

String- and Permutation-Coded Genetic Algorithms for the Static Weapon-Target Assignment Problem

Bryant A. Julstrom
Department of Computer Science
St. Cloud State University
St. Cloud, MN 56301 USA
julstrom@stcloudstate.edu

ABSTRACT

In the Weapon-Target Assignment Problem, m enemy targets are inbound, each with a value V_j representing the damage it may do. The defense has n weapons, and the probability that weapon i will kill target j is p_{ij} . The problem is to assign the weapons to targets so as to reduce as much as possible the total expected value of the targets. A greedy heuristic for this problem repeatedly assigns a weapon to a target to maximally degrade the target's value. Two genetic algorithms encode candidate assignments as strings of target labels indexed by weapon labels and as permutations of weapon labels decoded by a greedy algorithm, respectively. Both GAs can be seeded with the greedy heuristic's solution. In comparisons on fifteen randomly-generated problem instances, all the algorithms significantly reduced the hypothetical strikes' values, but the greedy heuristic was both effective and fast, while the seeded permutation-coded GA returned the best results. The times that all the GAs require grow quickly with problem size.

Categories and Subject Descriptors

G.2.1 [Mathematics of Computing]: Discrete Mathematics—*Combinatorics*; I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic Methods

General Terms

Algorithms

Keywords

Weapon-Target Assignment Problem, Maximum Marginal Return, coding, representations, permutations

1. INTRODUCTION

In a bunker, technicians monitor screens that depict an incoming strike at assets that they are tasked to defend.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
Copyright 2009 ACM 978-1-60558-505-5/09/07 ...\$5.00.

The strike consists of multiple targets, some of which appear unlikely to do serious damage, some of which threaten personnel and facilities. The defense has a supply of weapons with which to attack the targets, they know the probability that a weapon of each type in their inventory will destroy each target, and they must degrade the strike as much as possible. They face an instance of the Weapon-Target Assignment Problem; we describe here a greedy heuristic and two genetic algorithms that address it.

The greedy heuristic assigns the weapons to targets by repeatedly identifying an assignment of one weapon to a target that decreases the expected surviving value of that target the most. The two genetic algorithms encode candidate weapon-target assignments as strings of target labels indexed by weapon labels and as permutations of weapon labels, respectively. The permutation coding is motivated by the effectiveness of similar codings in evolutionary algorithms for other assignment problems; its chromosomes are decoded by a greedy algorithm. Both GAs can be seeded with an assignment generated by the greedy heuristic.

In comparisons on fifteen randomly-generated instances of the Weapon-Target Assignment Problem, all the algorithms significantly reduce the expected values of the hypothetical strikes. The greedy heuristic is both effective and fast, but the seeded permutation-coded GA consistently identifies the best assignments of weapons to targets. However, the times of all the genetic algorithms grow rapidly with problem size. Whether these times become infeasible depends on the time available before the engagement to prepare the defense.

Section 2 below describes the Weapon-Target Assignment Problem precisely. Section 3 specifies the formulation of it that the several algorithms address, and Section 4 relates this formulation to other assignment problems. Section 5 describes the greedy heuristic. Sections 6 and 7 present the string and permutation codings of candidate assignments, respectively, and Section 8 describes the genetic algorithms that use the codings. Section 9 presents the parameters of the test problem instances, and Section 10 the comparisons of the several algorithms on them. Section 11 considers the algorithms' wall-clock run times.

2. THE PROBLEM

In an enemy strike, n targets are inbound, each with value V_j representing its destructive potential. The defense has W_i weapons of each of m types, and the probability that a weapon of type i will destroy target j is p_{ij} . An assignment of weapons to targets designates a specific target for each

weapon. Zero or more weapons, of the same or of different types, may be assigned to each target.

When one weapon of type i engages target j with value V_j , the target will be destroyed with probability p_{ij} and will survive with probability $(1 - p_{ij})$, so the expected value of the target after the encounter is

$$p_{ij} \times 0 + (1 - p_{ij}) \times V_j = (1 - p_{ij}) \times V_j.$$

If several weapons engage target j , the expected value of the target after all encounters is

$$E(V_j) = \prod_i (1 - p_{ij}) \times V_j,$$

where the large product is taken over all the individual weapons aimed at target j . The initial value of the incoming strike is $\sum_{j=1}^n V_j$, and the expected value of the strike after the weapons have engaged their targets is

$$E(V_1) + E(V_2) + \dots + E(V_n) = \sum_{j=1}^n E(V_j). \quad (1)$$

The Weapon-Target Assignment Problem (WTAP) is to assign weapons to targets so as to minimize the expected surviving strike value $\sum E(V_j)$.

