Swarm Guidance using a Multi-Objective Co-evolutionary On-Line Evolutionary Algorithm

Evan J. Hughes
Department of Aerospace, Power and Sensors,
Cranfield University,
Royal Military College of Science,
Shrivenham, Swindon, England. SN6 8LA
ejhughes@iee.org

***Abstract*— This paper details the application of a Multi-Objective Cooperative Co-evolutionary On-Line EA (MOCCOLEA) to the guidance of a swarm of multiple missiles, against multiple targets. The multi-objective algorithm is used to develop a dynamic objective front which is used to trade the spatial distribution of missiles at impact, against the time to impact.**

Each missile optimises its own performance, based on limited information of the current intended actions of the other missiles. The decision making process is thus distributed between the missiles giving distributed coordination.

Results demonstrate the algorithm can form effective leaderless distributed control for multiple missiles against multiple targets in noisy and environments.

I. Introduction

Current defence procurement strategies are emphasising maximising performance while minimising cost. Traditionally, When engaging fast, manoeuvreable targets, single missiles with highly accurate and therefore expensive seekers have been employed. This paper discusses an approach where multiple missiles are used, each with a low-cost seeker. The aim is to use sophisticated guidance and control algorithms to offset the seeker performance short-fall.

The multiple low cost missiles are flown as a self coordinating swarm in an attempt to increase reliability and overall kill probability. Previous work [1], [2] has demonstrated that an On-Line Evolutionary Algorithm (OLEA) can be used to form the cooperative strategy of a command guided system against a single target.

In evolutionary guidance, the platform is first flown via a sequence of one or more points in space, before flying towards a predicted impact with the target, or a required rendezvous point. The points in space are evolved to generate a flight profile that is an optimal solution to a set of objectives and constraints. With multiple platforms, the flight profiles can be evolved simultaneously, each flight profile being evolved while accounting for the intended flight paths of the other missiles.

With the evolutionary guidance approach, for most of the engagement there is no direct, deterministic path between the sensors and the autopilot. Thus the initial stages of the flight path can be independent of the target position and motion, allowing different trajectories to be generated easily.

In the previous research, the swarm was developed from four missiles engaging a single moving target. The work used a command guided swarm with the processing for the on-line EA all performed in a ground-based central processor. Steering demands were communicated up to the missiles, with missile seeker information communicated back. In the command guided scenario, each chromosome of the OLEA describes a complete guidance scenario for all the missiles. All of the missiles are modelled together and the objectives can be assessed based on perfect knowledge of the anticipated behaviour of the missiles'.

This paper details the significant extension of the existing work to:

1) Distributed cooperative coevolution with local processing on each missile giving asynchronous parallel operation, rather than a command guided system with a single processing system 'on the ground', and
2) multiple missiles against multiple targets,
3) multi-objective operation in a dynamic co-evolutionary environment.

The MOCCOLEA allows complex swarm behaviours to emerge, yet only requiring low bandwidth communications, for example in this paper each missile broadcasts, ten times per second, the estimated time of impact, estimated impact angle, index of target being engaged and estimated speed at impact. The asynchronous nature of the parallel processing system makes the swarm robust to intermittent communication. With each missile carrying its own processing, heterogeneous swarms can be formed very easily using missiles with different performance capabilities, as the missiles only need detailed models of themselves and not the other members of the swarm.

Section II describes the missile and guidance model used in this paper, section III describes the structure and application of the MOCCOLEA, section IV describes example scenarios and demonstrates the behaviour of the MOCCOLEA swarm guidance and section V concludes.

II. Missile Model

A. Missile Model

The missile model used in this paper is simple, but may be representative of a low-cost missile. A simple fixed wide-field-of-view Imaging Infra-Red seeker is assumed, and simple bang-bang control, creating a non-linear system. The scenario is that the missiles will be gun launched with a muzzle velocity $V \approx 600m/s$ and have no propulsion of their own. Drag is assumed along with a velocity dependent drag coefficient. The magnitude of the lateral acceleration, a_l, when the fin is not in the zero position, is given by (1), where L is a constant coefficient of approximately $L \approx 98/330$ giving $10g$ lateral acceleration at a speed of Mach 1.

$$a_l = L|V_f| \tag{1}$$

It is assumed that although the drag is proportional to velocity squared, the drag coefficient decreases linearly with velocity. The resulting longitudinal acceleration, a_f, is given by (2), where C is a constant coefficient of approximately $C \approx 0.005$, and $L_d \approx 0.1$ is a lift-drag coupling coefficient.

