# Multi Sensor Data Fusion in Electronic Warfare Systems

Sergio Tortora, Andrea Sindico, Marco Guerriero, Gaetano Severino, Vittorio Rossi, Gabriele Falco
Elettronica S.p.A. Via Tiburtina Valeria Km. 13,700 Roma (Italy)

***Abstract* — Modern Electronic Warfare Systems have to deal with information retrieved from a number of heterogeneous sensors possibly belonging to different cooperating platforms. Sensor Data Fusion algorithms are therefore required to properly handle such concern. In this paper we describe a process of sensor data fusion introducing a new idea of combining a mathematical programming algorithm with a heuristic search techniques based on genetic algorithms.**

***Keywords* — Electronic Warfare, Multi Sensor Data Fusion**

## I. INTRODUCTION

Electronic Warfare [1] is a military discipline aimed at controlling the use of the electromagnetic spectrum (EM). It consists of two major subdivisions: *Electronic Attacks* (EA) and *Electronic Support* (ES) *Measures*. EA consists of all those techniques having the purpose to degrade the efficiency of the enemy to exploit the EM spectrum. ES is instead aimed at searching for, intercepting, identifying and locating source of intentional and unintentional radiated EM energy for the purpose of threat recognition, targeting and planning. ES data can therefore be used to produce signals intelligence, to provide targeting for electronic or destructive attacks and to produce measurement and signature intelligence. ES operations are performed by means of the use of passive sensors as: *Radar Warning Receivers* (RWR) for the detection of Radar threat emissions; *Laser Warning Systems* (LWS) for the detection of Laser threat emissions and *Missile Warning Systems* (MWS) for the detection of emissions in the IR spectrum.

Modern platforms (i.e. aircraft, helicopters, ships, etc.) are nowadays equipped with several sensors, of the aforementioned classes, ensuring detection for any kind of waveform radiated by the enemy. However the simultaneously presence of several sensors, either of the same or different type, that can be even be overlapped in their detection areas, produce a huge amount of information requiring a proper synthesis in order to be exploited. Such synthesis, also called *Situation Assessment*, was commonly demanded to the pilot that, continuously looking at different displays, typically one for each sensor, had the duty to understand and assess what entities refer to emissions coming from the same platform. This kind of effort reduces the pilot's time to take and actuate decisions. Things are even worst in the case of net-centric operations where data retrieved from on-board resources have to be compared and shared with those retrieved from other cooperating platforms.

In this paper we present an architecture of sensor data fusion techniques aimed at providing a continuously updated situation assessment from the data obtained from different sensors of an EW suite. This architecture represents a simplified version of a core component of an Elettronica S.p.A. product named EW-Manager. This is aimed at being an integrator of all the information provided by on-board and distributed sensors enabling sensor data fusion also in the case network centric operations.

## II. RELATED WORKS

Over the past two decades, significant attention has been focused on multisensory data fusion for both military and non-military applications [2][3][4]. Data fusion techniques combine data from multiple sensors, either of the same or different types in order to achieve more specific inferences than could be achieved by using sensors independently.

The Joint Directors of Laboratories (JDL) Data Fusion Working Group, established in 1986 by the US Department of Defence (DoD), has codified in [2] a process model for data fusion and a data fusion lexicon. Intended to be very general and useful across multiple application areas, the JDL process identifies: functions, categories of techniques and specific techniques applicable to data fusion.

It consists of four key sub-processes:

1. *Object Refinement*: it is aimed at fusing sensors' data in order to determine the identity and other attributes of entities and also to build tracks representing them. A track is usually directly based on detections of an entity, but can also be indirectly based on detecting its actions. The product from this level is called the *Situation Picture*;
2. *Situation Refinement*: dynamically attempts to develop a description of current relationships among tracks in the context of their environment;
3. *Threat Refinement*: projects the current situation into the future to draw inferences about enemy threats, friend and foe vulnerabilities, and opportunities for operations;
4. *Process Refinement*: it is a meta-process that monitors the overall data fusion process to assess and improve real-time system performance. This is an element of resource management.

