Control and Cybernetics

vol. **26** (1997) No. 3

The COMOGA method: constrained optimisation by multi-objective genetic algorithms

by

Patrick D. Surry and Nicholas J. Radcliffe

Quadstone Ltd, 16 Chester Street, Edinburgh, EH3 7RA, UK,
and: Department of Mathematics, University of Edinburgh,
King's Buildings, EH9 3JZ, UK,
e-mail: {pds,njr}@quadstone.co.uk

Abstract: This paper describes a novel method for attacking constrained optimisation problems with evolutionary algorithms, and demonstrates its effectiveness over a range of problems. COMOGA (Constrained Optimisation by Multi-Objective Genetic Algorithms) combines two evolutionary techniques for multi-objective optimisation with a simple regulatory mechanism to produce a constrained optimisation method. It shares the universal applicability of penalty-function approaches, but requires significantly fewer free control parameters.

COMOGA takes a dual perspective, considering a constrained optimisation problem sometimes as a constraint satisfaction problem, and sometimes as an unconstrained optimisation problem. These two formulations are treated simultaneously, using a single population, by basing each selection decision on the basis of either constraint violation or function value. A simple adaptive feedback mechanism couples the two formulations by adjusting the relative likelihood of these choices. Unlike penalty function approaches, COMOGA dynamically adapts the emphasis on constraint satisfaction and objective function value as the optimisation proceeds, usually yielding final populations which are both feasible and highly fit.

COMOGA has been successfully applied to real industrial problems with comparable performance to highly tuned penalty function approaches. On a test suite of constrained problems previously studied by Michalewicz, application of COMOGA required minimal effort but proved superior to all previous evolutionary methods known to have been applied; indeed it was the only method which found feasible solutions in every run for every problem.

Keywords: COMOGA, constrained optimization, multiobjective optimization, penalty functions, Pareto optimality, evolutionary algorithms, genetic algorithms

1. Introduction

It is a frequent criticism of evolutionary algorithms that published results are usually obtained with contrived problems without constraints, leading to the suggestion that evolutionary methods are unsuitable for tackling complex constrained optimisation problems. Within the community, there is a wide-spread perception that penalty function methods are a rather blunt instrument for handling general constraints (e.g. Michalewicz, 1992), exhibiting great sensitivity to the values of their many free parameters, and feeding rather too little information back to the algorithm to allow it to handle the constraints satisfactorily. While other methods are available for problems with explicit constraints (including repair methods, Davis & Orvosh, 1993a; smart decoders, Davis, 1987a;1991a; and special operators incorporating problem knowledge, Michalewicz & Janikow, 1991), these do not have fully general applicability, and tend to require significant work for each new class of problems tackled. There is thus a need for a method that combines the generality of penalty function approaches with a greater feedback of information to the underlying search algorithm about the way in which progress is being made with the various constraints under consideration.

In this paper we present such a method, COMOGA, based on ideas for multi-objective optimisation. Section 2 presents an introduction to constrained optimisation and constraint satisfaction problems, and surveys the evolutionary techniques which have been used to tackle such problems. Multi-objective optimisation is reviewed in Section 3, where the natural fit with population-based algorithms is discussed, and a link is made between multi-criterion optimisation and constraint satisfaction. In Section 4 these two strands are drawn together to motivate the COMOGA method, which is then described in detail. The technique is demonstrated for a gas-network problem in Section 5, and results for a test suite of constrained optimisation problems previously studied by Michalewicz arc summarised in Section 6.

2. Constrained optimisation

2.1. Formulation

Many optimisation problems can be phrased as the minimisation[1] of a given function f over a search domain S:

Minimise

$$f : S - d$$

sv,b.fect to [the sol11,tion x E S satisfying certain eqnalities or ineqv,alities).

[1] In this paper we assume, whenever appropriate, without loss of generality, that problems are cast as minimisation problems.

The equations or inequalities that the solution x must satisfy are known as constraints. Solutions which satisfy all constraints are said to be *feasible* and the set of all such solutions is called the *feasible region,* S_F. (Thus the set of optima, S^*, is a subset of S_F.) In a *constraint satisfaction problem,* the objective function f is discarded, with the goal being simply to find any solution in S_F.

Constraints can be characterised in various ways. An inequality constraint is said to be *active* at a point x if it is satisfied as an equality at x. It is typical in constrained optimisation that a number of constraints are active for optimal solutions, so that S^* is at the boundary of the feasible and infeasible regions (with the result that any weakening of the constraints would change S^*). Indeed, in highly constrained problems it is often the case that all feasible solutions are near this boundary, and that the volume of the feasible region is negligible when compared with that of the unconstrained search domain. (Although such problems arc typically difficult, the converse is clearly not true, with many difficult constrained problems having large feasible regions.)

The constraints on x are conveniently divided into two (imprecise) categories- implicit and explicit. Explicit constraints arc those which can be reduced to simple conditions on x and arc verifiable "by inspection" while implicit constraints are those which specify a condition on some function of x that requires significant computation to verify (comparable to, or perhaps much greater than, computing $f(x)$). For example, $x_1 < 3$ and $x_1 + x_2 = x_3$ would normally be regarded as explicit constraints while a condition such as 'the pressure on the wing should not exceed $3N/m^2$', where the pressure is computed by a fluid-flow simulation, would be regarded as an implicit constraint. The distinction is useful because genetic operators can usually be constructed that respect explicit constraints whereas this is impractical for implicit constraints.

2.2. Evolutionary approaches

There are several main approaches for tackling a constrained optimisation problem using an evolutionary algorithm, with varying degrees of generality (see for example the survey in Michalewicz, 1995b). Perhaps the simplest idea is to restrict the search to the feasible region. This can be done by rejecting infeasible solutions outright, by using greedy decoders or repair mechanisms, or by designing specialised operators that incorporate knowledge of the constraints. The search is thus reformulated as an unconstrained optimisation problem over the reduced space S_F (the *feasible* region), which is illustrated in Fig. 1. The diagram shows the image of the search space under the vector-valued function I: S----+ ($^+$)^2 x where I (c_1, c_2, f) with $c_i(x)$ measuring the degree to which x violates the ith constraint and $f(x)$ giving its cost (to be minimised).