In the static version of the WTAP, all the problem's inputs—inbound targets and their values, the weapons available to engage them—are known, and the weapons engage the targets at one time. In the dynamic version, the battle is divided into stages, each of which expends only some of the available weapons. After each stage, the results of the encounter are assessed and weapons are assigned to surviving targets. We consider here the static WTAP.

This problem was first described by Manne [13] and considered also by Day [3]. DenBroeder, Ellison, and Emerling [4] presented an exact solution for the case in which all the target values are equal, and exact algorithms are known for other special cases, but in general the WTAP is NP-hard [12] and thus an appropriate subject for heuristics.

Kolitz [9] described a greedy heuristic, called Maximum Marginal Return, that is reprised in Section 5 below. More recently, Ahuja et al. [1] presented branch-and-bound algorithms based on a variety of lower-bound schemes, including a relaxation of the Maximum Marginal Return algorithm.

Researchers have applied various “soft” computing techniques to the problem. Wacholder [16] presented a neural network for it, and both Lee, Lee, and Su [10] and Yanxia et al. [17] examined ant colony optimization. Zeng et al. [18] applied particle swarm optimization, Lee, Su, and Lee [11] implemented memetic search, and Shang et al. [15] investigated a genetic algorithm with an immune-system-based local search step.

3. A SIMPLER FORMULATION

The formulation of the weapon-target assignment problem with which the previous section began is the same as that used by Ahuja et al. [1] and others. However, an equivalent presentation that abandons the classes of weapons in favor of treating each weapon individually makes clearer the relationship of the WTAP to other assignment problems and simplifies coding in the algorithms that follow.

There are still n targets, each with value V_j , but now simply m weapons, and the probability that weapon i will kill target j is p_{ij} . In place of each class of W_i weapons,

we now have that number of identical weapons. The objective function remains the same, but the problem becomes the unconstrained assignment of individual weapons to targets so as to minimize the expected value of the surviving targets, rather than the constrained assignment of weapon types while not exceeding the available number of each type.

4. ASSIGNMENT PROBLEMS

The WTAP belongs to the class of assignment problems, in which elements of one set must be assigned to elements of a second set so as to maximize or minimize an objective function on the resulting arrangement. In the terminal assignment problem (TAP), for example, a communications network will connect m terminals, each with demand d_i , via n concentrators, each with capacity C_j . The cost of connecting terminal i to concentrator j is c_{ij} , and we seek an assignment of terminals to concentrators with minimum total cost but without any concentrator's capacity being exceeded by the demands of the terminals it serves. If we remove the TAP's capacity constraints and replace its objective function, we obtain the WTAP, with targets in place of concentrators and weapons in place of terminals.

The TAP is itself a case of the single-source capacitated plant location problem, in which we are given locations where plants can be opened, each with a capacity and a fixed cost; customers, each with a demand; and the costs of serving each customer from each location. The problem seeks a selection of locations at which to open plants and an assignment of customers to those locations with minimum total cost but without exceeding the location's capacity.

Codings that are permutations of terminals and customers have recently been effective in evolutionary algorithms for these and other assignment problems [6, 7, 8], so we expect similar success encoding candidate solutions to the WTAP as permutations of weapons. We will find that this is so but, when problems are large, at excessive computational cost.

5. A GREEDY HEURISTIC

In 1988, Kolitz [9] described a greedy heuristic for the weapon-target assignment problem. This heuristic assigns the m weapons in m steps; each step chooses a weapon to assign and a target to which to assign it so as to maximally reduce the expected value of the surviving targets. Because this algorithm always chooses the weapon and target that yield the largest marginal reduction in total target value, it is called the Maximum Marginal Return (MMR) algorithm. The following sketch summarizes the MMR heuristic; in it, V is the array of target values and p is the two-dimensional array of kill probabilities.

```

for  $j$  from 1 to  $n$ 
  decrease = 0.0
  for each unassigned weapon  $w$ 
    for each target  $t$ 
       $d \leftarrow V[t] * p[w][t]$ 
      if  $d >$  decrease
        decrease  $\leftarrow d$ 
         $w0 \leftarrow w$ 
         $t0 \leftarrow t$ 
  assign weapon  $w0$  to target  $t0$ 
   $v[t0] \leftarrow v[t0] -$  decrease
return the sum of  $V[\cdot]$ 

```

In each of n steps, the heuristic considers assigning each unassigned weapon, so the total number of such considerations is

$$n + (n - 1) + \dots + 2 + 1 = \frac{n(n + 1)}{2}.$$

Each consideration tentatively assigns the weapon to each of the m targets, so the heuristic's time is $O(n^2m)$.

