A Case Study of a Multiobjective Elitist Recombinative Genetic Algorithm with Coevolutionary Sharing

Martijn Neef[1], Dirk Thierens[2] and Henryk Arciszewski[3]

[1] Cognitive Artificial Intelligence, Dept. Of Philosophy, Utrecht University,
PoBox 80089, 3508 TB, Utrecht, The Netherlands (*neef@phil.uu.nl*)[*]
[2] Dept. Of Computer Science, Utrecht University,
PoBox 80089, 3508 TB, Utrecht, The Netherlands (*dirk.thierens@cs.uu.nl*)
[3] Command, Control and Simulation, TNO Physics and Electronics Laboratory,
PoBox 96864, 2509 JG Den Haag (*arciszewski@fel.tno.nl*)

Abstract: We present a multiobjective genetic algorithm that incorporates various genetic algorithm techniques that have been proven to be efficient and robust in their problem domain. More specifically, we integrate rank based selection, adaptive niching through coevolutionary sharing, elitist recombination, and non-dominated sorting into a multiobjective genetic algorithm called ERMOCS. As a proof of concept we test the algorithm on a softkill scheduling problem.
Keywords: genetic algorithms, multiobjective optimisation, coevolutionary sharing, softkill scheduling

1. Introduction

Characteristic to most engineering problems is that they are *multiobjective*. A problem is said to be multiobjective or *multicriteria* if it involves optimising multiple goals at once. It is clear that it is not always possible to find a solution which is optimal with respect to *all* objectives. A solution may be optimal regarding one objective, but at the same time be inferior regarding another objective. For instance, the cost and quality of a product are typically two objectives one cannot adequately optimise independently, nor can they be easily combined into a single objective function. In other words, usually the design goals are competing and many *trade-offs* are possible. Therefore the decision which design or solution is best is often taken by an external (human) *decision maker*. More recently evolutionary techniques have been applied to multiobjective problems as well. Because of their massive parallel search these techniques are especially suitable for multiobjective optimisation. These multiobjective evolutionary techniques have nevertheless scarcely been applied to real-world problems. In this paper we present a multiobjective genetic algorithm that is applied and tested on a practical optimisation case.

The next section contains a brief introduction on multiobjective problems and the evolutionary approaches that have been developed previously. In Section 3 a new multiobjective GA called ERMOCS is proposed and which is battle-tested against a real-world problem, softkill scheduling, in section 4. Finally, in section 5 a conclusion is posed and remaining issues are discussed.

2. Multiobjective Genetic Algorithms

2.1 Pareto-optimality

Multiobjective problems are special in the sense that they do not have a unique solution. Usually there is no single solution for which all objectives are optimal. The solution to a multiobjective problem therefore comprises a *set* of solutions for which holds that there are no other solutions that are superior considering *all* objectives. These

[*] please send all correspondence to the second author

solutions are called *Pareto-optimal*. Hence, optimising a multiobjective problem is comprised of finding Pareto-optimal solutions.

The notion of Pareto-optimality is defined in terms of *dominance*. Let's assume that a multiobjective problem has k objectives. Assuming that this is a minimisation problem, then a solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_k)$ is said to dominate another solution $\boldsymbol{y} = (y_1, y_2, \ldots, y_k)$ if $\forall i\ x_i \leq y_i$ and $\exists i\ x_i < y_i$. Solution $\boldsymbol{x}$ is a member of the Pareto-set, or said to be *non-dominated*, if there is no other solution $\boldsymbol{y}$ such that $\boldsymbol{y}$ dominates $\boldsymbol{x}$. The multiobjective problem can now be defined as finding solutions which are non-dominated.

Over the years, several evolutionary approaches to multiobjective problems have been introduced. The most commonly used approach is to combine the objective function into a single objective function using weighting coefficients and penalty functions. This problem-transformation enables the use of a simple single-objective genetic algorithm to find a single solution, which may be feasible, so requiring no further searches. Weights and penalty functions are generally hard to set accurately though, whereas both are very problem dependent (Richardson et al., 1989). As a result, the solution a GA comes up with, may not fulfil all the designer's needs. The solution may not even be non-dominated. Setting the weights correctly requires a certain amount of search space knowledge, which is often not available in advance. This way of dealing with multiobjective optimisation is therefore not always applicable or efficient.

Another, and perhaps more effective approach, is to use genetic algorithms to locate Pareto-optimal solutions. These solutions are, by definition, located on a boundary, known as the *Pareto-front*. We would like the solutions to cover the Pareto-front as well as possible, as to obtain a good representation of this front. This approach requires an extensive exploration of the search space and it is this requirement that makes evolutionary algorithms extremely applicable in this case. Their massive parallel exploration of search spaces is an invaluable advantage over other more conventional techniques in locating Pareto-optimal solutions. This Pareto-based approach has additional benefits as well. This approach offers multiple solutions from which a decision maker can select the solution that is best suited according to additional criteria, without requiring additional searches. Pareto-based optimisation is hence a more transparent and efficient way of dealing with multiobjective problem.

