Actas do Encontro Científico
3º Festival Nacional de Robótica - ROBOTICA2003
Lisboa, 9 de Maio de 2003.

PATH PLANNING OPTIMIZATION USING THE DIFFERENTIAL EVOLUTION ALGORITHM

Hélder Santos, José Mendes, P. B. de Moura Oliveira and J. Boaventura Cunha

Universidade de Trás-os-Montes e Alto Douro
Departamento de Engenharias
Quinta dos Prados, 5000 Vila Real – Portugal
e-mail: opt_trajectorias@mail.pt

Abstract: In this paper the Differential Evolution algorithm is deployed to the robotic path planning optimization problem for autonomous mobile vehicles. At this stage, the simulations consider the robot world with level surfaces and static obstacles, in which the optimization is performed off-line. The global objective is to evaluate the differential evolutionary algorithm in solving path planning problems, by finding the shortest path between start and goal points. Results obtained for three case studies with static circular and polygonal obstacles are presented.

Keywords: Evolutionary Algorithms, Differential Evolution, Optimization, Path Planning, Obstacles Avoidance, Vehicle Routing.

1. INTRODUCTION

One of the robotics main research field is the control algorithms development which allows robot mobility in a environment surrounded by obstacles. In this context, the path planning problem plays an important role by providing trajectories to avoid obstacles. There are several well known methods of path planning. Its choice depends on the complexity and quality of the world where the vehicle will move, as well as the vehicle's qualities. There are some criteria to evaluate the planning, such as path length, time spent, obstacle distance or the complexity and smoothness of the path.

The relevance of applying efficient optimization algorithms to solve trajectory optimization problems is unquestionable. This is the case of evolutionary inspired algorithms, particularly the popular genetic algorithms (Goldberg, 1989; Michalewicz, 1992). One more recently introduced algorithm is the Differential Evolution proposed by Price and Storn (1995) which provides good results in the context of function optimization (Storn, 1996).

This paper describes the off-line optimization of mobile robots trajectories using the Differential Evolution Algorithm, considering a world with level surfaces and obstacles represented by round and polygonal shapes. The path is formed by strait line segments. The overall objective of the expected work is to acess the Differential Evolution algorithm in the context of robotic path planning optimization using evolutionary inspired techniques. The optimization objective is to achieve the shortest path, between two points, avoiding any obstacle in the way.

The rest of this paper is organized as follows in section two an introduction to the Differential Algorithm is provided. Section three describes the off-line path planning problem considered. In section four simulation results are presented and finally in section five some conclusions are drawn.

2. DIFFERENTIAL EVOLUTION

The Differential Evolution (DE) algorithm was originally developed by Price and Storn (1995). The solution of this algorithm, to address the path planning problem, was based upon its efficiency, effectiveness and robustness in the optimization of functions (Storn, 1996). The DE can be easily implemented to solve several types of optimization problems, such as path planning.

Assuming that a vector $\vec{x}$ represented by (1) represents a candidate solution to the path optimization problem, in which t represents the current evolutionary iteration, a new solution vector $\vec{x}(t+1)$ can be evaluated from the previous vector by using equation (2), with $\vec{d}$ (3), representing an incremental vector.

$$\vec{x}(t) = x_1(t), x_2(t), \ldots, x_n(t) \tag{1}$$

$$\vec{x}(t+1) = \vec{x}(t) + \vec{d}\,(t) \tag{2}$$

$$\vec{d}\,(t) = d_1(t), d_2(t), \ldots, d_n(t) \tag{3}$$

Considering a set *pop* of potential solutions with size *m*, represented by (4), known as population (2) can be rewritten as (5) with *d* representing the dimension index.

$$pop_i(t) = \left(x_{i1}(t), x_{i2}(t), \ldots, x_{in}(t)\right) \quad 1 \le i \le m \tag{4}$$

$$x_{id}(t+1) = x_{id}(t) + d_{id}(t+1) \quad 1 \le i \le m \quad 1 \le d \le n \tag{5}$$