$$a_f = -C|V_f| - |a_l|L_d \tag{2}$$

To keep processing speed high, a simple fixed step integration process was used with a time step of 0.1 seconds for the missile simulations, and a 0.2 second step for the 'internal' trajectory predictions. If the large step size is used with conventional numerical integration of the lateral and longitudinal acceleration, a rapid build up of numerical errors occurs in the velocity and position, so (3) is used to try to mitigate many large errors by processing using turn rates rather than vector addition.

$$\begin{aligned} |V_{t+1}| &= |V_t| + a_f \delta t \\ \angle V_{t+1} &= \angle V_t + \frac{a_l}{|V_t|}\delta t \\ P_{t+1} &= P_t + V_t \delta t \end{aligned} \tag{3}$$

B. *Guidance Heuristics*

As the missile only has a simple Infra-Red seeker, only bearing to target can be generated, and so for simplicity, a *pursuit* guidance system has been used [3]. Here the missile simply steers towards the current target position, given the bearing information. When under the control of the MOCCOLEA, The missile will steer towards the way-point until the distance to the way-point increases. The missile will then steer towards the current selected target. If the internal missile model decides that if the missile steers towards the way-point, and then cannot possibly hit the target, the solution is marked as infeasible. If no solution is marked as feasible, the missile simply heads directly towards the target. Although the missile is assumed to be estimating target velocity and acceleration and so could easily use more complex laws such as Augmented Proportional Navigation, pursuit guidance often results in a near tail-chase scenario and as such, the MOCCOLEA will have to work much harder in order to provide any useful shaping for the trajectory.

The control surface is assumed to be a fin with bang-bang control. The fin is assumed to have a zero latax central position and only moves if the missile's forward velocity vector is more than 1° from the estimated line-of sight to the target (or way-point). The fin then pulls maximum available latax (1) until the line-of-sight to the target is within 1° of the velocity vector. The line-of-sight to the 'target' may not be equal to the the true seeker pointing angle if the missile is tracking one target but engaging another, or flying towards the way-point.

III. On-Line Evolutionary Algorithm (OLEA)

A. *Introduction*

Evolutionary Algorithms are optimisation procedures which operate over a number of cycles (generations) and are designed to mimic the natural selection process through evolution and survival of the fittest [4]. A *population* of *M* possible solutions is maintained by the algorithm. Each potential solution is represented by one *chromosome*. This is the genetic description of the solution and may be broken into *n* sections called *genes*. Each gene represents a single parameter in the problem, therefore a problem that has eight unknowns, for example, would require a chromosome with eight genes to describe it. The chromosome could be represented as vector $\overrightarrow{Q}$ where the elements of the vector are the genes. Each trial solution forms a single point in the parameter space.

The three simple operations found in nature: natural selection, mating and mutation are used to generate new chromosomes and therefore new potential solutions. Each chromosome is evaluated at every generation using an *objective function* that is able to distinguish good solutions from bad ones and the chromosome's performance is assigned a score. With each new generation, some of the old chromosomes are removed to make room for the new improved offspring. Despite being very simple to code, requiring no directional or derivative information from the objective function and being capable of handling a large number of parameters simultaneously, Evolutionary Algorithms can achieve excellent results.

B. *OLEA structure*

The basic OLEA is described in the algorithm:

1) $\chi = U(^L\chi, {^H\chi})$
2) $\mathcal{O}_t = F(\chi_t, Z_t)$
3) $Y_t = D(\chi_t, \mathcal{O}_t, Z_t)$
4) $\chi'_t = S(\chi_t, \mathcal{O}_t)$
5) $\chi_{t+1} = H(\chi'_t, Z_t)$
6) $t = t + 1$, if not end, repeat from step 2.

In the algorithm, χ_t represents the array of chromosomes at time t. In systems where the chromosome is a fixed length vector, χ would be a matrix with one row for each member of the population and one column for each gene. The function $U()$ is a random distribution (usually uniform) between the upper and lower bounds of each gene value. Thus the first line of the algorithm generates a random population for the OLEA.

The array of objective values and constraint satisfaction indices at time t is represented by $\mathcal{O}_t$. The objectives are calculated based on the model described by the function $F()$, the population χ_t and the observation of the environment Z_t. In general there will be multiple objectives calculated for each member of the population. The algorithm makes the implicit assumption that the measurement Z_t and the function $F()$ are subject to noise and uncertainty.

