Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning

Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté

***Abstract*—The development of autonomous unmanned aerial vehicles (UAVs) is of high interest to many governmental and military organizations around the world. An essential aspect of UAV autonomy is the ability for automatic path planning. In this paper, we use the genetic algorithm (GA) and the particle swarm optimization algorithm (PSO) to cope with the complexity of the problem and compute feasible and quasi-optimal trajectories for fixed wing UAVs in a complex 3D environment, while considering the dynamic properties of the vehicle. The characteristics of the optimal path are represented in the form of a multiobjective cost function that we developed. The paths produced are composed of line segments, circular arcs and vertical helices. We reduce the execution time of our solutions by using the "single-program, multiple-data" parallel programming paradigm and we achieve real-time performance on standard commercial off-the-shelf multicore CPUs. After achieving a quasi-linear speedup of 7.3 on 8 cores and an execution time of 10 s for both algorithms, we conclude that by using a parallel implementation on standard multicore CPUs, real-time path planning for UAVs is possible. Moreover, our rigorous comparison of the two algorithms shows, with statistical significance, that the GA produces superior trajectories to the PSO.**

***Index Terms*—Genetic algorithm (GA), path planning, particle swarm optimization (PSO), parallel computing, unmanned aerial vehicles (UAVs).**

I. Introduction

THE PATH planner is a key element of the unmanned aerial vehicle (UAV) autonomous control module [1]. It allows the UAV to autonomously compute the best path from a start point to an end point. Whereas commercial airlines fly constant prescribed trajectories, UAVs in operational areas have to travel constantly changing trajectories that depend on the particular terrain and conditions prevailing at the time of their flight.

In the past, the best path has been associated with the shortest path and deterministic search algorithms were used to find the very shortest path. The definition of the problem has since evolved and the best path is now associated with the path that minimizes the distance travelled, the average altitude, the fuel consumption, the radar exposure, etc. These are a few examples of the factors to be considered and clearly show that the complexity of the problem has grown. To cope with this complexity, researchers have slowly moved from using deterministic algorithms to using nondeterministic algorithms [2].

As an example, the authors of [3] proposed the use of a vibrational genetic algorithm (GA) to improve the exploration and better avoid local minima when searching for an optimal path. In [4], Bezier curves are used with a GA to produce trajectories that respect a minimum curvature and torsion in an effort to better consider the dynamic properties of the UAV. The authors of [5] also use a GA, but add the concept of artificial immune system in order to maintain superior population diversity throughout the evolution process. In [6], the authors use a PSO to compute the shortest path between targets in the context of surveillance UAV. In [7], a comparison of the PSO, the phase angle-encoded PSO and the quantum-behaved PSO in the context of UAV path planning is presented.

In this paper, we use two nondeterministic algorithms to develop an operational path planning module for fixed wing UAVs. Our research work presents three important contributions. First, we propose a comprehensive cost function which includes both the optimization and the feasibility criteria. This allows us to use a generic optimization algorithm (without modification) as the search algorithm. In our case, we use the GA and the PSO, but these could easily be replaced by other algorithms. Second, we present a technique to parallelize both the GA and the PSO while minimizing the communication between the processes in order to achieve a near linear speedup and fully exploit the computing power of today's multicore CPUs. Finally, we offer a statistically significant comparison between the quality of the trajectories generated by our GA-based and PSO-based path planners. Both algorithms have recently been widely used for UAV path planning [3]–[8] and [9]. However, to our knowledge, there exists no rigorous comparison between the two algorithms when applied to this particular problem. The results we present in this paper provide clear insight as to which of the two optimization algorithms is preferable for UAV path planning in complex 3D environments.

Nondeterministic algorithms have been used to cope with the complexity of UAV path planning; however, they could potentially produce, on some occasions, poor solutions. A subsequent re-execution of the algorithm would then be required. It is important to note that the path planner is a submodule of an UAV autonomous control system that includes a decision-making module to handle such situations [1]. For this reason, this paper does not cover the decision making process that would be required in the event of a poor trial run.

Manuscript received October 21, 2011; revised December 07, 2011, February 14, 2012; accepted April 02, 2012. Date of publication May 10, 2012; date of current version December 19, 2012. Paper no. TII-11-646.R2.

V. Roberge and M. Tarbouchi are with the Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON K7K7L6, Canada (e-mail: vincent.roberge@rmc.ca; tarbouchi-m@rmc.ca).

G. Labonté is with the Department of Mathematics and Computer Science, Royal Military College of Canada, Kingston, ON K7K7L6, Canada (e-mail: labonte-g@rmc.ca).

Digital Object Identifier 10.1109/TII.2012.2198665

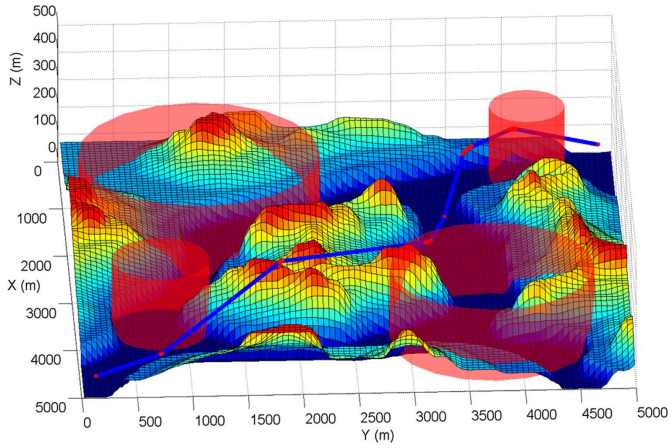
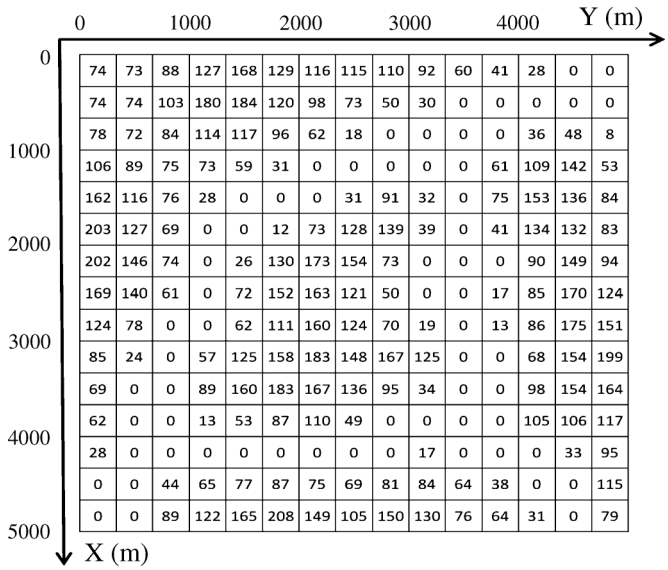


Fig. 1. UAV trajectory in a 3D environment. The red dots represent the waypoints of the UAV trajectory, the blue line represents the line segments connecting the waypoints and the red cylinders represents the danger zones to be avoided.



