JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 5, February 2008

Finding Good Dubins Tours for UAVs Using Particle Swarm Optimization

Richard J. Kenefic
Raytheon, 1010 Production Rd., Ft. Wayne IN 46808

DOI: 10.2514/1.35658

Unmanned aerial vehicles (UAVs) perform important surveillance functions on the battlefield. When a set of fixed targets for surveillance is known a priori it is desirable to find a minimum length tour subject to the kinematic constraints of the UAV and the terrain and threat constraints of the environment. The UAV path planning problem can be reduced to the traveling salesman problem (TSP) or the vehicle routing problem (VRP), both of which are known to be NP-hard. The problem of finding an optimal tour subject to kinematic constraints, the Dubins vehicle problem (DVP), is also NP-hard and several heuristics have recently been proposed. In this paper several instances of the DVP are posed and solved with a heuristic and the particle swarm optimization method. The particle swarm optimization (PSO) results are compared to another standard optimization method, and the best configurations found for these instances are reported.

I. Introduction

THE importance of autonomous vehicles in future military operations is generally recognized.[1–3] These vehicles include unmanned air vehicles (UAVs), unmanned ground vehicles (UGVs), and unmanned underwater vehicles (UUVs) with varying levels of autonomy and mission complexity. Both UAVs and UGVs have prominent roles in many planned military systems, and the control of multiple UAVs has been proposed as an AFOSR grand challenge.[1]

Some of the work on path planning for a fleet of UAVs[4] has noted that, in the absence of kinematic constraints, the problem can be reduced to the well known capacitated vehicle routing problem (CVRP). The inclusion of UAV kinematic constraints has been addressed[5,6] for a single stationary target and a single UAV using the Dubins set.[7] Using optimal control, it can be shown that the Dubins set will produce the minimum time trajectory for moving the UAV from an initial to a final configuration.[8] Path planning for UAVs following a moving target has recently been addressed[9] using a more general kinematic model for the UAV and optimal control methods to determine the acceleration and bank angle. A more complex path planning scenario has also been considered[10] where environmental factors, obstacles, and kinematic constraints were all considered for a single UAV.

Path planning for UAVs involves elements of both the traveling salesman (TSP) and vehicle routing problems (VRP). If the mission is a surveillance tour, and the UAVs are to be recovered, the depot and its heading constraints must be considered, and the problem is best modeled as a turn rate limited VRP. One of the most studied of the VRPs is the vehicle routing problem with time windows. In the VRPTW the objective is to first minimize the number of vehicles and then to minimize the length of the route plan. Recent heuristics for the VRPTW[11,12] have shown good performance against standard VRP benchmarks. To the best of this author's knowledge, no one has yet added turn rate constraints to a VRP.

If the mission is surveillance followed by the destruction of a fixed set of targets, then the UAVs are not recovered, and the problem becomes one of finding the least number of vehicles and the set of paths for each such that each

Received 13 November 2007; revision received 9 January 2008; accepted for publication 17 January 2008.

vertex is visited exactly once and each UAV path length is less than the UAV range. This problem is a generalization of orienteering,[13] which is closely related to TSP, and doesn't require that UAVs follow a closed path. An advantage of this model is that target values can be assigned when there are more targets to visit than there are vehicles available to execute the surveillance plan, and the objective then becomes one of maximizing the collected target value with the specified number of vehicles. Reference 13 has also shown good performance against standard benchmarks for orienteering. To the best of this author's knowledge, no one has yet added turn rate constraints to orienteering.

Other recent work in this area has considered multiple UAVs and has included the turn rate constraints.[14,15] These references treat the problem as a turn rate limited traveling salesman problem (kTSP) rather than a vehicle routing problem by omitting the UAV depot, and they restrict the target spacing so that the resulting objective function for the unknown headings will be continuous.[8] In practice such a restriction can be problematic because the turn radius for inexpensive loitering weapons, like LOCASS, can be large compared with the target spacing.[5] Typical methods for dealing with discontinuous multi-modal objective functions make use of evolutionary methods,[16] but these methods have not been applied to the problem.