2.2 Previous efforts

The first algorithm to exploit the parallel properties of genetic algorithms on multiobjective problems was the Vector Evaluated Genetic Algorithm (VEGA), designed by Schaffer (Schaffer, 1985). His algorithm also was the first to treat objectives separately in order to find multiple non-dominated solutions in a single run of the algorithm. In VEGA, each generation subpopulations are formed in turn from the existing population by using proportional selection according to each of the objectives. These subpopulations are then shuffled together again, forming a new population on which ordinary crossover and mutation can take place. This way offspring, which is created by parents from different subpopulations, is expected to perform well on both the objectives of the parents, and the population is expected to evolve towards the Pareto-optimal front. However, by independently selecting individuals that score well on a single objective there is consequently a bias against solutions that do not have high fitness values for all of the objectives, but are still non-dominated. VEGA hence performs well in locating the extreme parts of the Pareto-set, but has problems finding middling points. In short, all non-dominated points should have equal reproduction probabilities.

This notion was first recognised by Goldberg (Goldberg, 1989), giving rise to a alternative kind of multiobjective genetic algorithms, known as *Pareto-based* approaches. Goldberg suggested using a ranking procedure, in which all individuals obtain a rank, relating to the number of individuals that dominate them. After population domination sort, all non-dominated solution are set aside temporarily, giving rise to a new group of non-dominated solutions, which are assigned rank 2, after which this group is temporarily removed and so on. This procedure continues until all individuals have obtained a rank. This rank can function as a fitness-measure. A another way to assign a rank-based fitness to an individual (Fonseca & Fleming, 1993) is by carrying out a *single* full population domination sort, apply ranking and use a linear or non-linear scaling like $1/\text{rank}_i$ to obtain a fitness

measure. Hereafter the fitness values of the individuals with the same rank are averaged, as to ensure all equal individual will be sampled at the same rate.

Fonseca and Fleming used this ranking method in their Multi-Objective Genetic Algorithm (MOGA) (Fonseca & Fleming, 1993). Furthermore, they introduced a niching scheme that calculates distances in the *criteria* space (in contrast to distance measurements in *decision variable* space). The fitness sharing *niche counts* of individuals is in their scheme calculated within the various ranks. Horn and Nafpliotis (Horn & Nafpliotis, 1993) did not use explicit ranking, but focused on local *domination* tournaments. Their Niched Pareto Genetic Algorithm (NPGA) uses a tournament selection scheme that compares two individuals on their dominance status. Using a comparison set of randomly selected individuals each of the two candidates is classified as locally dominated or non-dominated with respect to the comparison set. In the case that one is non-dominated whereas the other is dominated, the non-dominated individual is selected for reproduction. If both or neither of the two candidates is non-dominated, the tournament is decided using a fitness sharing scheme. In this case the candidate with the lowest niche count wins. The size of the comparison set (t_{dom}) is used to control selection pressure. As Horn (Horn 1996) recognises, the main difference between the dominance tournament as used in NPGA and the ranking procedure as used in MOGA is that the former is locally calculated, whereas the latter is much more globally defined. Srinivas and Deb followed Goldberg's original ranking method more precisely in their Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas & Deb, 1993). The sharing procedures are performed within categories of individuals with the same ranking. After a non-dominance sort of the entire population, all non-dominated individuals are shared with a dummy fitness value, which is based on the number of individuals in the current population. After this procedure, the first class of non-dominated individuals is removed from contention and a next class of non-dominated individuals is determined and shared with a smaller dummy value and removed as well. In this way the entire population is classified and shared fitness-wise. This procedure gives a large reproduction chance to non-dominated solutions, and allows for quick convergence to non-dominated regions.

More recently, the focus has been on implementing multiobjective genetic algorithms in real-world situations. On the whole, these algorithms are based on either of the mentioned established algorithms with problem-specific enhancements. Examples can be found in (Todd & Sen, 1997), (Obayashi, Tsukahara, & Nakamura, 1997), (Cunha, Oliviera & Covas, 1997) and (Loughlin & Rajithan, 1997). Recently Shigura (Obayashi, Takahashi & Takeguchi, 1998) compared several sharing schemes in multiobjective evolutionary optimisation, including Coevolutionary Shared Niching, which is implemented in the algorithm that will be described in the next section. A comprehensive overview of evolutionary approaches to multiobjective optimisation can be found in (Fonseca & Fleming, 1995), and (Coello, 1998).

3. Elitist Recombinative Multiobjective GA with coevolutionary Sharing (ERMOCS)

The primary goal of the ERMOCS algorithm is an efficient and robust optimisation of multiobjective problems. To achieve this aim we have set out to integrate well-known evolutionary techniques, which have proven to be effective within their specific domain. Furthermore, we have adapted the techniques to cope with multiobjective problems. All these approached are integrated into a Pareto-based genetic algorithm. In this section we will discuss the several techniques that are used in the ERMOCS algorithm and clarify why they are, in our opinion, indispensable for our aims.

