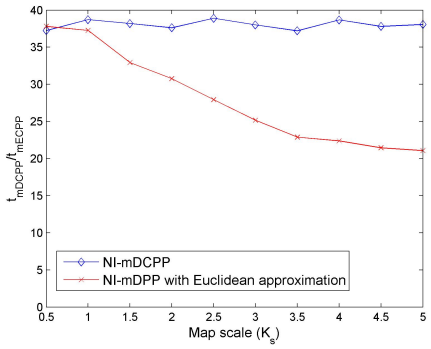
(c) $K_s = 3$ (d) $K_s = 4$

Figure 9. NI-mDCPP road search path with different map scale factors

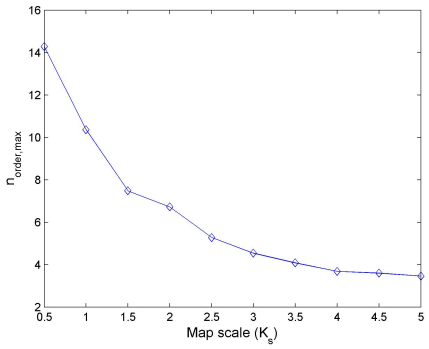
4.1.2. Multiple UAVs case

In the previous section, the performance of the proposed NI-mDCPP algorithm was compared to that of the NI-mECPP as well as NI-mDCPP without Euclidean approximation for single UAV. Now, let us compare the performance of algorithms for multiple UAVs case with different road map scales. One example of the search route planning results using the mDCPP for six UAVs is represented in Fig. 11. The initial position of each UAV is equally distributed around the road area. Figure 12 presents the normalised simulation results by the value of a single UAV. In particular, the longest path length (Fig. 12(c)) of the UAV is investigated since it is equivalent to the mission completion time of the entire UAV team. The normalised computation time and the longest path length of the UAV are decreased as the size of the UAV team increases regardless of the map scale since each UAV takes partial charge of the road search mission cooperatively using the auction-based task assignment. Unlike the others, the total path length (Fig. 12(b)) is affected by the map scale significantly. When the map scale is small, the total path length is decreased in proportional to the number of UAVs. Whereas in a relatively big map environment, the normalized path length remains nearly at one since each UAV should fly a long distance from the initial position or one road to the another road. Apparently, the simulation results show that the bigger the UAV team size is, the better performance it shows in term of the computation time and path length. However, using a large size of the UAV team requires more operational cost. Therefore, the performance index to determine the optimal size of the UAV team for the search mission can be proposed as Eq. (28) including an additional operational cost on each UAV represented as the normalised number of UAVs (by maximum seven UAVs in this paper), $\bar{n}_v$, and its weighting factor, w_{n_v}.

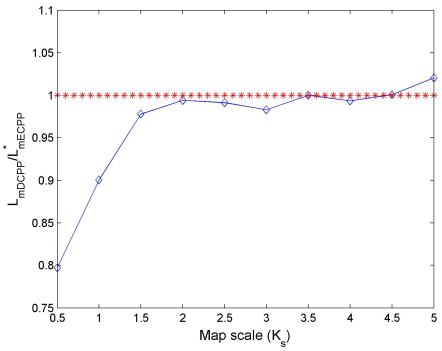
$$J = w_t\bar{t} + w_l\bar{l} + w_L\bar{L} + w_{n_v}\bar{n_v} \tag{28}$$



(a) Computation time ratio

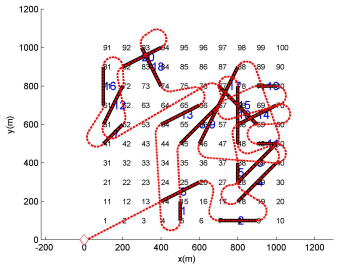


(b) $n_{order,max}$ for Euclidean approximation

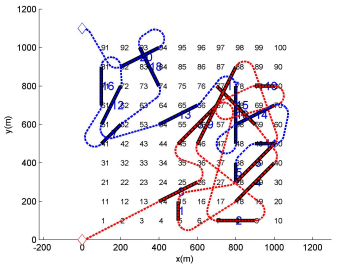


(c) Path length ratio

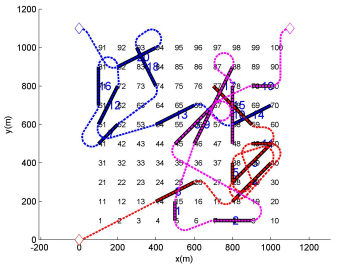
Figure 10. NI-mDCPP results with different map scale factors, average for 50 simulations