The turn rate limited TSP has also been addressed,[17,18] and some heuristics have been proposed and performance bounds derived. One of these heuristics, called the alternating algorithm, was based on selecting alternate legs of the TSP and filling in with the optimal Dubins paths. A few instances were presented in these references, but none of these benchmarks were published.

In this paper an algorithm for the turn rate limited TSP is proposed and tested. The target spacing constraint used in references 14 and 15 wasn't used here. Instead, to cope with the discontinuous objective function, the particle swarm optimization method[19–21] was used to determine the heading at each vertex, and TSP order was used to determine tour order. The algorithm is first tested on a simple benchmark where the length of the optimal tour can be tightly bounded from above and below, and it is shown that the algorithm produces a result that satisfies these tight bounds. The algorithm is then tested on four instances to show how it performs across a range of cases, from well separated vertices to strings of closely spaced vertices. The results are compared to those obtained using a standard technique. Following the practice common in the published studies of VRP and orienteering, these benchmarks are included to facilitate comparison with other methods.

II. Dubins Tours

When the vehicle has kinematic constraints of the form

$$\begin{aligned} \dot{x} &= V\cos(\theta) \\ \dot{y} &= V\sin(\theta), \\ \left|\dot{\theta}\right| &\le \Omega \end{aligned} \tag{1}$$

where (x, y) are position, V is velocity, and θ is heading, it can be shown[8] that the minimum time vehicle control, $\dot{\theta}(t)$, is a maximum effort (bang-bang) control. The vehicle turns at maximum rate left or right, or travels in a straight line. These class paths were first worked out by Dubins.[7] When the start and end point separation guarantees that the construction circles will not intersect, the paths are all C-L-C (circle-line-circle) class paths.[5] For closer spacing it's necessary to consider C-C-C (circle-circle-circle) class paths as well.[6] There will be no more than six feasible paths from $(x_1, y_1, \theta_1) \rightarrow (x_2, y_2, \theta_2)$, where (x, y) is the position and θ is the heading. Each of these paths is easy to construct using simple co-ordinate transformations and standard geometric techniques. The Dubins path is then the shortest of all these feasible paths. The UAV vehicle control changes at the tangent points along the Dubins path, and is one of $(+\Omega, -\Omega, 0)$ for a right turn, left turn, or straight path, respectively. A Dubins tour is composed by concatenation of the Dubins paths around the vertices to form a closed tour.

A good test case for any algorithm that constructs a Dubins tour is obtained by placing a set of vertices on a circle larger than the turn radius of the UAV.

It's easy to see that the Dubins tour must lie between the circle and the inscribed polygon. The proof follows by concatenating C-L-C paths between the vertices. Hence the circumference of the circle is an upper bound on the length of the Dubins tour, and the length of the polygon, which is the TSP tour length, is a lower bound on the length of the Dubins tour. Such a test case is shown for the algorithm proposed here in Fig. 1, where the

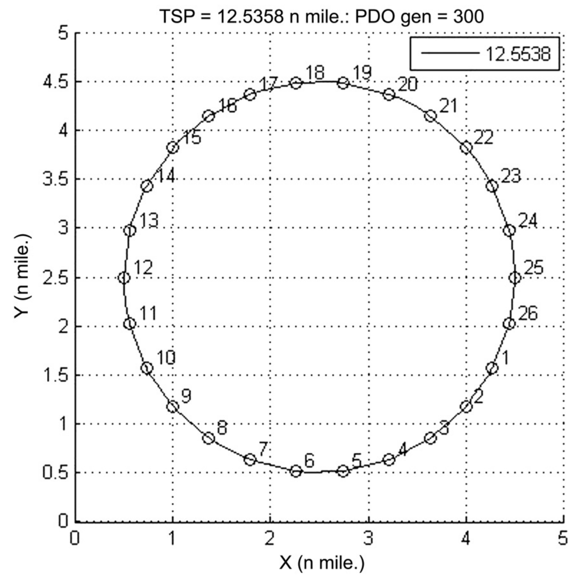

Fig. 1 A test case for the DVP algorithm used here shows that TSP < DVP < 4π, as expected.