3.1 Conventional techniques

First, since a Pareto-optimal set is defined in terms of its non-dominance, selecting a selection scheme that respects this notion is inevitable. A sensible choice therefore is a *rank-based* selection scheme instead of the established proportionate selection, which if often used in basic genetic algorithms. An additional ground for this decision is that multiobjective optimisation should be devoid of any criteria preference. Pareto optimisation depends on non-dominance instead of *raw fitness values*. Rank-based selection consequently seems to be the only valid choice in this matter. To this end in ERMOCS all individuals are given a rank, based on the number of other individuals that

dominate them. Individuals are given rank 1 + *n*, where *n* is the number of individuals that dominate them. Non-dominated individuals consequently obtain rank 1, others obtain a rank $1 + p_i$, where p_i is the number of individuals that dominate them. A fitness conversion is made by a simple linear transformation.

Regarding the selection scheme, we have chosen to use an elitist recombinative scheme instead of the usual tournament selection scheme. Elitist schemes have several interesting advantages (Thierens & Goldberg, 1994). First of all, by using an elitist scheme there is no need for specifying a crossover probability, since elitist schemes use crossover unconditionally. This reduces the overall number of parameters that have to be set, and consequently making it easier to apply the algorithm. Secondly, elitist schemes are less sensitive to undersized populations than tournament selection, also enhancing the overall performance in case of poorly set parameters. Finally, and also the most importantly, using elitist recombination ensures that good solutions are never lost during the search process. This is especially useful in time-constrained problem areas, since it allows the algorithm to come up with its best solutions at any given time.

To preserve diversity in the population, a *niching* scheme seems unavoidable as well. We need to maintain multiple optimal solutions, since we are looking for an optimal *set* of solutions. Almost all established multiobjective genetic algorithms have applied some form of niching technique, and niching has proven to be a efficient way to promote and maintain genetic diversity and to prevent genetic drift in multimodal function optimisation. Goldberg (Goldberg, 1989) was the first to propose a niching scheme in conjunction with the rank-based selection technique when optimising multiobjective problems. A niching technique promotes the development of stable niches along the Pareto-optimal front. The most commonplace and successful niche formation scheme is *fitness sharing* (Goldberg & Richardson, 1987). Fonseca and Fleming implemented fitness sharing in their MOGA, with distances being calculated in the objective domain. They also recognised the difficulties in setting niche size σ_{share} correctly, which is hard to do without prior knowledge of the fitness landscape. To this end, they provided a theoretical basis for estimating σ_{share}, which, although accurate, is not practically applicable, since it requires extensive knowledge of the search space.

The conjunction of tournament selection and sharing schemes causes chaotic behaviour in the dynamics of the algorithm and limits the number of stable niches that can be maintained (Oei, Goldberg & Shang, 1991). Since elitist recombination is in essence a tournament selection scheme, our algorithm will suffer from the conjunction as well. In Pareto-based optimisation it is essential that we can maintain a sufficient number of stable niches. The straightforward solution Oei offers is *continuously updated sharing*. The sharing information in the target population is constantly updated as new individuals are entered. During a tournament, the shared fitness of a competing individual is based on the number of niche-members it will have in the population that is being created. This simple modification of the standard fitness sharing scheme produces stable dynamics and thus allows for preservation of many niches. It is easy to use the elitist recombinative scheme with continuously updated sharing. After reproduction of the two competing parent individuals, the shared fitnesses of both the parents and the offspring should be calculated on the basis of the target population. From these four competitors the two top solutions can enter the target population. Since we employ an quasi-steady-state algorithm, we use a slightly modified kind of continuously updated sharing, in which the relevant sharing information *within* the population is constantly updated. This approach will be described more clearly later on.

3.2 Coevolutionary Shared Niching

As noted, one of the drawbacks of fitness sharing is setting the sharing radius σ_{share} properly. This parameter has a significant effect on the performance of the algorithm and should be set as accurate as possible. Ideally σ_{share} should be set adaptively, allowing the algorithm to optimally capture the characteristics of the fitness-landscape. There is a specific need for this in real-world problems, since in these cases fitness-landscapes are non-uniform and thus require non-uniform niche sizes. To this end, Goldberg and Wang (Goldberg & Wang, 1997) devised a sharing scheme, *Coevolutionary Shared Niching* (CSN), that eliminates this parameter and nevertheless allows the algorithm to adaptively find optimal niche locations. Their technique is inspired by the economic model of

monopolistic competition, which describes the geographical interaction between customers and businessmen. Businessmen will distribute themselves among the customers so as to maximise their profit, while at the same time customers will go to the shop that minimises their costs. Both populations, customers and businessmen, therefore have separate interests, which leads to the placement of shops where customers benefit the most.