74	73	88	127	168	129	116	115	110	92	60	41	28	0	0
74	74	103	180	184	120	98	73	50	30	0	0	0	0	0
78	72	84	114	117	96	62	18	0	0	0	0	36	48	8
106	89	75	73	59	31	0	0	0	0	0	61	109	142	53
162	116	76	28	0	0	0	31	91	32	0	75	153	136	84
203	127	69	0	0	12	73	128	139	39	0	41	134	132	83
202	146	74	0	26	130	173	154	73	0	0	0	90	149	94
169	140	61	0	72	152	163	121	50	0	0	17	85	170	124
124	78	0	0	62	111	160	124	70	19	0	13	86	175	151
85	24	0	57	125	158	183	148	167	125	0	0	68	154	199
69	0	0	89	160	183	167	136	95	34	0	0	98	154	164
62	0	0	13	53	87	110	49	0	0	0	0	105	106	117
28	0	0	0	0	0	0	0	0	17	0	0	0	33	95
0	0	44	65	77	87	75	69	81	84	64	38	0	0	115
0	0	89	122	165	208	149	105	150	130	76	64	31	0	79

Fig. 2. 2D matrix representation of the terrain displayed on Fig. 1. The number in each cell represents the terrain elevation at that location.

The remainder of this paper is organized into sections. Section II provides details of the representation of the terrain and the encoding of the trajectories. We present in Section III the cost function we developed to evaluate the quality of candidate trajectories. Sections IV and V, respectively, discuss the use of the GA and the PSO to find trajectories that optimize the cost function. Section VI briefly presents the method we used to smooth the generated path into a feasible path with no discontinuity in the velocity. Section VII explains the method used to parallelize the GA and the PSO in order to allow real-time path planning. Finally, we compare the quality of the trajectories produced by the GA and the PSO in Section VIII.

II. Environment and Trajectory Representation

The first step of path planning is to discretize the world space into a representation that will be meaningful to the path planning algorithm. This representation is closely related to a search algorithm and some algorithms will only perform well when coupled with a specific environment representation. An overview of the performance of different representations used with different algorithms is presented in [10]. In our implementation (see Figs. 1 and 2), we use an approximate cell decomposition of the terrain using a 2D grid where each element of the matrix represents the elevation of the terrain. This representation allows us to use digital elevation maps freely available from the GeoBase [11] repository with no further processing. Our representation of the environment also allows for the definition of cylindrical danger zones (or no-fly zones) to be kept in a separate matrix where each row represents the coordinates (x_i, y_i) and the diameter d_i of the ith cylinder as shown in (1). Complex no-fly zones can be built by partially juxtaposing multiple cylinders

$$\text{danger zones} = \begin{bmatrix} x_1 & y_1 & d_1 \\ x_2 & y_2 & d_2 \\ \cdots & \cdots & \cdots \\ x_n & y_n & d_n \end{bmatrix}. \tag{1}$$

The trajectories generated by the optimization algorithm are composed of line segments and encoded in a matrix where each row represents the (x_i, y_i, z_i) coordinates of the ith waypoint, as shown in (2). The trajectories are flown at constant speed

$$\text{trajectory} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \cdots & \cdots & \cdots \\ x_n & y_n & z_n \end{bmatrix}. \tag{2}$$

III. Cost Function

As previously stated, searching for the best path is often associated with searching for the shortest path. This is the case when solving the Traveling Salesperson Problem (TSP), which consists of finding the shortest path that visits all the given cities only once. In the case of UAV path planning, the optimal path is more complex and includes many different characteristics. To take into account these desired characteristics, a cost function is used and the path planning algorithm becomes a search for a path that will minimize the cost function. The cost of a path decreases with the degree to which the desired characteristics are being fulfilled. A path that fulfills all the characteristics to a high degree would result in a low cost. We define our cost function as follows:

$$F_{\text{cost}} = C_{\text{length}} + C_{\text{altitude}} + C_{\text{danger zones}} + C_{\text{power}} + C_{\text{collision}} + C_{\text{fuel}} + C_{\text{smoothing}} \tag{3}$$

where C_{length} penalizes longer paths, C_{altitude} penalizes paths with a higher average altitude, $C_{\text{danger zones}}$ penalizes paths going through danger zones, C_{power} penalizes paths requiring more power than the maximum available power of the UAV, $C_{\text{collision}}$ penalizes paths colliding with the ground, C_{fuel} penalizes paths requiring more fuel than initially available in the UAV, and finally, $C_{\text{smoothing}}$ penalizes paths that cannot be smoothed using circular arcs. C_{length}, C_{altitude} and $C_{\text{danger zones}}$ are optimization criteria and are used to improve the quality of the trajectory. Each of those three terms are defined to be in the range of [0, 1]. On the other hand, C_{power}, $C_{\text{collision}}$, C_{fuel}, and $C_{\text{smoothing}}$ are feasibility criteria that must be satisfied for the

final trajectory to be valid. Each feasibility criterion is defined to be equal to 0, when it is respected, or in the range $[P, P+1]$ when it is not. By choosing a P greater than 3, we ensure that unfeasible trajectories always have a cost greater than any feasible ones. It is important to note that our implementation defines $C_{\text{danger zones}}$ as an optimization criterion in the sense that it is preferable to avoid danger zones but not necessarily essential to the survival of the UAV.

This would simulate the scenario where the UAV should avoid enemy radars identified by the danger zones. If used in a different scenario, our cost function could easily be modified to include $C_{\text{danger zones}}$ as a feasibility criterion ensuring no danger zones are violated. Moreover, although we define danger zones as cylindrical regions, our approach also allows the representation of complex zones by overlapping multiple cylinders.

In our costs function, the term associated with the length of a path is defined as follows:

$$C_{\text{length}} = 1 - \left(\frac{L_{P1P2}}{L_{\text{traj}}}\right) \tag{4}$$

therefore

$$C_{\text{lenght}} \in [0, 1] \tag{5}$$

where L_{P1P2} is the length of the straight line connecting the starting point $P1$ and the end point $P2$ and L_{traj} is the actual length of the trajectory.

The term associated with the altitude of the path is defined as follows:

$$C_{\text{altitude}} = \frac{A_{\text{traj}} - Z_{\text{min}}}{Z_{\text{max}} - Z_{\text{min}}} \tag{6}$$

therefore

$$C_{\text{altitude}} \in [0, 1] \tag{7}$$

where Z_{max} is the upper limit of the elevation in our search space, Z_{min} is the lower limit and A_{traj} is the average altitude of the actual trajectory. Z_{max} and Z_{min} are respectively set to be slightly above the highest and lowest points of the terrain.

The term associated with the violation of the danger zones is defined as follows:

$$C_{\text{danger zones}} = \frac{L_{\text{inside } d.z.}}{\sum_{i=1}^{n} d_i} \tag{8}$$

with

$$C_{\text{danger zones}} \in [0, 1] \tag{9}$$

where n is the total number of danger zones, $L_{\text{inside } d.z.}$ is the total length of the subsections of the trajectory which go through danger zones and d_i is the diameter of the danger zone i. This definition ensures that a trajectory passing through a single danger zone is severely penalized on a map containing few danger zones and lightly penalized on a map containing many danger zones. Since it is possible for $L_{\text{inside } d.z.}$ to be larger than $\sum_{i=1}^{n} d_i$ (as in the case of a dog-leg path through a single danger zone), we arbitrarily bound $C_{\text{danger zones}}$ to 1.