The OLEA output at time t is denoted by Y_t and is a function of the chromosomes χ_t, the objective and constraint values $\mathcal{O}_t$, the observed state of the environment Z_t and the model function $D()$. In the multiple missile system, the output is the current best way-point and target index. In a system with multiple objectives that evolves a Pareto set, the function $D()$ is the decision maker that must choose one solution from the set to use as the current operating point.

The population update is performed by function $S()$. The function must perform the crossover, mutation and selection process, providing the selective pressure to drive the population towards a good solution. In a multi-objective system, the function must generate and maintain an estimate of interesting regions of the Pareto set. The output is a new population to evaluate in the next generation.

The fifth step uses $H()$ to create the population ready for the next generation by providing a state update if necessary. For example, if position and velocity and acceleration were being estimated from acceleration measurements, $H()$ would update the position from velocity, and the velocity from acceleration ready for the next iteration. Often $H() = 1$ is used where the chromosome remains unchanged between generations. The function $H()$ may also be a local optimisation process to allow hybrid and memetic algorithms to be created, where the chromosome is modified based on the results of a local optimisation process.

The algorithm is repeated, usually with one generation per time increment, until the estimated parameters are no longer required. The framework allows on-line, multiobjective, multi-species, parallel, memetic and hybrid algorithms, as well as very simple evolutionary algorithms to be represented with a wide range of crossover and mutation strategies.

C. Multi-Objective Cooperative Coevolutionary OLEA

With parallel cooperative coevolution, each missile runs its own OLEA. The chromosome within each missile is of the form $[x \quad y \quad z \quad T]$, where the vector $[x \quad y \quad z]$ is the location of the way-point for the missile and T is the index of the target to aim at after the way-point is passed. The function $F()$ of the OLEA contains a simulation of the dynamic behaviour of the missile, given the input chromosome set χ_t and environmental state Z_t. The environmental state consists of: the current estimated missile position, velocity and seeker state; the current estimate of the parameters of the target of interest; and a set of information from the other missiles indicating their current intended action. Each missile may have its own unique $F()$, allowing a set of heterogeneous missiles to be used to form the swarm, each with a very different dynamic model. It is assumed that a separate data fusion process is used to capture predicted target data from other missiles, and possibly other sensors, to create a unified air-picture, allowing targets to be identified uniquely by an index number.

Each missile broadcasts its current intended action to the other missiles in the swarm ten times each second. The data rate may be reduced if the action has not changed, but at the minimum, a 'keep alive' signal should be sent to indicate the missile is part of the swarm. The data transmitted is the vector $[M \quad \alpha_i \quad t_i \quad T_i \quad |V_i| \quad L_i \quad \delta d_i]$, where M is the missile identification number, α_i is the predicted impact angle, t_i is the predicted impact time, T_i is the index of the target being engaged, $|v_i|$ is the predicted impact speed, L_i is the predicted lateral acceleration at impact and δd_i is the predicted miss distance.

Each missile collects the broadcast information from the other members of the swarm and bases the score of each chromosome on the predicted collective behaviour of the entire swarm. Objectives such as *minimise spread of impact times, minimise longest engagement time, minimise worst latax at impact, maximise spread of impact angles for each target* etc. may be used to govern the behaviour of the swarm. Penalties may also be applied if the missile seeker is locked onto one target, but it intends to engage another. As the seeker will have to break-lock and reacquire the new target once the way-point is passed. There is a risk that the missile will not be able to acquire the new target in time and as such is undesirable behaviour that should be minimised, but not discouraged entirely. If all communication is lost with the other missiles, the platform defaults to heading towards the target that the seeker is currently locked to.

The function $D()$ which selects the solution to implement is simply to take the best performing chromosome and use it to describe the way-point and target to aim at. If the way-point is extreme and the prediction indicates that the missile cannot hit the target it it goes via the way-point, the way-point is ignored and the missile aims at the selected target directly. This default behaviour guarantees a minimum level of performance.

The missile demand is then derived by the guidance law steering towards the way-point or target if the way-point is not achievable. The state update function $H()$ is unity in this application as the way-point and target index are relative to fixed Earth coordinates, not to the missile body and orientation. In the initial population, $U()$ is a uniform random distribution within a region local to the estimated engagement envelope. A population of 100 has been used in all the simulations. The function $S()$ selects the best 20 solutions for breeding. An approach based on Differential Evolution [5] is used to generate 80 new solutions to replace the section of the population that was removed.