This economic model has some interesting properties which can be applied to an evolutionary optimisation scheme. The coevolutionary shared niching scheme is designed to form stable subpopulation of best solutions regardless of the solution spacing, extent and modality. The customer population may be viewed as the common population of solution candidates, searching for areas with a high fitness values through selection and recombination. The businessmen population has as primary aim to locate niches at optimal places. The businessmen population is a population of solution candidates as well, though the businessmen interact with the customer population to find those locations which yields them the highest payoff. Their fitness function enables them to place niches at highly fit regions of the search space. The interaction between the populations is accomplished through the separate fitness function for both populations. Keeping the original economic model in mind, we recall that customers will go to the store that is closest to them. This shop is defined by a businessman. Since an overcrowded shop is not desirable, customers will want to move towards other shops. This is actually a plain standard fitness sharing concept. The fitness function of the customers therefore is a modification of the standard fitness sharing scheme. Assuming that at generation t an individual c is served by businessman b, who has a total of $m_{b,t}$ customers, the fitness of an individual in the customer population is calculated as follows:

$$f'(c) = \left. \frac{f(c)}{m_{b,t}} \right|_{c \in C_b}$$

where C_b denotes the customer-set of businessman b. In short, a customer shares its fitness with the number of other customers that use its shop as well.

The businessmen determine the location and extent of the niches. They should therefore be located at peaks in the landscape with a certain minimal distance between them. This distance, denoted a d_{min}, is important to ensure a good distribution of the niches. A businessman's fitness is simply the sum of the raw fitness values of its customers and it is calculated as follows:

$$\phi(b) = \sum_{c \in Cb,t} f(c)$$

Goldberg and Wang suggested two schemes to accommodate evolution within the businessmen population, *simple CSN* and the *imprint* operator. In the simple CSN scheme, each businessman is chosen in turn, and a mutation on a randomly chosen site on the businessman string is performed. If the resulting businessman is an improvement over the original businessman, than the new individual replaces the original. Improvement is judged on two issues: a) the resulting individual have a higher fitness than the original and b) the resulting individuals should be at least d_{min} away from *all* other existing businessmen. This procedure is done up to a certain number of times. If none of the mutations prove to be an improvement, then the businessman is retained in the population in its original form. This scheme is effective on easy problems, but proves to be inadequate on hard and deceptive problems. An i*mprint* operator is suggested by Goldberg and Wang to enhance the performance. This operator simply chooses candidate businessmen from the customer population. If a customer proves to be an improvement over its competing businessman, the businessman is replaced by the customer, and the businessman is retained if none of the randomly chosen customer outperforms the businessman. This procedure is, again, performed up to a certain number of times for each businessman. This approach was more effective than the simple scheme and will therefore used in the

ERMOCS algorithm as well. The ERMOCS algorithm though uses a slightly modified version of the imprint operator, which we be described later on.

In (Goldberg & Wang, 1997) CSN has been tested and analysed on massive multimodal deceptive function (Deb, Horn & Goldberg, 1992). CSN (with imprint operator) proved to be a effective technique for optimisation on this complex landscape. In their paper, Goldberg & Wang proposed additional testing on other families of problems, notably multiobjective and real-world problems. When applied to multiobjective optimisation several issues have to be addressed. The original scheme uses the actual fitness values to calculate customer and businessman fitness. Of course, since solutions to multiobjective problems have several objective values, raw fitness values are not directly applicable. Consistent to the ranked-based selection we favoured earlier, a logical solution is to use ranks as fitness values. By doing so customer fitness is defined as the (transformed) rank divided by the niche count of the shop it attends. Consequently a businessman's fitness is the sum of the (transformed) ranks of all its customers.

A little less obvious is the measurements of distances in the multiobjective case. In this case it makes more sense to use phenotypic measurements in stead of genotypic measurements, like Goldberg & Wang used in their experiments. Pareto-optimality and dominance are defined in terms of objective fitness values. Therefore it is more logical to use phenotypic distance measurements. There is a slight issue here though. We have noticed that in our case simple Euclidean distance measurements suffice. This will only work, however, if both objectives are scaled to approximately the same domain size. If one objective is scaled to a much larger domain than the other objectives then consequently the movements in the direction of the larger-scaled domain will have less effect, and can disrupt the optimisation process.

The ERMOCS algorithm can be decomposed in four segments, *population creation, fitness calculations, recombinative schemes* and *imprint* operations. We will shortly describe these segments as to clarify the procedures within the algorithm.

a) create populations.

The algorithm uses integer strings for both populations. The genes correspond directly to the decision variables.

b) fitness calculation.

The algorithm uses ranked-based fitness assignment. The population is ranked in a single run, assigning a rank to each individual i according to 1 + pi, where pi is the number of individuals that dominate individual i. This rank is transformed into a fitness value by a simple linear transformation. Next for each individual in customer population a nearest businessman in population is determined. If all customers have been assigned a businessman, the shared fitness of each customer is calculated by dividing its rank-based fitness by the size of the customer-set of its businessman. The fitness of the businessmen is calculated by taking the sum of the rank-based fitness values of all the customer each businessman serves.

