Evolutionary Computation Methods for Synchronization of Effects Based Operations

Jeffrey P. Ridder
System of Systems Analytics Corporation
14900 Conference Center Drive, Suite 375
Chantilly, Virginia 20151
jridder@sosacorp.com

ABSTRACT

Effects based operations (EBO) is a concept based on the premise that a desired strategic outcome can be achieved through synergistic, multiplicative, and cumulative application of the full range of military and nonmilitary capabilities. Such synergism requires synchronizing a large set of operations in order to achieve a set of effects. The synchronizer develops a set of alternative plans by solving a multi-objective optimization problem subject to several constraints. The EBO synchronization problem is also dynamic since planning is a continuous activity in a dynamically changing environment. In this paper, we describe the operational concept for EBO synchronization, the requirements for algorithms to perform synchronization, and some algorithmic steps toward constructing a synchronizer.

Categories and Subject Descriptors: I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search – *Plan execution, formation, and generation*

General Terms: Algorithms

Keywords: Genetic algorithms, planning

1. INTRODUCTION

US Joint Forces Command (USJFCOM) has determined through experimentation that the synergism between units engaged in EBO is difficult to achieve without tools that enable synchronization, adaptation, and coordination of efforts across an entire force. USJFCOM's Project Alpha, a rapid idea analysis group, investigated synchronization, adaptation, coordination, and assessment (SACA) tools through their rapid assessment process (RAP). The SACA RAP developed a set of prototype tools, including a constrained, multi-objective genetic algorithm suitable for static fitness landscapes to select and schedule operations. While the SACA RAP is now complete, other programs within the Department of Defense recognize the requirement for synchronization of effects based operations.

GECCO'05, June 25–29, 2005, Washington, DC, USA.

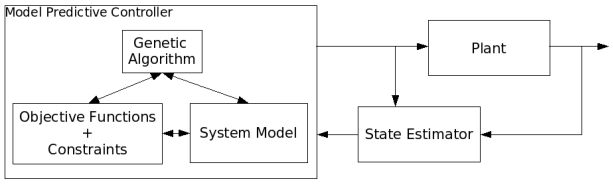

Figure 1: Architecture of a model predictive controller.

2. OPERATIONAL CONCEPT

The synchronizer is one component of what is, in effect, a control system for EBO. The synchronizer uses the current estimate of the state of the environment to select and schedule a plan from multiple possibilities enumerated by a hierarchical task network. The plan is issued in the form of an effects tasking order (ETO) that results in operations to manipulate the environment. The synchronizer must generate plans that are robust and flexible, and be capable of adapting plans as necessary.

One approach to this problem is to implement a genetic algorithm in a model predictive control framework such as that shown in Figure 1. A model predictive controller uses a model of the environment to predict the effects of control inputs over a rolling horizon as shown in Figure 2. The objective function evaluation for time T_1 includes estimates for the effectiveness of control inputs at future times T_2, T_3, and so on to the end of the horizon. These control inputs may be determined by the genetic algorithm, a simpler algorithm, or manual action.

In the model predictive control framework, the GA would execute continuously in order to dynamically respond to changes in the environment. Further, the genome would be dynamically modified as tasks are added and removed from the hierarchical task network.

3. GENETIC ALGORITHM

To meet the requirements of the operational concept, a synchronizing GA must execute continuously, be capable of responding to changes in the fitness landscape, adapt the genome to accommodate changes in the trade space, satisfy several constraints, discover multiple feasible regions if they exist, and select and schedule tasks subject to multiple objectives. Clearly, this is no small task and requires unifying the results of disparate areas of research interest in evolutionary computation.

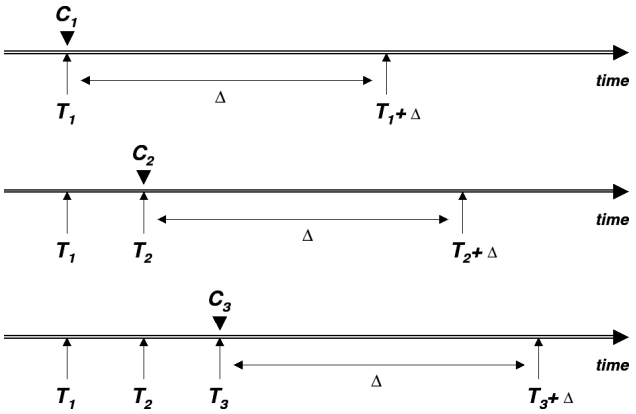

Figure 2: The model predictive control rolling horizon.

