A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks

C. E. Mariano
Instituto Mexicano de Tecnología del Agua
Paseo Cuauhnahuac 8532
Jiutepec, Morelos, 62550, MEXICO
cmariano@tlaloc.imta.mx

E. Morales
ITESM - Campus Morelos
Paseo de la Reforma 182-A
Temixco, Morelos, 62580, MEXICO
emorales@campus.mor.itesm.mx

Abstract

The difficulty in solving multiple objective optimization problems with traditional techniques, has urge researchers to use alternative approaches. Ant-Q algorithms have shown good results in the solution of combinatorial optimization problems, however little work has been done for multiple objective problems. This paper describes an Ant-Q algorithm called MOAQ, that can solve multiple objective optimization problems. MOAQ considers a family of agents for each objective function involved. Each family finds solutions that depend on solutions found by the rest of the families, creating a negotiation mechanism and finding compromise solutions for all the objectives involved. The compromise solutions are evaluated in the Pareto sense, assigning rewards to the non-dominated solutions fitting all problems constraints, and punishments to the solutions violating any of them.

We compare and contrast the solutions obtained with MOAQ with the solutions obtained with two recently developed genetic algorithm approaches in two artificial problems, demonstrating the ability of MOAQ to find and maintain the Pareto frontier. Furthermore, it is shown how MOAQ is applied to a complex real-world problem; the design of irrigation water distribution networks, with very promising results.

1 Introduction

Many real world applications require the solution of multiple objective optimization problems. Historically, the application of classical mathematical methods in the solution of this kind of problems involves the definition of a single objective function usually formed by a linear combination (weighted sum) of the multiple objectives, or by turning objectives into constraints. Aside from genetic algorithms, little research has been done, with the use of alternative optimization techniques to solve multiple objective problems. Genetic algorithms search for all possible trade-offs solutions among the multiple, conflicting objectives [8, 13, 14, 21, 22, 24]. These solutions are *non-dominated*, in the sense that there are no other solutions superior in all objectives and lie on a surface known as the *Pareto optimal frontier*. The goal is to find and maintain a representative sampling of solutions on the Pareto front. Although genetic algorithms have shown good behavior in the solution of multiple objective optimizations problems, codification is often a difficult procedure which may involve the simplification of some problem constraints.

On the other hand, reinforcement learning, especially Ant-Q, has shown good results in combinatorial optimization problems [5, 6, 10]. In reinforcement learning [23], an agent learns values trying to obtain the maximum reward based on proven actions that gave high rewards, while not forgetting to explore new states where reward values are unknown. In order to increase the exploration capabilities and to reduce convergence times, more than one agent can be used in the solution of a single problem, transforming the basic algorithm to a distributed reinforcement learning algorithm (Ant-Q). In this approach many agents try to satisfy a common objective; each agent proposes a solution which is evaluated and rewarded depending on its goodness. Ant-Q, however, has been used only in the solution of single objective optimization problems. Its distributed characteristics, theoretical background, and representation capabilities makes it a good candidate for the solution of multiple objective optimization problems.

In this work an algorithm for multiple objective optimization problems called MOAQ is presented. MOAQ was tested in the solution of two artificial problems, previously solved with genetic algorithms, with comparably good results. It is also shown how MOAQ is used in the solution of an extremely difficult real world problem, the design of irrigation water distribution networks, with overall very promising results.

Section 2 describes in detail the Multiple Objective Ant-Q Algorithm. Section 3 shows the results on the problems solved by MOAQ and compares them with other approaches. Section 4 discusses the algorithm and its performance, and finally conclusions and future research directions are given in section 5.

2 Multiple Objective Ant-Q

In reinforcement learning, an autonomous agent learns an optimal policy $\pi : S \rightarrow A$, that outputs an appropriate action $a \in A$, given the current state $s \in S$, where A is the set of all possible actions in a state and S is the set of states. The available information to the agent is the sequence of immediate rewards $r(s_i, a_i)$ for all the possible actions and states $i = 0, 1, 2, \ldots$. Given this kind of training information it is possible to learn a numerical evaluation function defined over states and actions, and then implement the optimal policy in terms of this evaluation function. The value of the evaluation function $Q(s, a)$ is the maximum discounted cumulative reward that can be achieved starting from state s and applying action a as the first action. In other words, the value of Q is the reward received immediately upon executing action a from state s, plus the value (discounted by γ) of following the optimal policy. Therefore the optimal policy for a given state s must select the action which Q function value is maximum $\pi^*(s) = \arg\max_a Q(s, a)$.

The idea is to learn $\widehat{Q}$ values that approach Q. The actualization of the $\widehat{Q}$ values is made according to the following rule:

$$\widehat{Q}(s, a) \leftarrow r(s, a) + \gamma \max_{a'} \widehat{Q}(s', a')$$

where $\widehat{Q}(s,a)$ is the updated $\widehat{Q}$ value for state s and action a, $r(s,a)$ is the reward the agent receives when it executes an action a in state s, and $\widehat{Q}(s',a')$ is the maximum $\widehat{Q}$ value for the next algorithm step.

More recently a distributed reinforcement learning algorithm called Ant-Q was proposed in [10]. The algorithm is in essence a reinforcement learning algorithm with few changes improving its exploratory capabilities and convergence time. The basic changes, which are also used in MOAQ for single objectives, are:

- Ant-Q considers m agents, rather than a single one, trying to find a solution.
- Ant-Q considers a domain-dependent heuristic value indicating how good is to go from a particular state (s) to another state (s'), $HE(s,s')$.
- Ant-Q considers an action choice rule or state transition rule to select the next state for each agent using a combination of the heuristic value $HE(s,s')$ and the $Q(s,a)$ value associated with state s and the action a that takes the agent to state s'. Let us call it $COMB(s,s')$. The transition rule has the following form:

$$s = \begin{cases} \arg\max\ COMB(s,s') & \text{if } q \leq q_0 \\ P & \text{otherwise} \end{cases} \tag{1}$$

  where q is a value chosen randomly with uniform probability over $[0,1]$, q_0 ($0 \leq q_0 \leq 1$) is a parameter such that the higher q_0 the smaller the probability to make a random choice, and P is a random state selected according to a probability distribution given as a function of the heuristic and the Q values.

- The delayed reinforcement is computed using the Q-learning rule [25]:

$$\widehat{Q}(s,a) \leftarrow (1-\alpha)\cdot\widehat{Q}(s,a) + \alpha\left[r(s,a) + \gamma\max_{a'}\widehat{Q}(s',a')\right] \tag{2}$$

  where $\widehat{Q}(s,a)$ is the new $\widehat{Q}$ value for the s,a pair, α is the learning step, s' and a' are the state and action in the next algorithm step, and γ is a discount factor. The learned action-value function, $\widehat{Q}$, directly approximates Q, the optimal action-value function, independent of the policy being followed. All that is required for correct convergence is that all pairs are visited and updated. The computation of the delayed reward depends on the problem, so details of its computation will be given later in the paper.

In one iteration of the algorithm every agent proposes a solution, all the solutions are evaluated and the best solution receives a reward. The new $\widehat{Q}$ values are then reinforced using equation 2.

The basic idea behind MOAQ is to perform a similar optimization algorithm but with a family of agents for each objective. Each family tries to optimize an objective considering the solutions found for the other objectives. So that all the agents from the different families act in the same environment proposing actions and expecting a reward value which depends on how their actions helped to find trade-off solutions between the rest of the agents.

In our algorithm, all the families must have the same number of agents, say m. The order given to the objectives (families) is arbitrary, although some problems may need a particular order in which the objectives should be tried. As in Ant-Q, all the agents in one family try to find a solution at the same time. The m solutions found for one objective in one cycle influence the m successive starting points of the following family (objective). This process continues with all the families (objectives). Once all the families have been tried, a reward is given to the non-dominated solutions (to all the agents in each family responsible for that solution) that satisfy all the constraints. Solutions violating constraints are punished while the rest of the solutions are neither rewarded nor punished. The whole process is repeated several times until only non dominated solutions (Pareto set) satisfying all the constraints are found or a predetermined number of iterations is satisfied (see table 1).

