HYBRID GAME EVOLUTIONARY ALGORITHM FOR MISSION PATH PLANNING OF AERIAL SURVEY TASKS

G. Rappa *, F. Gonzalez **, J. Kok **, F. Quagliotti *
*** Politecnico di Torino, Dipartimento di Ingegneria Meccanica e Aerospaziale, Torino, Italy.**
**** Australian Research Centre Aerospace Automation (ARCAA), School of System Engineering, Queensland University Technology (QUT), Brisbane 4001, Australia.**
giovanni.rappa@studenti.polito.it; felipe.gonzalez@qut.edu.au;
jonathan.kok@student.qut.edu.au; fulvia.quagliotti@polito.it

Keywords: *Hybrid Game, MOEA, Mission Path Planning, UAS, Infrastructure Inspection*

Abstract

The aim of this paper is to implement a Game-Theory based offline mission path planner for aerial inspection tasks of large linear infrastructures. Like most real-world optimisation problems, mission path planning involves a number of objectives which ideally should be minimised simultaneously. The goal of this work is then to develop a Multi-Objective (MO) optimisation tool able to provide a set of optimal solutions for the inspection task, given the environment data, the mission requirements and the definition of the objectives to minimise. Results indicate the robustness and capability of the method to find the trade-off between the Pareto-optimal solutions.

1 Introduction

Nowadays most of the world relies on complex frameworks of oil/gas/power distribution as well as goods delivery infrastructure such as highways and railways. In the last years, the use of Unmanned Aerial Systems (UASs) for civil applications had an increasing trend, so that today UASs can be considered as a valid alternative for long and monotonous civil missions such as the inspection of a large linear infrastructure. The use of UASs for linear infrastructure inspection has been explored by a number of researchers [3], [7], but the benefits of using UASs may be compromised if the mission planning is inadequate. Typically, monitoring missions of linear infrastructures alternate a number of on-service and off-service legs, and the total time spent for the off-service part of the mission may amount to a significant portion of the whole duration of the mission. Therefore, a well-designed mission should be targeted to minimise the costs due to the off-service legs with respect to the on-service legs.

The rest of the paper is organised as follows: in Section 2 the methodology adopted and the formulation of the problem are described. Section 3 presents an overview of NSGA-II, Nash-GAs and Hybrid Game. Section 4 describes how the Hybrid Game framework is implemented and details the strategies for the players of Hybrid Game. Section 5 presents some test bench cases. Section 6 details a novel formulation of the Multi-Objective Travelling Salesman Problem (MO-TSP) and the techniques adopted for its solution; the operators and the way they are used by the players of the Hybrid Game are detailed. Section 7, details the application of the MO-TSP solver to the mission path planning problem for aerial inspection of large linear infrastructures. Finally, the results obtained for two practical cases are presented. A discussion of the findings of this works is given in Section 8.

2 Methodology and problem formulation

The goal of the paper is to implement a novel MO Genetic Algorithm based on Hybrid Game strategy in Matlab. The core of the algorithm is an enhanced version of NSGA-II and original solutions for the strategy of Hybrid Game are proposed. The performance of the algorithm is evaluated against some bench test optimisation problems. Then, the Hybrid Game framework is used to solve Travelling Salesman Problems (TSP). The issues of such combinatorial optimisation problem are analysed and the definition of the chromosomes and the genetic operators are customised for the solution of the TSP. A MO formulation of the TSP (MO-TSP) is detailed and some bench instances of MO-TSP are created and tested.

The MO-TSP formulation is then used to define the two-dimensional environment of a large infrastructure inspection task. In order to reproduce

realistic environments the digital scenario includes both the infrastructure to inspect and a number of risk areas, thus allowing to formulate the mission path planning problem as a Multi-Objective Optimisation Problem (MOOP) involving the length of the trajectory and a risk function as the objectives. The model adopted to describe the trajectory of the UAS is based on Dubins curves. Finally, a number of scenarios reproducing the complete railway network of Queensland, Australia, are used to test the software. The hardware used for all the tests is a PC running Windows 7 64 bit with a 2.40 GHz Intel(R) Core(TM) i5-M450 CPU and 4GB of RAM.

3 Background

3.1 Multi-Objective Evolutionary Algorithms

Most real-world optimisation problems involve simultaneous minimisation of several conflicting objectives. Multi-Objective Evolutionary Algorithms (MOEAs) have been developed to find sets of optimal trade-off solutions for MOOPs. In this work we focus on NSGA-II, as it is a well-known algorithm and it showed good performance in many bench tests [1], [11]. The procedure of NSGA-II may be arranged into 7 steps [13], [14]:

1) the main evolutionary parameters (i.e. population size and maximum number of generations) are given and the objective functions are defined;
2) the population is randomly initialised; all objective functions are evaluated for each individual and is stored;
3) the population is sorted into fronts based on non-domination: a rank is assigned to each individual so that individuals having rank j dominate all individuals having rank $k>j$ and are dominated by all individuals having rank $i<j$. Then, within each front, all individuals are classified depending on their crowding distance, that is a measure of how close an individual is to its neighbours and it is used in order to preserve diversity in the population;
4) a mating pool is picked by means of a binary tournament selection based on individual rank and crowding distance;
5) generic operator on the mating pool is conducted to generate an offspring population; the evolution consists either in a Simulated Binary Crossover (SBX) or in a genetic mutation based on polynomial mutation [4];
6) the resulting intermediate population, which includes both parents and offspring, is in turn sorted based on the same criteria of non-domination and crowding distance;
7) a selection of the population is performed, i.e. only the individuals belonging to the first fronts survive while the others are discarded.

The number of generations evaluated is used as the stopping criterion, so steps 4, 5, 6, and 7 are cyclically run until the maximum number of generations is achieved.

3.2 Nash-Genetic Algorithms

A MOOP can be solved using as many players as the objectives of the problem: each player optimises one criterion keeping all other criteria fixed by the other players. Such a strategy is called *Nash-strategy* and tends to the so called *Nash-equilibrium*, i.e. the condition when no player can further improve its own objective [10], [29].

Nash-strategies can be implemented within a Genetic Algorithm in order to obtain Nash-equilibrium solutions of MO problems [15], [16], [18]. A technique to implement a Nash-GA for an M-Objective optimisation problem is to create M distinct populations assigned to M Nash-players; each player evolves its own population based on its own criterion and send its best solution to the other players, which in turn optimise their own criterion without changing any criterion optimised by the other players. The scheme proceeds generation after generation and the evolution can be considered completed when the Nash-equilibrium is reached. Figure 1 shows the flow-chart of a Nash-GA for a 2-objective problem.