This heuristic provides a standard against which we will measure the genetic algorithms that the following sections describe. We will see that it is remarkably effective.

6. A STRING CODING

The object of the WTAP is to assign each weapon to a target so as to degrade the expected value of the incoming strike as much as possible. An obvious coding of candidate solutions to this problem—that is, of candidate assignments of weapons to targets—is as strings of length m over the alphabet $\{1, 2, \dots, n\}$ of target labels. Each element of a chromosome $c[\cdot]$ indicates the assignment of one weapon: $c[i] = j$ indicates that weapon i will engage target j . The fitness of a chromosome, which we seek to minimize, is the expected surviving target value (1) after the weapons engage the targets the chromosome specifies.

Identifying this fitness is straightforward. Initialize an array with the initial target values. Each element $c[i]$ of the chromosome indicates weapon i 's target, so for each weapon i , multiply the value of its target $c[i]$ by one minus the probability that the weapon kills the target. The following sketch summarizes evaluation. As in the MMR sketch above, V is the array of target values and p is the two-dimensional array of kill probabilities. Note that the first step makes a local copy of the target values so that the original values are not changed.

```
locV ← a local copy of V
for each weapon  $w$  from 1 to  $m$ 
  target ←  $c[w]$ 
  locV[target] ←  $(1 - p[w][target]) * \text{locV}[target]$ 
return the sum of the values in locV
```

Building locV requires copying all n initial target values, and determining the expected surviving value requires scanning the chromosome to see where it directs each of the m weapons, so the time of evaluation is $O(\max\{m, n\})$.

This string coding is superficially appealing. It conforms to the traditional use in evolutionary algorithms of chromosomes that are strings over a binary or (as here) larger alphabet. It is easy to generate random chromosomes. We can apply to chromosomes familiar positional operators like k -point crossover and position-by-position mutation. We will find, however, that an evolutionary heuristic that uses this coding is not as effective as we might hope.

7. A PERMUTATION CODING

Section 2 suggested that since evolutionary algorithms that encode candidate solutions with permutations of problem elements have been effective on several other assignment problems, we might expect good performance in a permutation-coded EA for the weapon-target assignment problem. Imitating the approaches of the earlier algorithms, then, let chromosomes represent assignments of weapons to targets as permutations of the weapons. A greedy decoder

identifies and evaluates the particular assignment that a chromosome/permutation represents. The decoder considers the weapons in chromosome order and, imitating the MMR heuristic, assigns each in turn to a target whose expected surviving value will be reduced the most. As with the string coding of the previous section, a chromosome's fitness is the sum of these expected values (1), which we seek to minimize.

The sketch below summarizes the greedy decoder. Again, V is the array of target values and p is the two-dimensional array of kill probabilities.

```
locV ← a local copy of V
for  $i$  from 1 to  $m$ 
  weapon ←  $c[i]$ 
  damage ← 0.0
  for target from 1 to  $n$ 
     $d \leftarrow \text{locV}[target] * p[\text{weapon}][target]$ 
    if  $d > \text{damage}$ 
      damage ←  $d$ 
      best ← target
  locV[best] ← locV[best] - damage
return the sum of the values in locV
```

For each weapon, the decoder scans the targets to identify one whose expected value will be most degraded should the weapon engage it. Thus the decoder requires time that is $O(mn)$.

It is easy to generate random chromosomes in linear time. A good choice for the crossover operator is the Partially Mapped Crossover (PMX) of Goldberg and Lingle [5]. PMX builds one offspring from two parent chromosomes by first copying one parent into the offspring, then choosing two positions at random in the offspring and swapping elements of it until the segment between the positions matches the second parent. The order in which a chromosome lists the weapons in what matters, and this operator tends to preserve parental positions and orderings of the weapons. In preliminary tests on small WTAP instances, a genetic algorithm using PMX returned better results than did similar algorithms that used Reeves C1 [14], Davis' uniform order-based crossover [2], and alternation [7]. The latter merges two parent chromosomes and deletes all duplicate entries to build an offspring permutation. Mutation of these chromosomes swaps the entries at two randomly chosen positions.

