The Compact Memetic Algorithm

Peter Merz
University of Tübingen
Department of Computer Science - WSI-RA
Sand 1, D-72076 Tübingen, Germany
peter.merz@ieee.org

Abstract

Optimization by probabilistic modeling is a growing research field in evolutionary computation. An example is the compact genetic algorithm (cGA), in which the population of a genetic algorithm (GA) is represented as a probability distribution over the set of solutions. Both cGA algorithm and the order-one behavior of a simple GA with uniform crossover are operationally equivalent. The cGA is much easier to implement and requires less memory.

In this paper, memetic algorithms (MAs) are investigated in which the population is replaced by a probability vector analogously to the cGA. The resulting compact memetic algorithms (cMAs) hence require less memory, are easier to implement and require fewer parameters than other MAs. It is shown that cMAs with and without additional recombination perform comparable to or better than population-based MAs on a set of benchmark instances of the unconstrained binary quadratic programming problem.

1 Introduction

Recently, several optimization techniques have been proposed in which probability distribution is utilized to represent the state of the search. Examples of such approaches are *Bit-Simulated Crossover* (BSC) [1], *Population-based Incremental Learning* (PBIL) [2, 3], the *Ant System* (AS) [6], and the *Compact Genetic Algorithm* [7]. In these algorithms, a probabilistic model is used to generate solutions for a given optimization, which are, in turn, used to update the model. Thus, these algorithms repeatedly generate solutions with the model and update the model based on the solutions generated. They differ mainly in the learning rules that are applied to update the model. In this work, we concentrate on the learning rules used in the cGA. Other learning rules can be utilized with slight modifications of the proposed algorithms. Since the cGA as well as several others of the algorithms above have been proposed for a binary representation, we focus our studies on the binary quadratic programming problem (BQP). In the BQP, a symmetric $n \times n$ matrix $Q = (q_{ij})$ is given, and a binary vector x of length n is desired, which maximizes the objective function

$$f(x) = x^t\, Q\, x = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}\, x_i\, x_j, \quad x_i \in \{0,1\}\ \forall\, i \quad (1)$$

This problem is also known as the *(unconstrained) quadratic bivalent programming problem*, the *(unconstrained) quadratic zero–one programming problem*, or the *(unconstrained) quadratic (pseudo-) Boolean programming problem* [8]. Since it belongs to the class of $\mathcal{NP}$-hard problems, effective heuristics are required for solving large problem instances.

In this paper, compact memetic algorithms (cMAs) are proposed which utilize a probability distribution instead of a population of solutions analogously to the compact genetic algorithm (cGA). In experiments it is shown that for hard instances of the BQP, the cMAs perform better or comparable to previously proposed population-based MAs.

The paper is organized as follows. In section 2, the compact memetic algorithm for binary representations is introduced. Population-based MAs and new cMAs for the BQP are discussed in section 3. In section 4, results from various experiments with the cMAs are presented. Section 5 concludes the paper and outlines areas of future work.

2 The Compact Memetic Algorithm

In the compact memetic algorithms for binary-coded problems, a single probability vector $p \in P = I\!R^l$ is maintained which represents the current state of the search and replaces the population in population-based MAs. For each of the l bits in a solution vector $x \in X = \{0,1\}^l$, there is a real value in the probability vector p indicating the probability for assigning a 1 to that bit. For example, if $p[i] = 0.9$, the probability of assigning 1 to $x[i]$ is 0.9, and the probability of assigning 0 to $x[i]$ is $1.0 - p[i] = 0.1$. Hence, a new vector x is sampled from the probability model according to the formula

$$x[i] = \begin{cases} 1 & \text{if random}[0,1) < p[i] \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where random[0,1) represents a uniformly distributed random number between 0 and 1 exclusively. The pseudo–code is simply the following few lines:

```
function generate(p : P) : X

  begin
    for i:=1 to l do
      if random < p[i] then x[i] := 1.0
      else x[i] : = 0.0;
    return x;
  end;
```

Initially, all $p[i]$ are set to 0.5. After two new solutions x, x' have been generated and a local search has been applied, the model is updated with the following method:

```
procedure update(p : P, x' : X, x : X, n : N)

  begin
    for i:=1 to l do
      if x'[i] <> x[i] then
         if x'[i] = 1 then p[i] := p[i] + 1/n
                      else p[i] := p[i] - 1/n;
    for i:=1 to l do
      if p[i] > 1.0 then p[i] := 1.0
      else if p[i] < 0.0 then p[i] := 0.0;
  end;
```

In this method, x' represents the solution with higher fitness compared to x. In contrast to the cGA, in which simply two solutions are generated from the model, we considered four variants of cMAs, which differ in the generation of new solutions. The pseudo-code of the cMA is shown in Fig. 1. These variants are considered in our studies since they allow for the investigation of the influence of the selection pressure as well as the influence of sophisticated recombination operators usually employed in MAs.

```
function cMA(psize, gens, rec, ub) : X

  begin
    for i = 1 to l do p[i] := 0.5;
    x := generate(p);
    x := localSearch(x);
    x* := x;
    i := 1;
    repeat
      x: = generate(p);
      x: = localSearch(x);
      if rec then x' = recombine(x*, x);
      else x' := generate(p);
      x' := localSearch(x');
      if f(x') < f(x) then swap(x, x');
      if f(x') > f(x*) then x* = x';
      if ub then update(p, x*, x, psize)
      else update(p, x', x, psize);
      i := i + 1;
    until (i > gens);
    return x*;
  end
```

Figure 1: The Compact Memetic Algorithm

In the variant analogously to the cGA (rec = false, ub = false), two solutions x and x' are generated from the probability distribution and a local search is applied to the solutions. Afterwards, the model is updated with these solutions.