3.3 Hybrid Game Genetic Algorithms

Hybrid Games are advanced optimisation methods which couple different strategies within one single framework. A suitable implementation of Hybrid Game consists in one *Pareto-player* and a number of *Nash-players* exchanging information each other to produce Nash-equilibrium and Pareto-optimal solutions at one time [15].

The goal of coupling the Nash-strategy and

the Pareto-strategy into a Hybrid Game is to speed up the convergence of the solutions found by the Pareto-player towards the Pareto-optimal front. The role of the Nash-players within a Hybrid Game is therefore to explore the extreme zones of the objective space and seed useful information to the Pareto-player and to the other Nash-players.

The Hybrid Game developed in this work is a variant of the original Hybrid Game, [14] [15], [16]. The exchange of information is conducted by sending and seeding some elite members from each population to the other ones. Such a technique, further discussed in Section 4.3, is called migration and performed cyclically during the evolution.

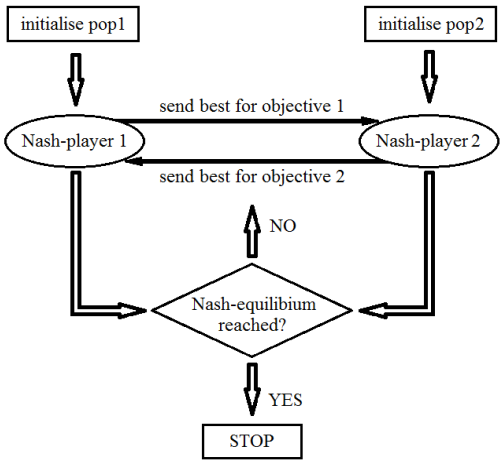

Figure 1. Flow-chart for a 2-objective Nash-GA.

4 Hybrid Game framework

4.1 Pareto-player's MO strategy

The Pareto-player of the algorithm presented in this paper consists of a MOEA based on the well-known NSGA-II algorithm. In its original formulation, NSGA-II works on a population of predetermined constant size [5]. The computational cost to evaluate one generation is $O(MN^2)$, being M the number of objective functions and N the population size [2]. The basic idea encouraging the new scheme proposed in this work is to reduce the computational cost of the MOEA by making it work on a population whose size varies in accordance with a given law type, in order to obtain better convergence towards the Pareto-front in lower computational time [8]. All the tests presented in this work are conducted with populations varying quadratically.

4.2 Nash-Players' single-objective strategy

Each Nash-player of Hybrid Game optimises one objective and produces optimal solutions speeding-up the convergence of the global Pareto-player towards the Pareto-optimal solutions set. The Nash-player's strategy is implemented following the usual structure of a generic EA. The parent-selection and the population-sorting operators used by Nash-players are different from those used by the Pareto-player. In particular, the mating pool of the Nash-players is obtained by merging two subsets of individuals, one filled up using an elitist criterion and one formed by randomly picked members. The purpose of such a combined parent-selection is to guarantee the involvement of the new members coming from other populations in the evolution of the Nash-populations.

Another important feature of the Nash-players strategy is that the application of the genetic operators occurs after the selection of the decision variables specific for that Nash-player. The action of a given Nash-player is hence restricted to just some genes of the whole chromosome while the other genes are kept frozen. An important issue in the development of a Hybrid Game algorithm is consequently the selection of the decision variables on which every Nash-player is allowed to operate on. The most pragmatic consequence of this feature of Hybrid Game is that the formulation of the model describing the optimization problem is a crucial matter, since the problem should be carefully formulated in such a way that the effect of the decision variables can be easily discerned and an efficient strategy can be implemented.

4.3 Elite migration technique

The exchange of information between the players of Hybrid Game occurs by means of cyclic migrations of some individuals (elite group) between the populations involved into the evolution process.

In the algorithm, the elite group sent from the Nash-population A to the Nash-population B is populated selecting the members of population A having the best fitness value from the standpoint of the Nash-player B. Analogously, the subgroup migrating from any Nash-population N to the Pareto-population P is composed by the members

of the populations *N* having the lowest rank. This mechanism aims to keep the exchange of information during the evolution as high as possible while simultaneously reducing the risk to get trapped in some local minima. Also the subgroup sent from the Pareto-population *P* to any Nash-population *N* is populated by the members of the origin population *P* having the lowest rank.

5 Test bench problems

5.1 Mathematical test cases: ZDT functions

The Zitzler-Deb-Thiele's (ZDT) functions are two-objective optimisation problems commonly used in literature [2], [12], [14]defined in the general form:

Minimise: $f_1(\boldsymbol{x})$ (1)

$$f_2(\boldsymbol{x}) = g(\boldsymbol{x}) \cdot h\big(f_1(\boldsymbol{x}), g(\boldsymbol{x})\big)$$

where: $\boldsymbol{x} = x_i \quad i = [1, n]$

Depending on the expressions assumed by the functions $f_1(x)$, $g(x)$, and $h(x)$, six problems were defined [12]. The ZDT functions used as bench tests in this work are the ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6. The value of the decision variable x_i is the i^{th} gene of the chromosomes used in the evolutionary optimisation. The only decision variables assigned to the first Nash-player is x_1, while all the other variables are assigned to the second Nash-player. The evolutionary parameters used for the optimisation of the ZDTs are summarised in Table 1.

	ZDT1	*ZDT2*	*ZDT3*	*ZDT4*	*ZDT6*
main pop.size	20:100	20:100	20:100	20:100	20:100
Nash-pop.size	15:50	15:50	15:50	15:50	15:50
growth exponent	2	2	2	2	2
elite group size	2	2	2	2	2
max time [s]	30	30	45	60	15
generations evaluated	1150	1124	1672	2395	1949

Table 1. Evolutionary parameters used for the bench **tests.**

The final Pareto-fronts of the ZDT functions were computed but only a few are shown here due to space restrictions.

5.2 MO Engineering problems

We also considered the *Gear train design*, the *Two-bar truss design* and the *Welded beam design* problems. [19], [20], [21] and the evolutionary parameters used for the optimisation of this set of problems are summarised in Table 2.