The main important points of the algorithm can be summarized as follows:

- The number of objectives is the same as the number of families.
- The number of agents is the same in all the families.
- The solution of one agent in one family affects the solutions of corresponding agents in other families.
- Each family is run once and rewards and punishments are given after all the families have reached a solution.

Table 1: MOAQ Algorithm.

Given: (i) a list of n families (objectives) and (ii) m agents for each family
Let N be the maximum number of iterations
Let $IN := 0$ (initialize iteration number)
Initialize all the Q values for all the families
Until only non-dominant solutions or $IN > N$
    Let $IN := IN + 1$
    Initialize parameters of family 1
    for $j = 1$ to m
        find a solution for objective 1
    for $i = 2$ to n
        Initialize parameters of family i
        for $j = 1$ to m
            Use the solution found with ant j in objective $i - 1$ to
            constrain the solution for ant j (of objective i)
            Find a solution for objective i
    Evaluate the found solutions
    for $j = 1$ to m
        if solution j violates any constraint,
            Apply punishment to all its components (corresponding
            agents)
        else if solution j is non dominated
            Apply reward to all its components
            Introduce solution j into the Pareto set
            Remove all dominated solution from the Pareto set
        else if solution j is dominated
            Neither apply reward nor punishment

- The whole process is repeated until only non-dominated solutions are obtained or after completing a pre-determined number of cycles.

3 Application to three problems

Distributed reinforcement learning for multiple objective optimization problems calls for further analysis of the behavior of this new algorithm. However, our preliminary empirical results are very promising. MOAQ found the Pareto frontier for an artificial, two objective function over bit strings (number of ones versus number of adjacent “01” or “10” pairs). MOAQ was also applied to a simple two objective function used to test Schaffer's multicriteria genetic algorithm (VEGA) [21]. MOAQ's performance was superior to Horn's *Niched Pareto Genetic Algorithm* [14] and slightly better to Valenzuela's *Non Generational Genetic Algorithm* [24]. Finally, MOAQ was tested on a real world open problem whose Pareto frontier is not known, with very promising results.

In the first two problems, MOAQ needs to find a binary string, where each agent has to decide whether to place a one or a zero in each bit position. The application of MOAQ to binary strings suggests that many problems that have previously been treated with genetic algorithms are susceptible to be solved with MOAQ.

3.1 Problem 1. A simple test function

We begin by testing MOAQ to a simple artificial problem proposed in [13] whose Pareto optimal front can be easily evaluated. The problem involves two objectives which are simple functions of finite bit string length l. Unitation, $Unit[s]$, is the number of ones in the string s, thus $Unit[01110010] = 4$. Pairs, $Prs[s]$, is the number of pairs of adjacent complementary bits, either 01 or 10. Thus $Prs[01110010] = 4$. The objective is to maximize both functions. Among strings of a particular unitation, those strings with the greatest “mixing” of ones and zeros dominates those who “clump” their ones (e.g., 01010 dominates 01100, although both have the same unitation).

For the solution of this problem each agent in both families has an associated string named $Tour$ of length l where it stores its solution and two variables named one_a^f and $pair_a^f$, where agent a of family f stores the number of ones and pairs, respectively, in its solution. As we mentioned earlier, we need two agent families, the agents belonging to the first family maximize $Unit[s]$, while members of the second family maximize $Prs[s]$ considering a string s of length l.

A solution is constructed by placing every agent from $family1$ on the left-most position of the string. At this point their corresponding one_a^f and $pair_a^f$ for all the agents are initialized to zero. Each agent decides, according to the transition rule in equation 3 (see below) whether a 0 or a 1 should be placed in its current string position, while the agent moves to the next string position. The process is repeated until all the agents complete their strings (move to the far end of the string). The transition rule combines a heuristic value with the associated Q value.

$$b' = \begin{cases} \arg\max_a \left[HE(b,a) \cdot Q(b,a)\right] & \text{if } q \leq q_0 \\ bit & \text{otherwise} \end{cases} \tag{3}$$

where $HE(b,a)$ is the heuristic value for agent k in $family1$ used to evaluate how good is to select, in string position b, any of the two options (0 or 1) taking the agent to the next string position. In this case the heuristic used is equal to the number of ones in the string, i.e., $HE(b,a) = one_k^1$, which means that it is incremented if the action chooses a 1 and remains the same otherwise.

$$HE(b,a) = \begin{cases} HE(b,a) + 1 & \text{if selection is 1} \\ HE(b,a) & \text{if selection is 0} \end{cases}$$

$Q(b,a)$ is the Q value associated to the pair b,a; the Q values are initialized to 1, q and q_0 are explained in section 2 (equation 1), bit is a random selection according to a uniform probability distribution:

$$\frac{HE(b,a) \cdot Q(b,a)}{\sum_{action} \left[HE(b,action) \cdot Q(b,action)\right]} \tag{4}$$

Once all agents in $family1$ have completed their strings, solutions are given to the corresponding agent in $family2$. Every agent in $family2$ maximizes $prs[s]$ following a process similar to that described for agents in $family1$. In this case, their heuristic is equal to the number of pairs, i.e.,

$HE(b,a) = pair_k^2$. The heuristic value increases if the number of pairs for $agent_k^2$ increases and does not produce more ones than those produced by $agent_k^1$. The intention is to relocate the ones proposed by $agent_k^1$ to positions where they contribute to maximize $prs[s]$. The heuristic value for agents in $family2$ is:

$$HE(b,a) = \begin{cases} HE(b,a)+1 & \text{if } pair_k^2 \text{ grows } \wedge one_k^2 \leq one_k^1 \\ HE(b,a) & \text{otherwise} \end{cases}$$

When all the agents from $family2$ have completed their strings an iteration has been performed and the solutions are evaluated. Every solution is compared against all other solutions in the iteration. Non dominated solutions, i.e., when there are no other solutions superior in all the objectives, are rewarded with $r(b,a)_k = 1$, and their Q values are updated using:

$$\widehat{Q}(b,a) \leftarrow (1-\alpha)\cdot\widehat{Q}(b,a) + \alpha\left[r(b,a)_k + \gamma\max_{a'}\widehat{Q}(b',a')\right] \tag{5}$$

Figure 1 shows the results obtained after 20 algorithm iterations, where the numbers indicate how many occurrences of that solution were found. We can see that MOAQ succeeds in finding all but one member of the Pareto set, maintaining the Pareto set while searching the solutions' space. We have plotted the solutions over many iterations noting that the algorithm maintains similar number of solutions for each Pareto point, as shown for 100 iterations in Figure 2.

We have observed similar behavior on larger problems, with corresponding larger families. Figure 4 and 5 show the agents behavior after 20 and 100 iterations for the 28 bit string case. It can be seen that the algorithm is also capable in this case to find and maintain the Pareto front. The values of the parameters used for both cases are showed in Table 2.

MOAQ solutions for this problem solution are better than the Niched Pareto Genetic Algorithm results reported in [13, 14], since MOAQ is able to find all the solutions in the Pareto front while the Niched Genetic Algorithm fails finding solutions in the left side of the front, [11,6] and [27,14] for 12 and 28 bit respectively (see figures 3 and 6).

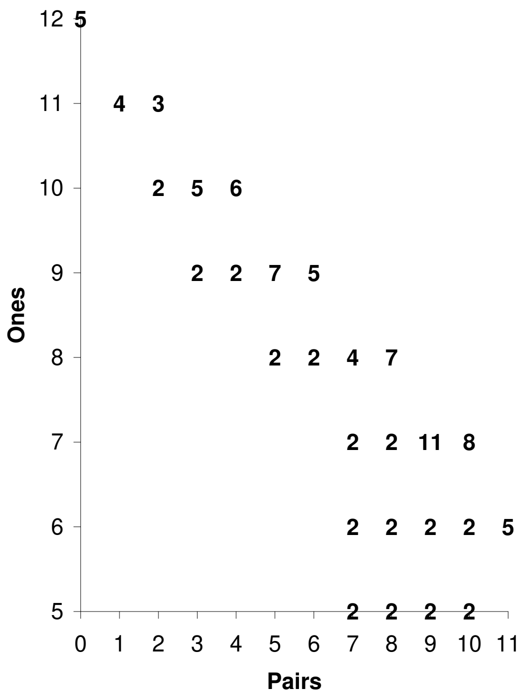

Figure 1: Solutions found after 20 iterations with a 12 bit string.