A trial $\vec{x}_v \in R^n$ vector is generated in each iteration for each population member using (6) and the increment d is evaluated using (7). In these expressions $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ and $\vec{x}_{r_3}$ are vectors selected randomly from the population set *pop*, in each iteration, and $F \in R^+$ is a constant defined prior to optimization.

$$\begin{gathered} x_{vid}(t+1) = x_{r_1 d}(t) + d_{id}(t+1) \\ 1 \le i \le m \quad 1 \le d \le n \quad r_1 \in [1,m] \ne i \end{gathered} \tag{6}$$

$$\begin{gathered} d_{id}(t+1) = F\left(x_{r_2 d}(t) - x_{r_3 d}(t)\right) \\ 1 \le i \le m \quad 1 \le d \le n \quad r_2 \in [1,m] \ne i \ne r_1 \ne r_3 \\ r_3 \in [1,m] \ne i \ne r_1 \ne r_2 \end{gathered} \tag{7}$$

The trial vector obtained with (6) and (7) is then crossed with the current population element $\vec{x}_i$ accordingly to a crossover scheme, binomial or exponential (Storn and Price, 1995), with a pre-defined probability defined by $CR \in [0,1]$ resulting in a new vector $\vec{x}_{ci}$ (8). If the value returned by the objective function f for the crossed trial vector is better or equal than the value obtained for the current vector, the latest is replaced by the former. This is represented by (9) for a minimization problem.

$$\vec{x}_{ci}(t+1) = crossover\left(\vec{x}_i(t), \vec{x}_{vi}(t+1)\right) \quad 1 \le i \le m \tag{8}$$

$$\begin{gathered} \vec{x}_i(t+1) = \vec{x}_{ci}(t+1) \quad if \quad f\left(\vec{x}_{ci}(t+1)\right) < f\left(\vec{x}_i(t+1)\right) \\ 1 \le i \le m \end{gathered} \tag{9}$$

In Figure 1 a pseudo-code is shown that can be used to implement the DE algorithm, in which comments are included between brackets.

```
[Initialization of the population vector]
Initialize pop_new;        [matrix with random values]
Initialize best;           [best vector]
Initialize best_value;     [best value]
From j = 0 until maximum number of iterations:
  [Copies the initialized matrix to a general one]
  pop = pop_new;
  From i = 0 until m:
    [Chooses randomly three numbers between 0 and
  m]
    r1 = a vector index different from i;
    r2 = a vector index different from i and r1;
    r3 = a vector index different from i, r1 and r2;
    [Saves the vector i ]
    temp = pop( i, :);
    [The following equation changes with the strategy]
      xv = pop( r1) + F*( pop( r2) - pop( r3) );
    [Evaluate the new temp vector]
    xc=crossover(xv,xi)
    Value_new = evaluation(xc);
    Value_old = evaluation(pop( i )) ;
    If (Value_new < Value_old) do:
      [It means that the new vector is better]
      pop_new( i, :) = xc;
      If (value_new < value_best) do:
        [Best value ever]
        value_best = value_new;
        best = temp;
    else
      [Keeps the old value]
      pop_new( i,:) = pop_old( i, :);
  i = i + 1;
  end
j = j + 1;
end
```

Fig. 1. A Pseudo-code for the DE algorithm.

The different strategies proposed for this algorithm are, basically, very similar, differing in the evaluation of the $\vec{x}_v \in R^n$ vector. Price and Storn (1995) proposed ten different options that can be classified using the notation DE/X/Y/Z, where:

- X: represents the methodology used to select vector r1, which can be random selected *(rand)* or the best population so far (*best*).

- Y: the number of vectors used to evaluate the differential perturbation:

- Z: Is the type of crossover operation used which can either be of the exponential (*exp*) type or binary (*bin*) type. Thus, ten DE algorithm variations can be summarized by: 1) *DE/best/1/exp*; 2) *DE/rand/1/exp*; 3) *DE/rand-to-best/1/exp*; 4) *DE/best/2/exp*; 5) *DE/rand/2/exp*; 6) *DE/best/1/bin*; 7) *DE/rand/1/bin*; 8) *DE/rand-to-best/1/bin*; 9) *DE/best/2/bin* and 10)*DE/rand/2/bin*, and are well described in (Price, 2002).