	Gear train	*Truss*	*Welded beam*
main pop.size	40:100	40:100	40:100
Nash-pop.size	15:50	15:50	15:50
growth exponent	2	2	2
elite groups size	2	2	2
penalty value on f_1	-	10^2	10^2
penalty value on f_2	-	10^7	10^{-1}
maximum time [s]	120	120	120
generations evaluated	2676	2487	2060

Table 2. Evolutionary parameters used for the Engineering optimisation test problems.

5.3 Results and discussion

The algorithm proved to be able to find the true Pareto-front of all the ZDT functions but the ZDT4. Such an anomalous behaviour is supposed to be due to intrinsic complexity of the ZDT4 function, which has 21^9 of local Pareto-fronts in the problem. Such a considerably high number of local minima makes the algorithm get stuck in one of them preventing the convergence towards the global Pareto-front.

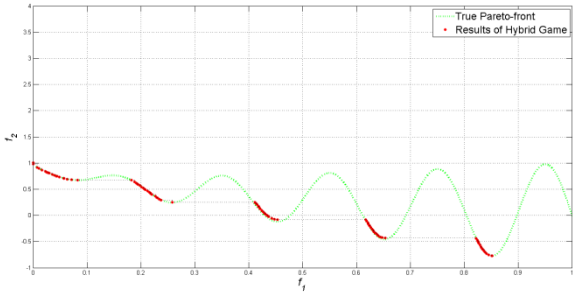

Figure 2. Pareto-front found for function ZDT3.

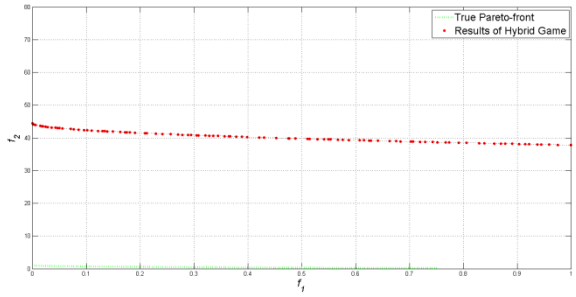

Figure 3. Pareto-front found for function ZDT4.

Amongst the Engineering optimisation problems evaluated, the true Pareto-front was available only in the gear design case. Some values of the true Pareto-front were not found by the algorithm, but a globally well spread set of optimal solutions was found. On the contrary, no direct numerical comparison of the results obtained for the truss design and the welded beam optimisation problems was available.

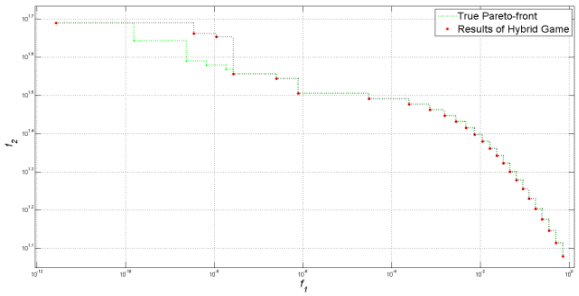

Figure 4. Pareto-front found for the gear train problem.

An intrinsic difficulty of the Engineering problems tested here is that the effects of each decision variable on the objective functions are not straightforward to identify. This probably prevented an adequate assignment of the decision variables to the Nash-players, leading to the adoption of a weak overall strategy of the Hybrid Game framework.

6 Multi-Objective TSP

The Travelling Salesman Problem (TSP) is a combinatorial optimisation problem stated as [22]:

$$\text{Minimise:} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}x_{ij} \qquad (2)$$

$$\text{subject to:} \quad \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1,\dots,N$$

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1,\dots,N$$

$$x_{ij} = \{0,1\}$$

The problem requires to minimise the total cost of a tour between a set of N cities, given the cost function d_{ij}=d_{ji} between each pair of cities i and j; x_{ij} is a discrete variable which can assume only value 0 or 1. The constraints indicate that the tour must visit each city only once, thus the solution of the TSP is the shortest *Hamiltonian cycle* (i.e. a close cycle visiting each node exactly once) of the graph whose nodes are the cities.

The solution of a TSP by comparison of all the possible $(N-1)!$ solutions becomes practically impossible for a few tens of cities. Many optimisation algorithms and heuristic techniques have been proposed during the last decades to solve the TSP. Currently, *Concorde* software is one of the best TSP solvers [22], and its solution is used in this work as the reference to estimate the quality of the paths found.

Many efforts were also done to solve the TSP by means of Evolutionary Algorithms [17], [23], [24]. The major issues concerning the application of Evolutionary Algorithms to the TSP are related to the chromosome definition and the genetic operators. In order to obtain suitable implementations, the chromosomes must be designed in such a way that their genes can represent the properties of feasible solutions; the generic operators must be able to combine the parents transferring useful information to the offspring in a non-destructive manner [25].

The algorithm presented here can be conceptually divided into three steps, which are run in sequence:

- *pre-processing*: the environment data is imported and decoded; the evolutionary parameters are set and the initial populations are initialised;
- *processing*: the populations evolve until a stopping criterion is met;
- *post-processing*: the chromosomes belonging to the final Pareto-front of the main population are stored in an external file and the results are shown graphically to the user.

In order to formulate a model for the implementation of a TSP solver, one of the major issues is the implementation of a method to describe a candidate solution. In this work the approach used is the so called *path representation*, i.e. the cities are labelled with progressive numbering, so that the chromosome is simply the ordered sequence of the cities which gives the candidate path.

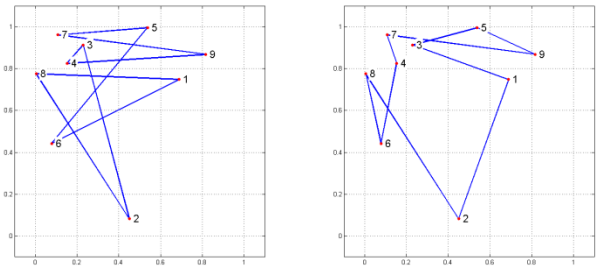

Figure 5. Representation of two example paths.