c) recombinative scheme

Pairs of individuals in the customer population are randomly selected to produce two offspring by crossover. During crossover mutation may occur at a rate of p_{mut}. The offspring are assigned a rank, based on the current customer population. This rank is transformed into a fitness value. Subsequently, for each offspring it is determined which businessman is nearest. The fitness values of the offspring is then divided by the number of customers the selected businessmen serve. The fitness values of the parents and offspring are compared to each other, and the two individuals with the highest fitness values are selected. This may imply replacement of parents by offspring. If however both offspring are inferior to the parents, both parents remain in the customer population. If an offspring enters the customer population, then the businessman to whom the parent belonged loses a customer and its fitness is consequently degraded. The businessman to who the offspring belongs gets one extra customer and consequently receives a higher fitness value. This procedure is done to keep the sharing information within the businessman population as accurate as possible.

d) imprint operation

If all customers have participated in an elitist tournament, an imprint operation is performed. For each customer in turn, random customers are chosen up to n times, and checked if they are an improvement over the competing businessman. Improvement is decided on the basis of dominance. If the selected customer dominates the businessman and is at least d_{min} away from all other existing businessmen, it is considered an improvement and replaces the original businessman. Unlike the previous segments, the fitness values of the businessmen and customers are not continuously updated. This is unnecessary, since in the next segment (*b*) all values are recalculated again.
After segment *d* the algorithm starts again at segment b until termination. The pseudo-code in figure 1 is a simple representation of the basic functions within the algorithm we described above.

ERMOCS algorithm

a. create customer population *C*
create businessmen population *B*

b. rank population C
find a businessman $b \in B$ for each customer $c \in C$
calculate fitness of each customer in population *C*
calculate fitness of each businessmen in population *B*

c. do for all individuals in customer population *C*:
select two individuals p1 and p2
perform crossover, producing o1 and o2
rank o1 and o2 on current customer population *C*
locate nearest businessman in population *B* for o1 and o2
calculate shared fitness for o1 and o2
perform elitist recombination between p1, p2, o1 and o2
if offspring is fitter than parents:
replace parent by offspring in population *C*
adjust fitness of affected businessmen in population *B*

d. perform imprint operation between customer population *C* and businessman population *B*
go to section b

Note that the fitness sharing scheme is imbedded in the fitness calculations of both populations. Furthermore, the selection scheme (section c) uses continuously updated sharing information. Each offspring that is created is assigned a rank and a shared fitness on the basis of the existing population, without actually entering the population. If an offspring is selected to be included in the population its rank and fitness are already up to date. The only adjustment that has to be made is a niche count correction: since one individual is replaced by another, a business loses one of its customers, while another gains a customer. On close inspection one can argue that the fitness information is not accurate at all times. This is essentially true, since during the selection procedure fitness values of existing, non-replaced individuals in the customer population are not fitness-wise corrected upon insertion of a new individual. This correction would require a complete re-ranking of the customer population and consequently a recalculation of all fitness values. and as an additional result recalculation of the fitness values of the businessmen population. To put in another way, procedure b in the algorithm should then be performed after every insertion of a new individual, which is computationally unacceptably expensive. Experimental investigation has proven that this is

not an extremely significant issue, since after each N/2 tournaments the fitness values of both population are recalculated by section b in the algorithm. Consequently the errors are reasonably limited.

The main focus of our algorithm is a robust and efficient optimisation of multiobjective real-world problems, which involves dealing with complex landscapes, hard constraints and specific requests regarding solutions. To this extend coevolutionary sharing and elitist recombination have been implemented in ERMOCS, and have empirically proven to be competent in practical multiobjective situations. In the next section we describe such an application, and look whether the algorithm lives up to our expectations.

4. Case Study: Scheduling of IR decoys

4.1 Problem description

The ERMOCS algorithm, as described above, has initially been designed to cope with the *resource scheduling* on board of naval ships, a time-limited and complex issue. The resource scheduling of a naval ship involves the resource management of weapons, both lethal and non-lethal, of the ship during one or more engagements. Its primary aim is such a deployment of the weapons that the threats are engaged as efficient as possible, i.e. with a maximised likelihood of 'killing' the threats (possibly distracting or deflecting them by means of 'electronic warfare') and a minimised expenditure of (scarce) resources. As the number and types of threats and the number of possible combinations and deployment times of weapons (and their possible positive and negative interactions) increases, the scheduling problem incurs a combinatorial explosion.

In this case study we focus on the optimisation of the deployment of *infrared decoys* (*flares*). This particular softkill scheduling problem is well understood in terms of goals and constraints, and the optimisation criteria are clearly defined. Decoys are intended to either minimise the target a ship presents to an enemy platform or to lure enemy weapons away from the ship. Flares present alternative targets to missiles guided by an infrared seeker. Since flares have quite a short lifetime, they must be positioned carefully and in sequence. An additional constraint is imposed by the small field of view of most infrared seekers, forcing a good alignment on the flare sequence with respect to the seeker's view.