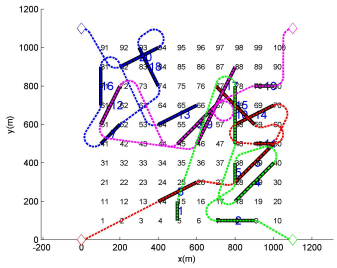
(a) 1 UAV



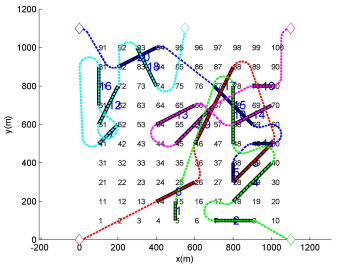
(b) 2 UAVs



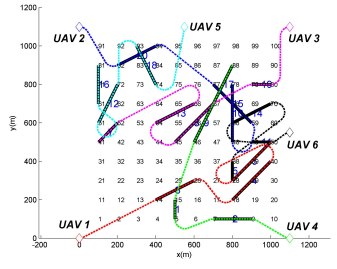
(c) 3 UAVs



(d) 4 UAVs

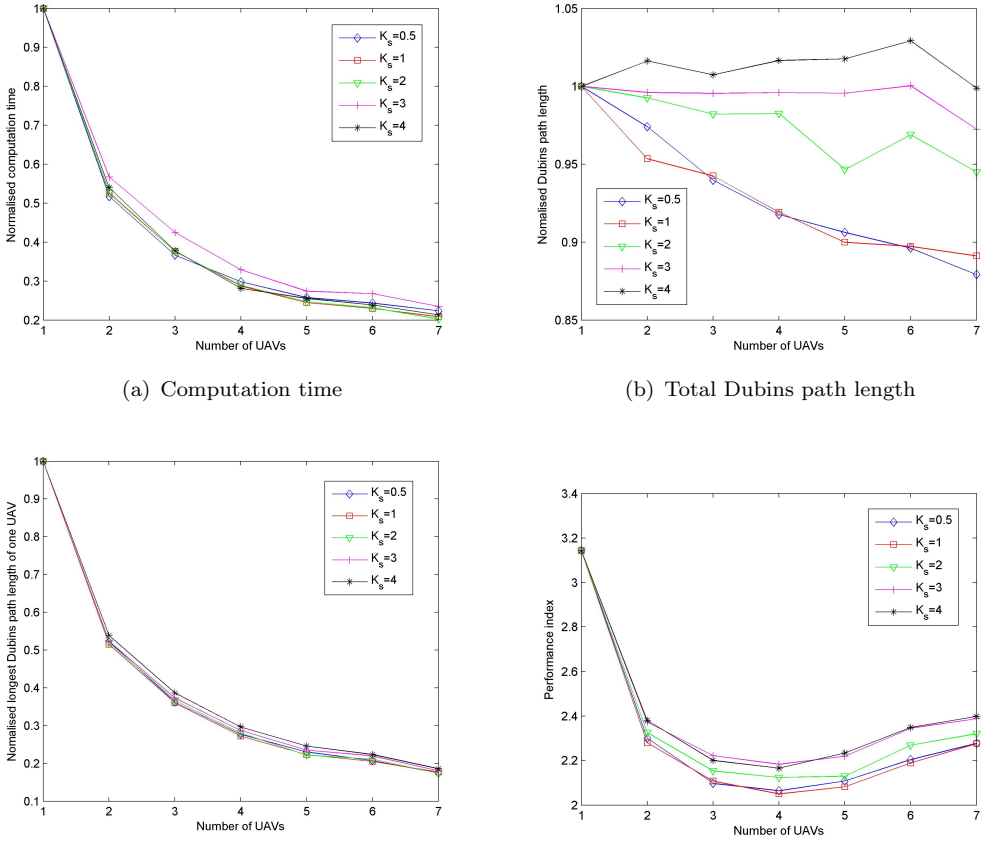


(e) 5 UAVs



(f) 6 UAVs

Figure 11. NI-mDCPP road search path with different number of UAVs ($K_s = 2$)