As an example, let us consider the simple TSP of 9 random cities (Figure 5). The path representation of two possible solutions *path1* and *path2* of such a TSP is then:

path1=[1 6 5 7 9 4 3 2 8]

path2=[8 6 4 7 9 5 3 1 2]

They are representative of the paths (a) and (b) of Figure 5 respectively. An important feature of the path representation is that a complete path through

the cities of a V-dimensional TSP can be expressed in $2V$ possible ways, due to the circularity of the solution of the TSP. Such a peculiarity is conveniently exploited by some of the genetic operators implemented in this work.

6.1 Objective functions definition

In order to build-up a model which can be later exploited as the basis for the aerial mission path planning, the MO-TSP formulation presented here involves the two following cost functions f_1 and f_2, where:

- $f_1 = L$ is the length travelled, i.e. in a two-dimensional environment the Euclidean distance between the coordinates of any couple of consecutive cities of the tour;
- $f_2 = R$ is an additional cost function referred to as the risk of the tour.

To introduce the risk function, the model includes a number of elements, called *dangerous points*, whose presence induces a cost to the paths passing in its vicinity closer than a certain threshold. The risk $f_2(i,j)$ of the edge between the cities i and j is then defined as the total number of dangerous points encountered during the travel from i to j on the left and right side of the edge. Figure 6 shows the 9-city example TSP described previously where 50 randomly placed dangerous points are considered. The tour *path1* is also shown as a dashed blue line. The yellow-shaded area plotted in the same picture identifies the region (threshold=0.05) within which any dangerous points affects the risk of an edge.

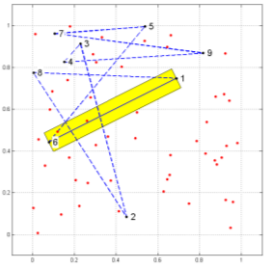

Figure 6. Example MO-TSP and risk evaluation of an edge.

The two cost functions just defined are additive functions, meaning that the values of the cost functions of a complete tour through the cities is given by the sum of the cost functions of each single edge forming the tour.

6.2 Look-up tables

The computational time needed for the evaluation of the cost function increases with the dimension of the problem (i.e. number of the cities to visit) as well as the total number of dangerous points considered [26]. In order to make the evolutionary process faster, a timesaving technique is adopted to save computational resources during the evolution: in the pre-processing phase, the cost functions of all the edges are evaluated and stored into tridimensional matrices used as look-up tables during the evolution. This solution makes the time required for the evolution absolutely independent of the time taken to evaluate the cost functions.

6.3 Genetic operators for MO-TSP

The MO-TSP belongs to the class of combinatorial problems. Due to the different nature of the problems, five crossover operators and four mutation operators were specifically designed for the TSP and implemented to be used by the MO-TSP solver instead of the SBX and the usual mutation operator.

6.3.1 Partially Mapped Crossover

This crossover performs a mapping between the set of consecutive cities which in the two parent chromosomes occupy the random positions from *i* to *j*, while keeping the order of the other cities unchanged.

6.3.2 Order-Based Crossover

This crossover keeps unchanged the order of a set of consecutive cities occupying the random positions from *i* to *j* of each parent and changes the order of the other cities in accordance of the order of those cities in the other parent.

6.3.3 Sub-path Crossover

A random city *X* is picked and the cities *Y1* and *Y2* following *X* in the parent paths are identified. The sub-path of the second parent *p2* going from *X* to *Y1* and the sub-path of the first parent p1 going from *X* to *Y2* are isolated and placed between *X* and *Y1* in *p1* and between *X* and *Y2* in *p2*. The rest of the cities are kept unchanged.

6.3.4 City-Centred Crossover

A random city *X* is picked and the path of *p1* from the first city to *X* is copied to the first child *c1*, and the path of *p2* from the first city to *X* is copied to the second child *c2*. The rest of *c1* is filled in

according to the order the remaining cities appear in *p2* and the rest of *c2* is filled in according to the order the remaining cities appear in *p1*.

6.3.5 Edge-Recombination Crossover

This crossover is targeted to obtain offspring chromosomes having as many edges as possible equal to the edges of the parents. The operator first evaluates the edges of each parent and identifies the ones common to both of them. The offspring chromosomes are built using the edges common to both parents; if no common edges are available the rest of the offspring chromosome is built randomly.

6.3.6 Shortest sub-path mutation

This mutation reorders heuristically the cities of the subpath of the parent *p3* from *i* to *j* while keeping the other cities unchanged. The method used to reorder the cities from *i* to *j* is to place after each city *X* the city *Y* of the remaining ones such that the cost of the edge (*X*,*Y*) is the lowest.

6.3.7 Single-point insertion mutation

This mutation picks a random city *X* and moves it between the consecutive cities *A* and *B* such that the total length of the subpath [*A*-*X*-*B*] is the shortest. The order of the other cities is kept unchanged.

6.3.8 Simple swap mutation

This mutation mutually exchange the position of two cities picked randomly.

6.3.9 Single-edge-insertion mutation

This mutation is the dual version of the single-point insertion operator: it randomly picks two consecutive cities *A* and *B* of the parent chromosome and interposes between them the city *X* for which the total length of the subpath [*A*-*X*-*B*] is the lowest. The order of the other cities is kept unchanged.

6.4 Sub-tour inversion operator

Besides the crossover and mutation operators detailed above, another type of operator was implemented. The operator can be conceptually considered as a mutation, since it gets as an input a single chromosome and returns as an output a single child. It is used as a regular mutation operator by all the players of Hybrid Game, but also as a specific optimiser of the offspring of the Nash-players. This operator, called *sub-tour inversion*, explores the possibility to reduce the cost of a complete tour [*A*…*Z*] by inverting any intermediate sequence [*X*…*Y*] of consecutive cities:

$p3=[A \dots X \dots Y \dots Z]$

$c3=[A \dots Y \dots X \dots Z]$

This operator is very effective, since it always returns as an output a chromosome whose fitness is not worse than the parent chromosome. Its drawback relies in the computational time, which is longer than any other operator implemented.

6.5 Hybrid strategy for MO-TSP

The algorithm presented in Section 4 is customised to solve MO-TSP cases. The framework imports the problem data (i.e. the position of the cities and of the dangerous points) and creates the matrix of look-up tables. Then the main population is initialised and the cost functions of all its members are evaluated. After the initialisation of the population operated by the Nash-players, the subtourinversion operator is applied to their members: each member of the population of the m^{th} Nash-player is linearised with respect of the m^{th} objective function, so that the useful action of the Nash-players be efficient from the first generation. Once all the populations are properly initialised, the evolutionary optimisation is conducted by following the general scheme of the Hybrid Game detailed in the previous section.