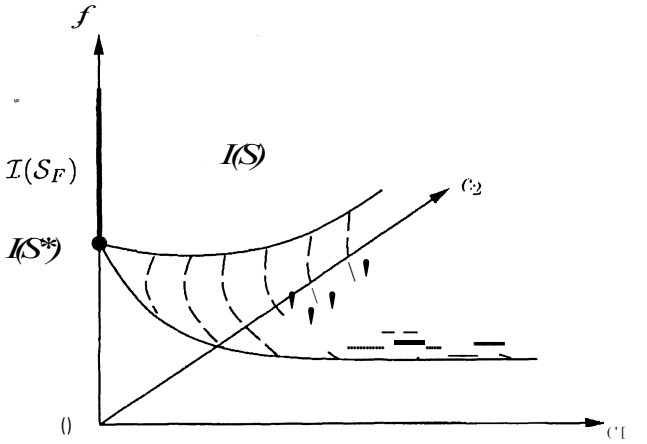

Figure 1. A constrained optimisation problem in which the objective is to minimise cost f while satisfying two implicit constraints is illustrated. Points in the search space S are shown under the three-dimensional mapping I (c1, c2, f). The c_1 and c_2 axes measure the degree of constraint violation, so that points in the feasible region, S_F, are mapped to the line where both are zero. The desired set of optima S^* is the set of feasible points with minimum cost. All solutions in S are mapped to points on and above the shaded surface.

2.2.1. Restricting search to the feasible region

Rejection of infeasible solutions is generally applicable, but is typically limited in its practical utility, due to the low density of feasible solutions in practical problems (e.g. densities of 1 in 10^{10} are not uncommon). In order to avoid generating and rejecting large numbers of infeasible solutions, *greedy decoders* can be used, in which a problem-specific growth function is designed. Here, the genotype does not directly encode a solution in S but rather a set of parameters which is used by the decoder to generate a feasible solution. Because the decoder must be guaranteed to never produce infeasible solutions (regardless of the provided parameters), it is often extremely difficult to design. Moreover, it is typically hard to generate a decoder that can be guaranteed to be capable of generating optimal or near optimal solutions.

A related approach is to use repair mechanisms to produce feasible solutions from infeasible ones, mapping $S \longrightarrow S_F$. (Here, genotypes directly represent solutions in S). This requires a problem-dependent operator which is able to modify any infeasible solution in such a way as to produce a (nearby) feasible solution. Again this is clearly difficult for many types of constraints. When such mechanisms are employed, a further choice is available, namely whether to write the repaired solution back to the genome, or to use it during evaluation, but then to leave the (infeasible) genome intact. The former approach, which is known as Lamarckism, has the advantage of generally allowing faster local improvement, but can make it harder for the search to traverse infeasible regions of the search space, particularly when S_F is strongly disconnected with respect to the genetic operators. The latter approach, (which has some parallels with the Baldwin effect) has converse advantages and disadvantages. Davis & Orvosh (1993a) present anecdotal empirical evidence that writing the repaired solution back to the genome probabilistically, about 5% of the time, is a good option, and the ideas are further explored by Whitley & Gordon & Mathias (1994a).

The final suggestion is to build and use genetic operators that "understand" the constraints, in the sense that the syntactic actions of the operators never produce infeasible solutions (ones that violate the constraints). This approach is advocated by Michalewicz (1992), and Radcliffe (1992d). Michalewicz & Janikow (1991) have shown how genetic operators can be built that "understand" linear constraints in this sense, and Schoenauer & Michalewicz (1996a) construct operators which maintain solutions on nonlinear analytical constraint surfaces; however, it is clear that for many types of constraints (particularly implicit ones) this approach is impractical.

2.2.2. Exploring the infeasible region

In a problem with implicit constraints, it is often at least as difficult to determine whether a solution is feasible as to evaluate its cost. In such a case it is typically impossible to utilise repair mechanisms or greedy decoders and

impractical to restrict search by simply rejecting infeasible solutions, and we must take an alternative approach. We arc forced to consider infeasible solutions during search, which has the advantage of simplifying the algorithm and the operators, but admits the possibility that no feasible solution is ever discovered. By exploring infeasible solutions, the goal is to drive the search towards the feasible region. Particularly in problems with many active constraints at optima, we hope to approach optima from 'both sides', that is to find nearly feasible solutions with better than optimal function values along with feasible solutions with nearly optimal function values (indeed, this is the explicit goal of the segregated genetic algorithm of LcRichc & Knopf-Lenoir & Haftka, 1995a). Thus, it has been stated that:

> [g]ood search should approach the optimal from both sides of the feasible/infeasible border (Richardson, Palmer, Liepins, Hilliard, 1989).

The problem then becomes how to compare feasible and infeasible solutions, since in the final analysis it is only feasible solutions which are acceptable. The most widely applicable approach is to employ a penalty function. Here, the search is treated as an unconstrained problem over S but the objective function is modified for infeasible solutions by adding terms which degrade their performance. In general, the size of penalty added reflects in some way the *degree* of constraint violation (for example the sum of the constraint violation for each constraint). It is also reasonably standard practice (e.g. Richardson, Palmer, Liepins, Hilliard, 1989; Michalewicz, 1992) to increase the size of penalties during the course of a run (reverse annealing), so that while a degree of violation is tolerated in early generations, this tolerance reduces over time. This ensures that, after sufficient time, the optimal solutions to the unconstrained problem using the modified objective function coincide with the optimal solutions to the original constrained problem.

Although penalty functions are essentially universally applicable, they exhibit a number of drawbacks. First, they are weak, in the (formal) sense that they do not provide any problem-specific information to the algorithm. This contrasts with repair mechanisms and problem-specific move operators that exploit understanding of the constraints to provide stronger guidance to the algorithm, but such techniques arc not applicable for general constraints. Secondly, the choice of weighting for the constraints is a somewhat subtle matter, particularly when there are many, and increases yet further the number of free parameters to the evolutionary algorithm. Because any choice of parameters determines a fixed form for the modified function in the infeasible region, it induces a fixed ranking on all infeasible solutions. This limits the way in which the search algorithm can explore the infeasible region, since fixed tradeoffs between the various constraints have been introduced. The resulting quality of solution obtained in fact, the likelihood of finding *any* feasible solution rnay be extremely sensitive to the values chosen.