Table 2: Parameter setting for 12 and 28 bit string problems.

Problem	q_0	α	γ	*Agents*	*Time*[a]
12 bit	0.7	0.1	0.3	100	288 s
28 bit	0.7	0.1	0.3	250	358 s

[a]MOAQ is written in Borland C++ V4.5 running on a 75 Mhz Pentium PC.

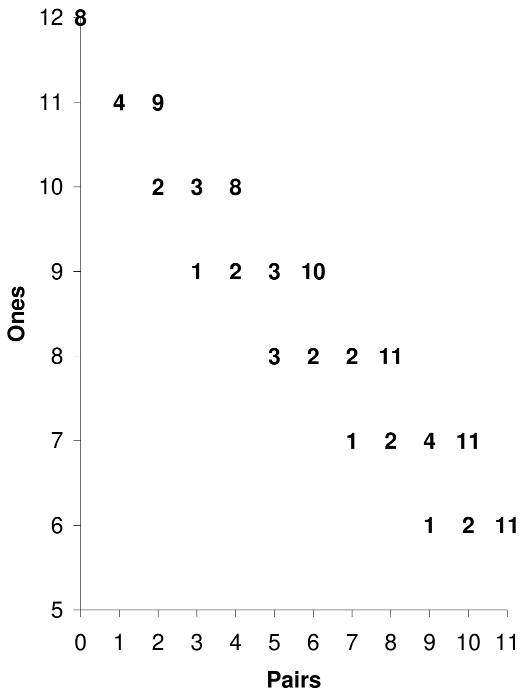

Figure 2: Solutions found after 100 iterations with a 12 bit string.

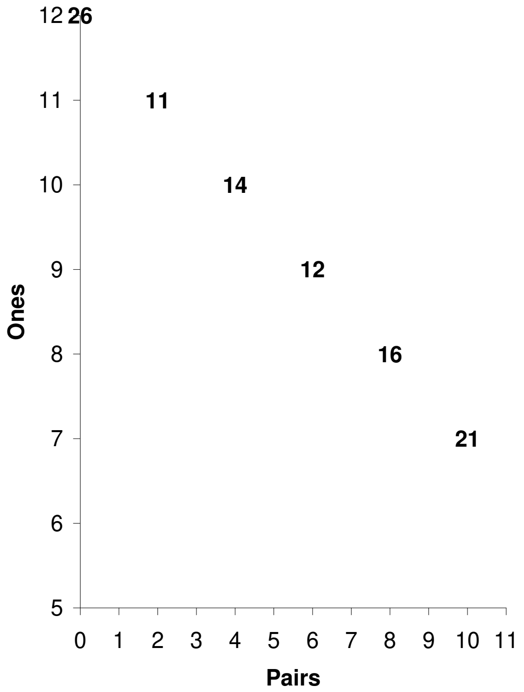

Figure 3: Horn's solution at generation 100.

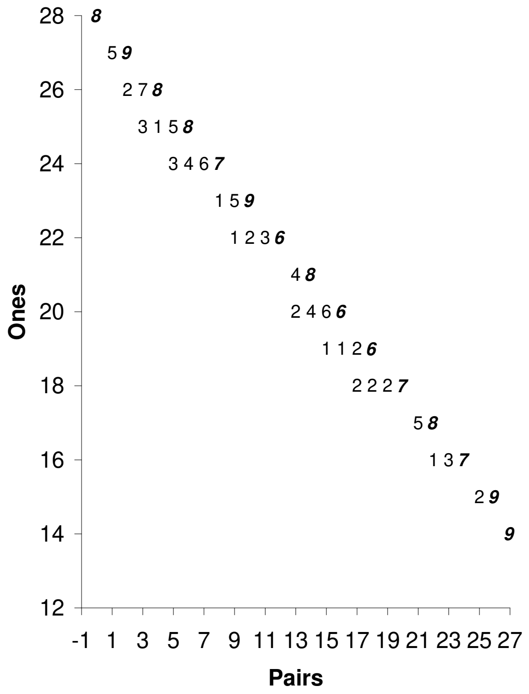

Figure 4: Solutions found after 20 iterations with a 28 bit string.

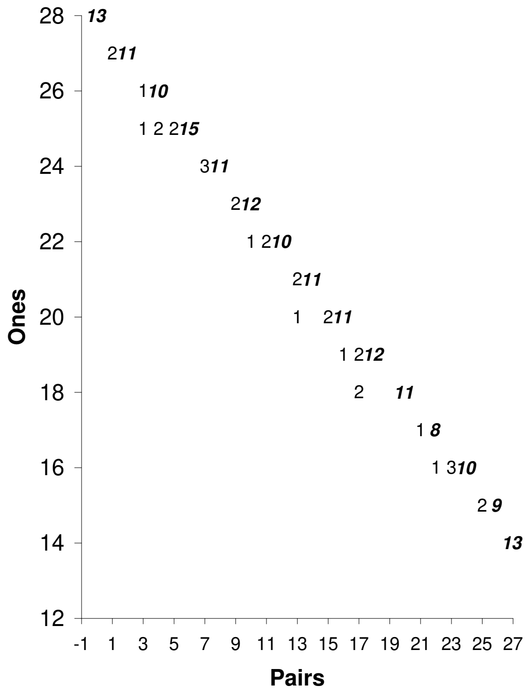

Figure 5: Solutions found after 100 iterations with a 28 bit string.

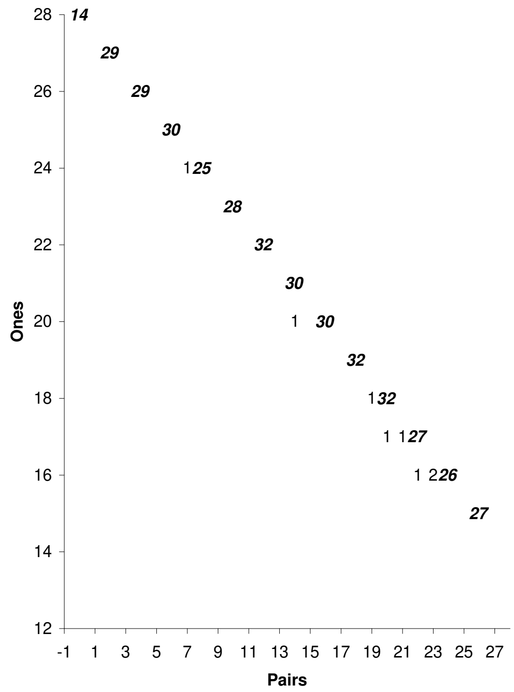

Figure 6: Horn's solutions for 28 bits at generation 200.

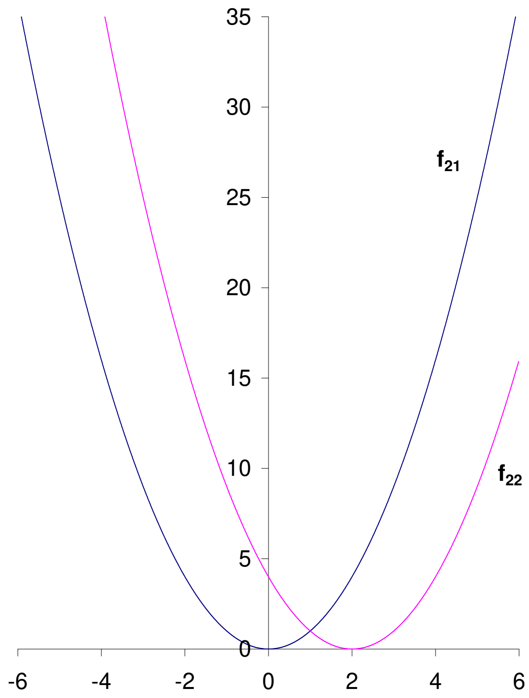

Figure 7: Shaffer's function F2.