(a) Computation time

(b) Total Dubins path length

(c) The longest Dubins path of one UAV

(d) Performance index cost

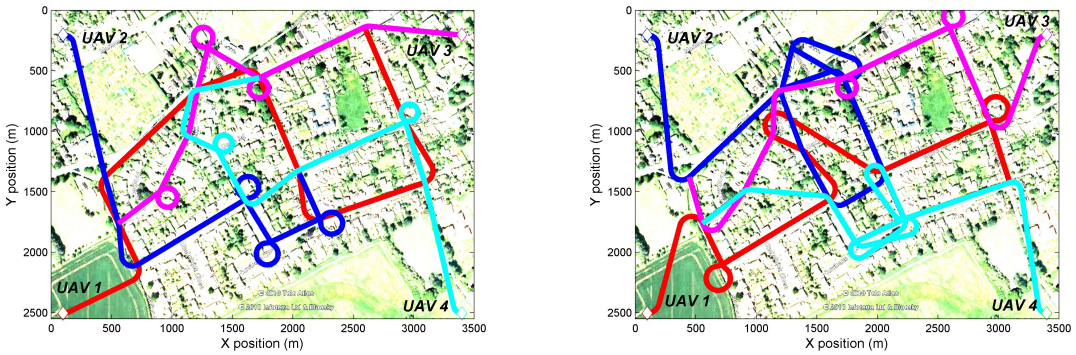
Figure 12. NI-mDCPP results with different number of UAVs, average for 50 simulations

where w_t, w_l and w_L represent the weighting factors of computation time, total path length and the longest path length of one UAV, respectively. Under the assumption that all the weighting factors are equally one, the number of UAVs to minimise the performance index J can be selected as four for all map scale factors ($K_s = 0.5 \sim 4$) consistently as shown in Fig. 12(d). Even though this sub-optimal number of UAVs can be changed according to different map or operational parameters, since the result comes from Monte carlo simulations with random maps with various map scale factors, this framework would help the operator to decide the reasonable number of resources (UAVs) in the initial phase of autonomous search mission. Besides, since this study used a simple operational cost in proportion to the number of UAVs, more rigorous approach to capture and incorporate the realistic operational cost should be followed depending on the characteristic of the mission and the vehicle such as fuel, communication resources, or surrounding environments as a future work.

4.2. *Performance Comparison*

To evaluate the performance of the proposed road-network search algorithms for multiple UAVs, numerical simulations are performed for a specific scenario with four UAVs and the road-network given in section 2. Each UAV has different dynamic constraints as:

- Minimum turning radius ρ_{min}: [100 90 80 70] m
- Maximum cruise speed $V_{c,max}$: [60 50 40 30] m/s



(a) MILP optimisation (b) NI-mDCPP

Figure 13. Road-network search route planning results using multiple UAVs

Table 2. Performance comparison between different algorithms)

Method	Computation time	Flight time (sec)
MILP	> 5 min	583.1
NI-mECPP	0.33 sec	681.0
NI-mDCPP*	1.15 sec	642.9
NI-mDCPP	0.78 sec	642.9

The maximum curvature κ_{max} of UAVs can be given by $\kappa_{max} = 1/\rho_{min}$. UAVs are assumed to have maximum cruising speed during the entire mission and the maximum petal size of the edge permutation is set to be five. Figure 13(a) shows the result of the road-network search using the MILP optimisation. The flight path is smooth and flyable due to the Dubins path planning, and since the UAV does not need to fly along the road only, the results include additional paths connecting some of the unconnected roads. Moreover, it can be observed that the faster UAV tends to have a longer flight path since the objective of the MILP is to minimise the total sum of flight time of all UAVs. The total flight length of all UAVs is 2798.1 metres, and its flight duration is 583.1 seconds. In this scenario, the total computation time exceeds a reasonable limit (> 5 minutes) using Matlab for the MMKP formulation and ANSI-C for the MILP computation using a normal PC system (Core 2 CPU, 2.0 Ghz, and 512 MB RAM). Meanwhile, Fig. 13(b) shows the search result of the NI-mDCPP using the cost of the flight time. Table 2 shows the performance comparison between MILP, the NI-mECPP, the NI-mDCPP* (without Euclidean approximation) and the NI-mDCPP algorithm. The maximum number $n_{order,max}$ is computed as six for this sample map environment. Note that the flight time of the NI-mECPP is computed by road search route from the NI-mECPP algorithm but connecting roads using Dubins paths with corresponding minimum turning radius. The proposed NI-mDCPP gives a solution within a second having about ten percents longer flight time (642.9 seconds) than that of the MILP optimisation. Considering both the computation time and performance, the proposed NI-mDCPP can be regarded as a preferable approach over the MILP optimisation for a given sample map or more complex scenario.

