A genetic based algorithm for the quadratic 0-1 problem

G. Schütz * F. M. Pires †

* Escola Superior de Tecnologia, Universidade do Algarve
gschutz@ualg.pt

† Faculdade de Ciências e Tecnologia, Universidade do Algarve
mpires@ualg.pt

Abstract

In this work we present an algorithm to the unrestricted binary quadratic program. This approach combines genetic operators with greedy and heuristic procedures. We started from a genetic based algorithm and replaced the random mutation by a greedy procedure based on each variable contribution to the objective function. We also introduced in the genetic population an individual obtained by a heuristic that finds a local star minimum point. Computational experience with a set of test-problems, known from literature, is reported and analysed. Our results are quite promising.

Keywords: Quadratic 0-1 programming, Genetic Algorithms, Hybrid Algorithms

1 Introduction

This work presents a genetic based algorithm for the unrestricted binary quadratic program (QP 0-1):

$$\begin{array}{c} max\ f\ (x)\ =\ x^t\, A\, x \\ x \in \{0,\, 1\}^n \end{array} \tag{1}$$

Since binary quadratic problems with an explicit linear component in the objective function can always be transformed in (1) we adopt this form, without loss of generality. We also consider A a symmetric matrix as any unsymmetric problem can be easily converted to this form.

This problem, known to be NP-hard [3], is very interesting since it arises in a large number of applications. Some of the QP 0-1 applications referred in literature deal with

capital budget and financial analysis problems [9], CAD problems [7], message traffic management problems [2] and machine scheduling problems [1]. Also some difficult graph problems (like the maximum clique problem) can be formulated and solved as a quadratic 0-1 programming problem [11].

The main purpose of this work is to combine the reasoning of genetic algorithms with greedy procedures and heuristic approaches in an algorithm for the QP 0-1, and study its behaviour. We describe our hybrid algorithm, in section 2. We present and analyse the computational results on a set of test problems known from literature, in section 3. Conclusions and final remarks are in section 4.

2 Algorithm implementation

In a genetic algorithm it is necessary to generate the individuals of the initial population accordingly to the codification that will be used. In each iteration selection, crossover and mutation operators are usually applied.

2.1 Initial population

In the QP 0-1, as it is well known, variables whose partial derivatives have fixed sign in the unitary hypercube can be fixed either to 0 or 1 according to that sign. Let,

$$m_i \;\leq\; \frac{\partial f(x)}{\partial x_i} \;\leq\; M_i \quad , \quad \text{for} \quad i \;=\; 1, ..., n$$

where the m_i and M_i values computed as:

$$m_i \;=\; 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}^{-} + a_{ii} \text{ and } M_i \;=\; 2 \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}^{+} + a_{ii} \tag{2}$$

with

$$a_{ij}^{+} \;=\; max\,\{0,\, a_{ij}\} \text{ and } a_{ij}^{-} \;=\; min\,\{0,\, a_{ij}\}\,,$$

provide narrow bounds.

So, we have an easy way to fix variables according to:

$$m_i \;\geq\; 0 \;\Rightarrow\; x_i^{*} \;=\; 1; \tag{3}$$

$$M_i \;\leq\; 0 \;\Rightarrow\; x_i^{*} \;=\; 0. \tag{4}$$

Table 1: Sample of a 10 individual population.

Individual	1	2	3	4	5	6	7	8	9	10
fit	1,78	0,89	1,33	0,22	2,00	0,44	0,00	1,56	1,11	0,67

Table 2: Sample of the Stochastic Universal Sampling selection.

roulette	1,78	2,67	4,00	4,22	6,22	6,66	6,66	8,22	9,33	10,00
pointers	0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4
selected	1	1	2	3	5	5	6	8	9	10

In order to reduce the problem size, this process of definitively fix some variables is initially performed.