1) Differential Evolution: Differential Evolution (DE) is an evolutionary technique that uses mutations that are related to the current spatial distribution of the population. The algorithm generates new chromosomes by adding the weighted difference between two chromosomes to a third chromosome. At each generation, for each member of the parent population, a new chromosome is generated. Elements of this new chromosome are then crossed with the parent chromosome to generate the child chromosome.

The size and direction of the difference between any pair of chromosomes is determined by the overall spread of the current population. Thus the DE algorithm self adapts to the fitness landscape, reducing the size of the mutations automatically as the search converges. In this way, no separate probability distribution has to be used for mutation which makes the scheme completely self-organising.

The trial chromosome $\vec{Q}_t$ may be described as in (4).

$$\vec{Q}_t = F(\vec{Q}_a - \vec{Q}_b) + \vec{Q}_c \tag{4}$$

Where chromosomes $\vec{Q}_a$, $\vec{Q}_b$ & $\vec{Q}_c$ are chosen from the population without replacement and F is a scaling factor.

The crossover process is controlled by a crossover parameter C. The crossover region begins at a randomly chosen parameter in the chromosome and then a segment of length G genes is copied from $\vec{Q}_t$ to the parent chromosome to create the child chromosome. If the segment is longer than the remaining length of the chromosome, the segment is wrapped to the beginning of the chromosome. The length G is chosen probabilistically and is given by (5).

$$P(G \geq k) = (C)^{k-1}, k > 0 \tag{5}$$

In general, the scaling parameter F and the crossover parameter C lie in the range $[0.5, 1]$. Small values of F mean that the population spread reduces faster implying faster convergence. However the faster convergence is more likely to result in the algorithm converging quickly at a *local* minima, rather than the *global* minima. We have found that values of $F = 0.7$ and $C = 0.5$ are suitable for this application.

D. Objective Value Calculation - MOPSEA

Multiple Objective Probabilistic Sampling Evolutionary Algorithm (MOPSEA) [6] has been used to rank the multiple objectives and constraints in order to minimise the objectives. MOPSEA is designed explicitly for developing Pareto sets in noisy environments and allows an estimate of the noise distribution to be included in the Pareto ranking in order to maintain the noisy Pareto surface. Like many other Pareto based ranking methods, MOPSEA is not well suited for many-objective problems where the Pareto dominance mechanism becomes the minor drive for evolution [7]. Thus only two objectives have been used in this example. To allow the objective surface to adapt, no elitism is used and the entire population is re-ranked each generation.

Constrained solutions are ranked lower than unconstrained solutions, and in an order based on the degree of constraint. Any ‘ties’ in constraint value are broken based on Pareto dominance. Thus the constraint rank is enforced over the Pareto rank.

Two objectives are presented in this paper:

1) Maximise the spread of impact angles over all missiles aiming at my target; if only one missile – one target, minimise the impact angle to force a head-on-engagement.
2) minimise the remaining flight time to impact.

The two objectives place conflicting requirements on the missile, leading to a trade-off surface. The solution is considered totally constrained if $\delta d_i > 50$ (i.e. definitely a miss). The solution is partially constrained if the missile is not targeted at the target it was tracking on the last time step. The solution is further partially constrained if the target aimed at, in conjunction with the information from the other missiles, does not cause all available targets to be covered. The more targets that could be engaged but are left uncovered, the larger the constraint imposed.

The function $D()$ is the decision maker function. In this paper the decision maker is a rule based system that applies additional preferences to the best 20 members of the population in order to derive a single solution to use as the waypoint. The preference rules are (applied in order, reducing the available set at each step):

1) If all the solutions are fully constrained, fly straight at the target, else remove all the solutions that are fully constrained.
2) If there is at least 1 solution that covers all the available targets with a missile, remove all solutions that do not.
3) If there is at least 1 solution that is heading towards the target that the seeker is locked onto, remove any that are tracking a different target.
4) Remove any solutions that are dominated (based on conventional non-domination rather than noisy classification)
5) Take the solution from the remaining set whose waypoint is closest to that used last.

Each missile makes its own decision based on the limited information passed from the other members of the swarm.

IV. Example Application

The example simulation has three missiles engaging two targets. Figure 1 shows the situation just after launch. The missiles are launched from the lower edge of the picture and are represented by crosses, circles and diamonds along the trajectory lines. The isolated symbols represent the top 20 waypoints of each missile, and the stars are the current location of the targets. The dotted lines are the predicted trajectories of the missiles and targets, the solid lines indicate the trajectories actually flown. Figure 2 shows the noisy objective surface at the same time instant (using the same symbol key as the missiles). It is clear that there is little convergence so far.