Figure 2 illustrates the DE basic concept (Lampinem J. and Zelinka, 1999), and it is important to note that this algorithm does not use any type of coding scheme operating directly in the optimizing parameters.

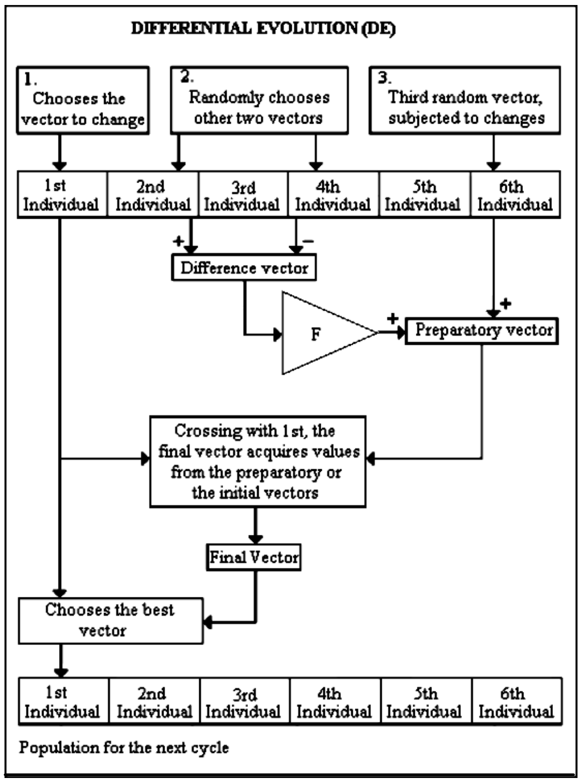
DIFFERENTIAL EVOLUTION (DE)
1. Chooses the vector to change
2. Randomly chooses other two vectors
3. Third random vector, subjected to changes
1st Individual
2nd Individual
3rd Individual
4th Individual
5th Individual
6th Individual
+
−
Difference vector
F
+
+
Preparatory vector
Crossing with 1st, the final vector acquires values from the preparatory or the initial vectors
Final Vector
Chooses the best vector
Population for the next cycle

Fig. 2. Global scheme for Differential Evolution.

3. OFF-LINE PATH PLANNING

In the off-line case, the simulated world is defined using static obstacles, and the optimization objective is to establish a full path, avoiding collisions. Thus, the DE algorithm is used as the optimization tool, and its performance evaluated. The robot world is defined by a square in the Cartesian plan, limited by the coordinates of its vertices. The robot position is represented by its coordinates corresponding to one point in the plane. The path between the start and goal positions is defined by line segments which link all the different points defined by the differential evolution.

In order to use the DE algorithm some parameters must be defined prior to the search procedure, such as:

- the number of optimization points *(d)*. The number of segments equals the number of points plus one. *(Segments=d+1)*. Thus, path flexibility depends on the number of points,

- initial and final positions for the robot *(Start and Goal)*,

- maximum number of iterations,

- DE strategy,

- population size *(m)*.

- amplification factor *F*. This parameter is multiplied by the difference between two vectors. It should be selected in order to prevent premature convergence ($0<F<2$);

- crossover probability (*CR*). Using a higher *CR* will increase the convergence speed. Values between *0* and *1* are usually used.

At this stage, it is important to underline the type of objective function used, which for every potential solution verifies if the corresponding trajectory intersects obstacles. It returns the length of the path connecting the starting and goal points. The global objective is to find the path which requires the minimum number of points and provides minimum length.

The defined static obstacles are defined using circular and polygonal shapes. For the circular case (Figure 3), the obstacles are defined by their circle centre and radius. Using this, a simple way to avoid collisions, consists on determining the minimum distance between each segment of the path and the centre of each obstacle. If this distance remains greater than the radius of each obstacle, there are no collisions.