3.2 Problem 2: Schaffer's F2

Next we tested our algorithm on Schaffer's function F2, with a single decision variable, the real valued x, and two objectives, f_{21} and f_{22} to be minimized.

$$f_{21}(x) = x^2 \qquad\qquad f_{22}(x) = (x-2)^2$$

The solution to this problem has been reported in [13, 14, 21, 24], where genetic algorithms need to perform Niches or use Non generational behavior to maintain solutions distributed over the Pareto front. We used the same representation used in these works where the decision variable is mapped to a 14 bit string as a binary-coded integer. Thus $00000000000000 = x_{\min}$ and $11111111111111 = x_{\max}$, where $x_{\min} = -6.00$ and $x_{\max} = 6.00$. f_{21} and f_{22} are shown over this range of x in figure 7. It is clear from the figure that the Pareto front is roughly where x takes values around $0 \leq x \leq 2.00$.

In this case, the agents from $family1$ minimize the function f_{21}, while agents from $family2$ minimize function f_{22}. A very similar approach, to the

Table 3: Parameter setting for Shaffer F2.

Problem	q_0	α	γ	*Agents*	*Time*
F2	0.9	0.3	0.5	200	658 s

previous problem, is used in this case. The heuristic value is the same for all the actions in the states, in this case, it is always equal to one ($HE(b,a) = 1$). So the string position value is decided considering only the Q values associated to every state-action pair $Q(b,a)$ using equation 6. The Q values are initialized to 1.

$$b' = \begin{cases} \arg\max\limits_{a} \left[Q(b,a)\right] & \text{if } q \leq q_0 \\ bit & \text{otherwise} \end{cases} \tag{6}$$

In this case the stochastic selection (bit) is performed considering the following uniform probability distribution:

$$\frac{Q(b,a)}{\sum\limits_{action} \left[Q(b, action)\right]} \tag{7}$$

When agents from both families have completed their string, the set of solutions is evaluated searching for non dominated solutions. Non dominated solutions are rewarded with $r(b,a)_k = 1$, and their Q values are updated using:

$$\widehat{Q}(b,a) \leftarrow (1-\alpha)\cdot\widehat{Q}(b,a) + \alpha\left[r(b,a)_k + \gamma \max_{a'} \widehat{Q}(b',a')\right] \tag{8}$$

Table 3 shows the parameter values used in the problem solution. In this case, a *greedier* selection was used to avoid a disperse Pareto set. More exploration (and diversity) can be obtained when q_0 is closer to 0.

Figure 8 shows the Pareto set obtained with MOAQ for Shaffer F2 problem after 100 iterations. Notice that almost all solutions lie on the Pareto front and some of them are very close to it. Figure 9 shows MOAQ's performance, and as can be seen, the solutions cover all the Pareto set. There

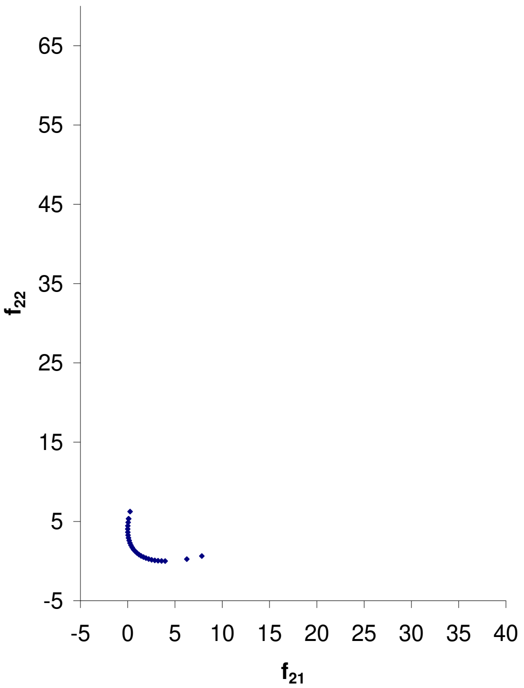

Figure 8: Pareto front for Shaffer's function F2.

are very few dominated solutions due to the exploration mechanism of the algorithm. The algorithm maintains the Pareto front over time and does not converge to a single solution, as often happens with some genetic algorithm approaches.

This is a slightly more difficult problem, but MOAQ reaches stability in just a few iterations with a relative small number of agents in each family. Results reported by [13, 14] show disperse solutions in the Pareto front, in contrast with the results reported in [24] with a Non Generational Genetic Algorithm, which are similar to the solutions obtained with MOAQ.

3.3 Problem 3: Design of irrigation water distribution networks

The design of irrigation networks involve the selection of pipe diameters for each pipe segment in the network, the design of the network layout to minize its length (summation of every segment length in the network), and the definition of the network parameters (required flux and pressure for each

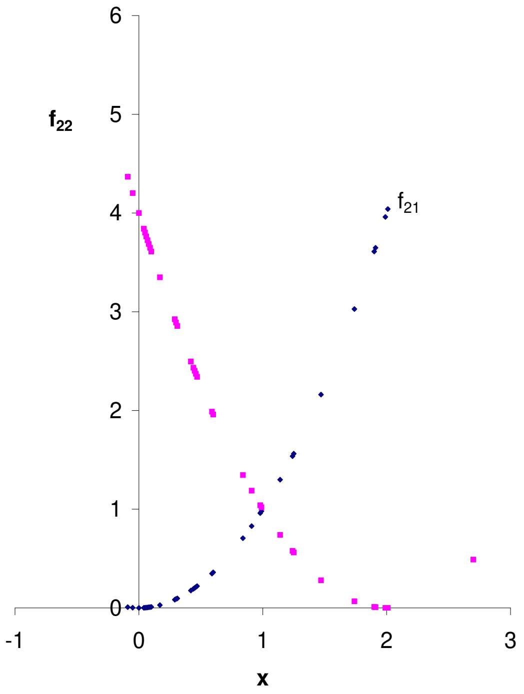

Figure 9: Expanded view of MOAQ's performance.

hydraulic node) looking for the minimal cost. A pipe diameter depends on the water flux and pressure at which water must be transported, i.e., bigger water fluxes require bigger diameters. There are pressure loses that depend on the pipe's length and diameter, so an adequate diameter selection is required to satisfy pressure and flux constraints. At the same time, the pressure and flux values depend on the crop selection to be irrigated from every hydraulic node. A careful crop selection must be performed in order to maxize profits.

Design of water distribution networks is often viewed as a least-cost optimization problem with pipe diameters being the decision variables. Pipe layout, connectivity, water requirements, and minimum pressure constraints imposed at pipe junctions (nodes) are generally considered as known. There are obviously other possible objectives, like reliability, redundancy and/or water quality, that can be included in the optimization process. However, problems with the quantifycation of these objectives within an optimization design problem, kept researchers centering on a single, least-cost objective function [11]. Even this restricted formulation of optimal network design represents a difficult problem to solve. It is also worth noticing that networks designed on a purely cost-effective basis and for a single loading condition (pressure in all nodes previously determined) tend to be driven to a branched layout. This simply means that for optimal least-cost design, loops will, in normal circumstances, disappear, leaving an underlying tree-like structure. However, reliability/redundancy considerations usually dictate that loops should be maintained. The inclusion of some measure of reliability/redundancy is beyond the scope of this work.

Let us consider a set of plots in a region, each one with its own hydraulic node. Every plot must be cultivated and irrigated. Irrigation water is supplied by a supply node located in the region. Water requirements in each node, as well as the plot profit, depend on the cultivated crop and on the corresponding cultivated area [4]. The irrigation water network design requires the selection of the crop in each plot trying to maximize the whole region profit and not exceeding the available water for irrigation. It is also necessary to define the network's layout, connecting all hydraulic nodes to the supply node, defining with these all segments length (a segment is considered as the link between two nodes), and consequently, the water discharge for each pipe segment (the sum of water requirements for each node located after the pipe segment). Once the length of the segments and water discharges are known, diameter selection can be performed for minimum cost.

In general, the irrigation water distribution network design problem can be stated as follows: *to select the best crop for each plot maximizing the region productivity, not violating water and land restrictions, considering that the water to each plot must be transported to each hydraulic node from the supply by a minimum cost and distance network that satisfies the pressure and flow requirements.*

The main reason to consider this problem as a multiple objective optimization problem, and not as a linear combination of objetives, is that it is very difficult to determine an adequate combination of objectives that can produce a good solution.