A knowledge base represented by a *support database* is also required. It contains *a priori* information aimed at supporting the data fusion process. Results obtained as output

of the process are maintained in a dedicate data base called Fusion Database.

## III. The EW-Manager Data Fusion Architecture

The EW-Manager data fusion component, from hereafter referred to as EWMDF in order to distinguish it from the generic sensor data fusion concept (DF), has an architecture compliant to the JDL process model and depicted in Fig. 1.

The *support database* entity is represented, in the EWMDF, by a *Mission Library*. The EWM Mission Library contains:

- a set of data representing the entities (i.e. waveforms and platforms) the pilot is supposed to meet during a mission. Waveforms are described by means of a numeric representation of their fundamental characteristics (i.e. emitted RF, PRI, PW, etc.). A *waveform model* is always defined as being associated to an *active sensor* installed on a *platform*. Each platform is also described by means of its possible behaviour. The *behaviour model* is described as a state machine of the different functions a platform can perform through its sensors and related waveforms. All these data are required to properly cue the data fusion algorithms.
- a set of models describing the measurement characteristics of the available sensors;
- a set of objective functions exploitable to properly cue the EWMDF behaviour.

The Fusion Database is instead represented by a track file, called *Fused Track File*. The EWMDF uses it in order to maintain:

- the *sensor tracks*: the original track data retrieved from the available sensors;
- the *domain tracks*: tracks obtained as output from the *Object Refinement* phase;
- the *platform tracks*: tracks obtained as output from the *Situation Refinement* phase.

With respect to the JDL DF process, the *Object Refinement* phase has been implemented, in the EWMDF, by means of four steps:

- The *Spatial Correlation*: is aimed at identifying clusters of tracks compatible in direction of arrival (DOA) and range (when this information is available). The process is described in section A. The resulting clusters, also called compatibility sets, define the sets of tracks that can be object of further data fusion analysis. Tracks belonging to different compatibility sets are not allowed to be fused together during the process.
- *Feature Correlation* and *Identification*: The *Feature Correlation* is responsible to fuse different tracks, belonging to the same compatibility sets, of the same type (i.e. RF, Laser, IR, etc.) but provided by different sensors (i.e. two or more RWR, etc.). The output of this process is a new set of tracks, called *domain tracks*: each domain track contains the best measurements of the involved sensors. The *Identification* process takes as input the domain tracks and compares them with the *waveform models* described in the *Mission Library*. The process aims to associate those tracks to intelligence data. At the end of the process, *domain tracks* together with their identifications (if any) are inserted into the *Fused Track File*.
- *Platform Recognition*: this process aims to assess what are the platforms present in the EW arena, starting from the data retrieved during the *Identification* process. When a track is compatible with more than one *waveform models*, or the identified *waveform* models are installable on several platforms according to the *Mission Library*, ambiguities arise. When this kind of ambiguity is present, the *Platform Recognition* process tries to resolve it according to the logic described in the following section. The result of this process is therefore a new set of tracks, called *platform tracks*, representing the platform that are supposed to be present in the environment.
- *Behaviour Prediction*: Once the information about the platforms (the *platform tracks*) and the related emitted waveforms (the *domain tracks*) are available and updated into the *Fused Track File*, the EWMDF process can exploit the mission library to infer what class of behaviour the detected platforms are currently performing. This information is useful to predict their possible future activities.

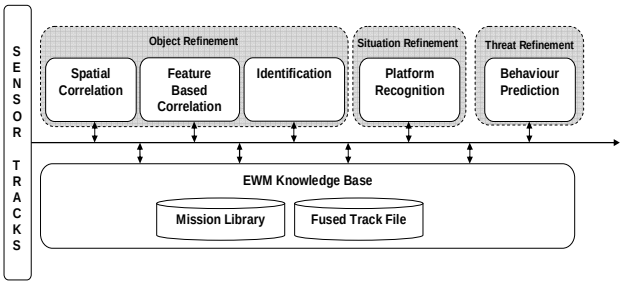


Fig. 1: The EWM Sensor Data Fusion Process

### A. The Spatial Correlation

The *Spatial Correlation* has the responsibility of identifying *compatibility sets* by grouping tracks which are similar for directions of arrival and ranges, when the range measurement is available.