8. TWO GENETIC ALGORITHMS

The string coding of Section 6 and the permutation coding of Section 7 were implemented in two generational genetic algorithms for the weapon-target assignment problem. Each GA initializes its population with appropriate random chromosomes. They select chromosomes to be parents in k -tournaments without replacement, and they generate each offspring via either crossover or mutation, never both. The GAs are 1-elitist; each copies the best chromosome of its current generation unchanged into the next generation. The GAs run through fixed numbers of generations.

The MMR heuristic can report both its assignments of weapons to targets and the order in which the weapons are assigned. Its effectiveness, reported below, suggested an extension of both GAs: Seed their populations with the MMR algorithm's solution. In the string-coded GA, this is a chromosome that replicates the heuristic's assignments. In the

Table 1: Results of the MMR heuristic and the unseeded and seeded versions of the string-coded genetic algorithm on the fifteen randomly-generated WTAP instances. For each instance, the table lists is number m of weapons, number n of targets, and sum $\sum V_j$ of initial target values. For the MMR heuristic, the table lists the expected surviving target value $\sum E(V_j)$ and the time MMR requires to return this value. For the two versions of the GA, the table lists the best result in the 30 trials, the mean of the 30 results, the standard deviation s of those values, and the mean time in seconds \bar{t} that the GA expended on each trial.

Instance			MMR		Unseeded				Seeded			
$m = W $	$n = T $	$\sum V_j$	$\sum E(V_j)$	t	best	mean	s	\bar{t}	best	mean	s	\bar{t}
20	10	475	8.2	<0.01	7.8	8.5	0.5	0.4	8.0	8.2	0.1	0.1
20	20	1338	173.2	<0.01	177.3	193.2	9.7	0.4	169.6	172.4	1.2	0.1
20	40	2576	1095.4	<0.01	1090.7	1124.8	15.3	0.5	1093.4	1095.3	0.4	0.1
40	20	1352	21.2	<0.01	21.7	24.6	1.5	2.6	20.2	20.9	0.3	0.7
40	40	2544	304.0	<0.01	363.5	387.7	14.4	2.8	299.7	303.2	1.4	0.7
40	80	4904	2128.1	<0.01	2200.6	2240.5	21.4	3.2	2123.5	2127.9	0.9	0.8
80	40	2431	31.1	<0.01	42.3	46.1	2.1	20.0	30.7	31.1	0.1	5.0
80	80	4809	517.6	<0.01	664.5	718.1	25.4	21.4	513.3	516.9	1.3	5.3
80	160	10293	4329.3	<0.01	4532.6	4585.4	30.4	24.6	4323.9	4329.0	1.0	6.2
100	50	2997	35.7	<0.01	51.3	55.7	2.1	38.9	35.0	35.5	0.2	9.6
100	100	6115	663.5	0.01	898.9	954.3	25.7	41.5	656.4	663.1	1.4	10.3
100	200	12790	5355.6	0.02	5628.1	5708.2	34.6	48.1	5354.8	5355.6	0.2	12.1
200	100	5987	65.8	0.03	108.9	119.8	4.7	298.8	65.2	65.7	0.1	74.7
200	200	12432	1287.5	0.06	1803.4	1914.3	44.9	339.0	1281.6	1287.2	1.2	84.8
200	400	24917	10469.7	0.11	11087.3	11230.7	60.2	395.6	10466.6	10469.5	0.6	97.9

permutation-coded GA, it is a permutation that lists the weapons in the order MMR assigned them; the GA’s decoder replicates MMR’s solution from this permutation. Each GA thus has two versions: unseeded and seeded with MMR’s solution.