radius of the circle is four times the turn radius (0.5 n mile.) of the UAV. For 26 vertices, the DVP tour length obtained by PSO is DVP = 12.5538. The TSP tour length is TSP = 26(4sin(360/52)) = 12.5358 n mile. Hence, $12.5358 < \mathrm{DVP} < 4\pi = 12.566$, as expected. Note that the vertex spacing is 0.482 n mile, well within the UAV turn radius, and that, for this set of instances, TSP tour order is also correct for DVP. The Matlab routine "fmincon.m" produced a tour length of 12.5432 n mile.

III. Particle Swarm Optimization

If the TSP tour order is accepted for an instance of DVP, the headings $(\theta_1, \ldots, \theta_N)$ remain to be determined. The results for PSO for a 10 vertex instance are presented in Fig. 2. The PSO results are compared to the results from a MATLAB routine* in the optimization toolbox as shown in Fig. 2.

A. PSO Equations

Particle swarm optimization was first proposed by Kennedy and Eberhart,[19] and several versions of PSO are commonly used.[20,21] The version implemented here is given by

$$\begin{aligned}\bar{v}_{m,k} &= \chi(\bar{v}_{m,k-1} + c_1 \cdot r_1(\bar{x}_{m,best} - \bar{x}_{m,k-1}) + c_2 \cdot r_2(\bar{x}_{gbest} - \bar{x}_{m,k-1}))\\ \bar{x}_{m,k} &= \bar{x}_{m,k-1} + \bar{v}_{m,k}\end{aligned} \tag{2}$$

where r_1 and r_2 are independent uniform random variables on [0, 1], χ is the contraction parameter, c_1 and c_2 are the cognitive and social parameters, $\bar{x}_{m,k-1}$ is the m^th particle position at the $k-1$st generation, $\bar{x}_{m,best}$ is the position at which the m^th particle found its shortest tour, $\bar{x}_{gbest}$ is the position at which the global optimum (shortest tour over all particles) was found, and $\bar{v}_{m,k-1}$ is the velocity of the m^th particle at the $k-1$st generation.

Each particle is an N-tuple of headings, and it's necessary to keep the headings between $-\pi$ and π as the swarm evolves. The particle velocities were limited by $|\bar{v}_{m,k}| \leq V_{\max}$. For the examples presented here, $\chi = 0.729, c_1 = 2.05$, $c_2 = 2.05$ and $V_{\max} = 0.2\pi$. The contraction parameter, cognitive parameter, and social parameter used here were suggested in reference 21. In all cases there were 20 particles and the PSO was run for 200–300 generations. Using

* fmincon requires a continuous objective function, which cannot be guaranteed in this application. The number of function evaluations and the number of iterations must be limited to keep this optimizer from getting trapped in an infinite loop.

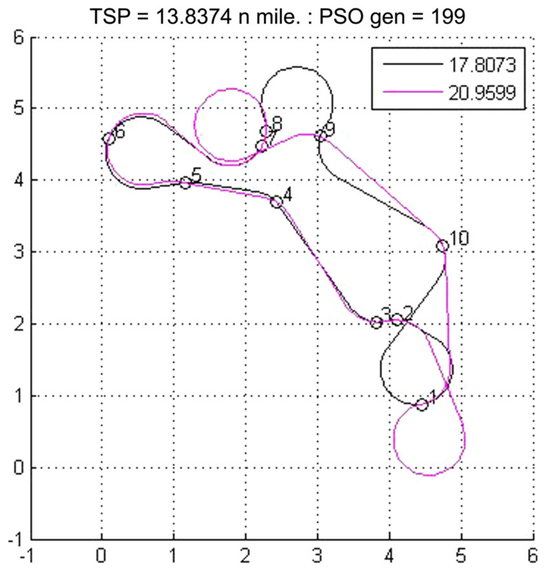

Fig. 2 PSO finds a tour of length 17.8 n mile. within 200 generations. The MATLAB routine "fmincon.m" finds a tour of length 21 n mile. when initialized with the same headings.

the initialization and heading constraints suggested here, a DVP configuration that's better than the "fmincon.m" configuration is found within a few dozen generations.