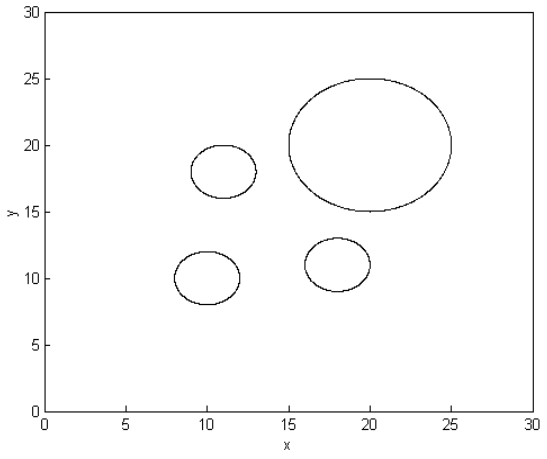

Fig. 3. Example of the defined world, with circular obstacles.

The distance between a point and a line, is given by the perpendicular segment that crosses the point. However, it is necessary to determine if the perpendicular line intersects the segment, otherwise the minimum distance is evaluated in relation to segment nearest extreme point. Assuming that the centre of the circular obstacle is represented by *P* and the line segment is represented by P_0P_1, an easy way to solve this problem is to consider the internal product of the angles between vectors P_0P and P_1P (Sunday, 2001a). If the result of this product is negative or positive, it means that the shortest distance between the central point and the segment is connected to one of the extreme points of the latest. If the product result is zero it means that the perpendicular line between the circle central point intersects the segment and it is the shortest distance (Figure 4).

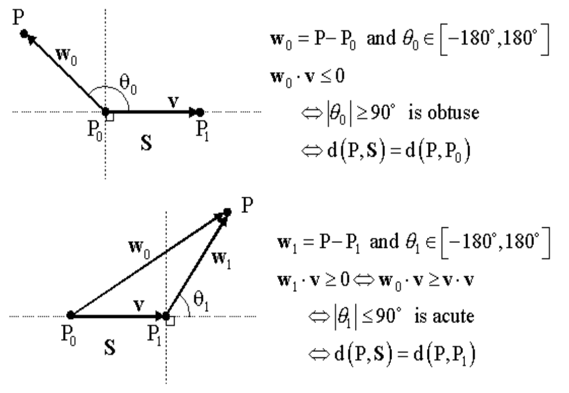
P
w_0
θ_0
v
P_0
S
P_1
$\mathbf{w}_0 = \mathrm{P} - \mathrm{P}_0$ and $\theta_0 \in [-180°, 180°]$
$\mathbf{w}_0 \cdot \mathbf{v} \le 0$
$\Leftrightarrow |\theta_0| \ge 90°$ is obtuse
$\Leftrightarrow \mathrm{d}(\mathrm{P}, \mathrm{S}) = \mathrm{d}(\mathrm{P}, \mathrm{P}_0)$
P
w_0
w_1
v
θ_1
P_0
S
P_1
$\mathbf{w}_1 = \mathrm{P} - \mathrm{P}_1$ and $\theta_1 \in [-180°, 180°]$
$\mathbf{w}_1 \cdot \mathbf{v} \ge 0 \Leftrightarrow \mathbf{w}_0 \cdot \mathbf{v} \ge \mathbf{v} \cdot \mathbf{v}$
$\Leftrightarrow |\theta_1| \le 90°$ is acute
$\Leftrightarrow \mathrm{d}(\mathrm{P}, \mathrm{S}) = \mathrm{d}(\mathrm{P}, \mathrm{P}_1)$

Fig. 4. Angles between the segments, (Sunday, 2001a).

The pseudo-code used to evaluate the minimum distance between a circular obstacle and a strait line segment is shown in Figure 5.

```
Distance (Point P, Segment P0P1)
  v = P1 – P0;
  w = P - P0;
  If ( (c1 = w · v ) <= 0 )
     Return distance from P to P0;
  If ( (c2 = v · v ) <= c1 )
     Return distance from P to P1;
  b = c1 / c2;
  base point = P0 + b*v;
  Return distance from P to base point;
```

Fig. 5. Distance between a point and a line segment.

In the case of polygonal obstacles, they are defined by the connection of their vertices by line segments, as it is illustrated in Figure 6.