In the next section, we present a method, previously discussed in the context of a specific optimisation problem (Surry, Radcliffe, Boyd, 1995a), which avoids this difficulty by appealing to the methods of multi-criterion optimisation. Some initial exploration has taken place in this area. For example the work of Schoenauer & Xanthakis (1994a) treats each constraint in turn to avoid amalgamating them. Richardson, Palmer, Liepins, Hilliard (1989) suggest the possibility of using multi-objective techniques (using fitness and either the sum or the number of constraint violations as two objectives) but claim to have been plagued by difficulties. In fact, Chu & Beasley (1995a) implement a scheme similar to this to deal with a single constraint (using what they term fitness and unfitness). However, they give no guidance in dealing with multiple, non-commensurate constraints, other than by combining them using what is essentially a penalty function (see Section 3). The method we propose in Section 4 presents a novel solution to this problem.

3. Multi-objective optimisation

3.1. Formulation

In many real-world optimisation problems there is not a single objective but a set of criteria against which a solution may be measured. Such problems are often known as *mv,lti-objective* or *multi-criterion* optimisation problems, and are defined by a set of objective functions Ji, h ... ,JN over the search space S each of which should ideally be minimised.

Perhaps the most common approach to multi-criterion optimisation is to form a new objective function F that is a weighted sum of the individual objectives,

$$F = \sum_{i=1}^{N} a_i f_i, \qquad a_i \in \mathbb{R}^+$$

and to seek to minimise this sum.

If there exists a solution $x^* \in S$ that simultaneously succeeds in minimising each of the Ii, this approach can be reasonably satisfactory, because in this case, successful optimisation of F will also optimise each fi- In the more general case, however, the component objectives fi will *compete,* in the sense that improvement against one will in some cases require a degradation against another. In this case, the approach of forming a weighted sum is less attractive, because the choice of weights ai will determine the trade-off between the various component objectives that optima of the combined function F will exhibit. This is particularly unsatisfactory in cases where the various objectives are *noncommensnrate,* in the sense that trade-offs between them are either arbitrary or meaningless. A good example of this might arise when seeking to maximise profit while minimising ecological damage, where most people would accept that any assignation of economic cost to ecological damage is at best arbitrary.

In the case of multi-objective problems with competing, non-commensurate criteria, a more satisfactory approach is to search not for a single solution but for that set of solutions that represent the "best possible trade-offs." Such solutions arc said to be *Pareto-optimal* (after Vilfrcdo Parcto who first advanced the concept), and arc characterised by introducing the notion of domination. A solution x is said to *dominate* another solution y if its performance against each of the objective functions is at least as good as that of y, and its performance is better against at least one objective, i.e. if and only if

$$\forall i \in \{1, 2, \ldots, n\} : \quad f_i(x) \leq f_i(Y)$$
$$\text{and} \quad j \in \{1, 2, \ldots, n\} : \quad f_1(x) < f_1(y).$$

Clearly in this case, x may reasonably be said to be a superior solution *toy*. The *Pareto-optimal set* (or *front) P* is the set of solutions that arc not dominated by any other solution in the search space, i.e.

$$P \quad \{x \in S \mid \quad y \in S : y \text{ dominates } x\}.$$

3.2. Evolutionary approaches

Although it is difficult to attack multi-criterion problems with traditional optimisation methods, it is relatively natural in population-based search algorithms to consider trying to use the population to hold solutions that represent different trade-offs. Reasonably simple modifications to the selection (and perhaps the replacement) method may be all that is required to effect this. A number of schemes have been proposed, most of which arc based around the notion of only allowing selective advantage between solutions when one dominates another. Fonseca & Fleming (1995a) provide an overview of many such techniques. The effectiveness of these methods is further enhanced when combined with some form of niching, to encourage greater diversity in the population. Niching methods include structured population models (e.g. Norman, 1988b; Mandcrick & Spicsscns, 1989a; Gorgcs-Schlcutcr, 1989a), sharing (Goldberg, Richardson, 1987a), and crowding (Cavicchio, 1970; DcJong, 1975).

3.3. Constraint satisfaction as multi-criterion optimisation

It is clear that the constraint satisfaction problem is (formally) equivalent to the simple class of multi-objective problems (discussed above) in which all objectives can be minimised simultaneously. We measure the degree of constraint violation for each constraint (or group of commensurate constraints) and treat each of these as an objective in a multi-criterion problem. (Consider again Fig. 1, where we arc now only required to find a solution in Sp. The dashed lines, if extended upwards as vertical manifolds, might indicate a series of progressively dominating surfaces, converging on $I(S_F)$ - the Parcto-optimal set in this case.) Although, in such a case, minimising a penalty function expressing the degree

of constraint violation would be the most common approach, we suggest that a more appropriate strategy when using evolutionary techniques is to use the simple techniques for general multi-criterion optimisation discussed above in order to exploit the ability of the population to hold many different possible trade-offs between the constraints. This allows the algorithm to dynamically "discover" an appropriate trajectory by which to approach the feasible region, rather than arbitrarily assigning the relative importance of different combinations of constraint violations.

In the next section we show how this idea can then be extended to the constrained optimisation problem, where not only must several constraints be satisfied, but a given objective function f must also be minimised.

4. The COMOGA approach

4.1. Motivation

It was pointed out in Section 2 that if we choose to use a penalty function with some given set of parameters to attack a constrained optimisation problem, we make an *a priori* decision about the relative importance of different degrees of constraint violation, regardless of their actual difficulty to satisfy. Further, by combining the degree of constraint violation with the objective function value we impose fixed choice concerning the tradeoff between constraint satisfaction and optimisation.

In Section 3, we showed that the methods of evolutionary multi-objective optimisation can be applied directly to the constraint satisfaction problem, and proposed that we could apply similar ideas to constrained optimisation. Obviously, the situation is complicated somewhat by the additional requirement of minimising some function over the feasible region. Here we think of f as an extra criterion which is of less importance than any of the "constraint criteria", i.e. there is no acceptable trade-off between minimising (satisfying) the constraints, and minimising f.

The approach encapsulated in COMOGA is to view a constrained optimisation problem alternatively as a constraint satisfaction problem (ignoring the objective function) and as an unconstrained optimisation problem (ignoring the constraints). We then decide adaptively which view to take at any instant based on the relative success with respect to the two formulations. In order to find near-optimal solutions, we must be careful to get neither "too far" from feasibility nor "too far" from optimal fitness, while also recognising that constraint satisfaction is more important than optimisation (as ultimately we are only interested in feasible solutions). We show that an adaptive population-based algorithm is ideal for this purpose.