The term associated with a required power higher than the available power of the UAV is defined as follows:

$$C_{\text{power}} = \begin{cases} 0, & L_{\text{not feasible}} = 0 \\ P + \left(\frac{L_{\text{not feasible}}}{L_{\text{traj}}}\right), & L_{\text{not feasible}} > 0 \end{cases} \tag{10}$$

therefore

$$C_{\text{power}} \in 0 \cup [P, P+1] \tag{11}$$

where $L_{\text{not feasible}}$ is the sum of the lengths of the line segments forming the trajectory which require more power than the available power of the UAV, L_{traj} is the total length of the trajectory and P is the penalty constant. This constant must be higher than the cost of the worst feasible trajectory which would have, based on our cost function, a cost of 3. By adding this penalty P, we separate nonfeasible solutions from the feasible ones. The power required and available depends on the altitude and the current weight of the UAV and is computed using the approach presented in [12] and based on [13].

The term associated with ground collisions is defined as follows:

$$C_{\text{collision}} = \begin{cases} 0, & L_{\text{under terrain}} = 0 \\ P + \left(\frac{L_{\text{under terrain}}}{L_{\text{traj}}}\right), & L_{\text{under terrain}} > 0 \end{cases} \tag{12}$$

therefore

$$C_{\text{collision}} \in 0 \cup [P, P+1] \tag{13}$$

where $L_{\text{under terrain}}$ is the total length of the subsections of the trajectory which travels below the ground level and L_{traj} is the total length of the trajectory. We compare the altitude of the terrain and the altitude of the trajectory in a discreet way using the Bresenham's line drawing algorithm [14].

The term associated with an insufficient quantity of fuel available is defined as follows:

$$C_{\text{fuel}} = \begin{cases} 0, & F_{\text{traj}} \leq F_{\text{init}} \\ P + 1 - \left(\frac{F_{P1P2}}{F_{\text{traj}}}\right), & F_{\text{traj}} > F_{\text{init}} \end{cases} \tag{14}$$

therefore

$$C_{\text{fuel}} \in 0 \cup [P, P+1] \tag{15}$$

where F_{P1P2} is the quantity of fuel required to fly the imaginary straight segment connection the starting point $P1$ to the end point $P2$, F_{traj} is the actual amount of fuel needed to fly the trajectory, F_{init} is the initial quantity of fuel onboard the UAV. The quantity of fuel required is computed using the approach presented in [12] and based on [13].

Because our representation defines a path as a series of line segments, the trajectory generated includes discontinuity in the velocity at the connections of those segments. As will be explained later in Section VI, we remove those discontinuity using circular constructions in the last step of our path planning algorithm. However, to ensure the smoothing of the optimal solution found is possible, we verify the feasibility of the smoothing in

our cost function. So finally, the term associated with the impossible smoothing of the path is defined as follows:

$$C_{\text{smoothing}} = \begin{cases} 0, & N_{\text{impossible}} = 0 \\ P + \left(\frac{N_{\text{impossible}}}{N_{\text{total}}}\right), & N_{\text{impossible}} > 0 \end{cases} \quad (16)$$

therefore

$$C_{\text{smoothing}} \in 0 \cup [P, P+1] \quad (17)$$

where $N_{\text{impossible}}$ is the number of connection that cannot be smoothed using circular constructions as explained in Section VI and N_{total} is the total number of connections in the trajectory.

During the optimization phase of our path planner algorithm, the search engine will be used to find a solution which minimize the cost function and, therefore, find a trajectory that best satisfies all the qualities represented by this cost function. Our cost function represents a specific scenario where the optimal path minimizes the distance travelled, the average altitude (to increase the stealthiness of the UAV) and avoids danger zones, while respecting the UAV performance characteristics. This cost function is highly complex and demonstrates the power of our path planning algorithm. However, this cost function could easily be modified and applied to a different scenario.

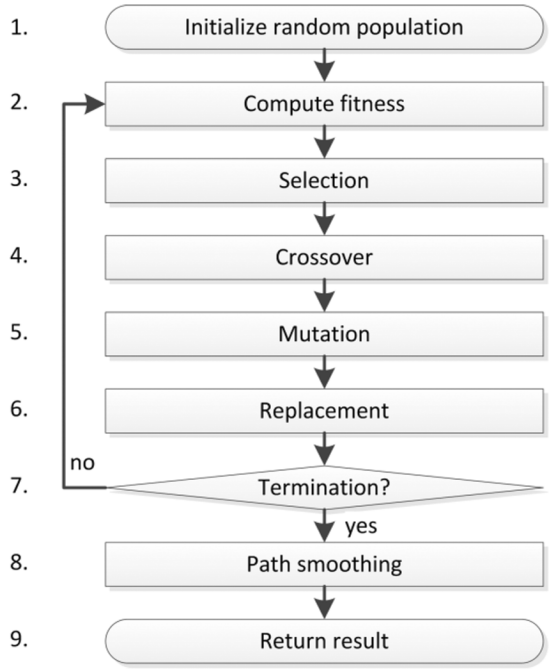


Fig. 3. Flowchart of the genetic algorithm.

IV. Genetic Algorithm (GA)

The GA is a population based nondeterministic optimization method that was developed by John Holland in the 1960s and first published in 1975 [15]. Based on the genetic theory of Darwin evolution, the GA simulates the evolution of a population of solutions to optimize a problem. Similarly to living organisms adapting to their environment over the generations, the solutions in the GA adapt to a fitness function over an iterative process using biology-like operators such as the crossovers of chromosomes, the mutations of genes and the inversions of genes. In recent years, the GA has been used for a wide range of applications such as the design of minimal phase digital filters [16] and the optimization of surface grinding process [17]. In this work, we use the GA to simulate the evolution of a population of trajectories adapting to the cost function defined in the previous section.

In our implementation, the initial trajectories are randomly generated within the 3D search space with no consideration of their feasibility. The fitness of each solution is then evaluated and the mating pool selected favoring fit parents using the stochastic universal sampling method presented in [18]. Children solutions are created from the selected parents using single point crossover also based on [18]. The children solutions are then subject to the following mutations: addition of a new waypoint, deletion of an existing waypoint and modification of an existing waypoint. Finally, the parent generation is replaced by the children generation and the evolution cycle continues until the termination criterion has been met. Our implementation allows the user to define the termination criterion as a maximum number of generations to be simulated (as used in Section VII) or a fixed execution time (as used in Section VIII).

We implemented the addition operator by randomly selecting one of the line segments of the trajectory and adding a random waypoint within the 3D neighborhood of its midpoint. Similarly, we implemented the modification operator by randomly selecting one of the waypoints of the trajectory and moving it to a random location within its 3D neighborhood. The size of this neighborhood is initially set based on the size of the search space and is linearly reduced throughout the evolution process improving the exploration at the beginning of the execution and the refinement towards the end. At every generation, each children solution is subject to a mutation based on the mutation rate parameter. The type of mutation is randomly selected between the three we implemented and the mutation affects the solution in a single point. Our implementation also uses the concept of elitism when replacing the old generation with the new one in order to improve convergence. The flowchart of the GA is shown in Fig. 3 and the different genetic operators used, in Fig. 4. The parameter values used in our implementation of the GA are listed in Table I. To reflect the good design of our GA and cost function, we plot in Fig. 5 the cost of the solutions generated by the GA against iterations. At iteration 1, the best solution has a cost of 9.17 and therefore violates a few feasibility constraints. The optimization process rapidly converges and finds a first feasible solution at iteration 30. The refinement of the solutions continues until the maximum number of iterations is attained. The cost of the best solution at the last iteration is 0.342.