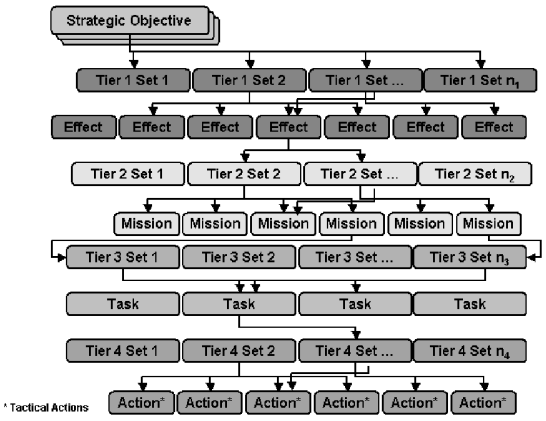

Figure 3: The SACA hierarchical task network.

3.1 Representation

The synchronizer representation includes variables for selection and scheduling of operations. The hierarchical task network for USJFCOM's SACA Tools RAP is illustrated in Figure 3. In SACA, each operation at every tier belongs to one or more tier sets. A set represents a complete option for performing a parent-tier operation, such that all sets within the same tier and with the same parent offer competing and complementing solutions for performing the parent-tier operation. The synchronizer selects one or more sets for each operation. If a tier set is selected, then all operations within that set are selected. The lowest tier contains only primitive operations. The SACA synchronizer selects the sets of operations that are best able to achieve higher tier operations subject to the objective functions and constraints.

Each GA individual represents a complete plan. Consistent with the task network, the representation for SACA includes binary genes for selection of sets, and integer genes for selection of discrete starting times of operations. Since an "off" selection for an upper tier set may result in deactivation of all subordinate sets and operations, many genes have no impact for a particular plan. The time genes may be specified in either absolute or relative time. We can use known sequences of operations, as specified through time and sequence constraints, to both reduce the search space of some time genes and to specify that the search is relative to another time gene, thereby ensuring that sequences and time slack between operations are preserved whenever a single operation's start time is modified by the GA. Relative time genes may form chains of genes that are grounded at some point by either an absolute time reference or a gene specified in absolute time.

3.2 Multi-Objective GA

Synchronization is inherently multi-objective, including objectives such as probability of success, resource cost, risk, and collateral damage. Robustness is another objective that can be implemented either with an explicit objective function such as schedule risk, or by introducing random variations in evaluation of other objective functions.

Multi-objective GA's are typically elitist, which works well for static fitness landscapes, but not for dynamic problems. We recently described a modification to Deb's NSGA-II algorithm [2] which eliminates the need for elitism [7]. Deb's NSGA-II algorithm performs a non-dominated sort of a combined population of elites with the evaluated population (creating a double-sized population), then discards the worst half, keeping the remainder as both the next elite population and as the parent population for the next generation. Our modification to NSGA-II, which we call the Dynamic Non-Dominated Sorting GA (DNSGA), is to implement a steady-state approach with 50 percent replacement. All individuals must be evaluated in each generation followed by a non-dominated sort (there is no separate elite population). The lowest ranking half of the population is discarded and replaced by children produced by parents selected from the best half.

The DNSGA algorithm removes the need for elitism in our multi-objective GA. However, multi-objective genetic algorithms also typically use diversity measures to promote exploration and coverage of the Pareto front. Diversity measures to explore the Pareto front may conflict with other niching methods to promote diversity which may be beneficial for discovering multiple feasible regions (see the next section). In order to avoid such conflicts between the operators needed for exploring and exploiting the Pareto front, and those needed for constraint satisfaction, we are investigating the two-population GA of Kimbrough, et al [3]. In this approach, the infeasible and feasible individuals are kept in separate populations with migration after every generation. That is, newly minted feasible individuals in the infeasible population are moved to the feasible population, and vice versa. This allows a completely different set of operators and diversity measures to operate on each population.

3.3 Constraints

The synchronizer must satisfy several constraints of various types, including constraints on the timing and sequence of operations, weather, resource consumption and availability, effect-on-effect (inhibiting one operation when another is taking place), and logical combinations of any simple constraints. The large numbers, types, and complexity of constraints are likely to result in multiple feasible regions, any one of which may contribute plans to a multi-objective Pareto front. Therefore, we require the synchronizer's constraint satisfaction method to be capable of locating multiple feasible regions in a dynamic environment.

We have performed some preliminary work investigating

Table 1: Locations of the 9 global minima for Levy #3.