To this end the process exploits a model of the sensors, from which the handled tracks are retrieved, in order to estimate measurement errors.

These models, defined by means of probabilistic curves representing the related measurement dispersions, are completely customizable (i.e. by choosing the kind of curve, the related variance σ, etc.) and contained in the *Mission Library*.

Sensor tracks contained into the *Fused Data Base* define a graph having as nodes the tracks and edges representing a compatibility relationships.

As an example suppose to have detected the following compatibility pairs: (T1,T2) (T1,T3) (T1,T4) (T3,T4) (T5,T6), a matrix representation can be provided by means of an NxN matrix having row elements (i) and column elements (j) representing the involved tracks and values at the intersection (i,j) equals to 1, if the compatibility pair (i,j) exists, 0 otherwise.

Fig. 2 depicts the matrix representation of an association graph. A graph representation of the same dataset is depicted in Fig. 3. Starting from this representation the track compatibility sets can be defined as the sub-graphs, of the association graph, that are completely connected.

In other terms as each compatibility set has to be composed by tracks that are all spatially associable, we are interested in finding the set of nodes in the association graphs that are all linked together.

In graph theory, this kind of graph is called *clique* [5]. More formally a *clique* of an undirected graph G = (V, E) is a subset of the vertex set, such that for every two vertices in C, there exists an edge connecting them.

The size of a clique is the number of vertices it contains. Several algorithms already exist aimed at finding the maximum cliques of a graph [5][6] and are exploited to address the *Spatial Correlation* problem modelled in the described way.

Following the example, the existing maximum cliques in the graph depicted in Fig. 3 are those linked by different type of edges: (T1,T2) (T1,T3,T4) (T5,T6).

A track Ti can belong to more than one clique, as in the case of the T1 track, and hence to more than one compatibility set.

In this cases both the hypothesis are kept, demanding to the following phases of the process the responsibility to disambiguate the multiple association relationships.

| | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** |
|---|---|---|---|---|---|---|
| **T1** | 1 | 1 | 1 | 1 | 0 | 0 |
| **T2** | 1 | 1 | 0 | 0 | 0 | 0 |
| **T3** | 1 | 0 | 1 | 1 | 0 | 0 |
| **T4** | 1 | 0 | 1 | 1 | 0 | 0 |
| **T5** | 0 | 0 | 0 | 0 | 1 | 1 |
| **T6** | 0 | 0 | 0 | 0 | 1 | 1 |

Fig. 2: An example of compatibility matrix

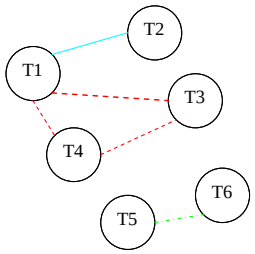


Fig. 3: An example of compatibility graph and related maximum cliques

### B. *Feature Based Correlation and Identification*

For each detected compatibility sets, the *correlation process* aims to associate tracks retrieved by different sensors of the same type (i.e. RWR, MWS, LWS, etc.) referring to the same entity (i.e. an RF waveform, an IR emission, etc.). This process is essentially based on a comparison of the parameters characterizing the involved entities aimed at finding tracks having strongly similar characteristics within the compatibility sets. When such compatibility is claimed a *domain track* is created by a proper merge of the involved tracks' parameters. *Domain tracks* are maintained into the *Fused Track File* where a relationship with the related sensor tracks is also kept. The resulting *domain tracks* are then object of an identification analysis. The identification is a process of comparison between the *domain tracks* parameters and the *waveform models* contained in the *Mission Library*. The comparison process produces an output value (called *identification score*), for each pair (*domain track*, *waveform model)*. The identification score is a value representing the degree at which the *domain track* has characteristics similar to those described in the *waveform model*. A list, called *identification list*, is therefore created for each *domain track* and consist of a set of references to each waveform model having received an *identification score* greater than a threshold when compared to the related *domain track*. If an identification list contains more than one entity, ambiguities arise.