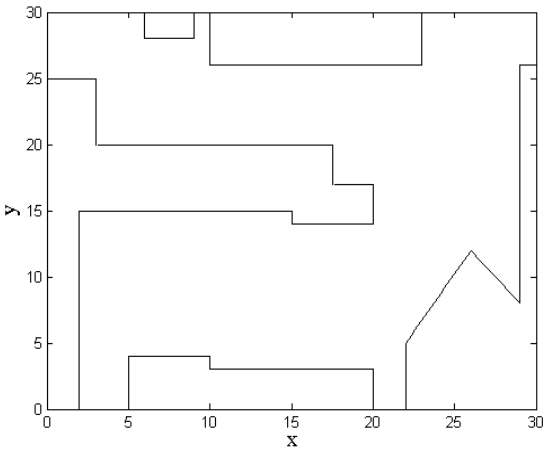

Fig. 6. Robot world with polygonal walls.

One way to avoid collisions is to check if every line segment of the optimized trajectory intersects the line segments defining the polygons (Sunday, 2001d). The pseudo-code used for this case is shown in Figure 7. It is clear that the optimized trajectory can not include line segments with are inside the polygonal shapes. To prevent this case it is necessary to determine for each solution if all the extreme points of all the defining segments are not inside the polygons. In figure 8 a pseudo code (Sunday, 2001b) is present that was used to solve this problem.

```
[Tests the intersection from two segments, returning: 0
        = none, 1 = intersection]
Intersection (Segment P0P1; Segment P2P3)

  Test (Local) if P0 and P1 are in the left, right or over
        P2P3;
  If (P0 and P1 are in the left or P0 and P1 are in the
        right) do:
     Return 0;

  Else:
     Test (Local) if P2 and P3 are in the left, right or over
        P0P1;
        If (P2 and P3 are in the left or P2 and P3 are in
        the right) do:
           Return 0;
        Else:
           Return 1;

[Checks if a point is in the left, right or over a segment,
        returning: >0 for left; =0 for over; <0 for
        right]
Local (Point Pa, Point Pb, Point Pc)
  Return (Pb.x – Pa.x) * (Pc.y – Pa.y) - (Pc.x – Pa.x) * (Pb.y –
        Pa.y)
```

Fig. 7. Intersection of two segments.

The algorithm shown in Figure 8, for each test point uses a parallel line to the *x* axis, determining the number of intersections between the polygon and points to the right of the test point. If the number of intersections is even it means that the point is outside the polygon, otherwise it is inside.

```
Checkpoint (Point P, Vector with vertices from polygon V)
  Counter = 0;
  Do for each segment of the polygon:
    If polygon segment intersects support line do:
      If P is left from the segment:
        Counter = Counter + 1;
      Else:
        If P is right from the segment:
          Counter = Counter – 1;
  Cycle end;
  Return Counter;
```

Fig. 8. Algorithm that checks if a point is inside or outside a polygon.

Another case is when one of the path segments and one of the polygon segments are collinear. To solve this problem the pseudo-code presented in Figure 9 was used (Sunday, 2001c).

```
Point in a Segment (Point P, Segment S)
  If (S.0.x ? S.1.x) do:
    If (S.0.x <= P.x) e (P.x <= S.1.x) do:
                          Return True;
    If (S.0.x >= P.x) e (P.x >= S.1.x) do:
      Return True;
  Else:
    If (S.0.y <= P.y) e (P.x <= S.1.y) do:
      Return True;
    If (S.0.y >= P.y) e (P.x >= S.1.y) do:
      Return True;

  Return false;      [If none of the conditions is possible]
```

Fig. 9. Detects if two line segments are collinear.

4. SIMULATION RESULTS

4.1 Case Study I

In the first case two circular obstacles were considered located in between the starting and goal points, (Figure 10). The strategy number 6 was selected for the DE algorithm and the crossover probability and amplification parameter were set to CR = 0.8 e F = 0.8, respectively. Different population sizes and number of points defining the trajectory are used and the number of iterations is 1000 per test. The results obtained with 20 runs for each condition are presented in Table 1.