So instead of representing each individual by a binary array of length n, we used a binary array of length l, with l equal to the number of the remaining free variables ($l \leq n$). For instance, considering a 10 variables problem we could have:

fixed variables	[	1	0	0	1	]					
variables indices	[	2	3	7	9	|	1	4	5	6	8	10	]
individual i						[	0	0	1	1	1	0	]

To obtain the initial population, with dimension *Pop*, we randomly generate *Pop* binary arrays of length l. We denote each individual by x^i, $i = 1, ..., Pop$, and its objective value is

$$f\left(x^i\right) = \sum_{j=1}^{n} a_{j\,j}\, x_j^i + 2 \sum_{j=1}^{n} \sum_{k=j+1}^{n} a_{j\,k}\, x_j^i\, x_k^i \tag{5}$$

The population is sorted in non increasing order by the objective values and the individuals fitness are assigned according to their ranking, $rank\ (i)$, in population, as follows:

$$fit\left(x^i\right) = 2 - \frac{2\ rank\ (i)}{Pop - 1} \quad , \ i = 1, ..., Pop \tag{6}$$

with $0 \leq rank\ (i) \leq Pop - 1$. This way the relative fitness of the best individual will be 2 and $\sum_{i=1}^{Pop} fit\left(x^i\right) = Pop$.

2.2 Selection and crossover

Individuals are selected by Stochastic Universal Sampling (SUS), which may be seen as the result of spinning a roulette wheel with slots proportional in width to the fitness of the individuals in the population, using multiple equally spaced pointers. So we construct the roulette wheel according to

$$\begin{aligned} &roulette\ (0) = 0\ , \\ &roulette\ (i) = roulette\ (i-1) + fit\left(x^i\right) \quad , \ i = 1, ..., Pop \end{aligned} \tag{7}$$

Next we randomly generate a number in [0, 1] and successively add 1 to form pointers. For instance, considering a 10 individual population with the fitness values of table 1 we would obtain the *roulette*, pointers and select individuals of table 2, supposing that 0.4 was the number randomly generated in [0, 1].

In order to combine the selected individuals we apply single-point crossover to individuals pairs. The pairs are randomly formed with different selected individuals.

2.3 Greedy procedure as a mutation operator

Afterwards, a new mutation operator is applied. It consists in a greedy procedure, based on the increment a variable will bring to the objective function if its value is changed (0 to 1 or 1 to 0). So, for each individual x^i , $i = 1, ..., Pop$, this procedure acts as follows:

(i) Compute

$$s_j = \sum_{\substack{k=1 \\ k \neq j}}^{n} a_{j\,k}\, x_k^i + a_{j\,j} \ , \ j = 1, ..., l$$

(ii) Order s_j , $j = 1, ..., l$, by its absolute values in a non-increasing way.

Let *perm* contain the ordered variables indices.

(iii) for $k = 1$ to l do

if $s_{perm(k)} > 0 \ \wedge \ x^i_{perm(k)} = 0$

then $x^i_{perm(k)} = 1$ if it improves the objective value

if $s_{perm(k)} < 0 \ \wedge \ x^i_{perm(k)} = 1$

then $x^i_{perm(k)} = 0$ if it improves the objective value.

For instance, consider the following quadratic form matrix

$$A = \begin{bmatrix} -1 & -1 & 1 & -1 \\ -1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix}$$

Applying (2), (3) and (4) to A we fix $x_3 = 1$. Consider, for example, the individual $x^i = [1\ 0\ 1]$, then the correspondent solution is (1, 0, 1, 1) with objective value equal to 0.

Applying mutation operator as defined we obtain:

$$s_1 = -1, \quad s_2 = 1 \text{ and } s_4 = -1$$

and

$$s_1 < 0 \ , \ x^i_1 = 1 \text{ and } f\ (0, 0, 1, 1) = 1 \Rightarrow x^i_1 = 0$$

$$s_2 > 0 \ , \ x^i_2 = 0 \text{ and } f\ (0, 1, 1, 1) = 4 \Rightarrow x^i_2 = 1$$

$$s_4 < 0 \ , \ x^i_4 = 1 \text{ and } f\ (0, 1, 1, 0) = 2 \Rightarrow x^i_4 = 1$$

So the individual become $x^i = [0\ 1\ 1]$ corresponding to (0, 1, 1, 1) solution with $f\left(x^i\right) = 4$.