### C. *Platform Recognition*

The *Platform Recognition* process has the responsibility to associate *domain tracks* into *platform tracks* by taking into account their identifications. After the *Feature Correlation* and *Identification* phase the EWM *Fused Track File* contains *domain tracks* each one related to a list of possible *waveform models* having an associated *identification score*. Because a *waveform model* is always described in the *Mission Library* as belonging to an emitting sensor installed on platform, the *identification score* also expresses the degree of confidence at which a *domain track* is supposed to belong to a platform. It can therefore happen that a *domain track* can be supposed to belong to several platforms with different confidence levels, depending on the *identification scores*. Exploiting these data it is possible to produce a matrix ([*domain tracks*],[*platforms*]) having as value of the *i-th* row and *j-th* column the *identification score* at which the track *i* is supposed to belong to the platform *j*, normalized with respect to the maximum *identification score*.

An example of this kind of matrix, called *Platforms Matrix*, is depicted in Fig. 4. The *Platform Recognition* process creates a *Platforms Matrix* for domain tracks of each compatibility set. Consequently it tries to find the set of track/platform associations maximizing an independently defined *objective function*. This approach makes possible to have a core algorithm process independent of the criterion by which a tracks/platforms association is evaluated. Such criterion can be in fact strongly variable depending on the strategic needs and has been therefore well encapsulated in a separated component, the *objective function* itself, that is part of the *EWM Mission Library*. The problem of the Platform

Recognition process is therefore an *optimization problem* based on a *multi-criteria objective function* [7].

In order to find a good enough solution while respecting real time constraints, an evolutionary heuristic approach based on genetic algorithms has been chosen.

Genetic algorithms have been proposed for the first time by John Holland (Michigan University) in 1975[8]. They consist of research algorithms based on the biologic evolution metaphor as they exploit mechanisms conceptually similar to the natural selection and sexual reproduction. The execution flow of a genetic algorithm basically consists of four steps. In the first step a set of elements is defined. This set is called a *population*. In the population each element represents a possible solution for the task the algorithm has to address. To this end it is coded as a string of bits which is called *genotype*.

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| **T1** | 0.0 | 0.0 | 0.1 | 0.3 | 0.3 | 0.3 |
| **T2** | 0.0 | 0.0 | 0.2 | 0.0 | 0.6 | 0.2 |
| **T3** | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **T4** | 0.3 | 0.0 | 0.0 | 0.7 | 0.0 | 0.0 |

Fig. 4: An example of Platforms Matrix

From the initial population the algorithm starts an iterative loop where, at each iteration:

- The objective function is evaluated for each element of the population. The objective function produces as output a value (called fitness) representing the degree at which a solution satisfies the task to address. The higher is the resulting value the better is the solution quality;
- If the best fitting element is considered a good enough solution, the algorithm successfully terminates.
- Otherwise the algorithm selects a subset of elements starting from the population preferring those with the highest fitness according to a reproduction probability rule;
- From this subset the algorithm creates new elements by applying genetic operators (i.e. *Crossover* and *mutation*);
- The algorithm calculates the fitness for these new elements and proceeds from the bullet 2.

A typical criterion to estimate the reproduction probability has been proposed by Holland in [8]. In this formulation the probability of an individual to be chosen is proportional to its fitness. Such probabilities are typically used in order to model a sort of roulette for the elements reproduction, called *reproduction roulette*. As an example Fig. 5 depicts four elements, namely A1, A2 A3 and A4 with probabilities 0.12, 0.18, 0.3 and 0.4 respectively of being selected for the reproduction. Each of these elements holds a place in the *reproduction roulette* which is proportional to the related probability. In the present example the selection operator generates a random number equal to 0.78 (c) so that the A4 element is chosen. Any time an element is selected a copy of it is introduced in a set named *mating pool*. As soon as the *mating pool* contains a predefined number of *n* elements, *m* new elements are created from them by the application of the crossover and mutation genetic operators.

The crossover operator (

Fig. 6) starts with the random selection of two elements of the mating pool and a point in their genotype which is called crossover point (single-point crossover).