The first difference with respect to the Hybrid Game algorithm described in Section 4 concerns the variable assignment at Nash-players. The results of the Engineering problems proposed in Section 5.2 revealed that the separability of the variables of a MOOP, and consequently a proper assignment of the decision variables to each Nash-player, is a complex aspect of the definition of an efficient Hybrid Game strategy. Hence, a novel automatic procedure for the variable assignment is proposed here. Such a procedure was made possible by the introduction of the look-up tables and it is adopted for the MO-TSP solver and for the infrastructure inspection task path planner as well. Keeping in mind that the cost function of the global trajectory is given by the sum of the cost of each edge, the contribution to the total cost of a complete tour due to the presence of the city *X* in the tour, is

given by the sum of the cost to go from the city A to *X* and from *X* to *B*, being *A* and *B* the cities preceding and following *X* respectively. The cost of visiting the city *X* of a *V*-dimensional MO-TSP is equal to the sum of the costs of the (*V*-1) edges connecting *X* to the other cities of the problem. According to this definition, the risk of visiting the city *X* can be evaluated by summing-up all the elements of the X^{th} row of the look-up table of the risk, since the table contains the costs of all the possible edges between *X* and any other city of the problem. The risk of each city is evaluated and the cities are sorted based on their risk. At this stage, the first N_{safe} and the last $N_{dangerous}$ cities are stored as the safest and most dangerous cities of the problem. The value of N_{safe} and $N_{dangerous}$ can be chosen by the user; in all the following cases it is assumed $N_{safe}=N_{dangerous}=0.25V$.

The subtourinversion operator is used by the m^{th} Nash-player to improve the fitness of its own offspring in the regards of the m^{th} objective. It is in this phase that the assignment of the decision variables takes place; the look-up table sent as an input to the subtourinversion operator contains the element of the m^{th} cost function for all the edges except the ones connecting the cities assigned to the other Nash-player. The value of the cost function of such edges is reduced by a certain amount in order to force the *sub-tour inversion* algorithm to restore the edges between the cities assigned to the other Nash-player.

Some operators perform heuristics estimations and they need the lookup table as an input. When a heuristic genetic operator is called by any Nash-player, the related look-up table is used. Otherwise, if a heuristic genetic operator is called by the Pareto-player, an equivalent look-up table is built averaging the cost functions of each edge.

6.6 Benchmark results

Hybrid Game MO-TSP algorithm was tested on a few benchmark problems, but only one (namely *MO_pma343*, where 343 cities and 3000 dangerous points clustered in six lines are located in the range [0,300]×[0,42]) is reported in this work. Other test cases can be found in [9].

The test case is a benchmark case proposed in the TSP toolbox [28] and adapted here to the MO-TSP formulation. The clusters of dangerous points are placed over some edges of the shortest path on purpose, with the intention to make the shortest path also the most dangerous path and force the algorithm to find alternative (i.e. longer but safer) solutions. The evolutionary parameters used for the test are summarised in Table 3 and the results obtained after the evolution are summarised in Table 4. The final Pareto-front of the main population is plotted in Figure 7, and the shortest, the safest and one trade-off path are shown in Figure 8 (a), (b) and (c) respectively.

Main pop.size	50:100
Nash-pop.size	20:50
Growth exponent	2
Elite groups size	2
Maximum time [s]	720
Generations evaluated	724

Table 3. Evolutionary parameters used for *MOpma343*.

	Shortest path found	Safest path found	Trade-off path
Length	1389.7938	1629.6626	1468.7392
Risk	1677	0	433

Table 4. Results of *MOpma343*.

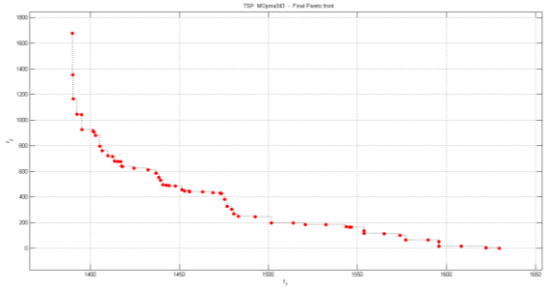

Figure 7. Final Pareto-front of *MOpma343*.

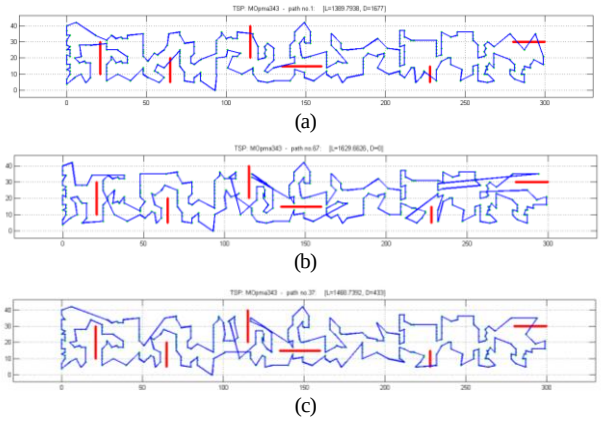

Figure 8. Shortest, safest and trade-off paths of *MOpma343*.

6.7 Discussion

A number of paths avoiding the clusters of dangerous points were found. This shows that the MO-TSP formulation presented in this work is a suitable model for representing environments with physical obstacles not to be crossed. Comparing the

length of the paths obtained by *Concorde* (1.3898) and Hybrid Game (1.3898) it was noticed that the Hybrid Game algorithm found paths shorter than the ones found by *Concorde*. These anomalous result (noticed for other bench cases as well) is due to the fact that *Concorde* evaluates distances rounding to the nearest integer [30].

Hence, the performance of the optimiser are satisfying. The Pareto-front found is well spread and a number of Pareto-optimal solutions found represent paths avoiding the obstacles, as demonstrated by the plot of the trade-off solution. Since the MO-TSP benchmark case evaluated are not present in literature, the true Pareto-front of the problem is unknown and any comparison with the Pareto-front found by the software is prevented.

7 Infrastructure inspection

7.1 Problem definition

One important application of the algorithm developed in this paper is the computation of the optimal trajectory needed to perform an aerial survey on large linear infrastructures like gas/oil ducts, power lines, railways or highways.