B. Initialization and Heading Constraints

An efficient initialization of the particle swarm is obtained by first computing a preferred set of headings and then adding independent random perturbations to initialize each particle position. The preferred headings are obtained by marking vertices (i.e. set_heading(n) = TRUE in Table 1) in tour order that lie within the vehicle turn radius of the next vertex in the tour. If only two consecutive vertices are found, the preferred heading at each is the heading of the line that joins them. If a longer sequence of these marked vertices is found, the AA heuristic is applied. If the vertex is unmarked (i.e. set_heading(n) = FALSE in Table 1), the preferred heading is set to the average of the inbound and outbound headings in the TSP tour. Some pseudo code that performs this initialization is presented in Table 1.

Unmarked vertices have an independent random variable uniform on $[-\pi/2, \pi/2]$ added for each particle, but marked vertices have an independent random variable uniform on $[-\phi, \phi]$ added, where $\phi = \pi/8$. Velocities were initialized with independent random variables uniform on $[-V_{\max}/2, V_{\max}/2]$. In addition, for any marked vertex in a particle, if the heading departs by more than ϕ from the preferred heading, that particle's vertex heading is re-initialized.

IV. DVP Instances

Standard benchmarks available for VRPs use 25, 50, or 100 (or more) vertices, and are unsuitable for the current study. For all instances considered in this section, vertices are generated independently and uniformly in a 5 by 5 n mile square. Random instances of this type are among the most difficult of the standard VRP instances. The cases considered here use 10, 20, and 30 vertices. In every instance, the UAV turn radius is 0.5 n mile. The MATLAB function "fmincon.m" was initialized in the same manner as the PSO, but no heading constraints were enforced.

C. Case One: 10 vertices

A 10 vertex instance is shown in Fig. 2. The best configuration found for this instance is presented in Table 2.

Table 1 Pseudo code for the particle initialization.

```
if set_heading(n)==FALSE
    preferred_heading(n) = average_heading;
    alt = 0; %--Length of the string. Used in AA heuristic
else
    if (set_heading(n-1)==FALSE)&(set_heading(n+1)==TRUE)
        preferred_heading(n) = heading_to(n+1);
        alt = 1;
    elseif (set_heading(n-1)==TRUE)&(set_heading(n+1)==FALSE)
        alt = alt + 1;
        if mod(alt,2)==0
            preferred_heading(n) = heading_from(n-1);
        else
            set_heading(n) = FALSE;
            preferred_heading(n) = average_heading;
            alt = 0;
        end
        alt = 0;
    else
        alt = alt + 1;
        if mod(alt,2)==0
            preferred_heading(n) = heading_from(n-1);
        else
            preferred_heading(n) = heading_to(n+1);
        end
    end
end
```

Table 2 Configuration of the best Dubins' tour found for the instance considered in Fig. 2.

10 Vertex Configuration (17.8073 n mile.)			
Vertex	X (n mile.)	Y(n mile.)	Heading (deg.)
1	4.4565	0.8813	8.8383
2	4.1070	2.0514	−172.2824
3	3.8105	2.0285	−186.3066
4	2.4299	3.6910	151.3849
5	1.1557	3.9597	179.4775
6	0.0925	4.5845	70.1655
7	2.2235	4.4682	62.2307
8	2.2823	4.6773	85.5215
9	3.0342	4.6091	−102.8277
10	4.7506	3.0772	−74.8803

D. Case Two: 20 well separated vertices

A 20 vertex example is shown in Fig. 3, where the TSP tour is 20.82 n mile long. The result from a heading constrained PSO is also shown in Fig. 3. The heading constrained PSO finds a Dubins' tour 32.41 n mile long within