There arc numerous approaches to unconstrained optimisation using a population-based algorithm, and various techniques for constraint-satisfaction based on multi-criterion evolutionary algorithms have been discussed above. The dif-

ference between these two types of algorithms can typically be ascribed solely to the selection (and replacement) regime-in the first case selective decisions are based on fitness (cost) while in the second they are normally based on some form of Pareto ranking.

This motivates an attractively simple scheme for attacking the combined constrained optimisation problem, in which we use a single algorithm but randomly decide each time a selective decision must be made whether to consider the problem as a constraint satisfaction problem or as an unconstrained optimisation problem. We then adjust the relative likelihood of adopting each view using a simple feedback mechanism that tries to maintain a fixed fraction of the population in the feasible region. Because individual solutions can be selected on the basis of either constraint satisfaction or cost, this results in an algorithm which aims to dynamically explore the boundary between feasibility and infeasibility *without* arbitrary penalty factors fixing the relative quality of the different achievable tradeoffs.

This can be seen as a generalisation of the scheme recently proposed by Chu & Beasley (1995a) which can importantly handle more than one constraint without having to amalgamate them.

4.2. Algorithm

To treat the constraint satisfactio.n aspect of the problem, we can conceptually label all members of the search space S with some measure of their Pareto ranking based on constraint violation, either by conceptually peeling off successive non-dominating layers (Goldberg, 1989a), or by assigning to each solution a "rank" equal to the number of solutions which dominate it (Fonseca & Fleming, 1993a). (The latter scheme has the advantage that it is easy to subtract the effect of a deleted individual and add the effect of a new individual without re-ranking the entire population.) Note that this ranking is a dynamic one, based on the current population of achievable constraint tradeoffs rather than a fixed ranking of any possible combination of constraint violations. We denote this *population-dependent* ranking function $R : (\ ^{+})^{N} \text{--t}\ z^{+}$, where N is the number of constraints.

From the unconstrained optimisation view, every solution has some cost value associated with it. Thus we are presented with a dual view of each solution in the population and can form the two-dimensional mapping $I_R : S \text{--t}\ z^{+}$ xJR, with 'IR (Ro (c1, . . . , CN), ff. This reduces the problem to the two-objective problem illustrated in Fig. 2. We must couple the two viewpoints in order to solve the combined constrained optimisation problem: in solving the constraint satisfaction problem we minimize along the R axis and in solving the unconstrained optimisation problem we minimize along the f axis. However, we desire not simply solutions on the Pareto-optimal surface P, but rather solutions in the intersection of the Pareto-optimal set with the feasible region (as constraint satisfaction is "more important" than cost minimisation).

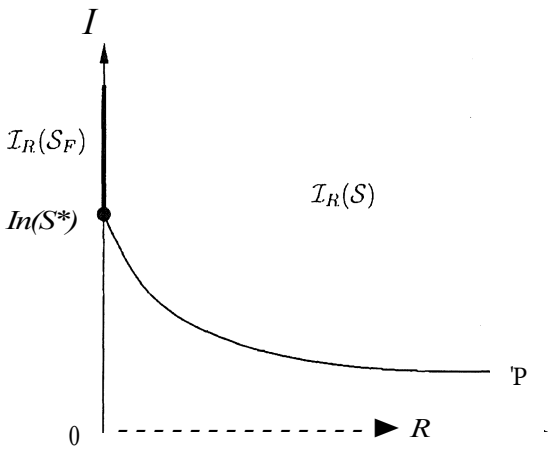

Figure 2. The constrained optimisation problem with N constraints can be recast as a two-objective problem by assigning a Pareto rank based on constraint violation. The Pareto-optimal set 'P is the set of non-dominated solutions under Γ_R (Ro(c $_{1,\ldots,}$ cN), The feasible set is mapped to the line segment Γ_R (S_F), and the desired set of optima is mapped to their intersection, Γ_R (S*). The search space S is mapped to points on and above P.

One possible approach is to use a sub-ranking scheme, where only solutions with equal Pareto rank for constraints arc distinguished on the basis of cost. However, this is likely to result in an evolutionary process which first concentrates on the constraint satisfaction problem (hence sampling points in the feasible region essentially at random) and only once this is solved tries to reduce cost. This "approach from above" not only lacks the desirable property of being able to combine low-cost, nearly-feasible solutions with higher-cost feasible ones, but may be an extremely poor way to search S_F if it is a highly sparse and disconnected subset of S.

An appealing alternative approach is to enlist the ideas of Schaffer (1985). In his *vector evaluated genetic algorithm* (VEGA), he selects some fraction (typically 1/k) of the population based on each of the k objective functions. When a fixed fraction is used for each objective (e.g. 1/k), this tends to favour the development of "specialist" subpopulations that excel in one objective function, particularly when fitness-proportionate selection is used (Richardson, Palmer, Liepins, Hilliard, 1989). However, COMOGA will actively exploit this tendency by adaptively changing the likelihood of selecting with respect to each objective.

The suggestion in our case is to use, for example, tournament selection (Goldberg, 1989a), sometimes basing the tournament on cost f and sometimes on the Pareto ranking R with respect to constraint violation. (In cases where the selected attributes are equal, the other attribute is compared.) A probability p_{cost}

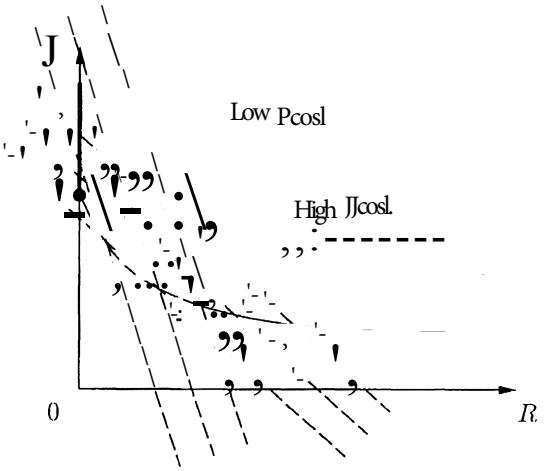

Figure 3. Using a VEGA-like scheme of selecting probabilistically with respect to one of the two objectives (cost or constraint rank), we induce a perceived fitness of some population-dependent weighted combination of the two objectives. As P_{cost} tends to zero, the scheme favours constraint rank more, and cost less. By adaptively changing P_{cost} based on the proportion of feasible solutions observed in the population, the algorithm dynamically discovers how to achieve constraint satisfaction and minimisation simultaneously.

is used to determine the likelihood of selection with respect to cost, and will be adapted as the algorithm progresses. Any fixed value of P_{cost} will induce an overall probability of reproduction equal to some linear combination of the reproductive probabilities with respect to the two attributes, with *population-dependent* weights. Although such a fixed P_{cost} may favour convergence to some non-feasible point on the Pareto-optimal curve, it is clear that as $P_{cost} \to 0$, the process increasingly favours constraint rank until in the limit of $P_{cost} = 0$ we are essentially solving the constraint-satisfaction problem; seeking feasible solutions regardless of cost (unless the constraint rankings arc equal this is equivalent to the sub-ranking approach described above). We thus hope that some intermediate non-zero value will allow us to find feasible solutions of low cost. This is illustrated in Fig. 3.