7.1.1 Digitalisation and discretisation

The map of the infrastructure was digitalised to be imported in Matlab and be effectively handled by the optimization algorithm. The best way to describe a linear infrastructure is to describe it as the combination of a number of lines whose extremes can belong to only one, two or more than two lines; in the first case, they are said *extreme waypoints*, in the second case *connection waypoints* and in the third case *nodes*. The lines to inspect are discretised as a list of intermediate waypoints between two extreme points of a line. The total number of the intermediate waypoints in which each line is discretised must be a compromise between the loss of detail due to the discretisation and the computational difficulty, which increases with the number of the waypoints.

The issue is solved by importing a high-density waypoint map into Matlab, but reducing the number of waypoints considered by the optimization algorithm by clustering some sequential waypoints of a global line together to form sub-lines whose extremes are connection waypoints of the lines of the global infrastructure. Such sub-lines are treated as unbreakable sub-paths, so that the goal of the optimisation algorithm is to find the best trajectory through those sub-lines.

This approach allows the optimisation algorithm to produce results only as feasible trajectories through the actual waypoints of the lines of the infrastructure. The discretization method described above can thus be applied to straight-line infrastructures, typically gas/oil pipelines and power lines, or curve-line infrastructures, i.e. railways, highways and even natural landscapes like rivers and coasts.

7.1.2 Off-service additional path

The goal of the optimisation algorithm developed in this paper is to compute a feasible trajectory above an infrastructure with an Unmanned Aerial Vehicle. Its trajectory is given by a composition of a certain number of sub-paths which can be generally classified as *on-service* and *off-service paths*. The first are the legs of the total trajectory in which the vehicle inspects the infrastructure, while the second ones are the legs in which all the mission sensors can be switched off because the vehicle is not overflying the infrastructure.

In this work, it is assumed is that the aircraft is able to follow any bend of the on-service path, as well as any trajectory made up by compositions of straight lines and arcs of circle tangent each other with constant turn radius R_{turn}. This hypothesis allows to model the additional paths as *Dubins curves* [6], i.e. sequences of a maximum of three straight or curve primitives.

7.2 Multi-objective analysis

The infrastructure inspection mission planning is handled as a MOOP with two objective functions $f_1=L$ and $f_2=R$, being the first one the length of the additional path and the second one the risk. Similarly to what was done for the MO-TSP, the scenario consists of a map of the infrastructure to inspect and a number of dangerous zones. The dangerous points are clustered into two-dimensional areas so that they can be used to reproduce realistic environments in which cities, restricted areas and/or bad-weather zones occur. The boundaries of the risk areas are imported by

the algorithm and the user is allowed to tune the density of dangerous points for each dangerous area that the algorithm generates. Once the density is set and the boundaries of the dangerous areas are imported, the algorithm automatically generates the grids of points which are taken into consideration during the optimisation process.

The solution implemented for the MO-TSP solver to hold down the computational cost needed for the evaluation of the cost functions is also adopted here: the length and the risk associated to all the possible additional trajectories are evaluated before the evolution process starts and stored into look-up tables.

7.3 Chromosome genes

As the problem was defined, any trajectory representing the solution of the inspection task path planning problem can be described as an ordered list of the sub-lines. Analogously to what has been done in the MO-TSP formulation, the sub-lines are numbered, so that any permutation of the vector [1,2,…,*V*-1,*V*], being *V* the number of sub-lines forming the global infrastructure, represents a candidate solution to the inspection problem.

While the cities in the MO-TSP had no dimensions, the sub-lines are defined by two different extreme waypoints, implying the need to specify which is the direction of travel over each sub-line. The technique implemented to univocally point out one of the eight feasible trajectories between any two sub-lines *i* and *j* is to define a flag binary variable which identifies the direction of each line. The eight feasible trajectories between the sub-lines *i* and *j* are thus defined by four variables: two indicating the order and two indicating the direction of inspection. Hence, the solution trajectory is fully described by the order of inspection of all the sub-lines and the direction of travelling direction over them. The chromosomes of the members involved in the evolutionary algorithm are made up of the composition of a permutation of [1,2,…,*V*-1,*V*] and the vector of the flag variables. Such a definition of the chromosomes implies that the dimension of the look-up tables is $[2V \times 2V]$.

7.4 Optimisation method

Thanks to the analogies between the inspection task problem and the MO-TSP, all the genetic operators implemented for the MO-TSP solver are used also for the infrastructure inspection path planner; however to take into account the direction flag variables, the genetic operators used for the optimisation of the inspection path are allowed to work only on the first part of the full chromosome, while the flag variables vector is changed in accordance to the result of the operator itself.

Thus, the functionality of all the genetic operators is basically unaffected by the presence of the direction flag variables; only the *sub-tour inversion* operator applied by the Nash-Players was strongly modified with respect to the one used for the MO-TSP solver. The *sub-tour inversion* operator used for the inspection path planning is indeed allowed to modify directly the flag variables vector. While for the MO-TSP the minimum sub-path which could be swapped was the sequence of two cities, for the infrastructure inspection path planning, any single sub-line has an orientation and can therefore be swapped too.

7.5 Queensland railway inspection

The infrastructure considered is the Queensland, Australia, railway network (Figure 9). In this scenario, the lines to inspect have variable lengths and curvatures and therefore they represent a complex linear infrastructure.

7.5.1 Environment definition

The map of the infrastructure was obtained by tracing a number of point-to-point paths matching all the available lines in the layer of the rail of Queensland on Google Earth. The coordinates (in the Earth-Centred–Earth-Fixed reference system) of the waypoints were imported in Matlab, where a reference system transformation (in accordance to the WGS84 ellipsoid model, [27]) was performed. The Cartesian coordinates of the waypoints were imported in *Rhinoceros 4.0*, where each railway line was rebuilt interpolating the available waypoints with polynomial curves of degree three. Such curves were finally split into 50-meter long segments and the equispaced intermediate waypoints exported in text files. At the end of this procedure, 251 files containing the Cartesian coordinates of 206180 waypoints were created.

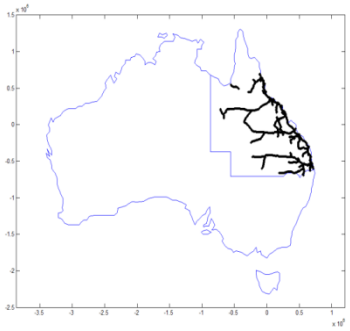

Figure 9. Overview of the railway network of Queensland, Australia.