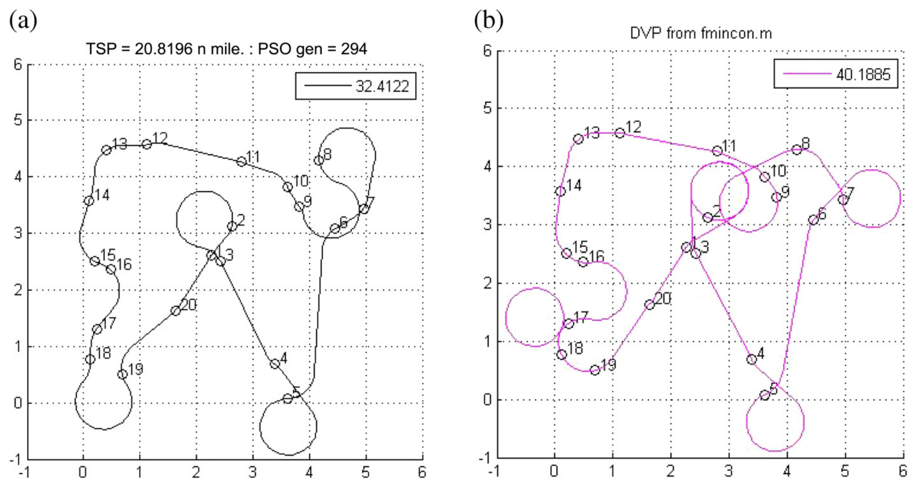

Fig. 3 A 20 vertex instance. The TSP tour is 20.82 n mile long. PSO finds a Dubins' tour of length 32.41 n mile within 300 generations. fmincon finds a tour of length 40.19 n mile.

Table 3 Configuration of the best Dubins' tour found for the instance considered in Fig. 3.

20 Vertex Configuration (32.4122 n mile.)			
Vertex	X (n mile.)	Y (n mile.)	Heading (deg.)
1	2.2546	2.6035	59.1261
2	2.6256	3.1251	76.0952
3	2.4228	2.5158	−58.6703
4	3.3907	0.6873	−32.7905
5	3.6114	0.074	1.8497
6	4.4581	3.0811	31.9315
7	4.9582	3.4337	55.6368
8	4.1738	4.2999	−82.1229
9	3.829	3.4633	123.2313
10	3.6094	3.8303	115.0049
11	2.7879	4.2613	164.6853
12	1.1198	4.5729	−161.565
13	0.4101	4.4763	−146.64
14	0.1032	3.5817	−116.557
15	0.1898	2.5123	−28.7057
16	0.4761	2.3575	−42.6881
17	0.2292	1.31	−127.001
18	0.1103	0.7765	−81.6674
19	0.6836	0.4954	102.7089
20	1.6244	1.6344	45.7732

300 generations. This instance has well separated vertices so that the AA heuristic wasn't applied during initialization. The best configuration found for this instance is given in Table 3.

E. Case Three: 20 vertices with a string of close vertices

Another 20 vertex instance is shown in Fig. 4. For this instance there is a long string of marked vertices found in the initialization step and the AA heuristic was applied. The PSO finds a Dubins tour of length 32.56 n mile within 300 generations. The configuration of this tour is shown in Table 4.

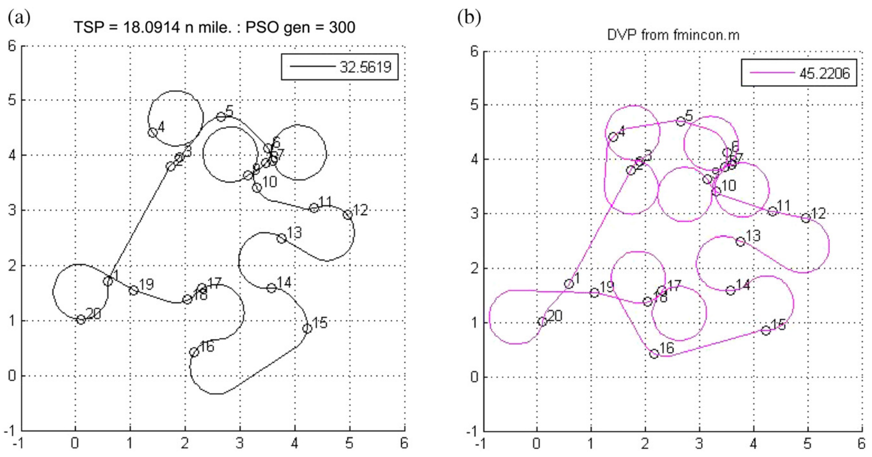

Fig. 4 A 20 vertex instance. The TSP tour is 18.09 n mile long. PSO finds a Dubins' tour of length 32.56 n mile within 300 generations. fmincon finds a tour of length 45.22 n mile.