In the following section, a mathematical formulation of the above optimization problem is presented.

3.3.1 Mathematical formulation

Consider a branching network supplied with a source. At each node, j, of the network a given demand of water, d_j, has to be satisfied. The layout of the network must guarantee the minimum network cost [1], that is:

$$min\ z_1 = \sum_{ij} C_{ij} L_{ij} \tag{9}$$

where, C_{ij} and L_{ij} are, respectively, the cost per unit of length and length of the pipeline in the link connecting nodes i and j.

The available pressure loss in the pipeline of the link connecting nodes i and j (ΔH_{ij}) is:

$$\triangle H_{ij} = J_{ij} L_{ij}$$

where $J_{i,j}$ is the hydraulic gradient that depends on the diameter and roughness of the pipe, and on the water discharge by the Hazen Williams equation:

$$J_{ij} = \alpha \left(Q_{ij}/C\right)^{1.852} D_{ij}^{-4.87}$$

where Q_{ij} is the water discharge for pipeline connecting node i and node j, C is the Hazen Williams coefficient, $D_{i,j}$ is the pipe diameter, and α is a coefficient, whose value depends on the units used (for example, for $Q_{i,j}$ in m^3/h and $D_{i,j}$ in centimeters, $\alpha = 1.526 \times 10^{10}$).

The head (available pressure) on each node H_j has to be between a given minimum $H_{min(j)}$ and a given maximum $H_{max(j)}$. Starting from any node s in the system, where the pressure is known in advance (for example, at the supply node or at some source), we can state for any node n:

$$H_{min(n)} \leq H_s \pm \sum_{ij} J_{ij} L_{ij} \leq H_{max(n)} \tag{10}$$

where the summation is over all links (i, j) connecting node s with node n. Equation 10 represents two non linear constraints. H_{min} is usually obtained from service performance requirements and H_{max} from performance requirements or from technological limitations on the pressure-bearing capacity of the pipes.

Minimization of equation 9 is subject to constraints of the form of equation 10 and non-negativity requirements:

$$L_{ij} \geq 0$$

The above formulation is used to determine the network of minimum cost. We now turn our attention to the other objective, that is to maximize the profits with an optimal crop selection:

$$max\ z_2 = \sum_j [A_j(c)(\$_c P_c - I_j - W \cdot d_{c,j})] \tag{11}$$

where, $A_j(c)$ is the cultivated area in plot j with crop c, $\$_c$ is the economic utility of crop c, P_c is the production per unit of area of crop c, I_j are the input costs for plot j, W is the water cost per unit of volume, and $d_{c,j}$ is the required water per unit of area cultivated with crop c. In this objective function a set of candidate crops is considered for each plot.

The volume of required water to supply the whole region, considering for each plot j a volume of water $V_j(c)$ necessary to satisfy the crop c requirements, must be less than or equal to the available water for irrigation in the region. ϕ is the amount of water from the total volume V_t that is available for irrigation, in the case where there are no water restrictions:

$$\sum_j V_j(c) \leq \phi V_t$$

The sum of cultivated areas ($A_j(c)$) in the region must not be greater than the available area in the region (A_t):

$$\sum_j A_j(c) \leq A_t$$

The required volume of water for plot j ($V_j(c)$) must not be greater than the required water per unit of area cultivated with the best crop option[1], $d_{max(j)}$, times the maximum cultivated area, $A_{max(j)}$:

$$V_j(c) \leq d_{max(j)} \cdot A_{max(j)}$$

The supply of water for crop (c) in plot $d_j(c)$, considering that the plot area, $A_j(c)$, is cultivated with it, must not be greater than the capacity to transport water through the pipe segment connecting plot j (Q_{ij}):

$$\sum_{ij} d_j A_j(c) \leq Q_{ij}$$

The profit for each plot must be greater than the profit obtained considering reducing the cultivated area, in all the plots, proportional to the water reduction ratio:

$$A_j(c)(\$_c P_c - I_j - W \cdot d_j) \geq \\ \left[A_{max(j)}(\$_{max} P_{max} - I_{max} - W \cdot d_{max(j)})\right] \phi$$

The profit reduction, ξ_j, with a crop selection for a plot, given by the following equation, must be similar in the whole region:

$$\xi_j = \frac{A_j(c)(\$_c P_c - I_j - W \cdot d_{c,j})}{A_{max(j)}(\$_{max} P_{max} - I_{max} - W \cdot d_{max)(j)}}$$

Water distribution network design has been treated with many traditional mathematical programming techniques, but only after making several strong assumptions. Some of them include considering the network layout independently of the overall network design, the linearization of the objective functions in segments, and the assumption that all the hydraulic parameters of the network are given (water and pressure requirements) [1, 7, 9, 12, 16, 17, 18, 19]

[1] Crop for each plot that generates the maximum profit without water restrictions.

3.3.2 Design of Irrigation Networks

For the solution of irrigation water distribution network design, two cooperating families of agents are proposed. The first family (*family1*) selects crops and defines the optimal area to cultivate for each plot in the region, while the second family (*family2*) searches for the least cost network layout. In particular, *family2* looks for the minimum distance network layout connecting all the hydraulic nodes that supply water to each plot in an agricultural region. The proposed network must guarantee that the bigger transport of water travels the minimal distance in order to minimize costs. Agents in *family1* select a crop for each plot region. This selection determines the required water for each plot and its productivity. This first family tries to maximize the global productivity without exceeding the irrigation water disposability and without violating the restrictions of the problem. To generate compromise solutions a relation is used between the distance that water must travel from the supply to the node and the required water in each plot. The idea is to connect the most demanding nodes closer to the supply reducing with this the water transport rate in the links. The definitions of how to decide where to move an agent in a family follows the approach given by [10] and is described in the following sections.

3.3.3 Optimization of the irrigated area

For each plot there is a list of possible crops. Each crop has its own productivity function that depends on the supplied water. A Q value is associated to each possible crop c of plot j ($Q(c,j)$). A heuristic value, ξ_j, is used to evaluate how good a crop, j, is with respect to the others. The profit distribution must be similar in all the plots, avoiding the existence of predominant crops.

$$\xi_j = \frac{A_j(c)(\$_c P_c - I_j - W \cdot d_{c,j})}{A_{max(j)}(\$_{max} P_{max} - I_{max} - W \cdot d_{max(j)})} \forall c \in list_k$$

where, $list_k$ is the list of crops that have not been selected for ant k in *family1*. A numerical value for ξ_j can be determined in many ways. In this paper, ξ_j is initially assigned a value close to one, which means that there is no water restriction and the crop that generates the greatest economical benefits is selected on each plot. Normally, this crop consumes large quantities of water producing infeasible solutions, so the ξ_j value is gradually reduced until feasible solutions are found.

The action choice rule has the following form:

$$c_j = \begin{cases} \max\limits_{c \in list} \arg[\xi_j + Q(c,j)] & \text{if } t \leq t_s \\ \frac{Q(c,j)+\xi_j}{\sum\limits_{i \in list}[Q(i,j)+\xi_j]} & \text{if } i \in list_k \end{cases}$$

The reward function is defined by:

$$r(c) = \begin{cases} profit \cdot \sum_j A_j(c) & \text{if } c \in list_{k-best} \\ 0 & \text{otherwise} \end{cases}$$

where $list_{k-best}$ is the crop configuration to be rewarded and *profit* is given by z_2 (see ec. 11).

Q-values are updated with the following actualization rule:

$$\widehat{Q}(c,j) \leftarrow (1-\alpha) \cdot \widehat{Q}(c,j) + \alpha \left[r(c) + \gamma \max_{c' \in list_k} \widehat{Q}(c',j) \right]$$

where α is the learning step and γ is the discount factor.

The algorithm starts with the same number of agents for each family. Agents in *family1* generate a feasible crop configuration and each one is assigned to a corresponding agent of *family2*. A crop selection determines the water demands from which a network layout is produced.