Table 1 Results from the case study I

d	m	Best Distance	Distance Average Value
4	30	30,7912	30,80322
	50	30,7911	30,79420
	100	30,7912	30,79429
5	30	30,7718	30,79900
	50	30,7716	30,78613
	100	30,7725	30,78533
6	30	30,7712	30,79046
	50	30,7741	30,79011
	100	30,7758	30,79304
7	30	30,7667	30,80538
	50	30,7684	30,79933
	100	30,7903	30,80562

The results achieved for different combinations between the number of points and population size are practically identical. In Figure 10, the path obtained with minimum distance (with 7 points and population size of 30) and the path obtained with maximum distance (with 4 points and population size of 30) are represented. While different, they correspond to very similar cost values in terms of distances attained.

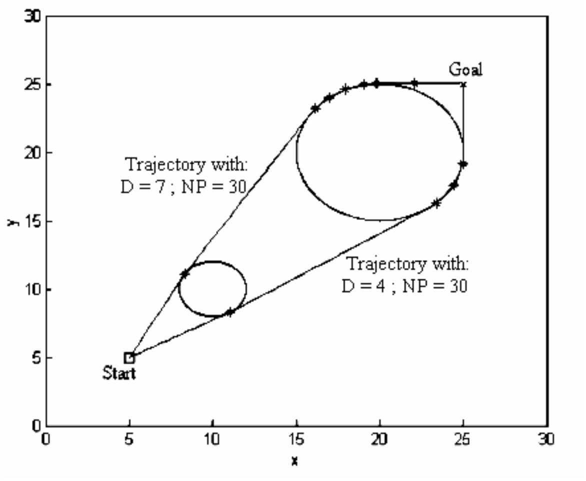

Fig. 10. Two optimized solutions for the case study I.

Figures 11 to 15 illustrate convergence plots for several optimization conditions.

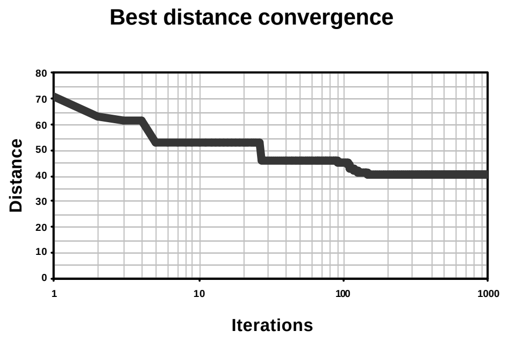

Fig. 11. Best distance convergence graphic with: 2 round obstacles; strategy 6; population of 100 and 6 points.

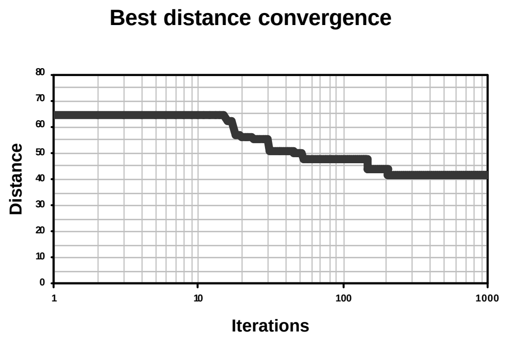

Fig. 12. Best distance convergence graphic with: 4 round obstacles; strategy 9; population of 50 and 7 points.

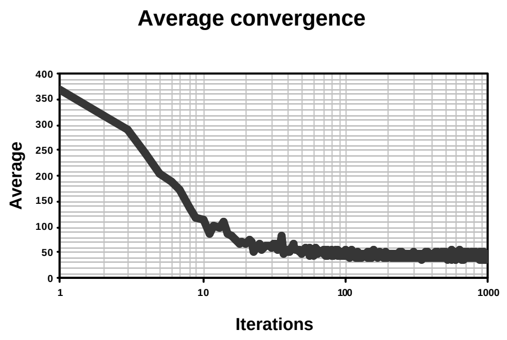

Fig. 13. Average convergence graphic with: 2 round obstacles; strategy 8; population of 30 and 5 points.