2.4 A local star minimum point heuristic

In order to introduce some diversification into the population we replace the worst individual by another obtained with a heuristic method. This is done, only once, after a fixed number of iterations. We use a heuristic, similar to the one described in Gulati et al (1984), based on finding a point that is better than all its adjacent points.

This heuristic can be described as follows:

Step 1:

Let

$$fv = 0,\ F_1 = \emptyset,\ F_0 = \{1, ..., n\},\ u = (u_1, ..., u_n)^t = 0$$

Step 2:

For $i = 1, ..., n$

If $2\,u_i + a_{ii} > 0$ do

$$\begin{array}{l} fv = fv + 1 \\ F_1 = \left\{ \begin{array}{ll} F_1 - \{\, i \,\} & ,\ i \in F_1 \\ F_1 \cup \{\, i \,\} & ,\ i \in F_0 \end{array} \right. \\ F_0 = \{\, 1, ..., n \,\} - F_1 \\ u_i = \left\{ \begin{array}{ll} -\left(a_{ii} + \sum\limits_{\substack{j \in F \\ j \neq i}} a_{ij} \right) & ,\ i \in F_1 \\ \sum\limits_{j \in F} a_{ij} & ,\ i \in F_0 \end{array} \right. \\ \begin{array}{l} u_k \\ k \neq i \\ k \in F_1 \end{array} = \left\{ \begin{array}{ll} u_k + a_{ik} & ,\ i \in F_0 \\ u_k - a_{ik} & ,\ i \in F_1 \end{array} \right. \\ \begin{array}{l} u_k \\ k \neq i \\ k \in F_0 \end{array} = \left\{ \begin{array}{ll} u_k + a_{ik} & ,\ i \in F_1 \\ u_k - a_{ik} & ,\ i \in F_0 \end{array} \right. \end{array} \qquad (8)$$

Step 3:

If $fv = 0$ then

$$\bar{x} \text{ such as } \bar{x}_j \;=\; \begin{cases} 1 & , \; j \in F_1 \\ 0 & , \; j \in F_0 \end{cases} \quad \text{is a local star minimum point}$$

$$f(\bar{x}) \;=\; \sum_{j \in F_1} - u_j$$

stop

Else do

$fv = 0$

go to Step 2

Theorem: The heuristic solution is a local star minimum point, that is,

$$f(\bar{x}) \;\geq\; f\left(x'\right) \;,\; \forall x' \;=\; (\bar{x}_1, ..., \bar{x}_{k-1}, 1 - \bar{x}_k, \bar{x}_{k+1}, ..., \bar{x}_n)$$

Proof:

This heuristic obtains a solution $\bar{x} \;=\; (\bar{x}_1, ..., \bar{x}_{k-1}, \bar{x}_k, \bar{x}_{k+1}, ..., \bar{x}_n)$ such that $2\,u_i \;+\; a_{ii} \;\leq\; 0\,,\; \forall\, i \;=\; 1, ..., n$, so, replacing u_i using (8) it follows that:

$$a_{ii} \;+\; 2 \sum_{\substack{j \,\in\, F \\ j \neq i}} a_{i\,j} \geq 0 \quad , \quad i \in F_1 \tag{9}$$

$$a_{ii} \;+\; 2 \sum_{j \,\in\, F} a_{i\,j} \leq 0 \quad , \quad i \in F_0 \tag{10}$$

Then

If $\bar{x}_k \;=\; 1 \;\Rightarrow f(x') = f(\bar{x}) - \left(a_{kk} + 2 \sum_{\substack{j \in F \\ j \neq k}} a_{kj} \right)$ and by (9) we have

$$f(\bar{x}) \;\geq\; f\left(x'\right)$$

If $\bar{x}_k \;=\; 0 \;\Rightarrow f(x') = f(\bar{x}) + \left(a_{kk} + 2 \sum_{j \in F} a_{kj} \right)$ and by (10) we have

$$f(\bar{x}) \;\geq\; f\left(x'\right)$$

2.5 A combined genetic and greedy algorithm

Now, we are able to describe, more formally, our algorithm approach combining genetic operators with greedy and heuristic procedures. In this algorithm $\tilde{x}$ represents the best solution, *Maxiter* contains the number of iterations to perform, $\alpha \in [0, 1[$ gives the fraction of *Maxiter* iterations to perform before introducing the heuristic individual and $\lceil y \rceil$ represents the smallest integer greater than y.