3.3.4 Optimal network layout

For *family2*, a heuristic value is used to evaluate how good is to connect node x to node y:

$$H(x,y) = \frac{Q_T}{\$_N \cdot L_N}$$

where Q_T is the total demanded flux, $\$_N$ is the network cost, and L_N is the network length, when node y is connected to the network. The heuristic favors connections that increment the total demanded flux in the network with the lowest cost and length increments. As in all the examples, a Q value is associated to each possible connection ($Q(x,y)$). A combination of these two parameters is used to select which node to connect.

$$s_k(x,y) = \begin{cases} \underset{y \in no\_net_k, x \in net_k}{\arg\max} \{Q(x,y)^{\beta} H(x,y)^{\delta}\} & \text{if } t \geq t_s \\ \frac{Q(x,y)^{\beta} H(x,y)^{\delta}}{\sum\limits_{z \in no\_net_k} [Q(x,z)^{\beta} H(x,z)^{\delta}]} & \text{if } y \in no\_net_k \end{cases}$$

where, k is the kth. agent of *family2*, *net* and *no_net* are lists of connected and disconnected nodes, respectively, for a particular agent. $t \in [0,1]$ is a stochastic value calculated in each iteration. t_s is initially given a t_{max} value and is decremented on each iteration with the following formula: $t_s(t) = \frac{t_s(t-1) * \lambda}{t_{max}}$, where $\lambda < 1$. During the first iterations, exploration (with uniform distribution) is favored over exploitation. This is gradually reversed on later iterations. β and δ are exponents representing the relative importance of their bases.

The reward function for connecting node x with node y is defined as follows:

$$r(x,y) = \begin{cases} \frac{1 \times 10^6}{L_{k-best} \cdot \$_{k-best}} & \text{if } (x,y) \in net_{k-best} \\ 0 & \text{otherwise} \end{cases}$$

where net_{k-best} is the network to be rewarded, L_{k-best} is its length and $\$_{k-best}$ its cost.

As in the previous examples, rewards are given after all the agents in both families have reached a solution. Q values are updated with the following formula:

$$\widehat{Q}(x,y) \leftarrow (1-\alpha) \cdot \widehat{Q}(x,y) + \alpha \left[r(x,y) + \gamma \max_{y' \in net_k} \widehat{Q}(x',y') \right]$$

The algorithm starts with the same number of agents in each family. Agents in *family1* generate feasible crop configurations and each one is assigned to a corresponding agent in *family2*. Given the crop configuration water demands can be stablished, and subsequently the network layout and pipe diameters selection for each link. At this stage, the network costs can be determined and the solutions evaluated. The evaluation is made considering non domination criteria in the Pareto sense. A reward, calculated with its corresponding equation, is given to the links and crops selected in all the non dominated solutions found on each iteration.

Two types of punishments were considered. If constraints of a single element (plot or link) are violated, only the element responsable for the voilation is punished. However, if generalized constraints are violated (constraints considering summation of values), all the elements are punished. The punishment is calculated considering the normalized difference between the value of the proposed solution and the boundary value of the constraint. For example, consider the irrigation ratio of available water $\phi = 0.6$, for region V_t=100 and a given plot p. Let us say, that the best crop selection requirements $d_{max(j)} \cdot A_{max(j)} = 2$. If selected crop requirements are $V_p(c) = 3$, then the punishment for the crop selected in plot p is equal to: $(2-3)/3$. Furthermore, if the water requirements in the region are: $\sum_j V_j(c) = 70$, then the punishment for every plot is equal to: $((0.6)(100) - 70)/(0.6)(100)$.

After the rewards and punishments are given, the whole process is repeated until only non-dominated solutions are found or a given number of iterations is reached.

MOAQ consider problem constraints in two ways: (i) constraints that are implicit in the design process. For example, the diameter selection of the agents is made according to the Hazen-Williams equation, depending on the flux, pipe characteristics and required pressure. Another example of this type of constrint, is the parameter xi_j which is used as the heuristic value for crop selection. (ii) Constraints that cannot be evaluated until certain decisions are made. These constraints are evaluated after solutions are proposed and are used to assign punishments.

3.4 Results

3.4.1 Least-cost pipe network design

Our first approach was to compare MOAQ to existing methods for water network design. As mentioned in section 3.3 the design network problem has been solved only after making strong assumptions and for a single objective (least-cost pipe network design).

MOAQ is applied with a single family of agents in the design of a two-loop network problem studied with mathematical proggraming techniques,

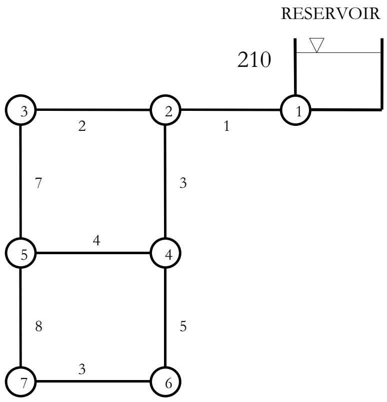

Figure 10: A two loop network.

by [1], later by [15], and only recently by [20] with genetic algorithms. In this problem only the first mathematical model described in section 3.3 to minimize the network cost is considered. The network's layout has a single source at a 210 mt. fixed pressure head, and 8 pipes arranged in two loops as shown in figure 10. The pipes are all 1,000 mt. long with an assumed Hazen-Williams coefficient of 130. The data for the nodes is summarised in Table 4. The minimum acceptable pressure requirements for nodes 2-7 are defined as 30 mt. above the ground level. There are fourteen commercially available diameters. Table 5 shows the total cost per meter of pipe length for different pipe sizes.

In this example only one family of agents and a heuristic value are used to evaluate how good is to select diameter i in pipe segment n:

$$H(i,n) = \frac{1}{\$_n(i) \cdot L_n}$$

where $\$_n(i)$ is the pipe segment n cost in the case diameter i is selected, and L_n is the segment length. The transition rule, as in the previous examples, combines a heuristic value with the associated Q value, if $q \geq q_0$, or makes a random selection using a uniform distribution:

Table 4: Node data for the two loop example.

Node No.	Demand m^3/h	Ground Level m
1(source)	-1120.0	210.0
2	100.0	150.0
3	100.0	160.0
4	120.0	155.0
5	270.0	150.0
6	330.0	165.0
7	200.0	160.0

Table 5: Cost data for pipes.

Diameter (Inches)	Cost (units)
1	2
2	5
3	8
4	11
6	16
8	23
10	32
12	50
14	60
16	90
18	130
20	170
22	300
24	550

$$s_k(i,n) = \begin{cases} \arg\max\limits_{i \in list_k}\{Q(i,n)^\beta H(i,n)^\delta\} & \text{if } q \geq q_0 \\ \frac{Q(i,n)^\beta H(i,n)^\delta}{\sum\limits_{z \in list_k} [Q(z,n)^\beta H(z,n)^\delta]} \end{cases}$$

where, k is the kth. agent, $list_k$ is a list of candidate diameters. β and δ are exponents representing the relative importance of their bases.

The rewards are evaluated as follows:

$$r(i,n) = \begin{cases} \frac{1 \times 10^6}{L_N \cdot \$_{k-best}} & \text{if } (i,n) \in net_{k-best} \\ 0 & \text{otherwise} \end{cases}$$

where L_N is the length of the network and $\$_{k-best}$ is the best network cost. Rewards are given after all the agents have reached a solution. Q values are updated using:

$$\widehat{Q}(i,n) \leftarrow (1-\alpha) \cdot \widehat{Q}(i,n) + \alpha \left[r(i,n) + \gamma \max_{i' \in list_k} \widehat{Q}(i',n) \right]$$

The parameters used in this example were: $\gamma = 0.3$, $\alpha = 0.1$, $\beta = 1.0$, and $\delta = 2.0$. Table 6 lists least-cost solutions, in terms of diameter and lengths, as reported by different researches as well as those obtained by MOAQ. The optimal pipe diameters D are given in inches and the pipe-segment lengths L in meters.

To assess the quality of the solutions obtained and compare them with the solutions from the literature, pressure heads in network nodes are compared in table 7. As can be seen from the table some of the solutions of Kessler and Shamir are infeasible since the pressure head is less than 30 (marked with *), consequently, MOAQ obtained the best feasible solution.