The greedy heuristic and the two GAs, each of the latter including a Boolean flag to turn the seeding step on and off, were implemented in C++ and executed on a Pentium 4 processor with 1GByte of RAM running at 2.53GHz under Red Hat Linux 9.0. In the comparisons that Section 10 reports, on instances of the WTAP with m weapons and n targets, the GAs’ populations contained $2m$ chromosomes. Each chose parent chromosomes in tournaments of size $k = 2$, and a tournament’s winner always became a parent. In the string-coded GA, tests on small WTAP instances suggested that crossover produce each offspring with probability 30%, mutation therefore with probability 70%. In the permutation-coded algorithm, similar tests prompted both these values to be set to 50%. The unseeded string-coded GA ran through $200m$ generations, the seeded version through $50m$ generations. The unseeded permutation-coded GA ran through $50m$ generations, its seeded version through $20m$ generations. The genetic algorithms are allowed different numbers of generations because of their varying effectiveness; the string-coded GA requires more generations to reach good solutions than does the permutation-coded GA, and the seeded versions of both converge to good results more quickly.

9. TEST PROBLEM INSTANCES

Real data for instances of the weapon-target assignment problem are classified, so the algorithms described above were compared on fifteen randomly generated WTAP instances, ranging in size from $m = 20$ weapons and $n = 10$ targets to $m = 200$ weapons and $n = 400$ targets. Follow-

ing the example of Ahuja et al. [1], the target values were drawn from the uniform distribution of integers from 25 to 100, and the kill probabilities from the uniform distribution of reals from 0.60 to 0.90. The leftmost columns of Tables 1 and 2 list the parameters of these instances and the sums of their target values.

10. COMPARISONS

The greedy MMR heuristic was run once and the four versions of the two genetic algorithms were each run 30 independent times on each of the fifteen randomly-generated WTAP instances. Table 1 summarizes the trials of the MMR heuristic and the unseeded and seeded versions of the string-coded GA. Table 2 reprises the trials of the MMR heuristic and summarizes the trials of the unseeded and seeded versions of the permutation-coded GA. In addition to the instances’ parameters and the results and elapsed clock times of the MMR heuristic on them, the tables present, for the unseeded and seeded versions of the two GAs on each instance, the best (i.e., smallest) result in the 30 trials, the mean of the 30 results, the standard deviation s of those values, and the mean time \bar{t} that the trials required.

Several observations stand out. First, all of the weapon-target assignment schemes seriously degrade the expected value of the incoming strikes. Even when the targets outnumber the weapons two-to-one, the expected value of the strike is always reduced by more than half. When the numbers of weapons and targets are equal, the strike’s expected value after the weapons engage it is no more than about 15% of its original value, and when the weapons outnumber the targets, the strike’s value is typically reduced by at least 95%.

The MMR heuristic is both fast and effective. Though various GA versions return more effective assignments of weapons to targets, the improvements are always small, and

Table 2: Results of the MMR heuristic and the unseeded and seeded versions of the permutation-coded genetic algorithm on the fifteen randomly-generated WTAP instances, as in Table 1.

Instance			MMR		Unseeded				Seeded			
$m = W $	$n = T $	$\sum V_j$	$\sum E(V_j)$	t	best	mean	s	\bar{t}	best	mean	s	\bar{t}
20	10	475	8.2	<0.01	7.8	7.9	0.1	0.2	7.8	7.8	0.1	0.1
20	20	1338	173.2	<0.01	159.3	160.1	1.3	0.4	159.3	161.4	1.7	0.1
20	40	2576	1095.4	<0.01	1059.1	1059.6	0.6	0.6	1059.1	1060.2	0.8	0.2
40	20	1352	21.2	<0.01	19.2	20.1	0.6	2.7	18.5	19.0	0.3	1.1
40	40	2544	304.0	<0.01	288.4	292.1	2.2	4.4	285.6	289.2	2.0	1.8
40	80	4904	2128.1	<0.01	2105.6	2109.8	2.0	7.2	2106.7	2108.8	1.1	2.9
80	40	2431	31.1	<0.01	31.9	33.6	1.0	34.6	29.6	29.9	0.1	14.2
80	80	4809	517.6	<0.01	507.3	511.8	2.6	57.7	504.7	506.8	1.1	23.5
80	160	10293	4329.3	<0.01	4288.6	4293.2	2.7	104.0	4287.3	4292.4	2.4	41.7
100	50	2997	35.7	<0.01	37.5	38.9	1.1	88.8	34.1	34.5	0.2	35.3
100	100	6115	663.5	0.01	645.1	650.6	3.0	155.7	640.2	643.5	1.6	63.1
100	200	12790	5355.6	0.02	5335.9	5341.2	2.5	291.3	5334.3	5339.9	2.6	114.9
200	100	5987	65.8	0.03	72.7	75.2	1.2	1257.2	64.3	64.6	0.1	499.0
200	200	12432	1287.5	0.06	1286.8	1295.5	5.7	2380.2	1270.2	1273.2	1.3	950.5
200	400	24917	10469.7	0.11	10466.4	10455.6	4.3	4630.7	10440.9	10444.8	1.9	1853.2

MMR requires much less time than any of the GAs.