Step 1:

Use (2), (3) and (4) to fix variables

Generate $Pop \times \quad l$ numbers randomly distributed in [0,1]
obtaining Pop individuals x^1, ..., x^{Pop}

$iteration = 0$

Step 2:

Compute $f(x^i)$, $i = 1,\ldots,n$, using (5)

$$sol = \min_i f\left(x^i\right)$$
$$\tilde{x} = x^i : f\left(x^i\right) = sol$$

Step 3:

Select individuals according to (6) and (7)

Apply single-point crossover to the random formed pairs of selected individuals

New individuals will replace the old ones only if
they produce a better objective function value.

Apply the greedy procedure, described in 2.3, only to the replaced individuals

$iteration = iteration + 1$

Step 4:

If $iteration < Maxiter$

If $iteration = \lceil \alpha \times Maxiter \rceil$

Replace the worst individual by the heuristic solution described in 2.4

Go to Step 2

Else

Stop

3 Computational Results

To test our algorithm we made use of a collection of problems retrieved from the Internet (http://mscmga.ms.ic.ac.uk/jeb/orlib). This collection contains 3 problems sets ("a", "b" and "c") of [10] and 3 sets ("d", "e" and "f") of [4]. We summarise in table 3 the characteristics of these problems, except the "f" set as we were not able to execute it in a PC, due to its problems sizes (500 variables). In fact, in [4] the authors refer that they had to solve them in a VAX Alpha 2100 model 300 computer.

Table 4 presents the results for these problems obtained in a 200 Mhz Pentium MMX PC with 16 MB RAM. In this table column "*sol**" gives the optimal solution (underlined), when known, or the best-published solution. Actually, the best solutions for these problems were obtained by the very efficient Tabu Search based algorithms of [4] and [5], as well as the evolutionary heuristic presented in [8].

We also present, in table 4, the local star minimum point heuristic solution obtained to produce an individual. This individual replaces the worst one after $^2/_3$ of the total number of iterations. This value of α was chosen in order to produce some diversification. In fact if this individual is introduced too early, the other individuals will converge to it. We run our algorithm three times with different population dimensions and number of iterations (*Maxiter*). So in table 4 we have 3 "GA (*Pop*, iterations)" columns. The columns "Time" and "*gap*" report, respectively, the total execution times in seconds and

$$gap \;=\; \left| \frac{f\,(x) \;-\; sol^*}{sol^*} \right| \;\times\; 100 \quad .$$

The best results obtained are in bold.

3.1 Heuristic results

It is possible to see, in table 4, that the local star minimum point heuristic usually performs quite well, mainly when considering the execution times, which are always less than 0.5 seconds. Nevertheless, in problems with many local optimums, as it happens with "b" set, this heuristic obtains poor results.

3.2 Combined genetic and greedy algorithm results

The results show that this approach is very efficient and consistent, obtaining good quality solutions and reduced execution time, even with an 80 individuals population and 60 iterations. The best results were obtained for "a", "b" and "c" problem sets, as it was expected. In fact, we knew from [4] that set problems "a", "b" and "c" were "easy" and only "c" problems were more time consuming.

For the "difficult" problem sets "d" and "e" the results were not so good. Nevertheless, when we failed to obtain the best-known solutions the *gap* was small except in one case, in each of these sets, where the *gap* was around 5%.

We cannot compare our execution times with those reported in [4], since different machines

Table 3: Summary of problems characteristics.