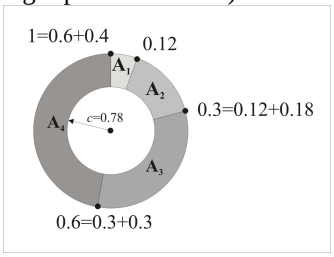


Fig. 5: An example of reproduction roulette

Two new elements can be then generated by exchanging the portion of both genotypes at the right of the crossover point. Dependently on the problem the algorithm is supposed to solve, the single point crossover approach can be modified by changing the way the genes are changed among the two parent elements. Once the descendants have been created a mutation can be applied to them depending on a mutation probability (*pm*). When mutation is applied to a genotype some of its bits are randomly selected and changed (i.e. from 0 to 1 and vice versa). As the crossover represents the sexual reproduction, the mutation models the genetic mutation that can occur in nature as a result of the sexual reproduction. The new generation of elements take the place of the previous one and the process continues as explained for a predefined number of times by providing, at each iteration, a better (or at least equal to the previous iteration) solution of the problem.

In order to address the *Platform Recognition* problem through genetic algorithms a binary representation for problem solutions has to be defined. As the process aims to find the tracks/platform associations maximizing an objective function, a solution can be modelled as a matrix admitting only boolean values where the (*i*,*j*) element is 1 if the track *i* has to be related to the platform *j*, 0 otherwise (Fig. 7). This kind of representation fits well with the genome representation needed by genetic algorithms. In fact such matrix can be easily encoded as a string of bits by simply concatenating all of its rows (Fig. 8).

Having adopted this binary representation for the possible solutions, the genetic flow resolving the *Platform Recognition* task has been designed in this way:

- First of all the initial population of solutions is created. In order to improve performances, the population is created avoiding the generation of solutions that are not applicable. For instance, considering that each track can be associated to

one platform, there cannot more than one 1 value for each row;

- An externally defined objective function is evaluated for each individual of the population providing as output a number representing the degree at which the solution satisfies the problem;
- If the best fitting element is considered a good enough solution the algorithm successfully terminates.
- Otherwise the most fitting individuals are chosen for the reproduction according to the reproduction roulette;
- From these individuals new solutions are created by applying specific cross-over and mutation functions. The cross-over function has been designed in order to avoid the creation of not valid individuals. This means that: starting from two parent individuals *i* and *j*, a son individual *k* is created by randomly choosing a row index (*g*) and assigning to *k* the former *g* rows of *i* followed by the latter *g* rows of *j*. The mutation function is then applied by randomly selecting a row of the resulting matrix; setting all its values to 0 and hence setting a randomly chosen value of the selected row to 1.
- A selected percentage of the previous generation most fitting individuals is always kept in order to avoid regression in the solution research. The others are instead replaced by the new solutions;
- The objective function is evaluated for the new solutions and the algorithm proceeds from the third bullet.

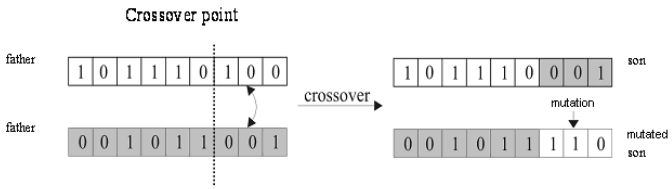


Fig. 6: the crossover operator

| | P1 | P2 | P3 | P4 | P5 | P6 |
|---|---|---|---|---|---|---|
| **T1** | 0 | 0 | 0 | 0 | 1 | 0 |
| **T2** | 0 | 0 | 0 | 0 | 1 | 0 |
| **T3** | 1 | 0 | 0 | 0 | 0 | 0 |
| **T4** | 0 | 0 | 0 | 1 | 0 | 0 |