3.4.2 Multiple Objective Irrigation Network Design

MOAQ was then tried on the multiple objective design problem. The algorithm was tested in the design of a 20 hydraulic node network configured as shown in Figure 8. The data for the plots under consideration are listed in table 8. Table 9 contains the data for the crops used in this problem. The

Table 6: Solutions of the two-loop network.

Pipe	Alperovits and Shamir		Kessler and Shamir		Savic (GA)		MOAQ	
	L	D	L	D	L	D	L	D
[−]	[m]	[in]	[m]	[in]	[m]	[in]	[m]	[in]
1	256.00	20	1000.00	18	1000.00	20	1000.00	18
	744.00	18						
2	996.38	8	66.00	12	1000.00	10	1000.00	10
	3.62	6	934.00	10				
3	1000.00	18	1000.00	16	1000.00	16	1000.00	16
4	319.38	8	713.00	3	1000.00	1	1000.00	4
	680.62	6	287.00	2				
5	1000.00	16	836.00	16	1000.00	14	1000.00	16
			164.00	14				
6	784.94	12	109.00	12	1000.00	10	1000.00	10
	215.06	10	891.00	10				
7	1000.00	6	819.00	10	1000.00	10	1000.00	10
			181.00	8				
8	990.93	6	920.00	3	1000.00	1	1000.00	1
	9.07	4	80.00	2				
Cost (Units)	497,525		417,500		420,000		419,000	

Table 7: Pressure head for the two-loop network.

Node i	Alperovits and Shamir	Kessler and Shamir	Savic (GA)	MOAQ
2	53.80	53.09	55.86	53.09
3	31.89	29.59*	30.30	30.45
4	44.71	43.45	46.38	43.59
5	31.65	29.38*	31.61	33.13
6	30.83	29.69*	30.50	30.11
7	31.11	29.59*	30.52	30.13

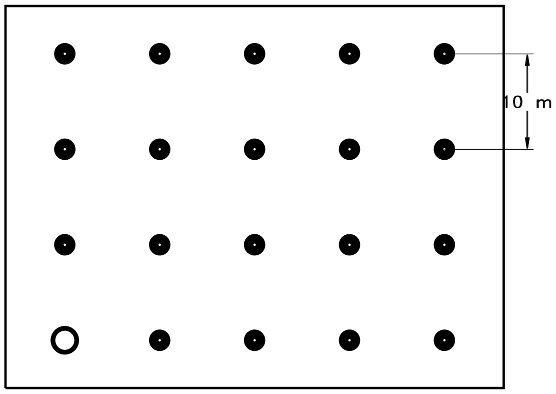

Figure 11: Hydraulic nodes configuration, where $\bullet$ are hydraulic nodes and $\circ$ is the supply node.

parameters used in this work were: $\lambda = 0.9$, $\gamma = 0.3$, $\alpha = 0.1$, $\xi_{initial} = 0.9$, $\beta = 1.0$, $\delta = 2.0, T_{max} = 100$ for network optimization, $T_{max} = 200$ for crop selection and profit optimization, considering 10 agents for both families. Initial Q values for links in the network optimization were initialized using $Q = (\sum_{ij}(L_{ij} \cdot \phi V_t))/(N_t^2 - N_t)$ where N_t is the number of hydraulic nodes under consideration. Q values for each crop in the plots were initialized considering the profit of the best available crop divided by the total irrigated area times the water demand for the whole region assuming no water constraints $(Q = \sum_{i=1}^{N_p} P_i/(\sum_{i=1}^{N_p} A_i * \sum_{i=1}^{N_p} V(i)))$ for each crop in all the plots (N_p).

Figure 12 shows the final Pareto set in the objectives space. A better understanding of the algorithm behavior can be seen in Figure 13, where the fraction of non-dominated solutions in the set versus the iteration number is shown. In this case, the plot was constructed by comparing each individual obtained after an iteration against all other solutions including the solutions in the Pareto set. The fraction of non-dominated solutions has a continuous growing behavior and then stabilizes. As the problem difficulty increases, more evaluations are required to get a larger fraction of non-dominated solutions in the Pareto set. Once the maximum is obtained, it is maintained over time, until the algorithm finishes. Thus, the algorithm behavior remains stable for some number of iterations.

Table 8: Plot data.

Plot No	x,y,z	Area (ha)
1	10,10,0	Supply
2	20,10,0	20
3	30,10,0	50
4	40,10,0	100
5	50,10,0	30
6	10,20,0	8
7	20,20,0	25
8	30,20,0	40
9	40,20,0	10
10	50,20,0	11
11	10,30,0	10.3
12	20,30,0	15.4
13	30,30,0	20.5
14	40,30,0	21
15	50,30,0	201
16	10,40,0	21
17	20,40,0	201
18	30,40,0	55
19	40,40,0	33
20	50,40,0	21

Table 9: Crop data (Taken from FAO 33).

Crop	FLUX (m^3/(s ha))	Yield (ton/ha)	Price ($/ton)
Alfalfa	0.00697	28	10
Banana	0.00958	60	200
Bean	0.00218	2	50
Cabbage	0.00218	35	40
Citrus	0.00523	45	80
Cotton	0.00566	5	100
Grape	0.00348	30	300
Groundnut	0.00305	3	20
Maize	0.00348	9	20
Olive	0.00261	20	300
Onion	0.00239	45	20
Pea	0.00218	3	30
Pepper	0.00392	25	100
Pineapple	0.00436	90	10
Potato	0.00305	35	10
Rice	0.03048	8	5
Safflower	0.00523	4	400
Sorghum	0.00283	5	200
Sugar beet	0.00327	60	50
Sugarcane	0.01088	100	45
Sunflower	0.00436	3.5	20
Tobacco	0.00261	2.5	20
Tomato	0.00261	65	50
Watermelon	0.00261	35	40
Wheat	0.00283	6	20

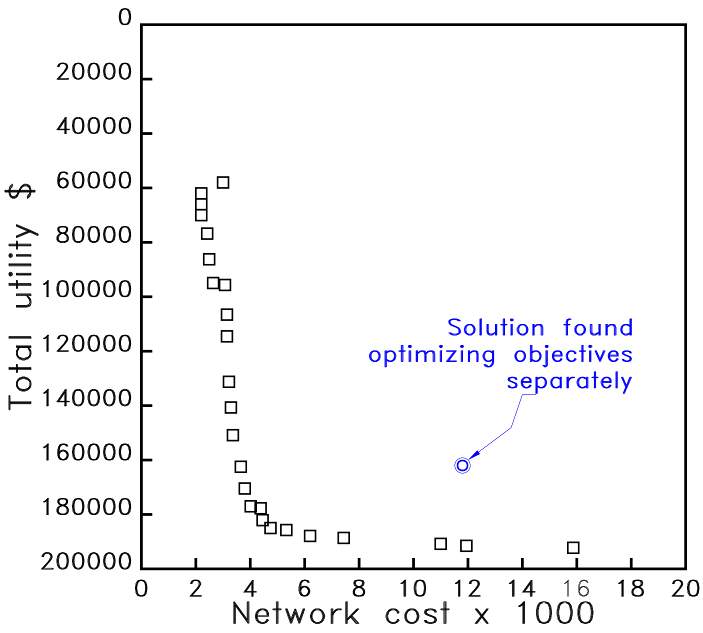

Figure 12: Final Pareto set.

The design of irrigation water distribution networks is an extremely complex problem and, as mentioned previously, it has been treated and solved only after considering very strong assumptions. With MOAQ, designs for irrigation networks were obtained considering not only the network layout design but also the crop selection and cultivated area. It is worth noting that all the solutions found in the Pareto set by MOAQ satisfied all the constraints, making them feasible in hydraulic and economic terms.

Figure 12 also shows the network design solution obtained using distributed reinforcement learning considering the optimization problem separately. The crop selection problem was solved before the network optimization and considering the hydraulic parameters obtained with the solution for crops. As it can be seen in figure 12 many solutions in the Pareto set are much better in both objectives than the solution obtained by considering the objetives separately.