The unseeded version of the string-coded GA returns the poorest results. It identifies a better assignment of weapons to targets than does the MMR heuristic on only the three smallest instances, and its performance relative to MMR and the other GAs degrades as the problem instances get larger. Seeding the string-coded GA, however, does enable it to improve on MMR’s solutions, on all the instances except the smallest, though the improvements are not large. For example, on the instance with $m = 100$ weapons and $n = 100$ targets, MMR identified an assignment with value 663.5. The unseeded string-coded GA did not do as well (898.9), but the seeded version improved MMR’s solution and identified one with value 656.4. Note, however, that the improvement is only slightly more than 1%. On two of the three smallest instances, the seeded GA’s best result is inferior to that of the unseeded version. This suggests that, on these instances, the seeding solution dominates the population and leads it to converge to a poorer solution.

The permutation-coded GA, on the other hand, consistently outperforms both the string-coded algorithm and the MMR heuristic. Its unseeded version always identifies better results than does MMR or either version of the string-coded GA, and seeding its population with MMR’s solution improves its results further, on all instances except the three smallest. For example, on the instance with $m = 100$ weapons and $n = 100$ targets, MMR’s result, as noted, was 663.5. The unseeded version of the permutation-coded GA found an assignment with value 645.1, and seeding its population improved this to 640.2. The decoder’s greediness tends to develop effective assignments. Note again that even the seeded GA’s result represents only about a 3.5% improvement on MMR’s solution, though such an improvement is worth seeking when “value” is measured in lives and assets.

11. A CONSIDERATION OF TIME

We have pointed out that the MMR heuristic identifies its greedy assignments quickly. On the instance just men-

tioned, with $M = 100$ weapons and $n = 100$ targets, it took just about 0.01 seconds of clock time on the implementing machine. The unseeded version of the string-coded GA, however, took more than 41 seconds, and the seeded version, running through a quarter as many generations, took about ten seconds. The permutation-coded GA was even worse. Its unseeded version, executing the same number of generations as the seeded string-coded GA, ran for more than two and a half minutes, and even its seeded version, again executing fewer generations, required more than a minute. Moreover, the GAs’ various times all grow even longer as the problem instances get larger. These times cast serious doubt on the practicality of evolutionary solutions to the weapon-target assignment problem, when the time available to assign weapons to targets may be very short indeed.

Can the GAs, and in particular the permutation-coded algorithm, be made faster? Most evolutionary algorithms spend most of their time evaluating chromosomes, and the present GAs are not exceptions. Any evaluation of candidate weapon-target assignments must at a minimum copy the m initial target values, consider the expected effect on those values of the n weapons, and sum the resulting expected values, so evaluation is $\Omega(\max\{m, n\})$; evaluation in the string-coded GA is as good as it gets.

As described in Section 7, the evaluation of permutation-coded chromosomes as given is $O(mn)$. For each weapon, in permutation order, the decoder must identify its best target, which requires computing the expected effect of engaging every target. If the weapons are distinct, this process is $\Omega(mn)$.

However, if the targets fall into a small number of types, with identical parameters within each type, then it is necessary only to identify a target *type* for each weapon. Because the number of types is constant, this is a constant-time operation, so the “assignment” phase of decoding is $O(n)$, and the entire evaluation becomes, as in the string-coded case, $O(\max\{m, n\})$.

Note that grouping the targets into types does not yield the original formulation of the problem as presented at the beginning of Section 2, in which the targets were distinct

but the weapons were of several types. Further, even if the targets within a type are physically identical, their values will also depend on where they are aimed and the damage they may do, so that their values will not be the same and this simplification will not apply.

Finally, even if the decoding/evaluation step can be made faster, it may still not be fast enough. Even the seeded string-coded GA took more than a minute to assign 200 weapons. Whether or not such a GA can supply an assignment quickly enough to be useful will depend on the engagement; in particular, on the time available to assess the attack and prepare a response.