V. Particle Swarm Optimization (PSO)

The PSO is a population-based nondeterministic optimization method that was proposed by Kennedy and Eberhart in 1995 [19]. Similarly to the GA, the PSO has recently been used in a wide range of applications such as the design and optimization of infinite impulse response digital filters [20]. The algorithm

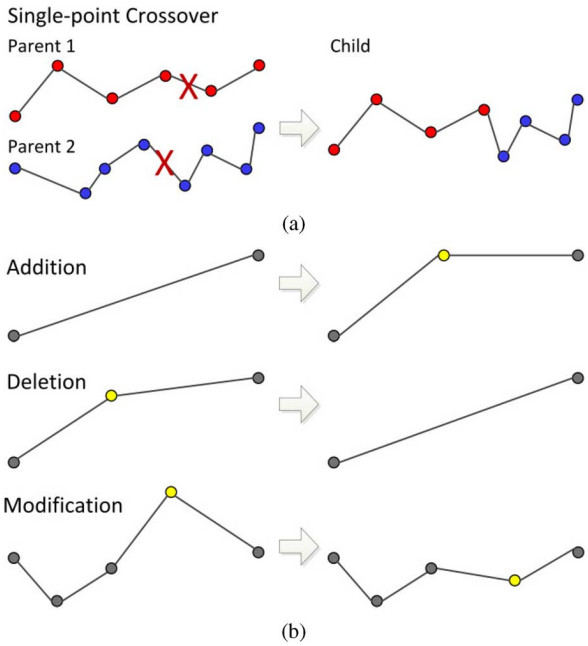


Fig. 4. Genetic operators used in our GA: (a) single-point crossover and (b) mutations: addition, deletion, modification.

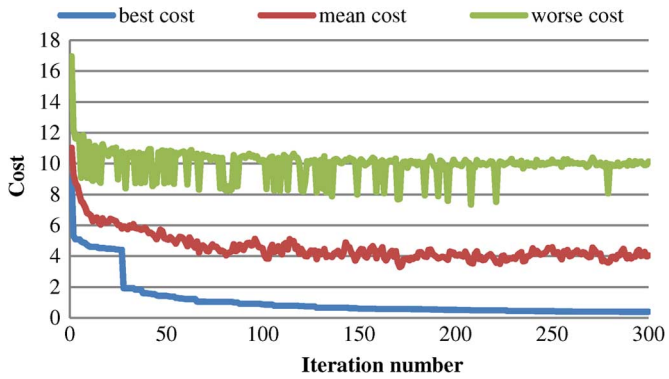


Fig. 5. Cost of the solutions generated by our GA at each iteration (256 chromosomes, 300 iterations, for the scenario shown in Fig. 1).

simulates the movement of a swarm of particles in a multidimensional search space progressing towards an optimal solution. The position of each particle represents a candidate solution and is randomly initiated. In our implementation, the position of a particle represents a complete trajectory using the same encoding as with the GA. At every step of the iterative process, the velocity of each particle is individually updated based on the previous velocity of the particle, the best position ever occupied by the particle (personal influence) and the best position ever occupied by any particle of the swarm (social influence). In line with the original PSO [19], our implementation does not allow any mutation type operations. As outlined in [21], the equations used to compute the velocity and position of a single particle at iteration t are as follows:

$$\boldsymbol{v}_{t+1} = \omega \boldsymbol{v}_t + c_1 \boldsymbol{r}_1 \cdot *(\boldsymbol{b}_t - \boldsymbol{x}_t) + c_2 \boldsymbol{r}_2 \cdot *(\boldsymbol{g}_t - \boldsymbol{x}_t) \quad (18)$$

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{v}_{t+1} \quad (19)$$

where variables in bold are vectors $\boldsymbol{v}$ is the velocity of the particle; $\boldsymbol{x}$ is its position; $\boldsymbol{b}$ is the best position previously occupied by the particle; $\boldsymbol{g}$ is the best position previously occupied by any particle of the swarm; $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are vectors of random values between 0 and 1; and ω, c_1 and c_2 are the inertia, the personal influence and the social influence parameters. When searching for an 8-waypoint trajectory, vectors in (18) and (19) have 24 elements and the particles are moving in a 24 D search space. Every element of the particles' velocity $\boldsymbol{v}$ and position $\boldsymbol{x}$ are updated independently as per the two equations. Still based on [21], the flow diagram of our implementation of the PSO is displayed in Fig. 6. The parameter values used in our implementation of the

TABLE I
ALGORITHM PARAMETER VALUES

Parameters	Values
Terrain resolution*	500 x 500
Number of waypoints per trajectories**	8
Number of generations (GA) or iterations (PSO)	100, 200 and 300
Number of chromo. (GA) or particles (PSO)	128 and 256
Number of migrations between sub-populations (outer-loop)	10
Crossover rate (GA only)	80 %
Mutation rate (GA only)	10 %
Elitism rate (GA only)	1 %
ω (PSO only)	0.7298
c_1 (PSO only)	1.4960
c_2 (PSO only)	1.4960

* Size of the 2D matrix representing the terrain as shown in Figure 2.

** In the case of the GA, the candidate solutions are initiated with 8 waypoints, but this number will vary for each solution throughout the evolution process due to the genetic operators. In the case of the PSO, the number of waypoints remains 8 throughout the optimization process.

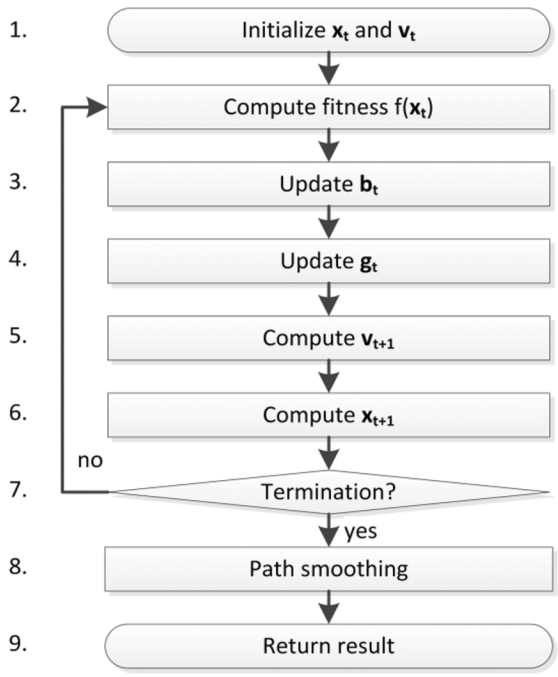


Fig. 6. Flowchart of the particle swarm optimization.

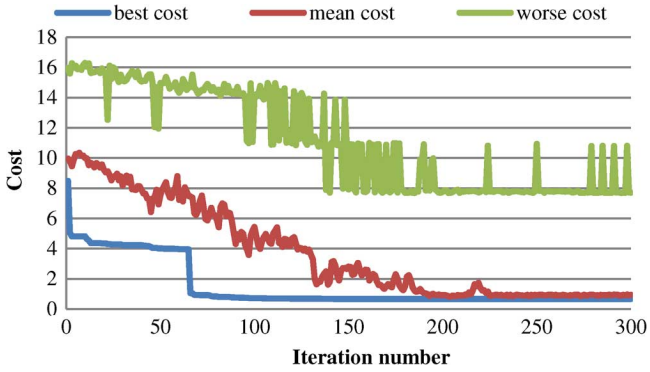


Fig. 7. Cost of the solutions generated by our PSO at each iteration (256 particles, 300 iterations, for the scenario shown in Fig. 1).