Table 4 Configuration of the best Dubins' tour found for the instance considered in Fig. 4.

20 Vertex Configuration (32.5619 n mile.)			
Vertex	X (n mile.)	Y (n mile.)	Heading (deg.)
1	0.5957	1.71	79.93
2	1.7292	3.8088	51.29
3	1.8923	3.967	45.2844
4	1.3977	4.4162	−59.3095
5	2.6509	4.7029	9.2357
6	3.5148	4.1214	−60.6423
7	3.5958	3.9031	−65.7977
8	3.4787	3.8699	−127.972
9	3.1414	3.6344	−140.488
10	3.3053	3.4091	−62.228
11	4.3429	3.037	11.8537
12	4.9637	2.9113	−49.4784
13	3.7632	2.4994	145.8692
14	3.5765	1.5957	−0.8902
15	4.2256	0.8587	−96.576
16	2.1701	0.4299	56.9641
17	2.3091	1.5805	−148.322
18	2.048	1.3763	−154.097
19	1.0603	1.5468	154.0959
20	0.1011	1.0249	1.0363

F. Case Four: 30 well separated vertices

For 30 vertices, long strings of marked vertices are common due to the density of vertices in the target space. For the instance shown in Figure 5 and Table 5, the vertices are well separated and the AA heuristic wasn't used in the particle initialization. As the target density increases, the particles are initialized near the headings found using the AA heuristic.

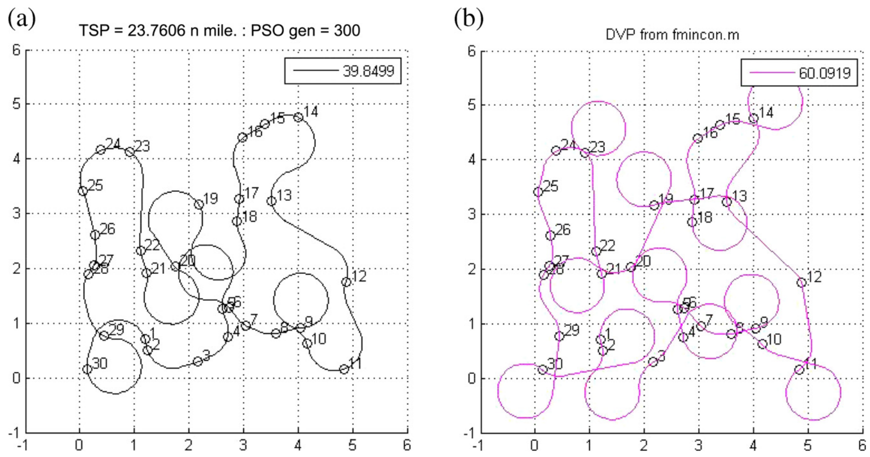

Fig. 5 A 30 vertex instance. The TSP tour is 23.76 n mile long. PSO finds a Dubins' tour of length 39.85 n mile within 300 generations. fmincon finds a tour of length 60.09 n mile.

Table 5 Configuration of the best Dubins' tour found for the instance considered in Fig. 5.