To avoid the problem of fixing a particular value for P_{cost}, we propose to change the value adaptively by setting a target proportion T of feasible solutions in the population (T is similar to the flip threshold of Schoenauer, Xanthakis, 1994a). We start by choosing some arbitrary value for P_{cost}, say 0.5, and some desired proportion of feasible solutions, e.g. $T = 0.1$. After each generation, if the number of feasible solutions in the population is not close to T, then we

adjust P_{cost} up or down accordingly: if the actual proportion is too low, we decrease P_{cost}, e.g.

$$P_{cost} \leftarrow (1 - c)P_{cost}, \tag{1}$$

and conversely, if the proportion is too high, we increase it, e.g.

$$P_{cost} \leftarrow 1 - (1 - Pcost)(1 - c). \tag{2}$$

This does, of course, introduce several new parameters to the algorithm (though notably fewer than a penalty function), which we were trying to avoid, but we find in practice that the scheme is remarkably robust to their values, in contrast to the sensitivity of penalty function parameters. This leads to the COMOGA method, which is summarised below.

The COMOGA Method

1. Calculate constraint violations $(c_1, \ldots, c_N)$ for all solutions.
2. Establish Pareto rank based on constraint violations (e.g. by counting the number of members of the population dominated by each solution).
3. Evaluate the cost (fitness) of solutions.
4. Select an (expected) proportion P_{cost} of parents based on cost, and the others based on constraint ranking.
5. Apply the genetic move operators (recombination, mutation etc.)
6. Replace an (expected) proportion P_{cost} of solutions based on cost, and the others based on constraint ranking.
7. Adjust P_{cost} if the proportion of feasible solutions in the population is not close to the target proportion, τ, according to equations (1) and (2). Lowering P_{cost} favours feasible solutions and raising it favours lower cost solutions.

Typical values for the parameters are $\tau = 0.1$ and $c = 0.1$, with $P_{cost} = 0.5$ initially.

4.3. Discussion

The COMOGA scheme has several attractive features. First, and foremost, it removes the necessity for the many parameters of a penalty function which must be determined empirically. Secondly, it turns the acknowledged weaknesses of VEGA to favour extreme solutions to advantage, as in our case we are only ultimately interested in solutions which excel at constraint satisfaction (have low constraint rank). Thirdly, the adaptive approach to specifying P_{cost} allows the algorithm to find its own trajectory to approach the desired optimal values.

In practice, it is important to incorporate explicit constraints (e.g. linear or other specific nonlinear ones) where possible, and to amalgamate multiple commensurate constraints in order to reduce the dimensionality of the Pareto-optimal front. It may also be advantageous to incorporate niching or other diversity promoting measures in the algorithm (in the work presented here we have simply enforced uniqueness which is not likely to be highly effective for real parameter optimisation).

5. An illustrative application

We will illustrate the application of the COMOGA approach to a gas-network pipe-sizing problem, contrasting the results with a penalty-function approach. This work has been previously reported in grnater detail (Surry, Radcliffe, Boyd, 1995a).

The problem involves determining the diameters of pipes in a fixed-topology network (with fixed supplies and demands), in order to minimise expenditure, while satisfying two implicit constraints defining the minimal pressure at each node along with an engineering requirement that each pipe should have at least one upstream pipe of the same or greater diameter.

In the particular problem considered, the network contained 25 pipes, each of which could be selected from six possible sizes, giving rise to a search space of size $6^{25} \quad 3 \times 10^{19}$. The network is a real one, which was actually built, with pipe sizes determined using a greedy heuristic method. (Both evolutionary methods outperformed the existing gTeedy heuristic for the problem, giving identical results.) The density of valid networks (ones which satisfy the constraints) in the search space is extremely low-random sampling of more than 3×10^7 points produced only a single admissible configuration.

Because of the implicit nature of the constraints in the pipe-sizing problem, the only applicable conventional approach is to use a penalty function, as it would be extremely difficult to construct genetic operators that respected them, and prohibitively expensive to use a repair mechanism (if indeed one could be constructed). In order to compare this approach with the COMOGA technique of modifying the selection regime, a standard steady-state elitist duplicate-free evolutionary algorithm using an integer-valued representation and standard recombination and mutation operators was employed (Surry, Radcliffe, Boyd, 1995a). Binary tournament selection with parameter 1.0 was used to select parents, and the resulting child was re-inserted using a replace-worst scheme. Tournament replacement was also investigated, but the more aggressive replace-worst strategy proved superior.

In the first case, an annealed penalty function which combined a time-dependent weighted sum of the degree of constraint violation for the two constraints along with the basic cost function value was used as the objective. This involved six control parameters to incorporate the two constraints. A wide range of penalty function parameters were tested to discover the typical relative values

of constraint violations, etc. As has been widely reported previously, the quality of the resulting algorithm is highly sensitive to these values, with small changes often resulting in runs in which no feasible solution was found.

The technique (with good parameters) produced consistently good results, although it did not always converge to the same optimal solution. In most cases it found networks which were better, often significantly so, than that determined by the heuristic approach. In almost all cases the algorithm found a valid network by the end of the run (i.e. one in which the penalty terms were zero). A snapshot of a single successful run using the penalty function is discussed in Fig. 4.

The same algorithm was then adapted to the COMOGA approach, as described in Section 4. Each member of the initial population was assigned a rank according to constraint violation by counting the number of members in the population by which it was dominated. The same selection and replacement regime was used, but with decisions based on cost value with probability P_{cost}, and otherwise on constraint ranking.