Probl.	matrix generation		n	density	elements (1)		diagonal (2)		total (3)	fixed
	off diagonal	diagonal			> 0	< 0	> 0	< 0		var. (4)
1a	[-100, 100]	[-100, 100]	50	0,1	58	48	25	25	156	16
2a			60	0,1	79	84	31	28	222	18
3a			70	0,1	113	110	35	34	292	9
4a			80	0,1	145	159	46	33	383	5
5a			50	0,2	119	112	22	28	281	3
6a			30	0,4	91	83	11	19	204	0
7a			30	0,5	107	104	13	17	241	0
8a			100	0,625	149	155	53	46	403	52
1b	[0, 100]	[-63, 0]	20	1	187	0	0	20	207	0
2b			30	1	429	0	0	30	459	0
3b			40	1	771	0	0	40	811	1
4b			50	1	1208	0	0	50	1258	1
5b			60	1	1751	0	0	60	1811	0
6b			70	1	2388	0	0	70	2458	0
7b			80	1	3125	0	0	80	3205	0
8b			90	1	3962	0	0	90	4052	1
9b			100	1	4902	0	0	100	5002	1
10b			125	1	7663	0	0	125	7788	3
1c	[-50, 50]	[-100, 100]	40	0,8	309	316	16	24	665	0
2c			50	0,6	358	405	26	24	813	0
3c			60	0,4	334	367	34	26	761	0
4c			70	0,3	349	371	23	46	789	0
5c			80	0,2	284	357	45	35	721	0
6c			90	0,1	190	210	44	45	489	8
7c			100	0,1	247	248	49	51	595	20
1d	[-50, 50]	[-75, 75]	100	0,1	235	258	49	51	593	4
2d				0,2	516	500	56	44	1116	0
3d				0,3	727	698	48	52	1525	0
4d				0,4	1022	978	43	57	2100	0
5d				0,5	1202	1212	58	41	2513	0
6d				0,6	1478	1470	50	50	3048	0
7d				0,7	1697	1737	43	55	3532	0
8d				0,8	1956	1951	50	50	4007	0
9d				0,9	2203	2143	46	53	4445	0
10d				1	2377	2520	44	55	4996	0
1e	[-50, 50]	[-100, 100]	200	0,1	967	957	88	112	2124	0
2e				0,2	1943	1984	97	103	4127	0
3e				0,3	2959	2891	101	97	6048	0
4e				0,4	3926	3991	85	115	8117	0
5e				0,5	4883	4974	104	94	10055	0

(1) = triangular matrix non zero elements (off diagonal)

(2) = non zero diagonal elements

(3) = triangular matrix non zero elements

(4) = number of fixed variables

Table 4: Genetic based algorithm results.