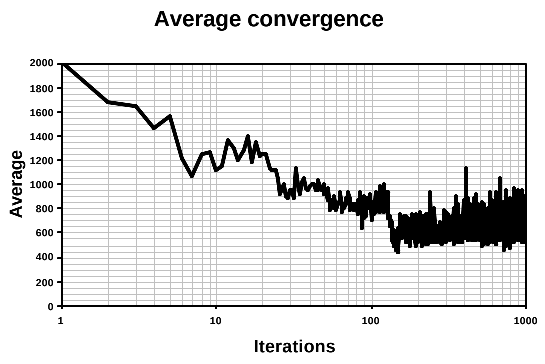

Fig. 14. Average convergence graphic with: 5 polygonal obstacles; strategy 6; population of 30 and 4 points.

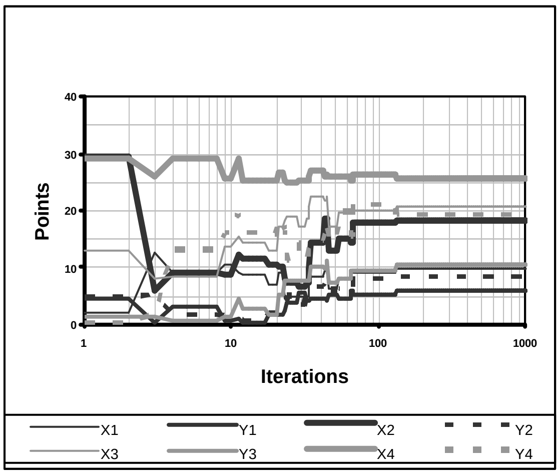

Fig. 15. Point coordinates convergence graphic with: 4 round obstacles; strategy 6; population of 30 and 4 points.

The results clearly indicate that there is no need to perform 1000 iterations, thus the simulation period is reduced to 500 iterations per test for the case study II.

4.2 Case Study II

Case study I was not hard enough to evaluate the DE algorithm. Thus in this case study the robot world has four circular obstacles as it is shown in Figure 16. The simulated results obtained for this case are presented in Table 2, in which several combinations between the number of points defining the path, population size and DE strategies were considered.

Table 2 Tests results with four case study II.

N	m	Str[1]	Best Distance	Distance average
4	30	6	32,8304	33,56297
		8	32,9924	33,89754
		9	33,2588	37,697395
	50	6	32,6951	33,868625
		8	32,8162	33,280825
		9	34,0119	37,59996
	100	6	32,5566	33,70214
		8	32,488	32,967025
		9	35,0931	37,7244
5	30	6	33,0608	33,822465
		8	33,1618	33,58607
		9	35,7624	40,066135
	50	6	32,7631	33,562595
		8	32,8727	33,8194
		9	37,0991	40,811525
	100	6	32,6107	34,35063
		8	32,4066	33,57655
		9	37,0881	40,36724
6	30	6	33,4941	34,086565
		8	32,9629	34,101175
		9	35,8821	45,009295
	50	6	32,9319	33,492605
		8	32,5999	34,33487
		9	39,4767	44,284785
	100	6	31,8105	37,43633
		8	32,8198	34,626335
		9	38,74	43,90211
7	30	6	33,2443	35,384755
		8	33,1747	35,15421
		9	37,5418	44,14192
	50	6	33,0859	35,211695
		8	33,3694	34,377645
		9	39,9851	45,45782
	100	6	32,9407	37,21735
		8	32,9297	33,830625
		9	41,0129	45,376245

[1] Strategy.

In this case the differences between the different test conditions are more significant. Strategy 6 is the one who delivered best results, as it can be verified by the best distances achieved. In Figure 16 some trajectories are plotted for a population size of 100 and different strategies and number of points.

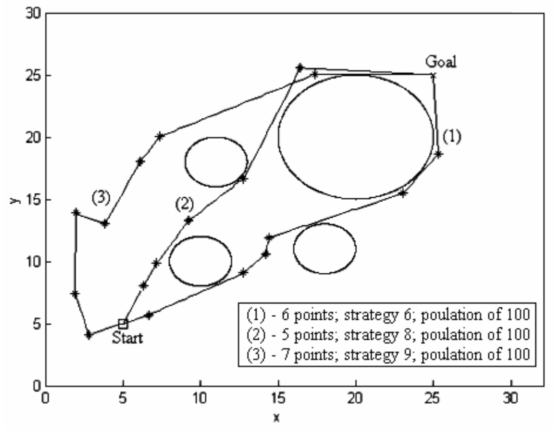

Fig. 16. Several paths in a world with four round obstacles.