Because electronic warfare is mainly a game of 'hide and seek', geometry and visibility play a fairly large part in it. So, the proper deployment of flares is an optimisation problem concerning the deployment times and angles of a *sequence* of flares, with the optimal deployment depending on the *geometry* of the current engagement (i.e. the bearing and velocity of the incoming missile, the heading and speed of the ship and the wind velocity). Because at least four to five flares must be launched in order to get a sufficient distance between the last flare and the ship, the number of dimensions of the search space is at least this large (it is quadrupled because there are four launchers available at four different angles). It can be further enlarged if we take manoeuvrable or trainable launchers in account. An exhaustive search of this search space (although it is bounded by maximum interval times between the flares) has proven to take too long so we are looking for ways of speeding up this search and genetic algorithms seem worthwhile to investigate for this speed-up. It goes without saying that with an on-line algorithm, a solution must be found before the missile hits the ship (and preferably a long time before that).

In an on-line scenario, the amount of search time that is available to any algorithm is at most a few seconds. The (sequential) ERMOCS genetic algorithm is in itself proficient to perform within such small amounts of time. If the problem domain is extended however, with for example manoeuvrable launchers, it may be necessary to employ a parallel version of the algorithm. Since genetic algorithms are by nature very suitable to be made parallel, this can be done relatively easy. This property of genetic algorithms is a major advantage over other non-evolutionary approaches.

4.2 Objectives and representation issues

A possible solution, a *deployment sequence*, consist of five variables, denoted as $<d_1, \ldots, d_5>$, which range from 0 to 20 seconds. These variables define the intervals between the deployments of flares, starting at the moment a

deployment is initiated. These intervals are our decision variables. The algorithm makes use of a simulation to calculate the effectiveness of a deployment. A deployment is judged on two objectives, a) the distance at which the missile passes the ship (known as Closest Point of Approach (CPA)) and b) a measure *P* of the actual deception of the missile.

A problem instance is defined by the bearing and velocity of the incoming missile, the heading and speed of the ship and the wind velocity. Each instance consequently has a different search space, each with different properties and location of the Pareto-optimal solutions. Additionally, each naval frigate has four possible launch-sites from which flares can be deployed. This adds up to a problem domain, in which exhaustive search is not an option. This search space has proven to be too large to be enumerated, especially since the scheduling has to be completed within a reasonably small amount of time.

The ERMOCS-algorithm has a few parameters which must be set correctly, notably population sizes, d_{min} and mutation rate. As for (customer) population sizing, Pareto-based multiobjective algorithms must have sufficiently large populations to be able to obtain a good sampling of the Pareto-optimal set. In our trials, a (customer) population size of 250 to 500 gave good results. Smaller populations may not be able to fill up all niches, which leads to incorrect representation of the Pareto-optimal front. On the other hand, large populations take up too much computational time due to the ranking procedures and since we are dealing with a time-constrained problem in this case, care has been taken not too use an extremely oversized population. With respect to the mutation rate, a value of approximately 0.005 proved to be efficient. Nevertheless, these values are educated guesses and may not be ideal in all situations. Further analysis should be done to find better ways to optimise the values. The d_{min} parameter, the minimal niche radius, is directly related to the size of the businessman population. For example, if we think of the Pareto-optimal set as solutions located an imaginary front of a certain length ***s***, the maximal number of businessmen that can be placed alongside this front is of course $\boldsymbol{s}/d_{min} + 1$. If the number of businessmen exceeds this number, some businessmen will be located at suboptimal places, and consequently multiple customers will be badly placed as well. If the number of businessmen too small, the Pareto-front will not be completely covered by solutions. Parts of the front will then be neglected, due to a too small number of allocated niches.

4.3 Experiments and Results

The algorithm has been tested on numerous instances of the softkill scheduling problem. As described earlier, a problem instance consist of values describing the bearing and velocity of the incoming missile, the heading and speed of the ship and the wind velocity. As a proof of concept we will apply the ERMOCS algorithm to a typical problem instance[1].

This particular, randomly chosen, instance is interesting, because it has some deceptive properties. We use an customer population of 250 and a business population size of 10 and the d_{min} is set at 10. The mutation is set at 0.005. We are only using one launcher this time, thus generating only a single problem instance. Figure 1[2] and figure 2 show respectively the customer and businessman population on generation 0 and generation 50. The number of times that possible candidates are selected from the customer population for imprint is set at 2. This proved to a reasonable value. A higher value slows down the algorithm.

[1] the problem instance variables are as follows: frigate speed: 5 m/s, frigate bearing: 0^0, wind speed: 10 m/s, wind bearing: -90^0, where a bearing of 0^o conforms to a eastbound direction; the missile bearing is -90^0 and is aimed directly at the frigate

[2] the area below the dashed line indicates areas where solutions are located; these areas have been determined by an exhaustive search