Fig. 7: An example of solution

0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0

Fig. 8: A Genetic representation of the solution depicted in Fig. 7

As an example suppose to define an objective function taking into account the *identification scores* and the number of tracks associated to a platform. This objective function evaluates the sum of the identification score values for each platform (by column with respect to a *Resources Matrix*) multiplied by the number of associated tracks. Fig. 9 depicts the trend of the best solution (blue line) and of the average fitness among the population (black line) obtained by an application of genetic algorithms to find a solution maximizing this objective function with respect to the *Platforms Matrix* depicted in Fig. 4. Approximately at the 90$^{th}$ cycle (generation) the algorithm finds the solution depicted in Fig. 10. Such solution remains the same up to the end of the algorithms (programmed to execute 200 generations). The same process, with the same input (the *Platforms Matrix* depicted in Fig. 4), produces a different output as soon as the objective function is changed. For instance if the objective function is changed in order to sum only the higher identification score associated to each platform, the results are depicted in Fig. 11 and Fig. 12. As it is possible to note in this case the algorithm try to find the association that maximize the number of most possible platforms.

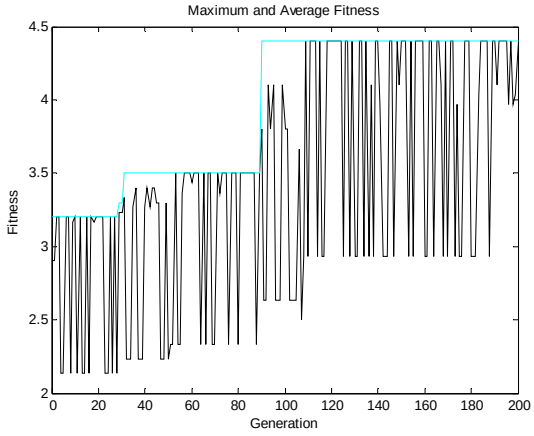


Fig. 9: An example of genetic solution search

(T1,P5), (T2,P5), (T3,P1), (T4,P1)

Fig. 10: An Example of solution to the Platform Recognition problem

The adoption of genetic algorithms have had several advantages. First of all it is strongly customizable as it is driven by externally defined objective functions the customer can define. The adopted objective function can be runtime changed dependently on the context. It always execute a fixed number of iterations in a fixed time which is a very important characteristic in real-time systems.

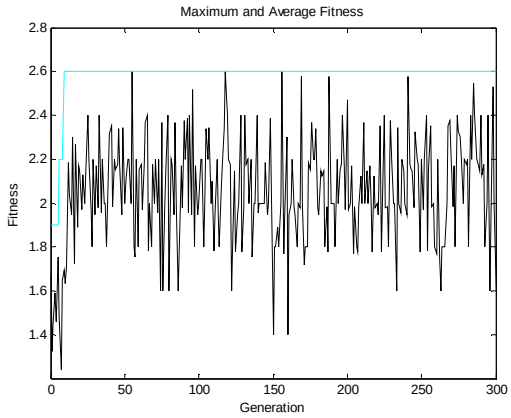


Fig. 11: An example of genetic solution search

(T1,P6), (T2,P5), (T3,P1), (T4,P4)

Fig. 12: An example of solution to the Platform Recognition problem

### *D. Improving on the genetic algorithms using the Auction algorithm as initial guess.*

In the previous subsection we proposed an iterative algorithm based on an evolutionary heuristic approach. Our belief is that there might be an improvement of the performance of such approach if we combine it with a rational choice of the elements of the population.

Our approach combines a mathematical programming algorithm with heuristic search techniques based on genetic algorithms. More specifically, we use the deterministic solution from a linear programming algorithm as an element of the population of the genetic algorithm.

To this end, we can recast the Platforms Matrix depicted in Fig. 4, as a gain matrix in the classical assignment problem [14].

The assignment problem is one of the fundamental combinatorial optimization problems in the branch of optimization. It consists of finding a maximum weight matching in a weighted bipartite graph.

In its most general form, the problem is as follows:

There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized.

We can immediately see the analogy with our problem replacing the platforms and tracks with the agents and tasks, respectively.

The mathematical formulation can be written in the minimum cost flow format

$$\max \sum_i \sum_j c_{ij} x_{ij}$$

subject to the constraints

$$\sum_i x_{ij} = 1 \quad \forall j$$

$$\sum_j x_{ij} = 1 \quad \forall i$$

$$x_{ij} \geq 0 \quad \forall i, j$$

The costs $c_{ij}$ represent the identification score while the variable $x_{ij}$ represents the assignment of platform j to track i, taking value 1 if the assignment is done and 0 otherwise (fitting to the genome representation needed by genetic algorithms).