4.3 Case Study III

In this case the simulated world has polygonal obstacles represented by walls, (Figure 17). Some preliminary results are presented in Table 3 using strategy 6, *CR*=0.8, *F*=0.8 and 500 iterations per test. A total of 20 runs per combination were made. The initial population was generated randomly. The results are not conclusive at this stage.

Table 3 Tests results with polygonal obstacles

Number of points	Number of population	Best distance	Distance average
4	30	41,3828	305,128895
	50	42,5877	204,55312
	100	43,2795	266,430845
5	30	41,9369	247,398085
	50	40,8407	285,943

While the DE algorithm managed to provide good path solutions for this case, it was clear that allowing randomly generated solutions which are not viable to incorporate the initial population deteriorate the algorithm performance. Some paths corresponding to the results presented in Table 3 are plotted in Figure 17.

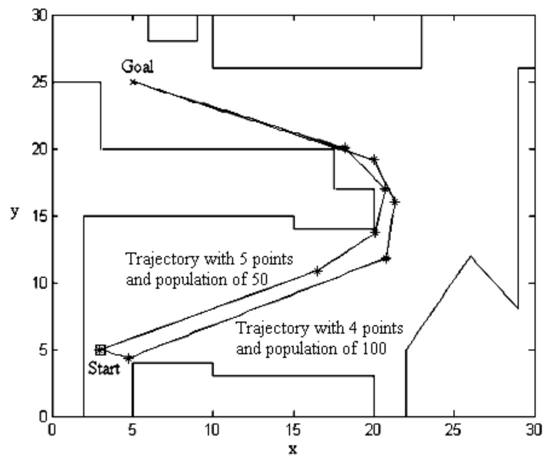

Fig. 17. Several optimized paths for a simulated world with polygonal obstacles.

5. CONCLUSIONS AND FURTHER WORK

The Differential Evolution Algorithm was proposed to address the path planning optimization problem. The obstacles were considered to be static and of circular and polygonal shapes and the path optimization performed off-line.

The results indicated that the Differential Evolution Algorithm could solve this problem satisfactorily, achieving the best results in terms of the minimization of the path length with strategy number 6.

Current research is being carried out in order to complete the optimization with polygonal obstacles. Future tests will implement the algorithms on-line assuming dynamic obstacles and comparisons with other evolutionary based algorithms, such as genetic algorithm and the particle swarm optimization algorithm.

REFERENCES

Goldberg, David E. (1989). *Genetic Algorithms in search, Optimization and Machine Learning.* Addison-Wesley, Reading (MA).

Michalewicz Z, (1992), "Genetic Algorithms + Data Structures = Evolution Programs", 2nd Edit., Springer-Verlag P.C..

Storn, Rainer and Price, Kenneth (1995). *Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces.* Technical Report TR-95-012, ICSI, March 1995.

Storn, Rainer (1996). *On the usage of Differential Evolution for Function Optimization.* NAFIPS 1996, Berkeley, pp. 519 – 523

Sunday, Dan (2001a). *About Lines and Distance of a Point to a Line (2D & 3D),*Geometry Algorithm Archive, February 2001.

Sunday, Dan (2001b). *Fast Winding Number Inclusion of a Point in a Polygon,* Geometry Algorithm Archive, March 2001.

Sunday, Dan (2001c). *Intersections of Lines, Segments and Planes (2D and 3D),*Geometry Algorithm Archive, April 2001.

Sunday, Dan (2001d). *The Intersections for a Set of 2D Segments, and Testing Simple Polygons,* Geometry Algorithm Archive, August 2001.

Price K. e Storn R. (2002), *Differential evolution homepage: http:/www.ICSI.Berkeley.edu/~storn/.*

Lampinem J. e Zelinka I., (1999), *Mixed Variable Non-Linear Optimization by Differential Evolution*, Em Zelinka I. (Editor), *Proceedings of Nostradamus'99, 2nd International Prediction Conference,* Zlin, República Checa, pp. 45-55.