12. CONCLUSION

The Weapon-Target Assignment Problem arises in planning to defend against an incoming enemy strike. The strike consists of n targets, each with a value V_j . The defense possesses m weapons, and the probability that weapon i will destroy target j is p_{ij} , so the expected value of target j after weapon i engages it is $(1 - p_{ij})V_j$. The defense seeks to assign its weapons to targets to minimize the expected surviving value of the strike.

The greedy Maximum Marginal Return heuristic repeatedly identifies the assignment of one weapon to a target that most degrades the target's value. Two genetic algorithms encode candidate assignment as strings of target labels indexed by weapon labels and as permutations of weapon labels. The latter chromosomes are decoded and evaluated by a greedy algorithm. Versions of both GAs are seeded with the MMR algorithm's solution.

In tests on fifteen randomly-generated WTAP instances of up to $m = 200$ weapons facing $n = 400$ targets, all the algorithms generated assignments of weapons to targets that seriously degraded the strikes' values. MMR was both effective and fast, and the seeded permutation-coded GA consistently returned the best results. However, the times all the GAs require grow quickly with problem size. Whether they are feasible will depend on the time available to prepare for the engagement.

13. REFERENCES

- [1] R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin. Exact and heuristic algorithms for the weapon-target assignment problem. *Operations Research*, 55(6):1136–1146, 2007.
- [2] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
- [3] R. H. Day. Allocating weapons to target complexes by means of nonlinear programming. *Operations Research*, 14:992–1013, 1966.
- [4] J. G. G. denBroeder, R. E. Elison, and L. Emerling. On optimum target assignments. *Operations Research*, 7(3):322–326, 1959.
- [5] D. E. Goldberg and J. Robert Lingle. Alleles, loci, and the traveling salesman problem. In J. J. Greffenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates.
- [6] B. A. Julstrom. An evolutionary algorithm for some cases of the single-source constrained plant location problem. In M. Keijzer. et al., editors, *Proceedings of the 2008 Genetic and Evolutionary Computation Conference*, pages 607–608, New York, 2008. The ACM Press.
- [7] B. A. Julstrom. A permutation coding with heuristics for the unconstrained facility location problem. In C. Cotta and J. van Hemert, editors, *Recent Advances in Evolutionary Computation for Computational Optimization*, volume 153 of *Studies in Computational Intelligence*, pages 295–307. Springer, Berlin, 2008.
- [8] B. A. Julstrom. Evolutionary codings and operators for the terminal assignment problem. In G. R. Raidl et al., editors, *Proceedings of the 2009 Genetic and Evolutionary Computation Conference*. The ACM Press, New York, 2009.
- [9] S. E. Kolitz. Analysis of a maximum marginal return assignment algorithm. *Decision and Control*, pages 2431–2436, 1988.
- [10] Z.-J. Lee, C.-Y. Lee, and S.-F. Su. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. *Applied Soft Computing*, 2:39–47, 2002.
- [11] Z.-J. Lee, S.-F. Su, and C.-Y. Lee. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33:113–121, 2003.
- [12] S. P. Lloyd and H. S. Witsenhausen. Weapons allocation is NP-complete. In *Proceedings of the 1986 Summer Conference on Simulation*, pages 1054–1058, Reno, NV, 1986.
- [13] A. S. Manne. A target-assignment problem. *Operations Research*, 6:346–351, 1958.
- [14] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 22(1):5–13, 1995.
- [15] G. Shang, Z. Zaiyue, Z. Xiaoru, and C. Cungen. Immune algorithm for weapon-target assignment problem. In *IITA '07: Proceedings of the Workshop on Intelligent Information Technology Application*, pages 145–148, Washington, DC, USA, 2007. IEEE Computer Society.
- [16] E. Wacholder. A neural network-based optimization algorithm for the static weapon-target assignment problem. *ORSA Journal of Computing*, 4:232–246, 1989.
- [17] W. Yanxia, Q. Longjun, G. Zhi, and M. Lifeng. Weapon target assignment problem satisfying expected damage probabilities based on ant colony algorithm. *Journal of Systems Engineering and Electronics*, 19:939–944, 2008.
- [18] X. Zeng, Y. Zhu, L. Nan, K. Hu, B. Niu, and X. He. Solving weapon-target assignment problem using discrete particle swarm optimization. In *Proceedings of the Sixth World Congress on Intelligent Control and Automation*, volume 1, pages 3562–3565, 2006.