		Heurisitc		GA (40 / 35)				GA (60 / 40)				GA (80 / 60)			
	*sol**	$f(x)$	*gap*	$f(x)$	*gap*	(1)	Time	$f(x)$	*gap*	(1)	Time	$f(x)$	*gap*	(1)	Time
1a	3414	3381	0,97	**3414**	0,00	3	1,8	**3414**	0,00	3	3,0	3404	0,29	1	5,8
2a	6063	5790	4,50	5986	1,27	11	2,9	**5989**	1,22	4	5,0	**5989**	1,22	5	9,5
3a	6037	6035	0,03	6035	0,03	8	3,9	6035	0,03	18	6,8	**6037**	0,00	13	12,3
4a	8598	7017	18,39	**8598**	0,00	12	5,1	8530	0,79	8	9,0	**8598**	0,00	20	14,0
5a	5737	**5712**	0,44	**5712**	0,44	6	2,5	**5712**	0,44	4	4,0	**5712**	0,44	4	7,7
6a	3980	**3980**	0,00	**3980**	0,00	5	1,2	**3980**	0,00	3	2,0	**3980**	0,00	5	3,8
7a	4541	**4541**	0,00	**4541**	0,00	1	1,2	**4541**	0,00	2	2,0	**4541**	0,00	2	4,0
8a	11109	**11109**	0,00	**11109**	0,00	24	7,3	**11109**	0,00	10	12,2	**11109**	0,00	6	23,6
Average			3,04		0,22	9	3		0,31	7	5		0,24	7	10
1b	133	85	36,09	**133**	0,00	1	0,3	**133**	0,00	1	0,6	**133**	0,00	1	1,0
2b	121	91	24,79	**121**	0,00	2	0,6	**121**	0,00	1	0,9	**121**	0,00	1	1,8
3b	118	102	13,56	**118**	0,00	1	0,9	**118**	0,00	1	1,7	**118**	0,00	1	3,1
4b	129	85	34,11	**129**	0,00	1	1,4	**129**	0,00	1	2,4	**129**	0,00	1	4,4
5b	150	**150**	0,00	**150**	0,00	3	2,0	**150**	0,00	1	3,4	**150**	0,00	1	6,3
6b	146	88	39,73	**146**	0,00	30	2,6	**146**	0,00	26	5,5	**146**	0,00	24	8,9
7b	160	**160**	0,00	**160**	0,00	1	3,6	**160**	0,00	1	5,8	**160**	0,00	1	10,6
8b	145	101	30,34	**145**	0,00	35	4,3	**145**	0,00	15	6,8	**145**	0,00	13	12,8
9b	137	75	45,26	**137**	0,00	3	4,9	**137**	0,00	3	8,1	**137**	0,00	3	14,5
10b	154	90	41,56	**154**	0,00	2	8,4	**154**	0,00	1	14,0	**154**	0,00	3	24,8
Average			24,88		0,00	8	3		0,00	5	5		0,00	5	9
1c	5058	**5058**	0,00	**5058**	0,00	2	2,3	**5058**	0,00	4	4,0	**5058**	0,00	3	7,2
2c	6213	**6213**	0,00	**6213**	0,00	6	3,9	**6213**	0,00	10	6,3	**6213**	0,00	7	11,9
3c	6665	**6649**	0,24	**6649**	0,24	16	4,8	**6649**	0,24	9	7,9	**6649**	0,24	11	14,1
4c	7398	**7398**	0,00	**7398**	0,00	2	6,0	**7398**	0,00	2	10,4	**7398**	0,00	2	18,3
5c	7362	7336	0,35	7336	0,35	13	0,3	**7342**	0,27	35	12,8	**7342**	0,27	41	22,1
6c	5824	**5805**	0,33	**5805**	0,33	6	6,5	**5805**	0,33	11	11,2	**5805**	0,33	12	18,6
7c	7225	7147	1,08	**7211**	0,19	6	8,2	**7211**	0,19	10	14,4	**7211**	0,19	10	24,9
Average			0,29		0,16	7	5		0,15	12	10		0,15	12	17
1d	6333	6077	4,04	6241	1,45	10	10,0	**6248**	1,34	27	15,8	**6248**	1,34	23	29,2
2d	6579	6240	5,15	**6286**	4,45	24	10,6	6240	5,15	11	15,2	6281	4,53	9	28,1
3d	9261	9158	1,11	**9247**	0,15	24	13,5	**9247**	0,15	17	22,9	**9247**	0,15	20	44,6
4d	10727	**10684**	0,40	**10684**	0,40	18	15,3	**10684**	0,40	13	25,2	**10684**	0,40	16	47,4
5d	11626	11162	3,99	11162	3,99	6	14,7	11390	2,03	18	26,5	**11511**	0,99	18	48,9
6d	14207	**14157**	0,35	**14157**	0,35	16	19,7	**14157**	0,35	23	35,2	**14157**	0,35	24	65,0
7d	14476	13969	3,50	13969	3,50	16	20,1	**14188**	1,99	10	35,6	13969	3,50	11	65,6
8d	16352	16308	0,27	**16352**	0,00	25	20,8	**16352**	0,00	30	37,8	**16352**	0,00	46	65,6
9d	15656	**15557**	0,63	**15557**	0,63	9	18,3	**15557**	0,63	9	31,2	**15557**	0,63	12	56,4
10d	19102	18391	3,72	**18821**	1,47	35	25,1	18772	1,73	30	40,1	18391	3,72	26	73,0
Average			2,32		1,64	18	17		1,38	19	29		1,56	21	52
1e	16464	16438	0,16	**16464**	0,00	29	46	**16464**	0,00	31	75	**16464**	0,00	42	144
2e	23395	22880	2,20	**23127**	1,15	28	69	23020	1,60	33	117	**23127**	1,15	51	209
3e	25243	23930	5,20	**23986**	4,98	32	73	23939	5,17	39	132	23930	5,20	31	241
4e	35594	**35594**	0,00	**35594**	0,00	20	111	**35594**	0,00	20	192	**35594**	0,00	49	429
5e	35154	34761	1,12	**34816**	0,96	29	102	**34816**	0,96	31	206	**34816**	0,96	26	347
Average			1,74		1,42	28	80		1,55	31	144		1,46	40	274
Average			8,09		0,66	13	16,37		0,63	13	29,16		0,65	15	54,51