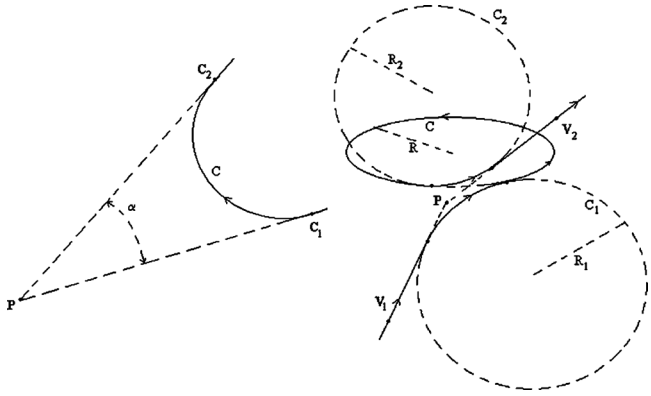


Fig. 8. Circular constructions used to smooth the final trajectory and remove all discontinuity in the velocity (left diagram shows a simple circular arc and the right diagram shows a connection using two circular arcs and a circle on an horizontal plane) (reproduced with permission from [22]).

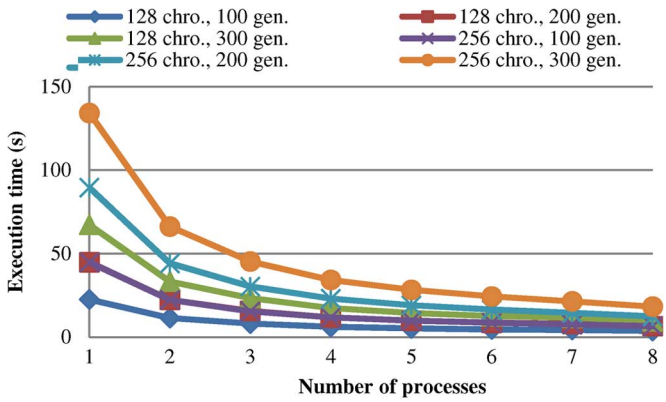


Fig. 9. Execution time of our parallel GA for different work sizes.

PSO are listed in Table I. Our implementation of the PSO allows the user to define the termination criterion as a maximum number of iterations to be simulated (as used in Section VII) or a fixed execution time (as used in Section VIII). As we previously did for the GA, we plot in Fig. 7 the cost of the solutions generated by the PSO against iterations. Compared to the GA, the PSO took more iterations before finding the first feasible solution. We also note that the refinement of the best solution is less visible than with the GA. This may be caused by a local minimum. Although the cost versus iteration graphs give clear

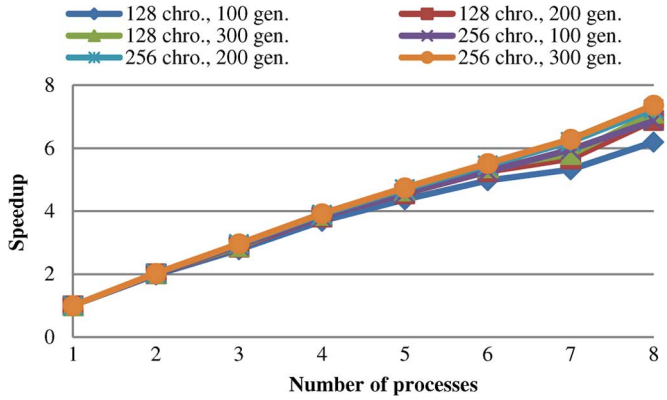


Fig. 10. Speedup achieved by our parallel GA for different work sizes.

TABLE II
COMPUTER SYSTEM USED FOR THE EXPERIMENTAL TEST

Computer	Dell PowerEdge 2900
CPU	2x Intel Xeon E5310, quad-core, 1.6 GHz
RAM	8 GB, DDR2, 667 MHz
OS	Windows 7 Enterprise, 64 bits
IDE	MATLAB 7.11.0.584 (2010 b) 64 bits Parallel Computing Toolbox v5.0

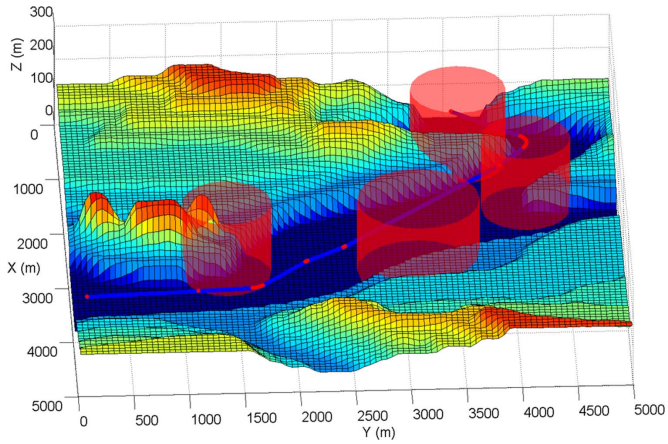


Fig. 11. 3D visualization of the computed path [fictitious map, 25 km^2, altitude ranging from 0 to 250 m above mean sea level (AMSL)].

insight of the good design of our GA and PSO, it is not sufficient to compare the two algorithms. A rigorous comparison in presented in Section VIII of this document.

VI. PATH SMOOTHING

Consistent with solutions proposed in the literature [22]–[24], our solution generates a path composed of line segments. This would be sufficient for multidirectional ground robots, but inadequate for a fixed-wing UAV. To remove all discontinuities in the velocity, we smooth the final path by connecting the line segments with simple circular arcs (when the power available is sufficient) or circles on a horizontal plateau (when the power available is not sufficient to fly a simple circular arc) based on [22], [24], [25], and as shown in Fig. 8. Our final stage module

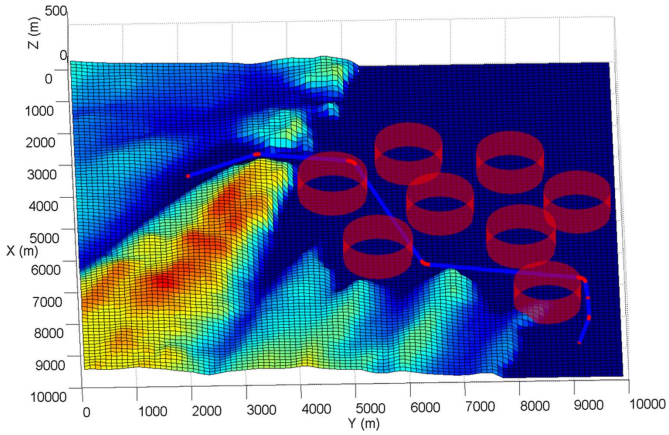


Fig. 12. 3D visualization of the computed path (St-John, NL, Canada, area of 100 km^2, altitude ranging from 0 to 246 m AMSL).