5. Conclusion

This paper has presented a practical approach for a road-network search route planning by which a team of autonomous aerial vehicles visit every road in the map of interest in the context of

the CPP. Firstly, the conventional CPP algorithm was explained and modified to be applied for the general type of roadmap including unconnected roads. Then, the MILP optimization and the nearest insertion algorithm along with the auction negotiation were proposed for multiple unmanned aerial vehicles. To realistically accommodate the manoeuvreing constraints of UAVs, the Dubins path planning was used for solving the modified CPP. The performance of the approximation algorithm was investigated via a Monte Carlo simulation framework by analyzing the effects of different map sizes, path planning methods and the number of UAVs. Based on these results, the efficient UAV team size and path planning method were suggested for the road search route planning, and this framework can be applied to a variety of autonomous search missions in the initial phase of mission planning. To clarify the benefit of the proposed algorithm, this paper compared the performance of the MILP optimisation and the approximation algorithm with each other in terms of computational load and flight time for a specific road-network scenario. Applying the proposed algorithm to the cooperative search and target tracking will be followed as a future work. Which and how many UAVs should be assigned to the tracking task can be determined in terms of a total path length or a mission completion time using the proposed approximation algorithm with a much lighter computation burden than that of the conventional techniques. Another future work will be a collaboration between aerial and ground vehicles, which can exploit complementary capabilities such as a broader field of view with a faster coverage by UAVs and a high resolution sensing over a short-range area by ground vehicles.

References

SYMPHONY 5.2.4, `http://www.coin-or.org/SYMPHONY` (2010).

Ahmadzadeh, A., Buchman, G., Cheng, P., Jadbabaie, A., Keller, J., Kumar, V., and Pappas, G. (2006), "Cooperative control of UAVs for search and coverage," in *Proceedings of the AUVSI Conference on Unmanned Systems*, Orlando, FL.

Ahr, D. (2004), "Contributions to Multiple Postmen Problems," Ph.D. dissertation, Fakultt fr Mathematik und Informatik, Institut fr Informatik, Heidelberg University.

Alighanbari, M., "Task assignment algorithms for teams of UAVs in dynamic environments," Master's thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology (2004).

Alspach, B. (2006), "Searching and sweeping graphs: a brief survey," *Matematiche (Catania)*, 59, 5–37.

Bellingham, J., Tillerson, M., Richards, A., and How, J., *Multi-Task Allocation and Path Planning for Cooperating UAVs, Cooperative Control: Models, Applications and Algorithms*, Kluwer Academic Publishers (2003).

Bertsekas, D.P. (1992), "Auction algorithms for network flow problems: A tutorial introduction," *COMPUTATIONAL OPTIMIZATION AND APPLICATIONS*, 1(1), 7–66.

Bertsekas, D.P., *Network Optimization: Continuous and Discrete Models*, Athena Scientific (1998).

Boissonnat, J., Crzo, A., and Leblond, J. (1994), "Shortest paths of bounded curvature in the plane," *J. Intell. Robot. Syst.*, 11, 5–20.

Casbeer, D., Kingston, D., Beard, R., and McLain, T. (2006), "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, 37(6), 351–360.

Chandler, P., Pachter, M., Swaroop, D., hwlett, J., Rasmussen, S., Schumacher, C., and Nygard, K. (2002), "Compexity in UAV Cooperation Control," in *American Control Conference*, Anchorage, AK.

Choi, H., Kim, Y., and Kim, H. (2011), "Genetic Algorithm Based Decentralized Task Assignment for Multiple Unmanned Aerial Vehicles in Dynamic Environments," *International*

Journal of Aeronautical and Space Sciences, 12, 163–174.

Choset, H. (2001), "Coverage for Robotics-a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, 31, 113–126.

Dubins, L. (1957), "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, 79(3), 497–516.