(1) - number of iterations needed to find the solution

Table 5: Summary of table 4 results.

Probl.		GA (40 / 35)		GA (60 / 40)		GA (80 / 60)	
		gap	time	*gap*	time	*gap*	time
"a"	Best	0	1	0	2	0	4
	Average	**0,22**	**3**	**0,31**	**5**	**0,24**	**10**
	Worst	1,27	7	1,22	12	1,22	24
"b"	Best	0	0	0	1	0	1
	Average	**0**	**3**	**0**	**5**	**0**	**9**
	Worst	0	8	0	14	0	25
"c"	Best	0	0	0	4	0	7
	Average	**0,16**	**5**	**0,15**	**10**	**0,15**	**17**
	Worst	0,35	8	0,33	14	0,33	25
"d"	Best	0	10	0	15	0	28
	Average	**1,64**	**15**	**1,38**	**29**	**1,56**	**52**
	Worst	4,45	25	5,15	40	4,53	73
"e"	Best	0	46	0	75	0	144
	Average	**1,42**	**80**	**1,55**	**144**	**1,46**	**274**
	Worst	4,98	111	5,17	206	5,2	429

were used. The worst case took around 7 minutes and refers to problem "4e" (200 variables and 8117 non zero elements).

In some of the problems, inserting an individual corresponding to the heuristic solution does not affect the results as its insertion is done after obtaining the best solution. But, in the others, when the best solution is not yet found, it brings some diversification to the population and the algorithm, generally, improves the heuristic solution.

Analysing columns "(1)" we see that our solution is obtained, generally, in few iterations, so with a smaller number of iterations the solution quality can be maintained, reducing the computational execution time.

We summarise in table 5 the results and show in figures 1 and 2 the *gap* and execution time averages.

We verify that, in average, the best results were obtained with the smallest population/iterations, except for set problem "d" where a population equal to 60 and 40 iterations obtained the best average results. As, in average, using an 80 individuals population and 60 iterations did not bring any improvement, we conclude that, at least, for these problem sets it is more appropriate to use populations of 40 or 60 individuals.

Figure 2 shows that the average execution times for sets "a", "b" and "c" are less than 1

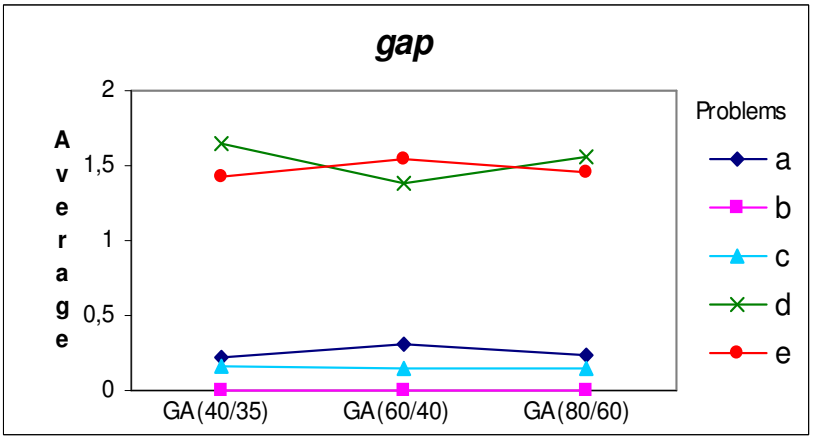

Figure 1: Average *gap* results.