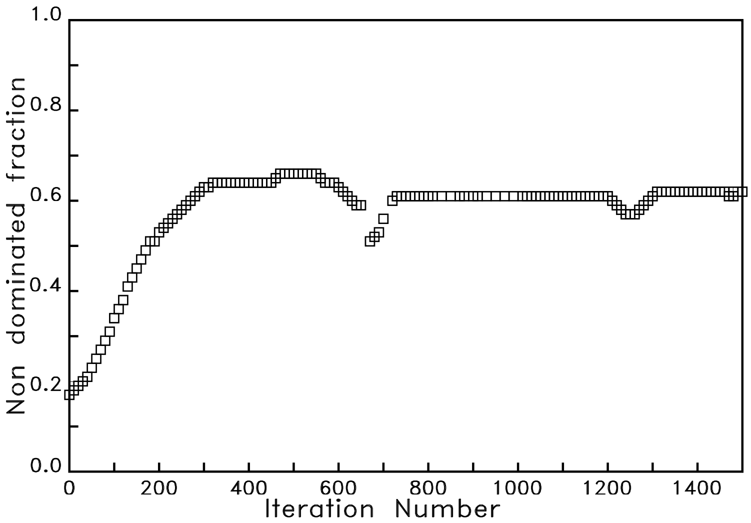

Figure 13: Non dominated solutions per iteration.

4 Discussion

From the results obtained in the previous section, we can conclude that MOAQ is adequate to solve some multiple objective optimization problems. In particular, the codification of the first two problems into MOAQ's framework suggests than many problems that have been treated with genetic algorithms can be solved with MOAQ. Furthermore, at least in these two problems, MOAQ showed comparatively better results than those obtained with genetic algorithms, although a more thorough comparison is needed.

A clear advantage of MOAQ over genetic algorithms approaches is that MOAQ does not need to codify the problem into a particular representation (i.e., binary). A large proportion of the time spent in a successful application of genetic algorithms is devoted to finding an adequate codification of the problem. In contrast, MOAQ's codification is much more transparent and does not loose expressivity, as it is made in terms of the mathematical constraints of the problem. This was evident when MOAQ was applied to the design of irrigation water distribution networks. MOAQ obtained comparatevely better results when applied to single objective and feasible solutions in the multiple objective case.

Theoretical analysis of the algorithm is still required in terms of scability, but as can be seen from the results, MOAQ looks as a promising alternative for the solution of multiple objective optimization problems.

A careful sensibility analysis is required for the parameters involved in the algorithm (α, β, γ, δ). In this work some variations of these parameters were tested and the values reported are those that gave us better results.

5 Conclusions and Future Work

Many real world applications require compromise solutions between several conflicting objectives. The use of traditional mathematical methods has been severely restricted and alternative techniques need to be tried. In this paper, an Ant-Q based algorithm called MOAQ for the solution of multiple objective optimization problems has been presented. MOAQ uses a family of agents for each objective and by sharing partial results between agents, global compromise solutions can be found. In particular, MOAQ was used to solve successfully some simple multiple objective optimization problems that have been solved with genetic algorithms with comparatively better results. Furthermore, MOAQ was used in the design of water distribution irrigation networks, a complex real-world problem.

As future work, we would like to test MOAQ with more objectives. In particular for the design of irrigation networks, we can try to minimize the length of the network with the following objective: $min\ z_3 = \sum_{ij} L_{ij}$. This is needed as not always the minimum cost network has the minimum length. We would also like to test MOAQ on other real multiple objective optimization problems, especially those presented in engineering design and problems which are considered difficult for genetic algorithms (e.g., [2, 3]).

Acknowledgements

This research was partly supported by CONACyT (México). The authors would like to thank Enrique Sucar and the anonymous referees who provided helpful comments on an earlier draft of this paper.

References

[1] Alperovits, E., Shamir, U. (1977). Design of Optimal Water Distribution Systems, *Water Research Resources* 13(6): 885-900

[2] Deb, K. (1998). Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, *Technical Report CI-49/98*, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany.

[3] Deb, K. (1999). Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In *Proceedings of Evolutionary Algorithms in Engineering and Computer Science* (EUROGEN'99).

[4] Doorenbos J., Kassam A.H. (1986). Yield Response to Water, *Food and Agriculture Organization of the United Nations, Irrigation and Drainage paper 33*, Rome Italy.

[5] Dorigo, M., Maniezzo, V., Colorni, A. (1996). The Ant-System: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics*, 26(1): 1-13 (Part B).

[6] Dorigo, M., Gambardella, L.M. (1997). Ant Colony System: Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*,1(1): 53-65, April 1997.

[7] Eiger, G. Shamir, U., Ben-Tal, A. (1994). Optimal Design of Water Distribution Networks, *Water Research Resources* 30(9): 2637-2646.

[8] Fonseca,C.M., Flemming, P.J. (1993). Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. In S. Forrest (Ed.), Genetic Algorithms: *Proceedings of the Fifth International Conference* (pp. 416-423). San Mateo, CA: Morgan Kaufmann.

[9] Fujiwara, O., Khang D.B. (1990). A two phase decomposition method for optimal design of looped water distribution networks, *Water Resources Research* 26(4): 539-549.

[10] Gambardella, L.M., Dorigo, M. (1995) Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. In *Proceedings of the 12th. International Machine Learning Conference* (ML-95), pp., 252-260, Morgan Kaufmann.

[11] Glouter, I.C., (1987), Current and Future use of Systems Analysis in Water Distribution Network Design, *Civil Engineering Systems*, 4(4), 175-184.

[12] Granados, A. (1986). *Infraestructura de Regadios: Redes Colectivas de Riego a Presión E.T.S. de Ingenieros de Caminos, Canales y Puertos*, ISBN: 84-7493-104-5.

[13] Horn J., Naftopolis, N. (1993) Multiobjective optimization using the niched Pareto genetic algorithm. IlliGAL Report 93005, University of Illinois at Urbana-Champaign.

[14] Horn J., Naftopolis, N., Golberg, D.E. (1994). A niched Pareto genetic algorithm for multi-objective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1 (pp. 82-87). New York: IEEE.

[15] Kessler, A. and Shamir, U. (1989), Analysis of the linear programming Gradient Method for Optimal Design of Water Supply Networks, *Water Resources Research*, 25(7), 1469-1480.

[16] Labye, Y. Olson, M.A. Galand, A., Tsiourtis, N. (1988). Design and Optimization of Irrigation Distributions Networks, *FAO Irrigation and Drainage Paper 44*, Rome.

[17] Loganathan, G.V. Greene, J., Ahn, T.J. (1995). Design Heuristic for Globally Minimum Cost Water-Distribution Systems, *Journal of Water Resources Planning and Management* 121(2): 182-192.

[18] Ormsbee, L.E., Wood, D.J. (1986). Hydraulic Design Algorithms for Pipe networks, *ASCE Journal of Hydraulic Engineering* 112(12): 1195-1207.

[19] Quindry, G.E., Brill, E.D., Liebman, J.C. (1981). Optimization of Looped Distribution Systems, *ASCE Journal of Environmental Engineering* 107(EE4): 665-679.

[20] Savic, D.A. and Walters G.A., (1995). An Evolution Program for Optimal Pressure Regulation in Water Distribution Networks, *Eng. Opt.*. 24(3): 197-219.

[21] Schaffer, D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J.J. Grefenstette (Ed.), *Genetic Algorithms*

and Their Applications: Proccedings of the First International Conference on Genetic Algorithms (pp. 93-100). Hillsdale, NJ: Lawrence Erlbaum.

[22] Srinivas, N., Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3), 221-248.

[23] Sutton, R.S., Barto, A.G. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

[24] Valenzuela-Rendón, M., Y Uresti-Charre, E. (1997). A non generational genetic algorithm for multiobjective optimization. In *Proceedings of the seventh international conference on genetic algorithms*, pp. 658-665.

[25] Watkins, C.J.C.H. (1989). *Learning from Delayed Rewards.* Ph.D. thesis, Cambridge University.

[26] Yates, D.F., Templeman, A.B., and Boffey, T.B., (1984), The computational complexity of the Problem of Determining Least Capital Cost Designs for Water Supply Networks, *Engineering Optimization*, 7(2), 142-155.