The data fused sensor system of the missiles’ was simulated by generating perfect knowledge of the target location, velocity and acceleration, and then corrupting the data with Gaussian noise. Thus the forward predictions of the missiles vary considerably with each time step. The aerodynamic coefficients used in each of the local simulation models were corrupted by up to 10% from the true values used in the main missile simulation, thus introducing further errors and uncertainty into the predicted impact conditions.

Figures 3, 5, 7 & 9 and 4, 6, 8 & 10 show the scenario and objectives at stages through the engagement. It is clear that the missiles are engaging both targets. The intercept angles are not as wide as we would have liked, it appears this is partly due to the range of the engagement as the missiles are stretched to near their maximum limit, and partly due to only

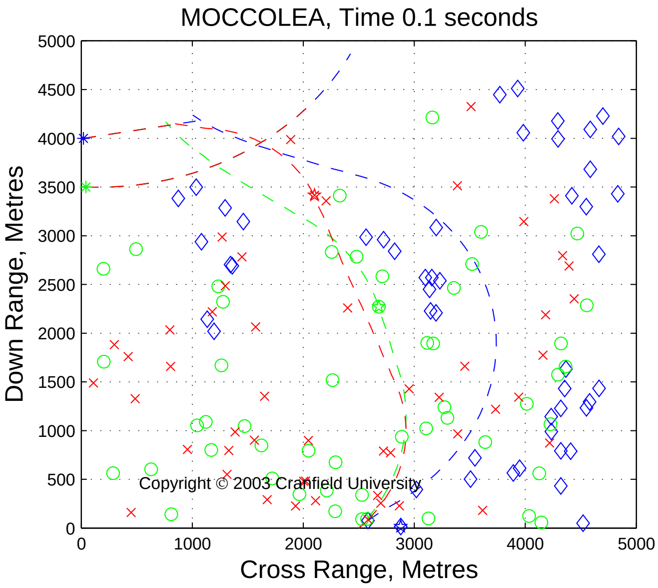

Fig. 1. Scenario just after launch

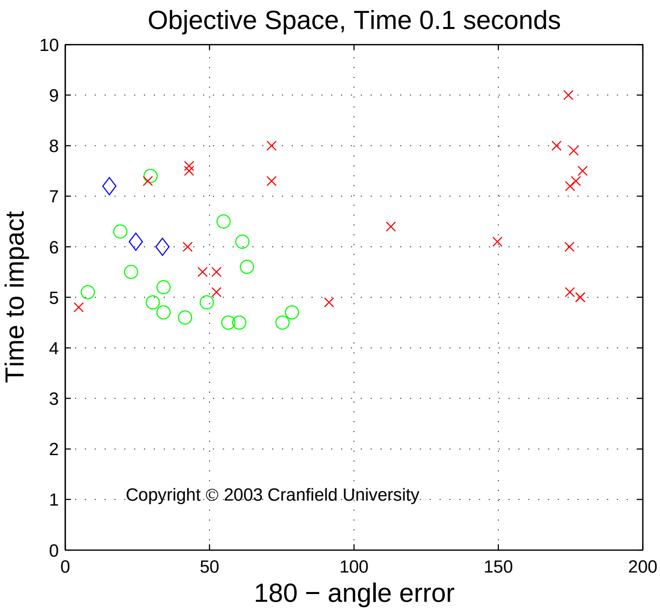

Fig. 2. Objectives just after launch

using pursuit guidance. The range limit and pursuit guidance is the reason for the second (turning) target being engaged as a tail chase, rather than head on.

It should also be noted that in the initial phase of the engagement, all the missiles are targeting the same target. Quickly, the missiles represented by a circle and a diamond aim at the upper target, with the missile represented by crosses aimed at the lower target. This situation persists for a short while, but by 3 seconds, the missiles represented by the diamond and the cross have swapped. By 4.5 seconds, the diamond and the circle have swapped, remaining in this formation to impact. The spread of possible way-points is quite wide for all the missiles throughout the engagement: a product of the multi-objective ranking process.

These results show that even in adverse scenarios, the MOCCOLEA is able to adapt to find a workable solution, even if it is not wholly desirable.