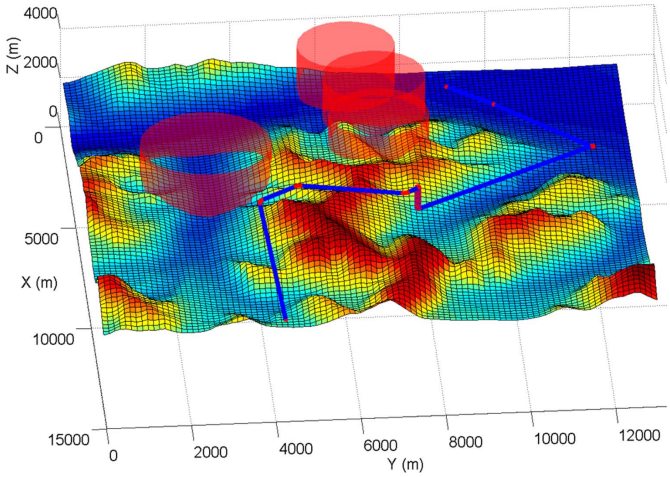


Fig. 13. 3D visualization of the computed path (Lake Louise, AB, Canada, area of 192 km^2, altitude ranging from 1 515 to 3 491 m AMSL).

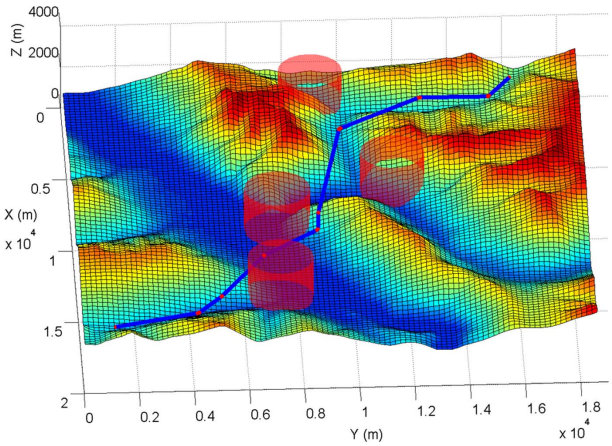


Fig. 14. 3D visualization of the computed path (Kinbasket lake, BC, Canada, area of 348 km^2, altitude ranging from 750 to 2 746 m AMSL).

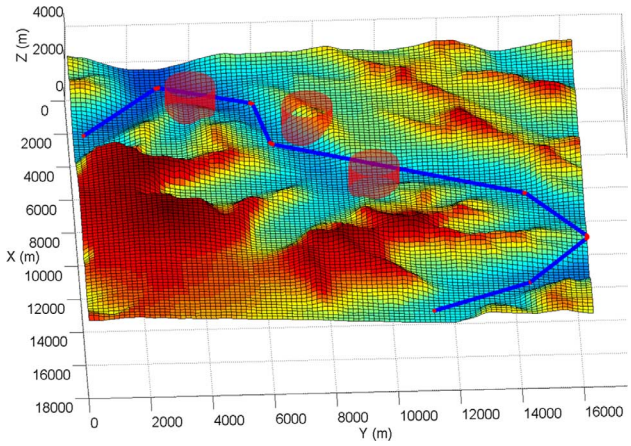


Fig. 15. 3D visualization of the computed path (Columbia Icefield, AB, Canada, area of 269 km^2, altitude ranging from 1 621 to 3 486 m AMSL).

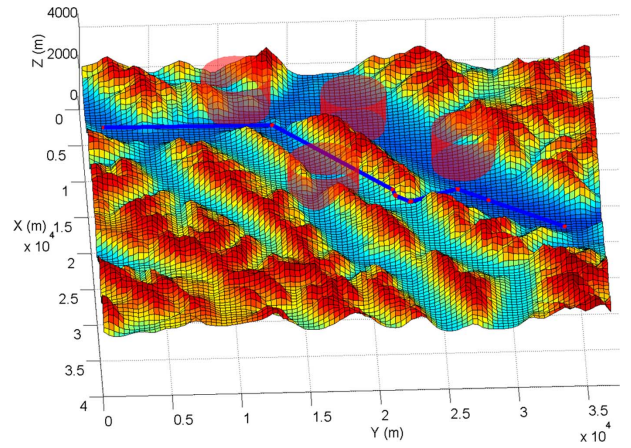


Fig. 16. 3D visualization of the computed path (Banff, AB, Canada, area of 1 360 km^2, altitude ranging from 1290 to 3079 m AMSL).

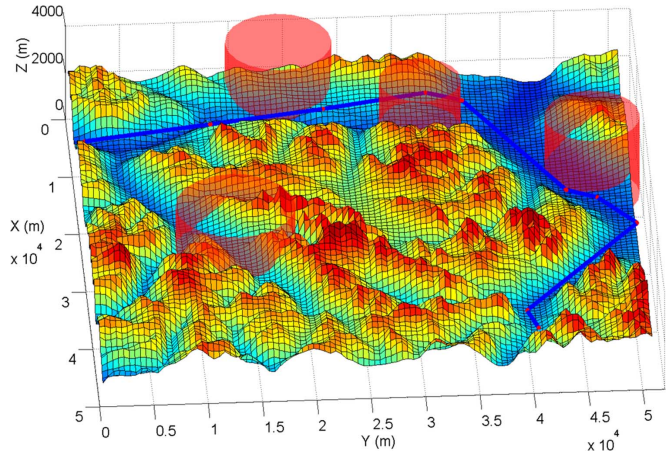


Fig. 17. 3D visualization of the computed path (Jasper, AB, CA, area of 3 717 km^2, altitude ranging from 1020 to 3308 m AMSL).

also replaces any line segment requiring more power than available with a vertical helix as in [22]. Although smoothing of the path is performed after the optimization process, the feasibility of this operation is verified in our cost function to ensure the smoothing of the final trajectory is always possible.

VII. Parallel Implementation

We have now discussed all the elements required to build a complete path planning module for UAVs. Although the generated trajectories are feasible and nearly optimal, the computation time remains too long for real-time applications. To address this problem, we developed parallel versions of