As with the penalty function approach, a variety of population sizes, mutation and crossover rates, and so forth were investigated. Results were best with populations of about 100 individuals, and with the same stopping conditions, runs lasted for similar numbers of evaluations, and produced similar quality solutions (the same "best" solution from the penalty-function approach was found consistently). In contrast to the penalty-function approach however, algorithm performance was much less sensitive to these control parameters. Most importantly, the COMOGA scheme was not particularly sensitive to the method used for adapting the P_{cost} parameter, nor to the target proportion of feasible solutions, τ. A sample run of the COMOGA algorithm is discussed in Fig. 5.

Although the overall performance of the COMOGA algorithm was very similar to that of the best penalty function approach found, both in terms of computational effort required and frequency of finding the best solutions, significantly less experimentation was required to find values for COMOGA's parameters that work well than was the case with the penalty function method.

6. Experimental results

In order to validate the COMOGA method it has been applied to a series of known test problems in constrained optimisation for which other evolutionary methods have been applied. The first such problem is the so-called "bump problem" of Keane (1996a), a highly multi-modal maximisation problem defined for an arbitrary number of variables, n, and two non-linear constraints, with an unknown optimum value. For comparative purposes, we studied the problem with $n = 50$. Using an untuned implementation of the COMOGA method, we achieved results superior to any generic evolutionary algorithm previously studied, with fitnesses in the range of 0.814 to 0.828 after 200 000 evaluations. Keane (1996) reports that the best result known to him is 0.832, produced by

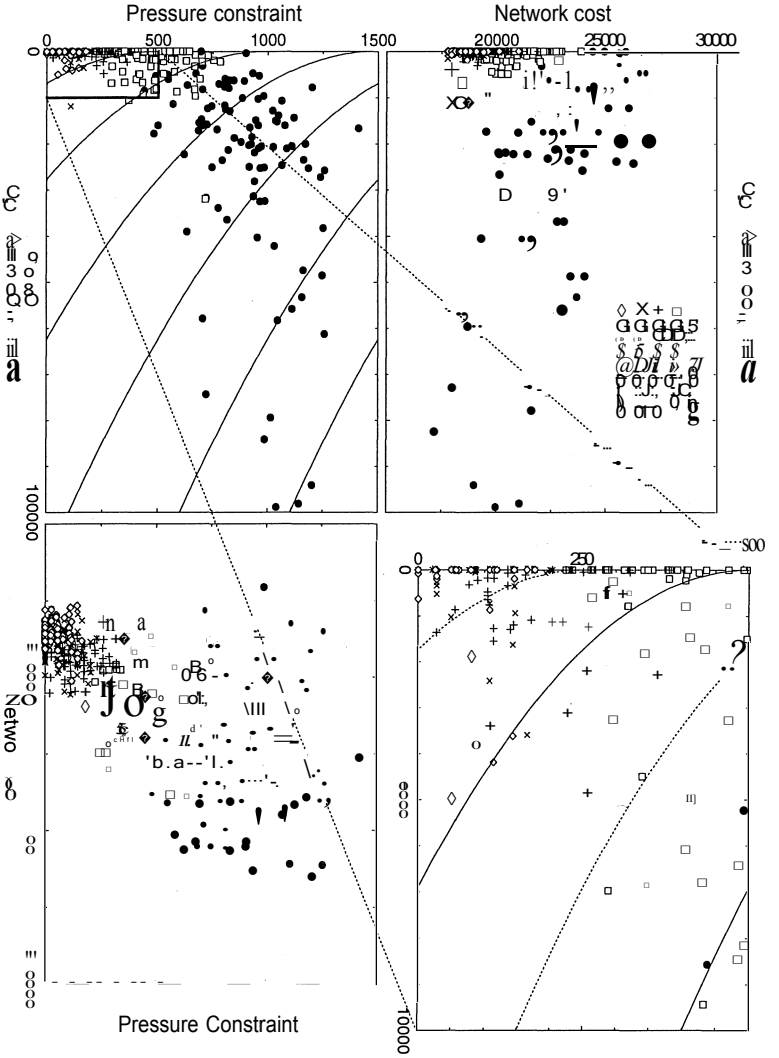

Figure 4. The figure shows the pattern of convergence for a single run of an algorithm based on a penalty function. A snapshot of the population is shown every five generations, plotted with respect to the degree of violation of the two constraints and to unpenalised cost (to be minimised). An enlarged section of the region near zero constraint violation is also shown, along with curves which show lines of equal penalty when constraint violations are incorporated with the cost. Note that the population approaches the minimum by first satisfying the constraints and then minimising in the feasible region.