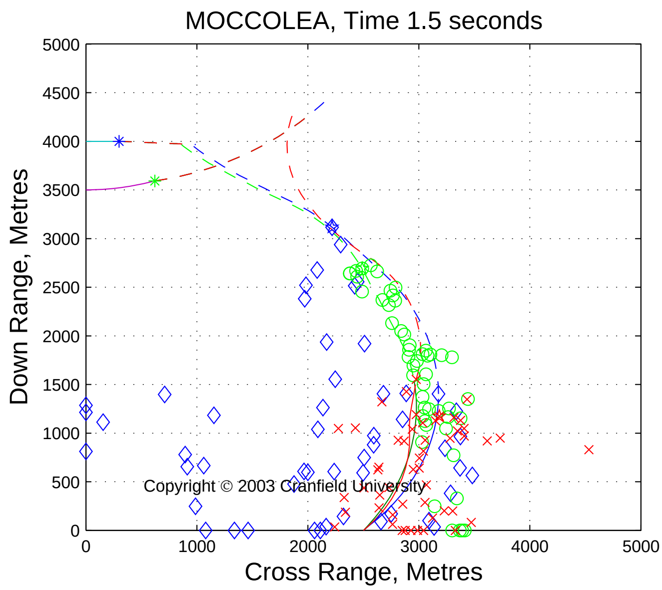

Fig. 3. Scenario during engagement at 1.5 seconds

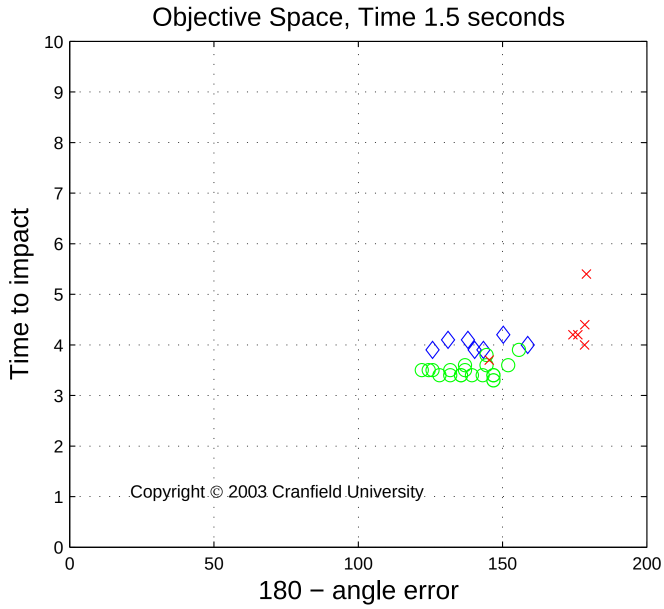

Fig. 4. Scenario during engagement at 1.5 seconds

It has been observed that although for each missile the noisy Pareto set is not very crisp or widespread, there are often two