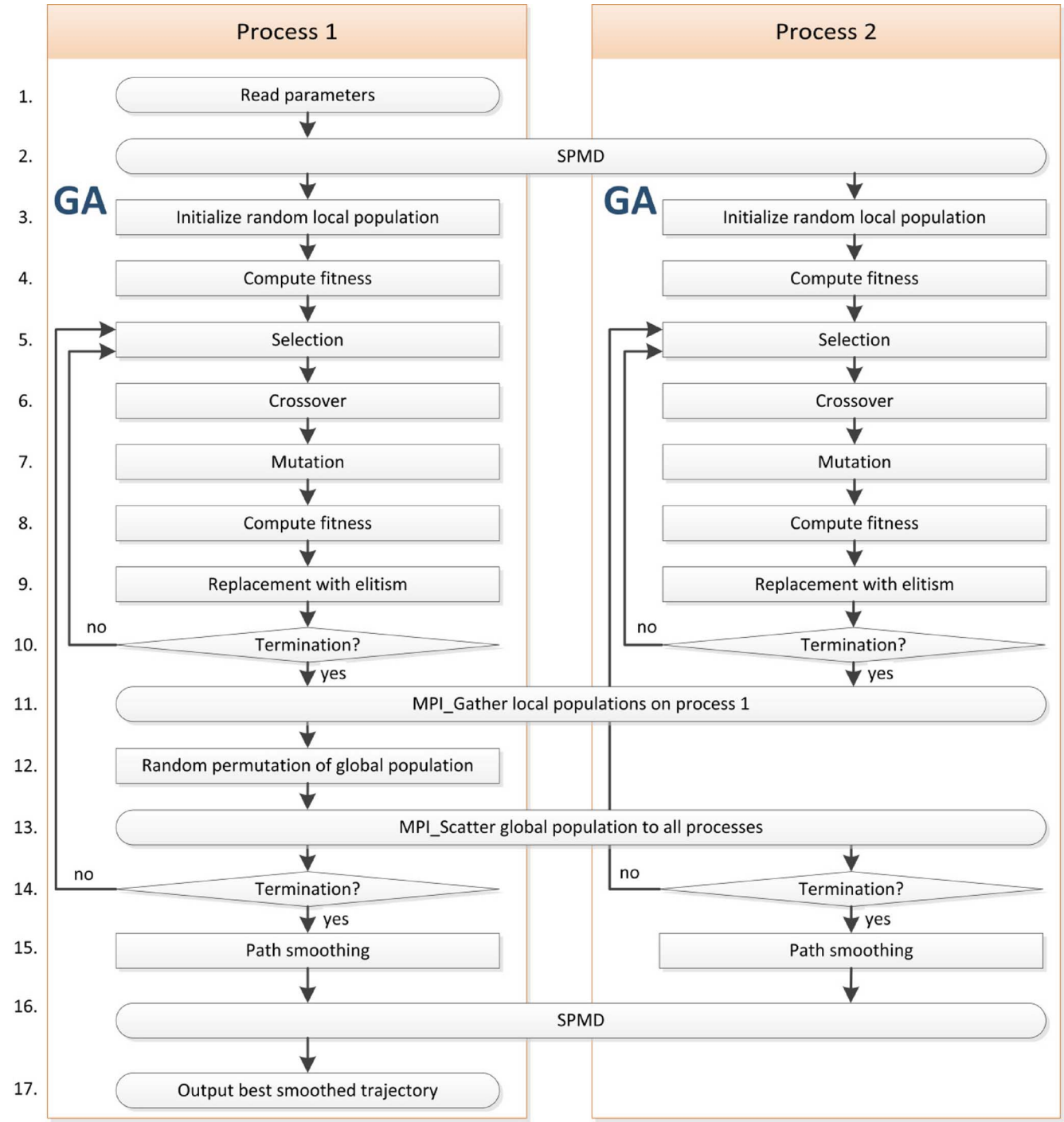


Fig. 18. Flowchart of our parallel genetic algorithm.

our GA and PSO using the "Single-program, multiple-data" parallel programming paradigm. Our implementation was done in MATLAB™. It can be classified as a multiple-deme parallel GA where sub-populations evolve independently while allowing some level of migration between the demes [26]. In our implementation, the migrations occur synchronously, ten times throughout the optimization process and involve all the solutions. Our approach maintains a good level of collaboration between the subpopulation, minimizes the communication between the processes, offer a better speedup than a master-slave implementation [26] and allows full use of today's multicore CPUs. The flowchart of our parallel GA is shown in Fig. 18. Although drawn for two processes, our implementation allows any number of processes. Our parallel version of the PSO is not shown in this paper but follows the same principle. The execution time and the speedup of our parallel GA were measured using the parameters in Table I, on the system described in Table II and are plotted in Figs. 9 and 10. The execution time and the speedup of our parallel PSO are not presented here, but almost identical.

Although we achieved a computation time of 3.63 s for 128 chromosomes and 100 generations, we arbitrarily configured

TABLE III
AVERAGE COST OF THE 60 TRAJECTORIES GENERATED FOR EACH OF THE 40 SCENARIOS USING OUR GA-BASED AND PSO-BASED PATH PLANNING ALGORITHM (SHOWING RESULTS FOR THE FIRST 20 SCENARIOS DUE TO SPACE LIMITATION)

Terrain	Scenario	Average cost and standard deviation of generated trajectories (60 trials)		p-value (test-T)	Averages are statistically different	Winner	
		GA	PSO			GA	PSO
1	1	**0.103 ± 0.003**	0.119 ± 0.045	0.011	yes	X	
	2	**0.353 ± 0.020**	0.368 ± 0.058	0.064	no		
	3	**0.519 ± 0.022**	0.988 ± 1.575	0.025	yes	X	
	4	**0.512 ± 0.029**	0.503 ± 0.044	0.184	no		
	5	**0.559 ± 0.040**	0.718 ± 0.393	0.003	yes	X	
2	6	0.342 ± 0.043	**0.253 ± 0.020**	0.000	yes		X
	7	0.414 ± 0.028	**0.357 ± 0.047**	0.000	yes		X
	8	**0.550 ± 0.059**	0.597 ± 0.149	0.027	yes	X	
	9	**0.543 ± 0.036**	0.571 ± 0.148	0.157	no		
	10	**0.612 ± 0.031**	0.917 ± 0.258	0.000	yes	X	
3	11	**0.200 ± 0.011**	0.740 ± 2.453	0.093	no		
	12	**0.327 ± 0.143**	0.529 ± 0.139	0.000	yes	X	
	13	**0.733 ± 0.098**	0.815 ± 0.158	0.001	yes	X	
	14	0.460 ± 0.035	**0.437 ± 0.060**	0.014	yes		X
	15	**0.595 ± 0.056**	0.772 ± 0.159	0.000	yes	X	
4	16	**0.731 ± 0.030**	1.442 ± 1.791	0.003	yes	X	
	17	**1.029 ± 0.902**	5.745 ± 2.034	0.000	yes	X	
	18	**1.292 ± 0.014**	3.049 ± 2.405	0.000	yes	X	
	19	**1.734 ± 0.007**	3.119 ± 2.283	0.000	yes	X	
	20	**4.156 ± 2.301**	8.670 ± 5.905	0.000	yes	X	
				TOTAL	**28/40**	**25/40**	**3/40**

our path planning module for a fixed execution time of 10 s. This allows the simulation to continue for few hundreds of generations (318 for the GA and 213 for the PSO) and generate very high-quality trajectories in a wide range of complex 3D environments. Ten seconds may seem too long for a real-time application, but easily allows the UAV to compute the next path, while flying the current one. As an example, the Global Hawk (one of the fastest UAVs currently in use) has a cruising speed of 635 km/h and travels 1 764 m in 10 s [13]. Our path planning module can easily generate trajectories longer than this distance and therefore allows computation while in flight.

VIII. COMPARISON OF THE GA AND THE PSO

Finally, we compare the performance of the GA and the PSO using 40 different scenarios from two fictitious terrain elevation maps and six real terrain elevation maps from the Canadian coast, the Canadian Rockies and the Canadian ice fields. The digital elevation maps for the six real terrains were taken from the GeoBase repository [11]. Each scenario has a unique start and finish point and a varied number of danger zones. Six of the 40 scenarios are shown in Figs. 11–17. These scenarios are representative of the complexity and realism of all 40 scenarios used. The average costs of 60 trajectories generated using our parallel GA and parallel PSO for a fixed computation time of 10 s and population size of 256 are listed in Table III. Each run used a different initial random population of solutions. To compare our implementations of the GA and the PSO, we used the T-test [27] on the cost distributions of the trajectories obtained by each algorithm. In Table III, the p-value obtained by the T-test represents the probability that the difference between the two averages is due to chance. In other words, when the p-value obtained is lower than 0.05, the two averages are significantly different with a 95% confidence. In that case, we can compare the two averages and identify which algorithm performed the best. Base on the results presented in Table III, we conclude that:

1) the GA produced trajectories significantly better than those generated by the PSO for 25 of the 40 scenarios;
2) the PSO produced trajectories significantly better than those generated by the GA for 3 of the 40 scenarios;
3) the GA and the PSO produced trajectory of similar quality for 12 of the 40 scenarios.