In such a scenario, a number of risk-related restrictions can be specified:

- overflying cities for a long period may be unsafe due to airspace restrictions;
- flight in the outback far from airports or populated areas is also a consideration;
- according to forecasts, flying through some weather zones should be avoided.

The restrictions listed above are conveyed into the digital environment as risk-inducing clusters of dangerous areas located over the major cities of Queensland, throughout the outback and far from the railway. Different levels of risk are assigned to the bad-weather forecasts. Over this environment, a number of test cases (on different scales) were evaluated and two of them are presented here:

- *Test case 1*: inspection of the Brisbane, Gold Coast, Ipswich and Toowoomba (BGIT) area railway in good-weather conditions;
- *Test case 2*: inspection of the BGIT area railway with two zones of bad-weather zone are considered.

The density of the dangerous points used for the cities is 800km^{-2}, while for the bad-weather zones a density of 1000km^{-2} was used.

7.5.2 Results

The software was run for 1200 seconds; 1249 and 940 generations were evaluate for the *test case 1* and for the *test case 2* respectively. All the evaluations are performed using R_{turn}=150m as the constant turn radius of the aircraft and 2km for the range of risk influence. A trade-off path, as well as its position on the final Pareto-front are plotted and a summary of the cost functions for each trajectory is presented. Table 5 summarises the results obtained.

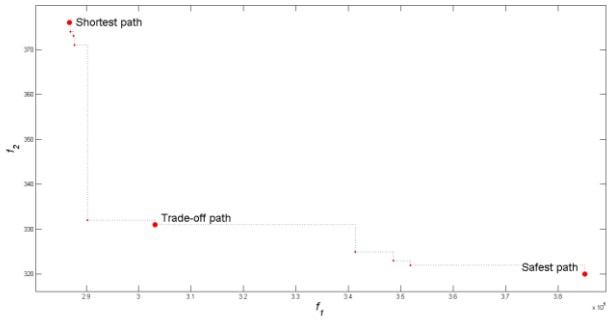

Figure 10. Pareto-front for test case 1.

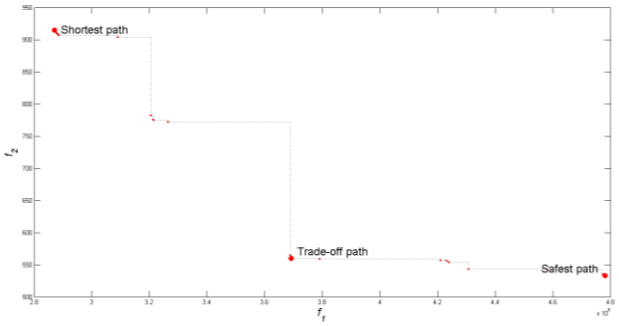

Figure 11. Pareto-front for test case 2.

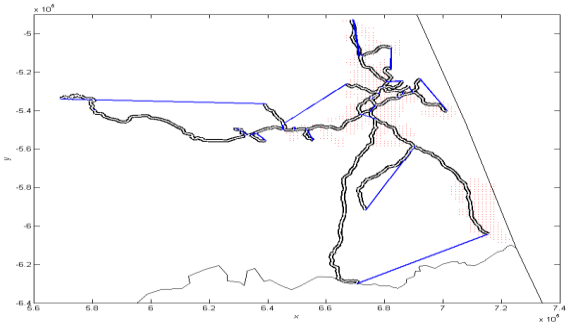

Figure 12. Intermediate path over the BGIT area railway in good-weather conditions.

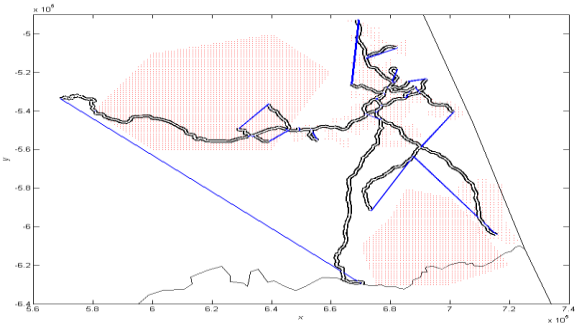

Figure 13. One trade-off path over the BGIT area railway in bad-weather conditions.

	Shortest path found		Safest path found		Trade-off path	
	length	risk	length	risk	length	risk
Test case 1	287km	376	385km	320	303km	331
Test case 2	287km	915	478km	533	369km	560

Table 5. Summary of the results for the test cases.

7.5.3 Discussion

The results obtained for the railway inspection task confirm the good performance of the optimisation algorithm displayed for the benchmark cases: in the real-world cases evaluated, the difference between the maximum and minimum of off-service lengths (ΔL) and risk (ΔR) found for are:

	Test case 1	*Test case 2*
ΔL	$98km$	$191km$
ΔR	56	382

Hence, the optimal trade-off paths chosen are representative of paths which allow a reduction on the risk by $[80\%, 93\%]$ with respect to the shortest path paying only $[16\%, 43\%]$ in terms of length.

Finally, the off-service to total length ratio is low for all the trade-off missions planned: the on-service lengths of the railway to inspect is 575km, hence the off-service to total length ratio metric is $[0.345, 0.390]$.

The shortest paths found in the two cases are characterised by the same length although the risk related to each one of them is significantly different. These results prove that the algorithm is able to adapt efficiently to the environment definition and the risk constraints considered.

8 Conclusions

This paper presented a novel formulation of the problem of the mission path planning of aerial survey tasks and an advanced Hybrid Game Evolutionary Algorithm was implemented for its solution. The mission path planning was handled as a MOOP where the objectives to minimise are the distance travelled and a risk function. These two objectives are defined for the purpose of taking into account any element of the real environment which can generally reduce the safety of the mission; thus, the scenarios considered include a number of dangerous zones representing restricted airspace, populated regions or bad weather areas. The risk function was defined in such a way that the risk induced by each dangerous zone is adjustable, thus allowing to define complex environment involving zones having different levels of risk.

The problem of the mission path planning was formulated as an enhanced MO version of the TSP. The technique proposed for the discretisation of the lines of the infrastructure is applicable to networks consisting of either straight and curve lines.