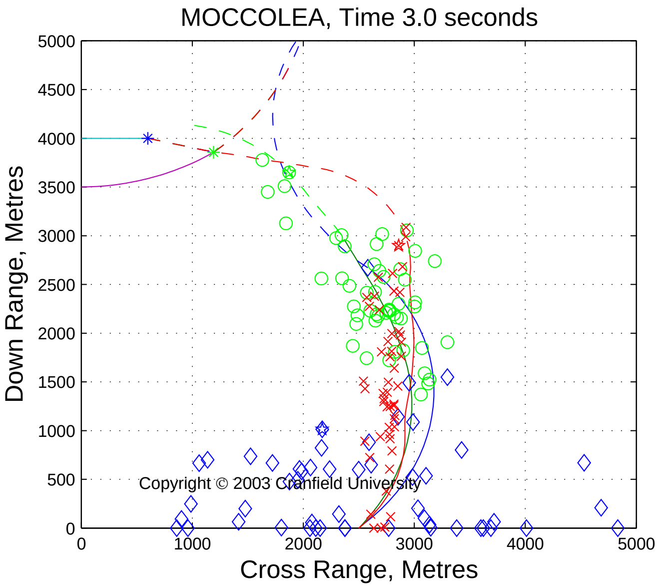

Fig. 5. Scenario during engagement at 3 seconds

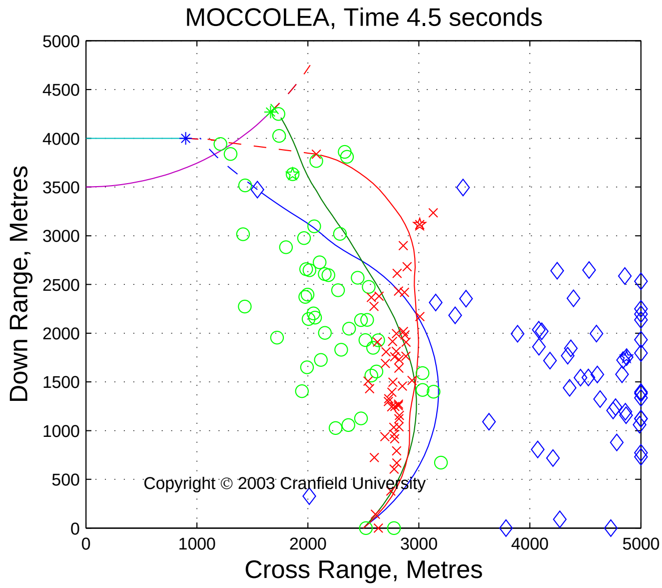

Fig. 7. Scenario during engagement at 4.5 seconds

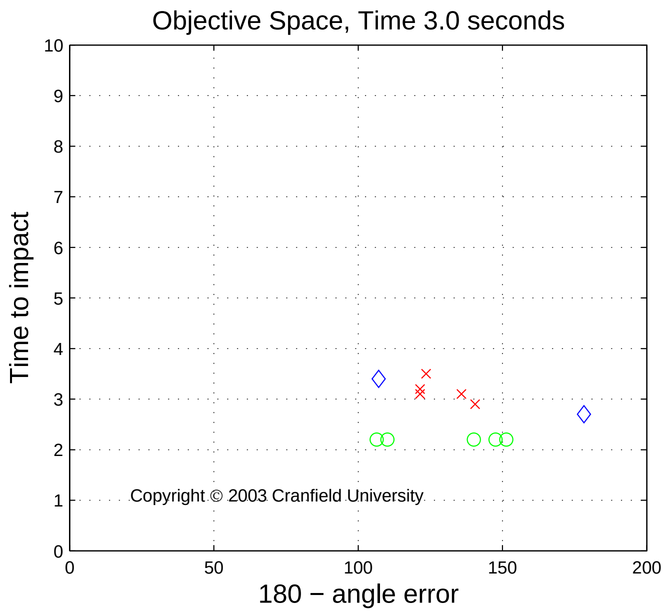

Fig. 6. Scenario during engagement at 3 seconds

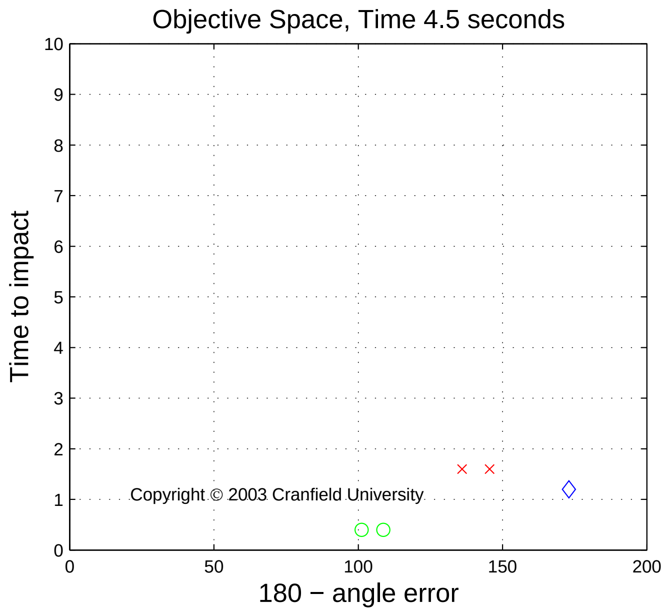

Fig. 8. Scenario during engagement at 4.5 seconds

regions of valid solutions existing in the objective space, one for each target. By maintaining solutions from each region, the missile is able to swap targets very rapidly and still have a near optimal guidance solution. Figure 4 shows this behaviour with a set of crosses to the right, and another set amongst the circles and diamonds. This bi-modal behaviour often results in one of the sets being dominated and being gradually removed from the population. Having the decision making process gradually refine the set of *interesting* targets and then selecting the way-point closest to the last point, causes the trajectories to be quite smooth with little switching of the actuator. In practical systems, low actuator switching reduces power consumption and improves actuator reliability.

V. Conclusions

Even with highly non-linear systems subject to high levels of noise and uncertainty, the MOCCOLEA approach provides a comprehensive framework allowing multiple missiles to coordinate attacks on single or multiple targets. This shows clearly that the method is tolerant of complexity and many sources of error.