This formulation allows also fractional variable values, but there is always an optimal solution where the variables take integer values. The first constraint requires that every platform is assigned to exactly one track, and the second constraint requires that every track is assigned exactly one platform. There is a large class of algorithms for solving the assignment problems. Earlier assignment algorithm, such as the Hungarian method proposed by Kuhn [15] were only applicable to square assignment matrices. Newer faster methods include the Jonker-Volgenant (JV) relaxation [16] techniques as well as Bertsekas's Auction algorithm [17]. In our opinion, the Auction algorithm is the most efficient assignment algorithm currently available and we decided to use it in order to solve the assignment problem for our application. An example of the Auction algorithm solution using the costs $c_{ij}$ in thePlatforms Matrix depicted in Fig. 4 is the same of that depicted in Figure 6. In Figures 12 and 13 the genetic solutions obtained without and with he Auction initialization are plotted versus the number of generations. As we can see from these figures, the genetic algorithm with/without a good initialization is characterized by a high/low fitness level as well as by a quick/slow rate of convergence.

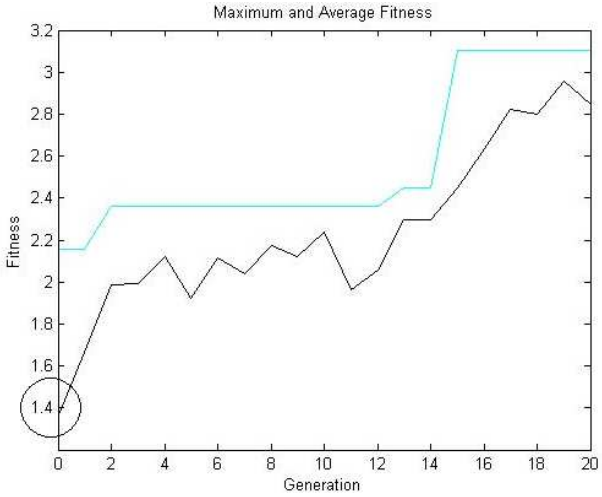


Fig. 12: An example of a genetic solution search without the Auction initialization

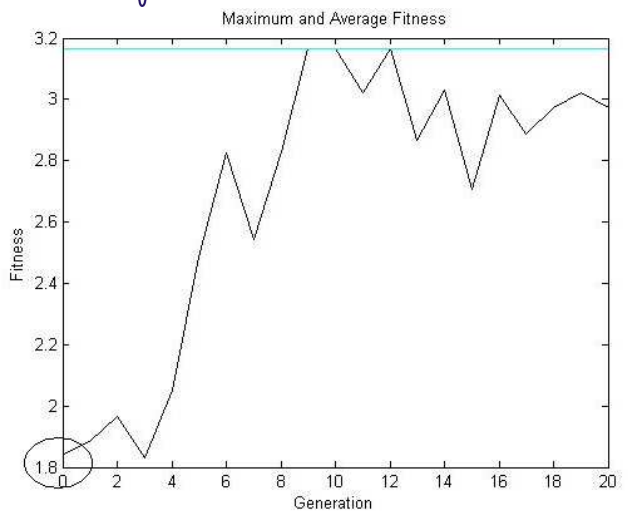


Fig. 13: The example as in Figure 12 of a genetic solution search with the Auction initialization

We should also mention that the improvement of the Auction initialization over the "plain" genetic algorithm is due to the particular choice of the objective function. There might be situations in which the multi-criteria objective function does not satisfy the Auction algorithm constraints (every platform is assigned to exactly one track and every track is assigned exactly one platform). We intend to investigate these cases and possibly modify the Auction algorithm in our future works.

### *E. Behaviour Prediction and User Feedbacks*

Once assessed what platforms are currently present in the environment and what are their related domain tracks, the *Behaviour Prediction* process can finally be executed in order to project the observed situation into the future to draw inferences about next enemy's actions.