30 Vertex Configuration (39.8499 n mile.)			
Vertex	X (n mile.)	Y (n mile.)	Heading (deg.)
1	1.1911	0.7115	−71.6412
2	1.2374	0.509	−70.7085
3	2.1711	0.2931	27.3418
4	2.7083	0.7542	77.8969
5	2.6155	1.2714	97.5068
6	2.7278	1.2912	−44.391
7	3.0369	0.9614	−28.7775
8	3.5959	0.8155	11.6599
9	4.0521	0.914	8.4693
10	4.1593	0.6179	−75.0998
11	4.8351	0.1563	18.1219
12	4.8802	1.7582	69.8832
13	3.5147	3.2198	99.2982
14	4.0037	4.7611	156.8801
15	3.3971	4.6369	−156.333
16	2.9884	4.3889	−140.407
17	2.929	3.2667	−105.478
18	2.885	2.8489	−92.4509
19	2.1726	3.1754	122.9362
20	1.7456	2.0322	−43.5484
21	1.2278	1.9087	101.0396
22	1.1101	2.3345	97.7063
23	0.9214	4.1212	147.0868
24	0.3907	4.1604	−161.54
25	0.0539	3.4127	−64.0537
26	0.2801	2.618	−80.7304
27	0.2546	2.0509	−116.599
28	0.1493	1.8902	−123.383
29	0.4396	0.7674	−27.5271
30	0.1343	0.1513	97.7949

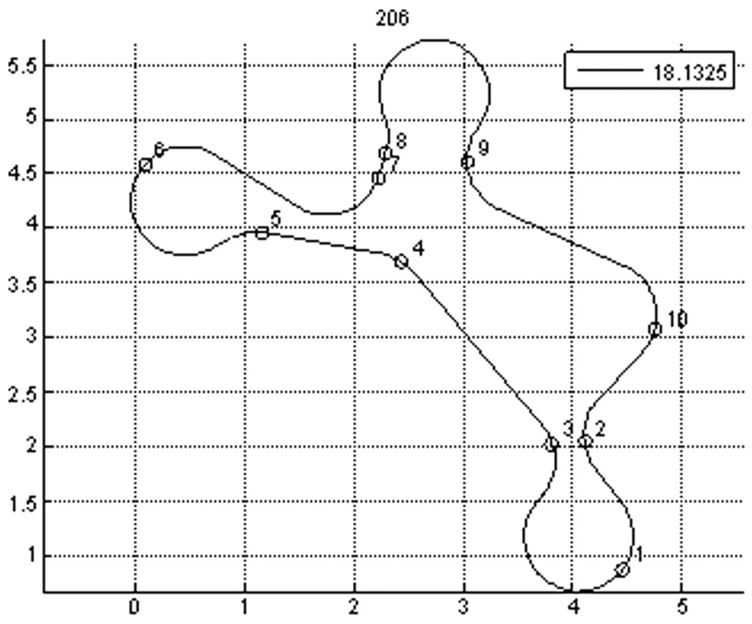

Fig. 6 A Dubins' tour with no crossing paths for the 10 vertex instance presented in Fig. 2.

V. Conclusions

A heuristic that used TSP tour order along with a heading constraint heuristic and particle swarm optimization to find vehicle headings was proposed and tested for the Dubins vehicle problem. Several instances of DVP were examined and the best configurations found for 10, 20, and 30 vertices uniformly distributed within 25 n mile^2 were reported. As expected, PSO outperforms "fmincon.m" from the MATLAB optimization toolbox due to the discontinuous and multimodal nature of the objective function. These instances can be used for comparison with other approaches to DVP. As the density of targets increases, the preferred headings used in the particle initialization reduce to those found from the alternating algorithm (AA) heuristic.[17]

It is interesting to consider permutations of the TSP tour order after the PSO has completed its processing. Currently, there is no reason to expect the TSP tour order to be optimal for large values of turn radius. For the instance presented in Fig. 2 it is possible to remove the self-crossing in the path by exchanging vertices 1 and 2 in the TSP tour order, as shown in Fig. 6. However, the tour length is not reduced by this permutation.

Adjusting tour order could shorten the tour length, and a heuristic that examines permutations near these crossings might be worth trying. So far, a Dubins tour shorter than the 10 vertex tour presented in Table 2 hasn't been found for this instance. Some theoretical work on this problem has been reported in reference 18.