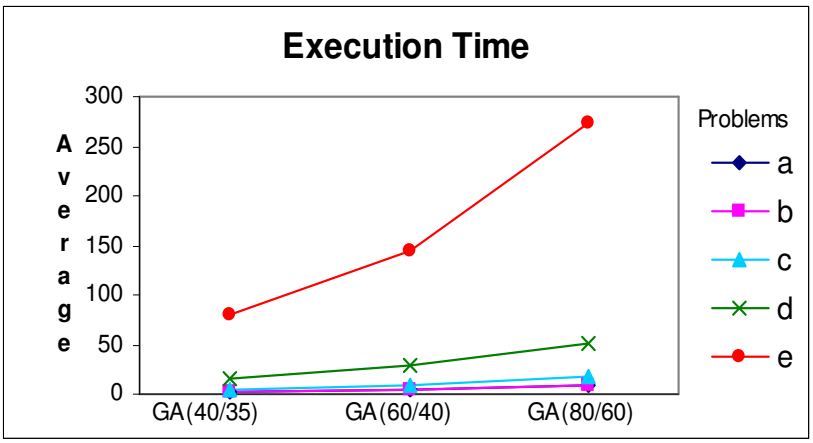

Figure 2: Average execution times in seconds.

minute. For set "d" the results were not so good but in average they are less than 5 minutes, which is a quite reasonable execution time for problems of these dimensions.

4 Conclusions

Our combined genetic and greedy approach efficiently found good or optimal solutions to the QP 0-1. Moreover, the solutions were obtained very quickly even for large problems (200 variables) and large genetic population dimensions.

In this approach we combine genetic ordinary operators, like single point crossover and stochastic universal sampling selection, with a fast greedy procedure replacing the usual mutation. As in previous computational tests with a similar algorithm with the usual mutation operator we had achieved worse results (an average gap that was almost 4 times the average gap obtained now) we can conclude that this greedy procedure contributes, meaningfully, for a better quality solution and to obtain the best solution earlier. So it also contributes to speedup the algorithm.

We suppose that the described algorithm approach can be improved using more efficient heuristic based procedures as genetic operators. The introduction of heuristic individuals, in the population, a few more times should also improve the results.

5 References

[1] Alidaee, B., G. Kochenberger and A. Ahmadian (1994), "0-1 quadratic programming approach for the optimal solution of two scheduling problems", *Int. J. Systems Sci.*, 25, pp. 401-408.

[2] Gallo, G.; P. L. Hammer; B. Simeone (1980), "Quadratic knapsack problems", *Math. Prog.*, 12, pp. 132-149.

[3] Garey M. R.; S. Johnson (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, S. Francisco.

[4] Glover, F.: G. A. Kochenberger; B. Alidaee (1998), "Adaptative memory tabu search for binary quadratic programs", *Manag. Sci.*, 44, pp. 336-345.

[5] Glover, F.: G. A. Kochenberger; B. Alidaee; M. Amini (1999), "Tabu search with critical event memory: an enhanced application for binary quadratic programs" in Meta-Heuristics Advances and Trends in Local Search Paradigms for Optimization, S. Voss, S. Martello, I. H. Osman and C. Roucairol (eds).

[6] Gulati, V. P., Gupta, S.K., Mittal, A.K. (1984), "Unconstrained quadratic bivalent programming problem", European Journal of Operational Research, 15, pp.121-125.

[7] Krarup, J.; P. A. Pruzan (1978), "Computer aided layout design", *Math. Prog. Study*, 9, pp. 75-94.

[8] Lodi, A., Allemand, K., Liebling, T.M. (1999), "An evolutionary heuristic for quadratic 0-1 programming", European Journal of Operational Research, 119, pp. 662-670.

[9] McBride, R. D.; J. S. Yomark (1980), "An implicit enumeration algorithm for quadratic integer programming", *Manag. Sci.*, 26, pp. 282-296.

[10] Pardalos, P. , G. Rodgers (1990), "Computational aspects of a branch and bound algorithm for quadratic 0-1 programming", *Computing*, 45, pp. 131-144.

[11] Pardalos, P., G. Rodgers (1992), "A branch and bound algorithm for the maximum clique problem", *Comp. and Ops. Res.*, 19, pp. 363-375.