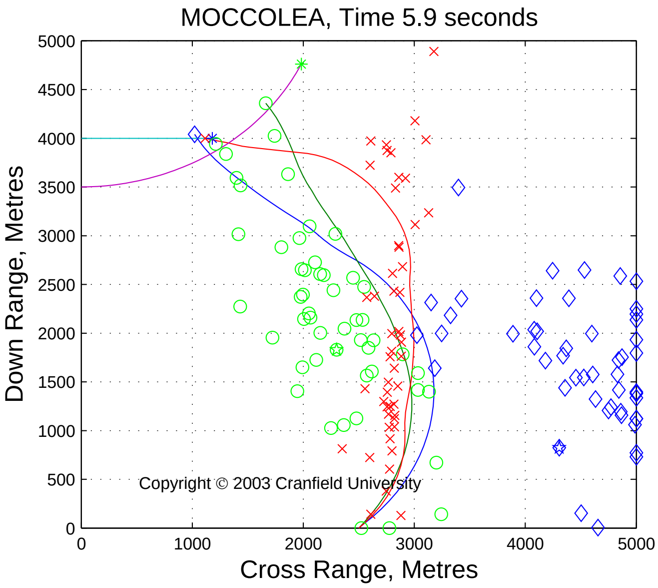

Fig. 9. Final trajectories at 5.9 seconds

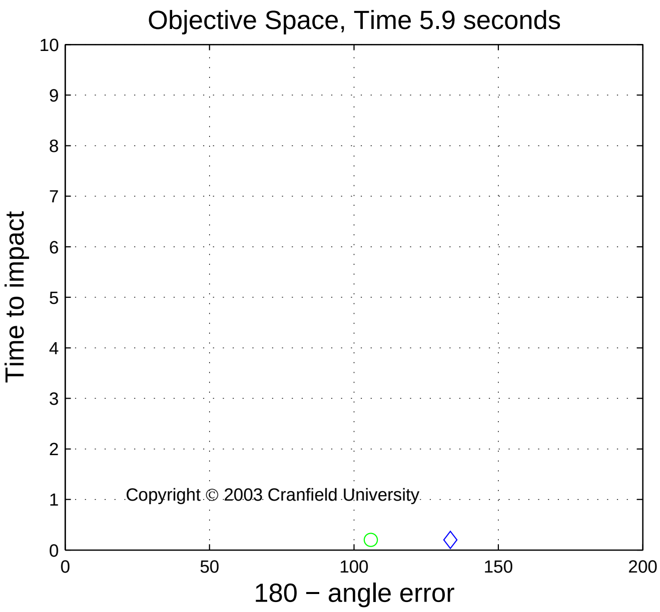

Fig. 10. Final trajectories at 5.9 seconds

The parallel distributed processing allows todays high-speed processors to be exploited to the full. The larger the population size used, and the more accurate the missile simulations, the better the guidance will perform. As each missile only has to predict its own behaviour, swarms of heterogeneous swarms can be formed very easily.

The use of the multi-objective formulation allows a spectrum of possible guidance solutions to exist within the population. The on-line decision making process is then used to select one single solution as the operating point for the current generation. This decision making framework is unusual as many decisions need to be made every second, rather than one single decision overall. The results show that rapid on-line decision making is indeed possible, but not a trivial problem.

References

[1] Evan J. Hughes. Evolutionary guidance for multiple missiles. In *IFAC World Congress Conference*, page Paper 1801, Barcelona, Spain, 21-26 July 2002.

[2] Evan J. Hughes. Multi-objective evolutionary guidance for swarms. In *World Congress on Computational Intelligence*, pages 1127–1132, Honolulu, Hawaii, May 12-17 2002. IEEE.

[3] Paul Zarchan. *Tactical and Strategic Missile Guidance*. American Institute of Aeronautics and Astronautics Inc., 3rd edition, 1997. ISBN 1-56347-254-6.

[4] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[5] Rainer Storn and Kenneth Price. Differential Evolution- a simple and effective adaptive sche me for global optimization over continuous spaces. http://http.ICSI.Berkeley.edu/storn/code.html. last accessed Oct 2003.

[6] Evan J. Hughes. Constraint handling with uncertain and noisy multi-objective e volution. In *Congress on Evolutionary Computation*, volume 2, pages 963–970. IEEE, May 27–30 2001.

[7] R. C. Purshouse and P. J. Fleming. Evolutionary many-objective optimisation: An exploratory analysis. In *Congress on Computational Intelligence*, Canberra, Austrailia, 8-12 December 2003. IEEE. Paper 1390.