To this end this process exploits a specific section of the *Mission Library* in which classes of behaviours can be modelled by means of state machines of functions each platform can perform by means of its available waveforms, weapons, etc. Taking as input the recognized platforms and the observed domain tracks, the *Behaviour Prediction* process can therefore recognize what is the class of behaviour that is most fitting with respect to each platform and, depending on it, identify its possible evolution. Moreover an information about the degree of threatness can be associated to each state of the modelled behaviours. Exploiting all these information the EWM is capable to depict a complete situation assessment to the system user, enriched with information about the enemy's possible future actions and related threatness.

## IV. CONCLUSIONS

In this paper we have presented a simplified overview of a data fusion (DF) architecture for the new Electronica S.p.A. product called EW-Manager.

Following the "*Separation of concerns principle*" [9] the presented architecture has been designed by means of a set of parts each one performing a specific function with respect of the whole DF process.

Each part of the DF component exploits different classes of algorithms depending on the task it has to address. As an example the *Spatial Correlation* sub-process adopts graph research algorithms to find subsets of compatible tracks while the *Platform Recognition* exploits genetic algorithms in order to find a proper disambiguation of the possible detected platforms. The resulting product is a software component capable to scale seamlessly with respect to the amount of data and the user needs.

The development and refinement of the presented component has been done exploiting a distributed simulation environment based on STAGE [10] and HLA [11]. This simulation framework has been properly extended by us in order to make it strongly representative of the data flow the EWMDF will handle in real operations. This approach has enabled us to test the product with respect to several scenario conditions.

## REFERENCES

[1] S. A. Vakin, L. N. Shustov, R. H. Dunwell, "*Fundamentals of Electronic Warfare*," Artech House Radar Library, 2001.

[2] U.S. Department of Defense, "*Data Fusion Subpanel of the Joint Directors of Laboratories*," Technical Panel for C3, "Data fusion lexicon," 1991;

[3] O. Kessler, "*Functional Description of the Data Fusion Process*," Office of Naval Technology, Naval Air Development Center, Waminster, PA, 1991;

[4] H.W. Sorenson, "Least-squares estimation: From Gauss to Kalman," IEEE Spectrum, 8, 63-68, July 1970;

[5] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo, *The maximum clique problem*, in *Handbook of Combinatorial Optimization*, Suppl. Vol. A, D.-Z. Du and P.M. Pardalos,eds., Kluwer, Dordrecht, The Netherlands, 1999, pp. 1–74.

[6] P. M. Pardalos and A. T. Phillips, "*A Global Optimization Approach for Solving the Maximum Clique Problem*," Int. J. Computer Math., 33:209 216, 1990.

[7] M. Ehrgott, "*Multicriteria Optimization*," Springer Verlag, Sept. 2000.

[8] J. Holland, "*Adaptation in Natural and Artificial Systems*," Ann Arbor: The University of Michigan Press, 1975.

[9] W. Hürsch and C.Videira Lopes, "*Separation of concerns. Technical report*," Northeastern University, February 1995

[10] STAGE: http://www.presagis.com/products/simulation/stage_scenario/

[11] High Level Architecture (HLA): IEEE 1516-2000 - Standard for Modeling and Simulation High Level Architecture - Framework and Rules

[12] E. Waltz and J. Llinas, "*Multisensor Data Fusion*," Artech House, Norwood, MA (1990)

[13] S. Giompapa, F. Gini, A. Farina, A. Graziano, R. Croci and R. Distefano, "*Maritime border control multisensor system*," IEEE Aerospace and Electronic Systems Magazine, Vol. 24, Issue 8 (2009)

[14] Munkres, J., "*Algorithm for the Assignment and Transporation Problems*", J. SIAM, Vol. 5, 1957, pp. 32-38.

[15] Kuhn, H. W., "*The Hungarian Method for the Assignment Problem*", Naval Research Logistic Quarterly, No. 2, 1955, pp. 83-97.

[16] Jonker, R and Volgenant A., "*A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems*", Computing, Vol. 38, 1987, pp.325-340.

[17] Bertsekas, D. P., "*The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial*", Interfaces, Vol. 20, 1990, pp.133-149.