References

[1]McLaughlin, A. J. et al., *Basic Research in Information Science and Technology for Air Force Needs*, The National Academies Press, 2006.

[2]Deyst, J. J. et al., *Autonomous Vehicles in Support of Naval Operations*, The National Academies Press, 2005.

[3]Albus, J. et al., 4D/RCS: A Reference Architecture for Unmanned Vehicle Systems, Version 2.0, NISTIR 6910, August 2002.

[4]Russell, M. A., and Lamont, G. B., "A Genetic Algorithm for Unmanned Aerial Vehicle Routing", *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, Washington DC., pp. 1523–1530.

[5]Yang, G., and Kapila, V., "Optimal Path Planning for Unmanned Air Vehicles with Kinematic and Tactical Constraints", *41st IEEE Conference on Decision and Control*, December 2002, pp. 1301–1306.
doi: 10.1109/CDC.2002.1184695

[6]Wong, H., Kapila, V., and Vaidyanathan, R., "UAV Optimal Path Planning using C-C-C Class paths for Target Touring", *43rd IEEE Conference on Decision and Control*, December 2004, pp. 1105–1110.

[7]Dubins, L. E., "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", *American Journal of Mathematics*, Vol. 79, 1957, pp. 497–516.
doi: 10.2307/2372560

[8]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006.

[9]Geiger, B. R. et al., "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming", *AIAA GNC Conference*, August 2006.

[10]Peot, M. A. et al., "Planning Sensing Actions for UAVs in Urban Domains", *Proceedings of SPIE*, Vol. 5986, 2005.

[11]Bent, R., and Van Hentenryck, P., "A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows", *Transportation Science*, Vol. 38, No. 4, November 2004, pp. 515–530.

[12]Braysy, O., Hasle, G., and Dullaert, W., "A Multi-Start Local Search Algorithm for the Vehicle Routing Problem With Time Windows", *European Journal of Operational Research*, pp. 586–605, 2004.
doi: 10.1016/S0377-2217(03)00435-1

[13]Fischetti, M., Gonzalez, J. J. S., and Toth, P., "Solving the Orienteering Problem Through Branch-And-Cut", *INFORMS Journal on Computing*, Vol. 10, No. 2, 1998, pp. 133–148.

[14]Rathinam, S., Sengupta, R., and Darbha, S., "A Resource Allocation Algorithm for Multivehicle Systems with Nonholonomic Constraints", *IEEE Transactions on Automation Science and Engineering*, Vol. 4, No. 1, pp. 98–104, January 2007.

[15]Yadlapali, S., Malik, W.A., Darbha, S., and Pachter, M., "A Lagrangian Based Algorithm for a Multi-Depot, Multiple Travelling Salesman Problem", *Proceedings of the 2007 American Control Conference*, pp. 4027–4032.

[16]Haupt, R., and Haupt, S. E., *Practical Genetic Algorithms*, Wiley, 2004, pp. 151–154.

[17]Salva, K., Frazzoli, E., and Bullo, F., "On the Point-to-Point and Traveling Salesperson Problems for Dubins' Vehicle", *2005 American Control Conference*, Portland, OR, pp. 786–791.

[18]Ny, J. L., and Feron, E., "An Approximation Algorithm for the Curvature-Constrained Traveling Salesman Problem", *43rd Annual Allerton Conference on Communications, Control, and Computing*, September 28–30, 2005.

[19]Kennedy, J. and Eberhart, R., "Particle Swarm Optimization", *Proceedings of the IEEE International Conference on Neural Networks"*, December 1995, pp. 1942–1948.
doi: 10.1109/ICNN.1995.488968

[20]Clerc, M., and Kennedy, J., "The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space", *IEEE Transactions on Evolutionary Computation*, February 2002, pp. 58–73.
doi: 10.1109/4235.985692

[21]Parsopoulos, K. E., and Vrahatis, M. N., "Unified Particle Swarm Optimization for Tackling Operations Research Problems", *IEEE 2005 Swarm Intelligence Symposium*, Pasadena, CA, pp. 53–59, 2005.

Christopher Rouff
Associate Editor