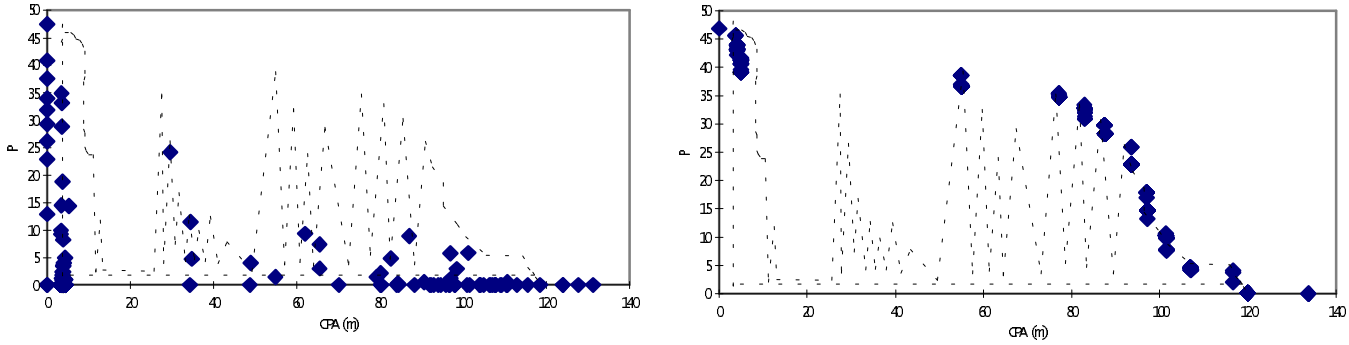

figure 1 - customer population at generation 0 (left) and at generation 50 (right)

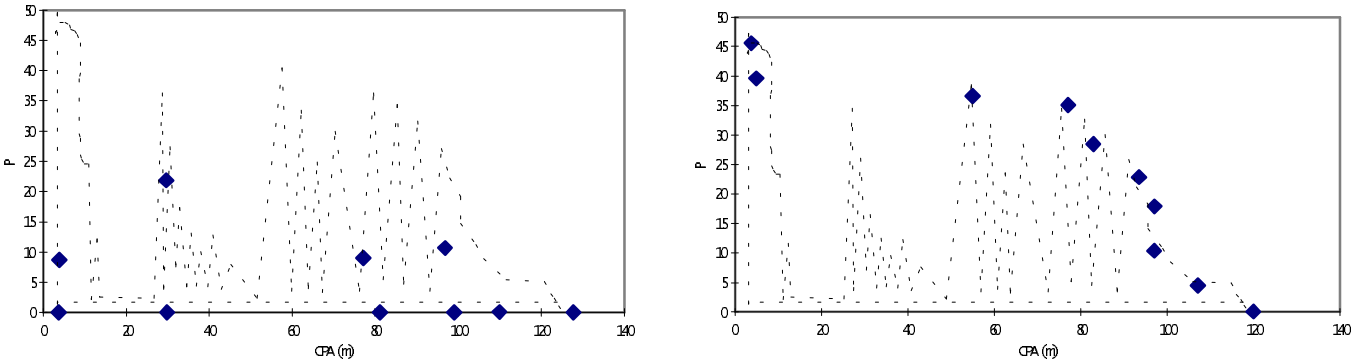

figure 2 - businessmen population at generation 0 (left) and at generation 50 (right)

Note, as one can see in the final population in figure 1, that the customer population has converged to the locations close to the centres of the optimal niches, as defined by the businessmen. This is caused by the elitist recombinative scheme, which results in a clean representation of the actual Pareto-set. The solution set the ERMOCS algorithm has found is an accurate representation of the actual Pareto-optimal set. The businessman population has, as expected, found the optimal niches which are located on top of the peaks. This has as a result that the customer population is optimally distributed over this area as well. The subpopulations are stable as well.

We have applied the ERMOCS algorithm on numerous instances of the problem. Other problem instances cause very different solution spaces, each having its own characteristics in terms of number of optimal niche locations and distribution. The ERMOCS algorithm has invariably shown to come up with, at least, a reasonable coverage of the Pareto-optimal front.

The d_{min} parameter is of great effect on the performance. In this experiment, several peaks are still uninhabited. If we increase d_{min} beyond the number of peaks, we have noticed increasingly more customers located at suboptimal niches. A d_{min} which is too small results in a less uniform distribution along the Pareto-optimal frontier and less coverage of this frontier. The setting of this parameter in conjunction with optimal number of businessmen, is where more research has to be done. These two parameters should be made more adaptive. For example, the amount of convergence of the customer population could be used as a measure in this matter. If after an amount of time still a lot of individuals are located at very much dominated regions, the number of businessmen may have to be reduced, for this may imply that there are some businessmen located suboptimally. It may interesting to use a combined d_{min}/business population size measure, since these two parameters are directly related to each other. A

single parameter of this kind would make an adaptive setting of d_{min} and business size easier as well. Nevertheless, the setting of d_{min} and the businessman population size is more comprehensive and translucid than setting the niche size σ_{share} and thus enhances the applicability of the technique.

5. Conclusion & discussion

This paper introduces a new multiobjective algorithm, which integrates rank based selection, adaptive niching through coevolutionary sharing, elitist recombination, and non-dominated sorting, and is called ERMOCS. The primary aim of the algorithm is to provide a robust and efficient way to optimise real-world problems and it seems that ERMOCS performs quite well in these environments.