A specific chromosome definition was necessary as well as the implementation of genetic operators peculiar for combinatorial optimisation. In order to speed up the evolutionary process, a novel technique for the evaluation of the fitness of the individuals involved into the evolution process was developed: the length and the risk of the edges are read on look-up tables created in the pre-processing phase of the software.

Some of the test cases used to test the algorithm were presented. The software was finally applied to the mission path planning problem of the inspection task of the railway of Queensland, Australia. The results obtained were satisfying in all the cases: in all the cases evaluated, the software produced well spread sets of optimal trade-off solutions, proving that the software developed in this paper is a suitable and versatile tool for the offline mission path planning of aerial inspection tasks of large network infrastructures.

Even though the algorithm was initially meant as an offline path planner for UASs, a possible future development based on this work could be the implementation of a Graphical User Interface to run the software on the UAV Controller Station. The risk areas could be updated downloading the information of the on-board Weather Surveillance Radar and the software could be used as a real-time re-planner in case of sudden change of the environmental data. Future works will also consider the airfield locations and aircraft endurance to determine an optimal path into package missions.

Acknowledgements

The second and third authors would like to acknowledge the CRCSI.

References

[1] Coello Coello, C. A. (1999). A comprehensive survey of evolutionary-based multiobjective optimization. *Knowledge and Information Systems*, 1(3):269–308.

[2] Deb, K., (2001) Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Ltd.

[3] Jones, D. (2005) Power line inspection - a UAV concept. *The IEE Forum on Autonomous Systems 2005*, London.

[4] Deb, K. and Agrawal, R. B. (1995) Simulated binary crossover for continuous search space. *Complex Systems*, 9 115–148.

[5] Deb K, Pratap A, Agarwal S, and Meyarivan T (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation* 6:

182-197

[6] L. E. Dubins (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, *American Journal of Mathematics*, 79 (1957), 497-516.

[7] Hausamann, D., Zirnig, W., Schreier, G., Strobl, P., (2005) Monitoring of gas pipelines - a civil UAV application. *Aircraft Engineering and Aerospace Technology: An International Journal*, Volume 77, No. 5, pp. 352-360(9)

[8] T. Hu, et al. (2010). Variable population size and evolution acceleration: a case study with a parallel evolutionary algorithm. *Genetic Programming and Evolvable Machines.*

[9] Rappa, G. (2011). Hybrid Game Evolutionary Algorithm for Mission Path Planning of Aerial Survey Tasks. *MSc thesis.*

[10] Sefrioui, M., and Periaux, J., 2000, "Nash Genetic Algorithms: Examples and Applications," *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, IEEE, La Jolla, CA, pp. 509–516

[11] Van Veldhuizen, D A, Lamont, G B. (2000) Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation* 8(2)

[12] Zitzler, E., Deb, K., and Thiele, L. (2000) Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2) 173-195.

[13] Seshadri, A., "A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II".

[14] Gonzalez, L.F., Lee, D.S., Perlaux, J. "UAS Mission Path Planning Systems (MPPS) Using Hybrid-Game Coupled to Multi-Objective Optimizer", *Journal of Dynamic Systems, Measurements, and Control*, July 2010, Vol. 132

[15] Lee, D.S., Gonzalez, L.F. (2011). Efficient Hybrid-Game Strategies Coupled to Evolutionary Algorithms for Robust Multidisciplinary Design Optimization in Aerospace Engineering. *IEEE Transactions on Evolutionary Computation* 15(2):133-150

[16] Gonzalez, L.F., Lee, D.S, Walker, R.A. (2009) Optimal mission path planning (MPP) for an air sampling unmanned aerial system. *In Scheding, S. (Ed.) Proceedings of the 2009 Australasian Conference on Robotics & Automation, Australian Robotics & Automation Association, Sydney*, pp. 1-9.

[17] Michalewicz, Z. (1992). Genetic algorithms + data structures = evolution programs. *Artificial Intelligence. Springer-Verlag, New York.*

[18] Lee, D.S., Periaux, J., Gonzalez, L.F., Srinivas, K., Onate, E. (2010) Active flow control bump design using hybrid Nash-game coupled to evolutionary algorithms. *In Pereira, J.C.F., Sequeira, A., & Pereira, J.M.C. (Eds.) Proceedings of V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010, ECCOMAS CFD, Lisbon, Portugal*, pp. 1-14

[19] Deb, K., Pratap, A, Moitra, S. (2000) Mechanical Component Design for Multiple Objectives Using Elitist Non-dominated Sorting GA. In Proceedings of the Parallel Problem Solving from Nature VI Conference.

[20] Palli, N., Azram, S., McCluskey, P., and Sundararajan, R. (1999). An interactive multistage ϵ-inequality constraint method for multiple objectives decision making. *ASME Journal of Mechanical Design*, 120(4). 678–686.

[21] Kannan, B. K. and Kramer, S. N. (1995). An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design*, 116, 405–411

[22] Hahsler, M, Hornik, K, (2007) TSP - Infrastructure for the Traveling Salesperson Problem. *Journal of Statistical Software, vol. 23, issue 2.*

[23] Goldberg, D.E., (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley

[24] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. VanGucht, (1985) "Genetic algorithms for the traveling salesman problem," *in Proceedings of International Conference Genetic Algorithms and Their Applications*, pp. 160–168

[25] Whitley, D, Starkweather, T, Fuquay, D'A, (1991) Scheduling Probems and Travelling Salesman: The Genetic Edge Recombination Operator, *in Davis, L (Editor), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York*, pp.133-140

[26] Mathworks. (2011). *User's Guide* (r2011b)

[27] Etkin, B, (1196) *Dynamics of Atmospheric Flight, Third Edition*, John Wiley&Sons, INC, New York

[28] http://www.tsp.gatech.edu/vlsi/index.html

[29] Wang, J. F., and Périaux, J., 2001, "Multi-Point Optimization Using GAs and Nash/Stackelberg Games for High Lift Multi-Airfoil Design in Aerodynamics" *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, Vol. 1, Issue 2001, pp. 552–559.

[30] Fekete, S. P., Meijer, H., Rohe, A., Tietze, W., 2001, Solving a "Hard" Problem to Approximate an "Easy" One: Heuristics for Maximum Matchings and Maximum Travelling Salesman Problems, in *Algorithm Engineering and Experimentation, Buchsbaum, A. L., Snoeyink, J. (Eds.), LNCS* 2153, pp. 1-16

Copyright Statement