Point	x_1	x_2
1	4.97648	4.85806
2	4.97648	-1.42513
3	4.97648	-7.70831
4	-1.30671	4.85806
5	-1.30671	-1.42513
6	-1.30671	-7.70831
7	-7.58989	4.85806
8	-7.58989	-1.42513
9	-7.58989	-7.70831

Table 2: Results of algorithm tests for constrained Levy #3.

Point	#1	#2	#3	#4	#5	#6	#7	#8
1	2	5	99	98	36	43	20	29
2	7	11	100	100	96	96	84	83
3	15	7	100	100	77	81	79	78
4	7	11	100	99	100	100	100	99
5	8	26	100	100	100	100	100	100
6	23	21	100	100	100	100	100	100
7	3	5	99	99	72	54	54	42
8	10	8	100	98	100	100	100	100
9	25	6	99	100	95	85	89	89
Total	100	100	897	894	776	759	726	720

constraint satisfaction methods capable of identifying multiple feasible regions in a multimodal search space. Consider the nonlinear problem known as Levy #3 [4]:

$$f(x) = \sum_{i=1}^{5} i \cos[(i-1)x_1 + i] \sum_{j=1}^{5} j \cos[(j+1)x_2 + j] \quad (1)$$

where $-10 \leq x_i \leq 10, i = 1, 2$. There are 760 local minima and 9 global minima. The global minimum is -176.542 and is located at the points listed in Table 1.

We modified Levy #3 to make it a constraint satisfaction problem by creating a circular feasible region of radius 0.5 around each global minimum. The goal is to find those techniques that result in discovering each of the global minima in every trial. We used Deb's constraint satisfaction method [1] as a baseline for comparison with Kimbrough's two population method. Deb's method uses tournament selection, where feasible solutions are always preferred over infeasible solutions. Feasible solutions are ranked by their fitness, and infeasible solutions are ranked by their constraint error. We test both Deb's and Kimbrough's methods with and without deterministic crowding for promoting diversity [5].

Table 2 shows the number of times, out of 100 trials, that each global minimum was found in the last generation for a variety of algorithms and conditions. A perfect score is 100 for each point (900 overall). For each test, we used Michalewicz' arithmetical crossover and nonuniform mutation operators [6]. The first four tests were run for 1000 generations per trial and used a static fitness landscape. Tests #5 and #6 were run for 1000 generations per trial and considered a dynamic landscape where the feasible regions begin in non-optimal locations and shift to the global minima for the last 500 generations. Tests #7 and #8 are identical to #5 and #6, except that they are run for 2000 generations, where the last 1000 generations have feasible global minima. Test #1 used Deb's method. Test #2 used Kimbrough's two-population method. Tests #3, #5, and #7 used Deb's method with deterministic crowding. Tests #4, #6, and #8 used Kimbrough's two-population method with deterministic crowding.

The results of these tests show that constraint satisfaction methods without niching tend to locate only one feasible region, while niching methods, such as deterministic crowding, can lead to discovery of multiple feasible regions. Performance degrades for dynamic environments, with some points being discovered significantly fewer times than others. The two-population GA performs well with niching and compares well with the single population method. This opens the door to further development use multi-objective niching in the feasible population, and other methods like deterministic crowding in the infeasbile population. Further work is needed to test these methods against problems with more variables, and to extend the results to discrete optimization problems.

4. CONCLUSIONS

In this paper we have described the problem of synchronization of effects based operations. Preliminary investigations of algorithms for this problem have already been conducted via the US Joint Forces Command SACA Tools RAP, and we expect to further develop the synchronizer for application to dynamic environments in a model predictive control framework in the near future by further exploring and developing the ideas discussed here.

5. REFERENCES

[1] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186:311–338, 2000.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. Technical Report 200001, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, 2001.

[3] S. O. Kimbrough, M. Lu, D. H. Wood, and D. Wu. Exploring a two-population genetic algorithm. In *Genetic and Evolutionary Computation Conference Proceedings*, pages 1148–1159. Springer, July 2003.

[4] A. Levy, A. Montalvo, S. Gomez, and A. Galderon. Topics in global optimization. In *Numerical Analysis, Lecture Notes in Mathematics, vol. 909*, pages 18–33. Springer-Verlag, 1981.

[5] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, Illinois, 1995.

[6] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third edition, 1996.

[7] J. P. Ridder and J. C. HandUber. Mission planning for joint suppression of enemy air defenses using a genetic algorithm. In *Genetic and Evolutionary Computation Conference Proceedings*. ACM, June 2005.