Application of Graph-Bound Genetic Operators to the Generation of Master Air Attack Plans

Darrin Taylor, Sc.D., Paula deWitte, Ph.D., Sherry E. Marcus, Ph.D., Rose Davis

21st Century Technologies, Inc.
11675 Jollyville Rd., Ste. 300 Austin, TX 78759
dtaylorz@21technologies.com

Abstract. Constructing a Master Air Attack plan requires the negotiations of many constraints. Typical of these constraints are limitations on the munitions that can be delivered to each target, limitations on the platforms that can carry the munitions, as well as limitations on the availability of the platforms from a specified unit. A schedule needs to be constructed that respects the unit contracts, the planner enforced attack waves, and the packaging preferences. In the process, we seek to maximize the targets that can be attained with available resources subject to the priorities of the targets. The initialization of the genetic algorithm is accomplished by randomized greedy operations that quickly find local minima in the solution space. These greedy algorithms are implemented as genetic operators and are used not only to initialize the chromosomes with valid solutions but also to "finish off" the good solutions that result from the genetic algorithm to ensure that any remaining resources are used and that the solution can only be improved by de-allocating some subset of the existing solution.

INTRODUCTION

Planners assemble Air Tasking Orders (ATO) consisting of a plan describing what type planes are being used, their destinations, and with what ordinance the planes are loaded. ATOs are usually planned on a daily basis and establish the order of battle for that day. During the process for constructing ATOs, planners must simultaneously optimize a number of different objectives. First, the planner may want to maximize the Probability of Destruction (PD) of a target while simultaneously minimizing attrition and mid-air refueling. The planner may also want to prioritize these objectives based on the mission at hand. Alternatively, a planner may consider minimizing attrition to be the top priority in one mission, while minimizing refueling may be of primary importance in another mission. This results in multiple competing objectives during mission planning for the ATOs. Further, plan generation systems must also be **flexible** in that they must allow for the planner to insert new objectives (and constraints) on the fly; thereby eliminating classic optimization solutions, since it would require the algorithm to be completely re-written. In addition, the solutions provided to the planner must be credible. In many current planning systems, much of the 'smart technology' is not used by planners, because they don't have confidence in the solutions. By allowing the planner to prioritize objectives in a hierarchical manner, one would provide the planner with increased visibility into the criteria in which solutions are being computed.

ATO-Link, our resource allocation software that computes Master Air Attack Plans, is sponsored by AFRL at AFRL Information Directorate (Rome, NY) and is being inserted into the Fast Maap Toolkit developed at the USAF C2 Battlelabs. This paper discusses aspects of the revolutionary approach taken by 21st Century Technologies utilizing a genetic algorithm (GA) approach that minimizes the generation of invalid chromosomes by constraining the creation of chromosomes, whether by crossover or by mutation to represent valid solutions. One of the approaches we use represents the constraints, as much as possible, as a weighted, directed graph. We create genetic operators that implement crossover and mutations in the context of this constraint graph. The result is that generated chromosomes all represent feasible solutions and faster convergence results.

SYSTEM DESCRIPTION

OBJECTIVE: The generation of the plan requires criteria to select from a large number of possible plans. There are multiple objectives and these objectives must be managed simultaneously. ATO-Link chooses a plan that optimizes the following metrics based on a hierarchical and prioritized ordering.

- Maximize number of targets serviced. As many targets as possible should be serviced.
- Minimize the number of unpackaged aircraft.
- Minimize attrition by seeking to increase the mutual support that each aircraft has.

OUTPUT: Using the above constraints, ATO-Link will compute a plan containing at least the following:

- Allocate (aircraft, munitions, Time On Targets) tuples for each target.
- Compute the mission composition and packaging needed for the planes to provide the most mutual support.

INPUT: ATO-Link processes the following non-exhaustive list constraints:

- Locations of bases and targets
- Aircraft availability at each base
- Availability of fuel, munitions, fins and fuses at each base
- Location of Launch baskets. Bombing operations need to be out of the way of long-range cruise missiles and the platforms that fire them. This minimizes the chance of collisions and the general hazard caused to manned vehicles from automated air vehicles.
- Aircraft load capability, including tradeoffs between munitions and additional fuel tanks.
- Aircraft fuel consumption at different stages in flight
- Threat information (location of SAMs, etc.)
- Turn around times for aircraft
- Pre-existing TOTs
- Unit Contract dictates the numbers of sorties a unit can generate during a specified time period.
- Munition constraints (time of day, weather). TV guided munitions cannot be used when visibility is poor.
- Pre-existing airspace control constraints
- Available pilots
- Available ground crew

IMPLEMENTATION

We represent the Chromosome as an array of java objects (nucleotides) instead of an array of bits or characters or genes. Our implementation, therefore, is to use a singly indexed array of nucleotides (objects) and perform minor algebra to convert gene indices and nucleotide indices into an index in the chromosome array.