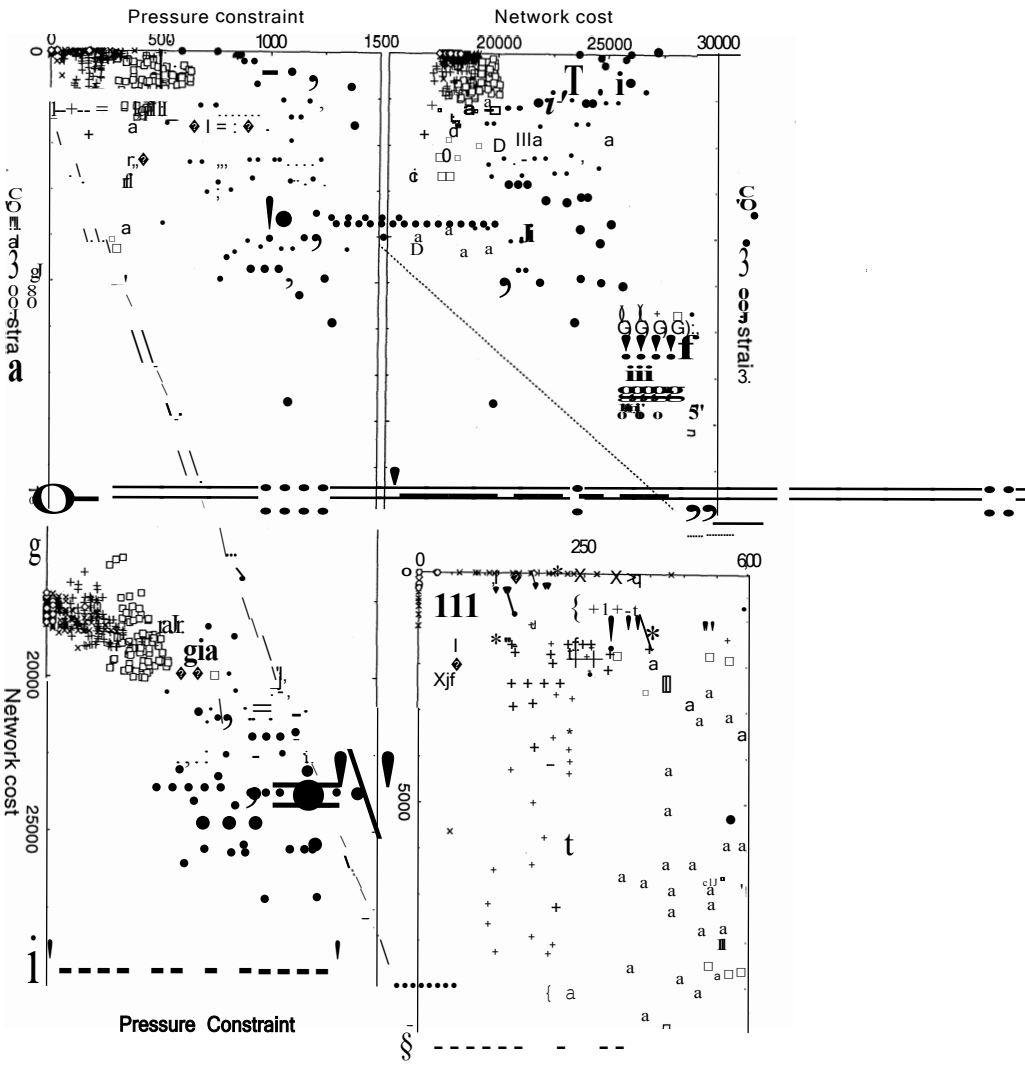

Figure 5. The figure shows the pattern of convergence for a single run of the COMOGA method. The population is shown every five generations with respect to the two constraints and the cost. In contrast to the penalty function run, the population explores a variety of tradeoffs between constraint satisfaction and cost, approaching the feasible solutions with minimal cost from both above and below.

a non-genetic technique, but more recently Michalewicz & Schoenauer (1996a) have achieved 0.833 with a population of size of 30 over 30 000 generations using a problem-specific crossover operator to search the constraint surface, and in this volume Wodrich & Bilchev (1997a) report results of 0.826 after only 30 000 evaluations, also employing a problem-specific heuristic.

Michalewicz (1995a) has proposed a test set consisting of five constrained optimisation problems, for which he has compared six existing evolutionary techniques. This test set was coded as a library for the *Reproductive Plan Language,* RPL2 (Surry, Radcliffe,1994a; Radcliffe, Surry, 1994d), and an untuned implementation of COMOGA was applied to the problems. Table 1 summarises the results of these experiments in the same form presented by Michalewicz (based on the same number of total evaluations per run). Here all runs resulted in feasible solutions so Michalewicz's summary by degree of violation has been dropped.

For problem #1, it has failed only to accomplish the final 'polishing' having achieved 4+ figure accuracy in all of the parameter values for every run. This is more suggestive of shortcomings in the genetic operators used (e.g. perhaps the mutation rate or loss of diversity) rather than a failure in the COMOGA technique itself. Note also that no operators which 'understood' these linear constraints were used for the purposes of this initial comparison.

For problem #4, by treating the equality constraints as hard inequality constraints (using the arbitrary 0.001 factor for allowable degree of violation) we create a sharp artificial boundary between the feasible and infeasible regions. Since this region is essentially a two-dimensional subspace of the five-parameter domain it is perhaps not surprising that it is difficult to "explore" it effectively. In fact, we typically observe that the trajectory of the initial population to find the first feasible solution has a large impact on the quality of the best feasible solution found. This suggests that perhaps this is not the best way in which to incorporate equality constraints within the COMOGA framework, but this requires further investigation.

7. Summary

A new approach of general applicability to constrained optimisation-the COMOGA method-has been presented. This technique treats a constrained optimisation problem sometimes as a constraint satisfaction problem and sometimes as an unconstrained optimisation problem, using a single population, by switching between selection regimes. A simple feedback mechanism is used to determine the (expected) proportion of selective decisions which arc made with respect to the two viewpoints, depending on the relative progress observed on each.

The COMOGA method uses the memory implicit in the population to "discover for itself" the relative utility of different achievable combinations of constraints and objective function value. The population thus forms not just a

Case	Form	Vars	LI	NI	Eq	Optimum	Best/Median/Worst Previous	COMOGA	Rank
1	quadratic	13	9	-	-	-15.000	#4 -15.000	-14.997	5=
							-15.000	-14.996	
							-15.000	-14.994	
2	linear	8	3	3	-	7049.331	#4 7377.976	7081.43	1
							8206.151	7556.85	
							9652.901	8322.51	
3	polynomial	7	-	4	-	680.630	#4 680.642	680.663	1=
							680.718	680.690	
							680.955	680.755	
4	nonlinear	5	-	-	3	0.054	#4 0.054	0.058	3
							0.064	0.205	
							0.557	0.570	
5	quadratic	10	3	5	-	24.306	#2 25.486	24.340	1
							26.905	24.509	
							42.358	24.710	

Table 1. The table shows the results of applying the COMOGA method to the test set of Michalewicz (1995a), in which he compared six evolutionary constraint handling methods. The form, number of variables and number and types of constraints (linear inequalities, LI, non-linear inequalities, NI, and non-linear equalities, Eq) are shown for each test problem, along with the optimal value. Results from a set of ten runs are shown for both the previous best method (numbered) and the COMOGA method, along with the overall ranking on minimum, median and maximum for this new scheme (where this is ambiguous a tie has been awarded). COMOGA performs well on most of the problems, and was the only method to produce a feasible solution in every run for every problem. It performs worst with the linear constraints of problem 1, though this is essentially only a failure to sufficiently polish the parameter values. In problem 4 the equality constraints are treated by essentially converting them to inequality constraints specifying an allowable violation of the equality (a tolerance of 0.001 was used, as this defined feasibility in Michalewicz's study). This creates an artificial division between the feasible and infeasible region.

pool of good solutions among which recombination takes place, but a context in which to determine the fitness of any one member-the effective weighting of the various constraints is determined by the population, as is the relative weighting of constraint satisfaction and cost minimisation. This contrasts with a penalty function approach, where both are determined *a priori,* and appears to carry the significant benefit of reducing both the sensitivity of the genetic algorithm to the values of the free parameters, and the number of those parameters.