Based on these results, we conclude that the GA is preferable to the PSO when solving the path planning problem for UAVs in a fixed computation time of 10 s on multicore commercial off-the-shelf processors.

IX. CONCLUSION

This paper presents a path planning solution for UAVs which considers the dynamic properties of the UAV and the complexity of a real 3D environment. We used two nondeterministic algorithms, the GA and the PSO, to attack this complexity and produce solutions in a relatively short computation time. We further reduced the execution time by developing parallel versions of our algorithms. After achieving a quasi-linear speedup of 7.3 on 8 cores and an execution time of 10 s for both algorithms, we conclude that by using a parallel implementation on standard multicore CPUs, real-time path planning for UAVs is possible. Moreover, our rigorous comparison of the two algorithms shows, with statistical significance, that our implementation of the GA produces superior trajectories than our implementation of the PSO when using the same encoding.

REFERENCES

[1] H. Chen, X.-M. Wang, and Y. Li, "A survey of autonomous control for UAV," in *Proc. 2009 Int. Conf. Artif. Intell. Comput. Intell.*, 2009, vol. 2, pp. 267–271.
[2] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning—A chronological review," *World Academy of Science*, vol. 29, pp. 101–106, 2007.
[3] P. Y. Volkan, "A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV," *Aerosp. Sci. Technol.*, vol. 16, no. 1, pp. 47–55, Jan. 2012.
[4] D. G. Macharet, A. A. Neto, and M. F. M. Campos, "Feasible UAV path planning using genetic algorithms and Bezier curves," in *Proc. SBIA'10*, Berlin, Germany, 2010, pp. 223–232.
[5] Z. Cheng, Y. Sun, and Y. Liu, "Path planning based on immune genetic algorithm for UAV," in *Proc. Int. Conf. Electric Inform. Control Eng.*, 2011, pp. 590–593.
[6] Y. Bao, X. Fu, and X. Gao, "Path planning for reconnaissance UAV based on particle swarm optimization," in *Proc. 2nd Int. Conf. Comput. Intell.Natural Comput., CINC'10*, Wuhan, China, 2010, vol. 2, pp. 28–32.
[7] Y. Fu, M. Ding, and C. Zhou, "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV," *IEEE Trans. Syst., Man, Cybern.*, vol. 42, no. 2, pp. 511–526, 2012.
[8] D. M. Pierre, N. Zakaria, and A. J. Pal, "Master-slave parallel vector-evaluated genetic algorithm for unmanned aerial vehicle's path planning," in *Proc. Int. Conf. Hybrid Intell. Syst.*, 2011, pp. 517–521.
[9] G. Wang, Q. Li, and L. Guo, "Multiple UAVs routes planning based on particle swarm optimization algorithm," in *Proc. 2nd Int. Symp. Inform. Eng. Electronic Commerce*, 2010, pp. 1–5.
[10] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in *Proc. 4th Student Conf. Res. Develop.*, Piscataway, NJ, USA, 2006, pp. 183–188.
[11] GeoBase—Canadian Digital Elevation Data Accessed: Apr. 3, 2012. [Online]. Available: http://www.geobase.ca/geobase/en/index.html

[12] G. Labonté, "Mathematical relations to analyze aircraft performance for trajectory planning" Defense Research and Development, Ottawa, ON, Canada, Contract Rep. CR 2010-249, Dec. 2010.
[13] J. D. J. Anderson, *Introduction to Flight*, 6th ed. New York: McGraw-Hill, 2008.
[14] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.
[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1975.
[16] A. Slowik, "Application of evolutionary algorithm to design minimal phase digital filters with non-standard amplitude characteristics and finite bit word length," *Bull. Polish Acad. Sci.: Tech. Sci.*, vol. 59, no. 2, pp. 125–135, 2011.
[17] A. Slowik and J. Slowik, "Multi-objective optimization of surface grinding process with the use of evolutionary algorithm with remembered Pareto set," *Int. J. Adv. Manuf. Technol.*, vol. 37, no. 7–8, pp. 657–669, 2008.
[18] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. London, U.K.: Springer, 2010.
[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, Perth, WA, Australia, 1995, vol. 4, pp. 1942–1948.
[20] A. Slowik and M. Bialko, "Design and optimization of ir digital filters with non-standard characteristics using particle swarm optimization algorithm," in *Proc. ICECS'07*, Piscataway, NJ, 2007, pp. 162–165.
[21] M. Clerc, *Particle Swarm Optimization*. Paris, France: Lavoisier, 2005.
[22] G. Labonté, "On the construction of dynamically feasible aircraft trajectories using series of line segments," *Comput. Intell. Res. Lab, RMC*, Nov. 2009. [Online]. Available: http://cirl.dyndns.org/en/downloads/internal-publications/58-g
[23] I. Hasircioglu, H. R. Topcuoglu, and M. Ermis, "3-D path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms," in *Proc. 10th Annu. Conf. Genetic Evol. Comput.*, Atlanta, GA, 2008, pp. 1499–1506.
[24] E. P. Anderson, R. W. Beard, and T. W. McLain, "Real-time dynamic trajectory smoothing for unmanned air vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 3, pp. 471–477, May 2005.
[25] C. L. Bottasso, D. Leonello, and B. Savini, "Path planning for autonomous vehicles by trajectory smoothing using motion primitives," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 6, pp. 1152–1168, 2008.
[26] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. New York: Springer, 2008.
[27] MATLAB Two-Sample T-Test Accessed Apr. 3, 2012. [Online]. Available: http://www.mathworks.com/help/toolbox/stats/ttest2.html

Vincent Roberge received the B.Eng. and M.A.Sc. degrees from the Royal Military College of Canada, Kingston, ON, Canada, in 2005 and 2010, respectively. Currently, he is working towards the Ph.D. degree at the Department of Electrical and Computer Engineering, the Royal Military College of Canada.

In July 2011, he joined the Department of Electrical and Computer Engineering, Royal Military College of Canada, where he is currently Lecturer . His current research interests include parallel computation, optimization and real-time applications.

Mohammed Tarbouchi received the M.Sc. and Ph.D. degrees from Laval University, Laval, QC, Canada, in 1993 and 1997, respectively.

In September 1997, he joined the Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON, where he is currently an Associate Professor. His current research interests include analysis and design of electrical machines, variable speed drives and fault diagnosis of electric machines.

Gilles Labonté received the B.Sc. and M.Sc. degrees in physics from the Université de Montréal, Montreal, QC, Canada, and the Ph.D. degree in theoretical physics from the University of Alberta, Edmonton, AB, Canada.

He is presently a Professor with the Department of Mathematics and Computer Science and the Department of Electrical and Computer Engineering, Royal Military College of Canada. He has contributed to relativistic quantum mechanics, symbolic algebraic computations, artificial neural networks and soft-computing. He presently works on automatic target recognition, intelligent control systems for unmanned vehicles, and applications of the theory of complex systems.