Easton, K., and Burdick, J. (2005), "A Coverage Algorithm for Multi-robot Boundary Inspection," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain.

Gross, J., and Yellen, J., *Handbook of Graph Theory*, CRC Press (2003).

Gunetti, P., Thompson, H., and Dodd, T. (2011), "Autonomous mission management for UAVs using soar intelligent agents," *International Journal of Systems Science*, DOI:10.1080/00207721.2011.626902.

Hifi, M., Michrafy, M., and Sbihi, A. (2006), "A reactive Local Search-Based Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem," *Computational Optimization and Applications*, 33, 271–285.

King, E., Kuwata, Y., and How, J. (2006), "Experimental demonstration of coordinated control for multi-vehicle teams," *International Journal of Systems Science*, 37, 385–398.

Kingston, D., and Schumacher, C. (2005), "Time-dependent cooperative assignment," in *American Control Conference*, Portland, OR.

LaValle, S., *Planning Algorithms*, Cambridge University Press (2006).

Maza, I., and Ollero, A. (2007), "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," *Distributed Autonomous Robotic Systems 6*, 5, 221–230.

Nigam, N., and Kroo, I. (2008), "Persistent Surveillance Using Multiple Unmanned Air Vehicles," in *Aerospace Conference, 2008 IEEE*, Big Sky, MT.

Ny, J.L., Feron, E., and Frazzoli, E. (2012), "On the Dubins Travelling Salesman Problem," *IEEE Transactions on Automatic Control*, 57, 265–270.

Oh, H., Shin, H., Tsourdos, A., White, B., and Silson, P. (2011), "Coordinated Road Network Search for Multiple UAVs Using Dubins Path," in *1st CEAS Specialist Conference on Guidance, Navigation and Control*, Munich, Germany.

Pearna, W.L., and Chiub, W.C. (2005), "Approximate solutions for the maximum benefit chinese postman problem," *International Journal of Systems Science*, 36, 815–822.

Perrier, N., Langevin, A., and Campbell, J. (2007), "A Survey of Models and Algorithms for Winter Road Maintenance," *Computers and Operational Research, Part IV: Vehicle Routing and Fleet Sizing for Plowing and Snow Disposal*, 34, 258–294.

Rathinam, S., Sengupta, R., and Darbha, S. (2007), "A Resource Allocation Algorithm for Multi-vehicle systems with Non holonomic Constraints," *IEEE Transactions on Automation Sciences and Engineering*, 4, 98–104.

Rosenkrantz, D., Stearns, R., and Lewis, P. (2009), "An Analysis of Several Heuristics for the Traveling Salesman Problem," *Fundamental Problems in Computing*, 1, 45–69.

Ryan, J., Bailey, T., Moore, J., and Carlton, W. (1998), "Reactive Tabu Search in Unmanned Aerial Reconnaissance Simulations," in *30th Conference on Winter Simulation*, Washington, DC.

Salva, K., Frazzoli, E., and Bullo, F. (2008), "Traveling Salesperson Problems for the Dubins Vehicle," *IEEE Trans. on Automatic Control*, 53(6), 1378–1391.

Shanmugavel, M., Tsourdos, A., White, B.A., and Zbikowski, R. (2007), "Differential geometric path planning of Multiple UAVs," *Journal of Dynamic Systems, Measurement, and Control*, 129, 620–632.

Shanmugavel, M., Tsourdos, A., White, B.A., and Zbikowski, R. (2010), "Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs," *Control Engineering Practice*, 18(9), 1084–1092.

Shima, T., and Rasmussen, S., *UAV Cooperative Decision and Control: Challenges and Practical Approaches*, SIAM, Philadelphia (2010).

Shkel, A., and Lumelsky, V. (2001), "Classification of the Dubins set," *Robotics and Autonomous Systems*, 34, 179–202.

Tang, Z., and Ozguner, U. (2005), "Motion Planning for Multitarget Surveillance with Mobile Sensor Agents," *IEEE Transactions on Robotics*, 21, 898–908.

Tomlin, C., Mitchell, I., and Ghosh, R. (2001), "Safety Verification of Conflict Resolution Manoeuvres," *IEEE Transactions on Intelligent Transportation System*, 2(2), 110–120.

Yu, J., and Wang, L. (2012), "Group consensus of multi-agent systems with directed information exchange," *International Journal of Systems Science*, 43, 334–348.