A series of experiments has validated the COMOGA approach, based on the simple yet powerful idea that by merely alternating between two selection (and replacement) regimes we can couple solution of the constraint satisfaction problem to the simultaneous solution of the unconstrained optimisation problem, and discover good solutions to the constrained optimisation problem. Because the scheme is based only on modifying the selection regime, it is possible to use whatever representation, genetic operators and update strategy is appropriate for a given problem within the COMOGA framework.

References

CAVICCHIO, D.J. (1970) *Adaptive Search Using Simv,lated Evolution.* PhD thesis, Univerity of Michigan.

CHU, P.C. and BEASLEY, J.E. (1995A) A genetic algorithm for the set partitioning problem. Technical report, The Management School, Imperial College. (Revised 1997).

DAVIS, L. and ORVOSH, D. (1993A) Shall we repair? Genetic algorithms, combinatorial optimization, and feasibility constraints. In: S. Forrest, ed., *Proceedings of the Fifth International Conference on Genetic Algorithms,* 50. Morgan Kaufmann, San Mateo.

DAVIS, L. (1987A) *Genetic Algorithms and Simulated Annealing.* Pitman, London.

DAVIS, L. (1991A) *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

DE JONG, K.A. (1975) *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan.

FONSECA, C.M. and FLEMING, P.J. (1993) Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Forrest, S., ed., *Proceedings of the Fifth International Conference on Genetic Algorithms,* 416-423. Morgan Kaufmann, San Mateo.

FONSECA, C.M. and FLEMING, P.J. (1995A) An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation,* 3, 1, 1-16.

GOLDBERG, D.E. and RICHARDSON, J. (1987A) Genetic algorithms for multimodal function optimisation. In: *Proceedings of the Second International Conference on Genetic Algorithms,* 41-49. Lawrence Erlbaum Associates, Hillsdale.

GOLDBERG, D.E. (1989A) *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading, Mass.

GORGES-SCHLEUTER, M. (1989A) ASPARAGOS: an asynchronous parallel genetic optimization strategy. In: *Proceedings of the Third International Conference on Genetic Algorithms,* 422-427. Morgan Kaufmann, San Mateo.

KEANE, A.J. (1996A) (personal communication).

KEANE, A.J. (1996B) *Modern Heuristic Search Methods,* chapter A Brief Comparison of Some Evolutionary Optimization Methods, 255-272. J. Wiley.

LERICHE, R.G., KNOPF-LENOIR, C. and HAFTKA, R.T. (1995A) A segregated genetic algorithm for constrained structural optimization. In: Eshelman, L.J., ed., *Proceedings of the Sixth International Conference on Genetic Algorithms,* 558-565. Morgan Kaufmann, San Francisco.

MANDERICK B. and SPIESSENS, P. (1989A) Fine-grained parallel genetic algorithms. In: Schaffer, J.D., ed., *Proceedings of the Third International Conference on Genetic Algorithms,* 428-433, San Mateo. Morgan Kaufmann Publishers.

MICHALEWICZ, Z. and JANIKOW, C.Z. (1991) Handling constraints in genetic algorithms. In: *Proceedings of the Fourth International Conference on Genetic Algorithms,* 151-157. Morgan Kaufmann, San Mateo.

MICHALEWICZ, Z. and SCHOENAUER, M. (1996A) Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Compv,tation,* 4, 1, 1-32.

MICHALEWICZ, Z. (1992) *Genetic Algorithms + Data Ŝtrnctv.res = Evol11tion Programs.* Springer Verlag, Berlin.

MICHALEWICZ, Z. (1995A) Genetic algorithms, numerical optimization, and constraints. In: Eshelman, L.J., ed., *Proceedings of the Sixth Intern.ational Conference on Genetic Algorithms,* 151-158. Morgan Kaufmann, San Francisco.

MICHALEWICZ, Z. (1995B) A survey of constraint handling techniques in evolutionary computation methods. In: *Proceedings of the 4th Annual Conference on Evolutionary Programming,* 135-155. MIT Press, Cambridge, MA.

NORMAN, M. (1988) A genetic approach to topology optimisation for multiprocessor architectures. Technical report, University of Edinburgh.

RADCLIFFE, N.J. and SURRY, P.D. (1994D) The Reproductive Plan Language RPL2: Motivation, architecture and applications. In: Stender, J., Hillebrand, E. and Kingdon, J., eds., *Genetic Algorithms in Optimisation, Simulation and Modelling,* 65-94. IOS Press, Amsterdam.

RADCLIFFE, N.J. (1994) The algebra of genetic algorithms. *Annals of Maths and Art ficial Intelligence,* 10, 339-384.

RICHARDSON, J.T., PALMER, M.R., LIEPINS, G. and HILLIARD, M. (1989) Some guidelines for genetic algorithms with penalty functions. In: *Proceedings of the Third International Conference on Genetic Algorithms,*

191-195. Morgan Kaufmann, San Mateo.

SCHAFFER, J.D. (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: *Proceedings of an International Conference on Genetic Algorithms,* 93-100. Lawrence Erlbaum.

ScHOENAUER, M. and MICHALEWICZ, Z. (1996) Evolutionary computation at the edge of feasibility. In: Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel, H., eds., *Parallel Problem. Solving from Nature TV,* 245-254. Springer-Verlag, LNCS 1141.

SCHOENAUER, M. and XANTHAKIS, S. (1993) Constrained GA optimisation. In: Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms,* 573-580. Morgan Kaufmann, San Mateo, CA.

SuRRY, P.D. and RADCLIFFE, N.J. (1994) RPL2: A language and parallel framework for evolutionary computing. In: Davidor, Y., Schwefel, H.-P. and Manner, R., eds., *Parallel Problem. Solving from Nature III,* 628-637. Springer-Verlag, Lecture Notes in Computer Science 866.

SURRY, P.D., RADCLIFFE, N.J. and BOYD, I.D. (1995) A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA method. In: Fogarty, T.C., ed., *Evol11,tionary Computing: AISB Workshop,* 166-180. Springer-Verlag, Lecture Notes in Computer Science 993.

WHITLEY, D., GORDON, V.S. and MATHIAS, K. (1994) Lamarckian evolution, the baldwin effect and function optimization. In: Davidor, Y., Schwefel, H.-P. and Manner, R., ed., *Parallel Problem. Solving from Nature III,* 6-15.

WODRICH, M. and BILCHEV, G. (1997) Cooperative distributed search: the ants' way. *Control and Cybernetics,* in this volume.