We have tested the algorithm on a softkill scheduling problem. The algorithm performs like intended, with a steady efficiency and robustness. It has shown be efficient, considering the quality of non-dominated solutions it produces. Furthermore it has demonstrated its robustness, because of its capabilities on numerous problem instances, each with varying properties in terms of search space properties. Still, the results are only an indication of the performance of the algorithm since it has not been thoroughly analysed and the experiments have been rudimentary. Nevertheless, the algorithm has exhibited promising properties and results. It has shown to be capable to produce a large set of non-dominated solutions, which can be effectively processed and judged by an external decision maker. Coevolutionary Shared Niching (CSN) in particular appears to be a beneficial addition to the field of multiobjective optimisation. Its capability to adaptively form niches of varying extent and location has a beneficial effect on the performance of this algorithm.

There are still some issues though that have to be resolved. As mentioned before, the d_{min} parameter in conjunction with the businessman population size setting should be made more adaptively, since these still have a large effect on the performance of the algorithm. Furthermore, extensive testing should be done to investigate performance under more complex domains. In the context of the case study, we could, for example add manoeuvrability to the launchers, which would increase the complexity extensively. Another issue that should be studied is the optimisation of the speed of the algorithm. In its current sequential form it can be employed in an on-line situation. With increasing complexity of the problem domain, one should consider using more efficient programming and perhaps a parallel version of the ERMOCS algorithm.

References

Coello, C.A., *An Updated Survey of GA-Based Multiobjective Optimization Techniques*, Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México, december 1998.

Fonseca, C.M., and Fleming, P.J., Genetic Algorithms for Multiobjective Optimization: formulation, discussion and generalization. *Proceeding of the fifth International Conference on Genetic Algorithms*, pp. 416-423, Morgan-Kauffman, 1993.

Fonseca, C.M. and Fleming, P.J., *An overview of evolutionary algorithms in multiobjective optimization*, Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.

Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

Goldberg, D.E. and Richardson, J.E., Genetic Algorithms with Sharing for Multimodal Function Optimization, *Proceedings of the Second International Conference on Genetic Algorithms*, 41-49, 1987.

Goldberg, D.E. and Wang, L., Adaptive Niching via coevolutionary Sharing. In Quagliarella, D., Periaux, J., Poloni, C. and Winter, G. (eds.), *Genetic Algorithms and Evolutionairy Strategies in Engineering and Computer Science*, pp. 21-38, John Wiley and Sons, Chichester, 1998.

Horn, J., Multicriteria Decision Making, In *Handbook of Evolutionary Computation*, Edited by Thomas Baeck, University of Dortmund, Germany, D B Fogel, Natural Selection Inc, USA, and Z Michalewicz, University of North Carolina at Charlotte, USA, 1996.

Horn, J., Nafpliotis, N. and Goldberg D.E., A Niched Pareto Genetic Algorithm for Multiobjective Optimization, *Proceeding of the first IEEE Conference on Evolutionary Computation*, 82-87, 1994.

Cunha G., Oliveira, P., and Covas J.A., Use of Genetic Algorithms in Multicriteria Optimization to Solve Industrial Problems. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 682-688, San Mateo, California, Michigan State University, Morgan Kaufmann Publishers, 1997.

Loughlin, D.H. and Ranjithan, S., . The Neighborhood constraint method: A Genetic Algorithm-Based Multiobjective Optimization Technique. *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp. 666-673, San Mateo, California, Michigan State University, Morgan Kaufmann Publishers, 1997.

Obayashi, S. Takahashi, and Y. Takeguchi. Niching and Elitist Models for MOGAs, In A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature -- PPSN V*, pages 260-269, Amsterdam, Holland, 1998.

Obayashi, S, Tsukahara, T., and Nakamura, T., Cascade Airfoil Design by Multiobjective Genetic Algorithms. *Proceedings of the Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 24-29, September 1997.

Oei, C.K., Goldberg, D.E. and Chang, S., *Tournament Selection Niching and the Preservation of Diversity*, Technical Report No. 91011, University of Illinois, Urbana-Champaign, 1991.

Richardson, J.T., Palmer, M.R., Liepins, G., and Hilliard, M., Some guidelines for genetic algorithms with penalty functions. *Proceedings of the third International Conference on Genetic Algorithms*, pp. 191-197, Morgan-Kaufman, 1989.

Shaffer, J.D., Multiple objective optimization with vector evaluated genetic algorithms, *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 93-100, Lawrence Erlbaum, 1985.

Srinivas, N. and Deb, K., Multiobjective Optimization Using Non-dominated Sorting in Genetic Algorithms, *Evolutionary Computation*, volume 2(3), pp. 221-248, fall 1994.

Thierens, D., and Goldberg, D.E. (1994), Elitist recombination: an integrated selection recombination genetic algorithm, *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 508-512, 1994.

Todd, D.S. and Sen, P., A Multiple Criteria Genetic Algorithm for Containership Loading, In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 674-681, San Mateo, California, Michigan State University, Morgan Kaufmann, 1997.