Our implementation uses the fast `System.arraycopy` command to enhance the speed of the genetic operators. It is a native method and not written in Java. Our standard genetic operators have been written using `System.arraycopy` and are in two classes: those that respect gene boundaries and those that don't. Since the granularity of the representation is at the nucleotide level, all of the operators insure that even if gene boundaries are not respected the internal objects are. For example, the 3rd object in a gene can be replaced with the 3rd object obtained from some other gene so that an Integer will never be replaced with a Double via a genetic mutation.

The allocation of times for Time On Targets (TOTs) is facilitated by discretizing time into small intervals for this time, typically 1 or 5 minute intervals. Available times are represented by objects. Attack waves are implemented by restricting the set of available TOT objects.

Customer requirements change continually and, as we learn more of what may be appropriate in the future, we need a flexible representation of the constraints. We chose a graph based representation of the constraints so that traversal of constraints will be fast due to the connectivity of the graph. In addition, respecting the connectivity of the graph insures that infeasible solutions are not generated from genetic crossover or mutation. Finally, the nucleotides in the chromosomes can be elements of the graph so that traversal does not have to begin at the root of the graph.

Graph based queries are fast, since queries amount to traversal along the edges of the graph and our representation of the graph store all associative information in the queried node. Moreover, a pointer (nucleotide) in the chromosome can point directly to a relevant node that may represent an aircraft type, a unit, a mission, or a TOT. We can easily see which units use a given plane type or which munitions are carried by the specific plane. Not only will this permit faster evaluation of fitness, but it will also permit the easy creation of more intelligent genetic operators that understand the structure of the constraints and can quickly produce valid mutations that reflect the structure of the constraints.

We adapted our simple graph representation to have strong typing to represent the concepts important to these constraints:

- **Root Nodes** represent one access point to get to all of the constraints.
- **Top Level Nodes** provide easy access to lists of munition types, aircraft types, targets, units and weaponeering options that are used in the processing.

The links between the nodes are useful to specify the type of relationship that exists between the nodes. For example, inventory links have slots for the number of planes that exist at a given base. The links used are:

- **InventoryEdge** contains a slot indicating how much of the child the parent contains. Genetic operators use these edges to track the consumption of resources in a sample solution.
- **ScoreEdge** contains a quantity used in scoring such as PD. It indicates for example, the PD of an SCL associated with a target.
- **InstanceEdge** indicates that the child is a unique instance.
- **CategoryEdge** indicates that the child is a category and not an instance.

The genetic operators that we use include the entirely stochastic mutations and crossovers. Other algorithms have been encoded into our genetic operators. For example, a greedy operator is one that greedily looks to see if there are any obvious allocations that have not been made and greedily matches the remaining resources to targets. This greedy operator “leaves no food on the table.” It’s important because at whatever point the user desires an answer or the algorithm converges, it is statistically likely that there will be spare resources that can be applied to un-hit targets. From the perspective of the user, this is highly undesirable. Instead of having the user match up the remaining unused resources with targets by hand, we use this greedy operator to remove all obvious remaining resources. Finally, this operator becomes the basis of a class of other mutation and crossover operators that takes parts of the parents and finishes off the resulting chromosome to insure a valid allocation of results.

CONCLUSION

The use of a graph to represent the constraints permits a control of the combinatorial explosion that would otherwise result from using the constraints in the scoring of the chromosome. Furthermore, by using the graph representation that keeps the associations (edges) listed in each node, genetic operators are able to quickly traverse the graph, seeking valid resource allocations for generating Master Air Attack Plans. By maintaining viability from generation to generation, we have obtained good performance for the datasets against which we have tested. As a result, this work has provided increased planning effectiveness of the ATO process as it has been demonstrated to maximize the number of targets and PD. Additionally, the customer has demonstrated confidence in the results and uses the system.

FUTURE WORK

Our investigations have shown that different genetic operators influence the speed of convergence. We will investigate further the variety of genetic operators that can be defined on graphs to see if there are operators that stand out for improving convergence. In addition, future system enhancements include automating the planning with the plane inventories, improving seeding and genetic mutations and incorporating TOT, fuel utilization, turn around times, and fleet utilization rates.

ACKNOWLEDGEMENT

This work was performed under USAF Contract F30602-02-C-0029. The authors wish to thank the COTR, Pat O'Neill, of the Air Force Research Laboratory Information Directorate (Rome, NY). The 21st Century Technologies, Inc. team included Rose Davis, Abe DeLaO, Henzil Browne, Michael Anderson and John Beyerle of C3I Associates, Inc.

REFERENCES

[1] Fast Maap Initiative, C2Battlelabs http://www.hurlburt.af.mil/tenant/c2battlelab/fastmaap.htm
[2] McCarter, Mickey, Toolkit Aids Air Attack, Military Information Technology, Vol. 7, Issue 8, Oct 13, 2003