If rec=true, the second solution x' is generated by recombination of the best solution found so far $x*$ and the first solution x.

If ub=true, the model is updated with the best solution found so far $x*$ and the worst of the two generated solutions x and x'.

Before p is updated, $x*$ may be updated and x' and x may be swapped, to ensure x' has higher fitness than x. The parameters of the algorithm are the number of generations (gens) and the size of the simulated population psize in respect to the equivalent GA as well as the parameters rec and ub to specify the cMA variants.

Since only the probability vector has to be stored in memory and not a population of solutions, cMAs are especially well–suited for solving very large combinato-

```
function MA(psize, gens, recs) : X

  begin
    for i = 1 to psize do
       P[i] := Initialize();
    for i = 1 to psize do
       P[i] := localSearch(P[i]);
    sort(P);
    x* := P[1];
    j := 1;
    repeat
      for i = 1 to recs do begin
        x := select(P);
        x' := select(P);
        x := recombine(x, x');
        x := localSearch(x);
        append(P, x);
      end;
      sort(P);
      reduce(P, psize);
      if f(P[1]) > f(x*) then x* = P[1];
      j := j + 1;
    until (j > gens);
    return x*;
  end
```

Figure 2: The (population-based) Memetic Algorithm

rial optimization problems such as graph partitioning problems with more than 1.000.000 nodes or traveling salesman problems with more than 1.000.000 cities. Furthermore, cMAs appear to be well–suited in multi–agent frameworks where each agent represents a cMA algorithm. Several agent may be executed on a single machine due to the low memory profile.

3 Memetic Algorithms for the BQP

The population-based MA as well as the compact MAs for the BQP used in our studies are described in the following.

3.1 Population-based MA for the BQP

Recently, MAs have been proposed for the BQP that are highly effective [9]. A slightly modified algorithm is considered in this paper as described in the following. The MA is population-based and utilizes a simple panmictic population. The outline of the MA is provided in Fig. 2. The selection for reproduction is performed randomly without bias towards fitter individuals. The selection for survival is essentially as in the $(\mu+\lambda)$-ES: in each generation, recs offspring are appended to the population. The psize+recs individuals are sorted and the population is reduced to its original size.

The problem specific components of the MA are:

Representation and Fitness Function:

The solutions are represented simply as binary vectors. No search parameters for self-adaptation are part of the individual. The fitness function utilized in the MA is the objective function shown in equation (1).

Initialization and Local Search:

The population is initialized by randomly generating solutions and applying local search. The local search used in the MA is the k-opt local search proposed in [10].

Variation Operators:

The recombination operator used in the MA is the innovative variation operator proposed in [9]. The MA considered here does not utilize additional mutation operators.

3.2 Compact MAs for the BQP

The cMAs proposed in this paper use the same representation, fitness function, local search and recombination operator as the population-based MA described above. This allows us to concentrate in our studies on the influence of the different population models on the overall performance.

4 Experimental Results

In order to compare the performance of the cMA with the (population-based) MA, we conducted several experiments. To enable a fair comparison, the number of local searches was kept constantly to 2000. This was achieved by setting psize=40,recs=20,gens=99 for the MA and gens=1000 for the cMAs. The set of BQP instances used in the experiments is denoted by kb-g and consists of 10 instances of size $n = 1000$. The matrix Q densities of the instances in the problem set are between 0.1 and 1.0 [11]. The results are displayed in Table 1. In the table, the percentage excess below the best-known solution (excess) and the success rate of finding the best-known solution (success) is provided for each instance. Furthermore, the averaged success rate over all ten problem instances is shown in the last row of the table. Compared to the cMA with parameters psize=1000,rec=true,ub=false (denoted cMA (rec)), the MA appears to be slightly inferior in terms of robustness: the average success rate is slightly

Table 1: Comparison of an MA four cMA variants

	MA		cMA		cMA (ub)		cMA (rec)		cMA (rec,ub)	
Instance	excess	success	excess	success	excess	success	excess	success	excess	success
kb-g01	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %
kb-g02	0.00 %	96.7 %	0.00 %	86.7 %	0.01 %	53.3 %	0.00 %	66.7 %	0.01 %	50.0 %
kb-g03	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %
kb-g04	0.00 %	100.0 %	0.04 %	53.3 %	0.01 %	96.7 %	0.01 %	96.7 %	0.02 %	93.3 %
kb-g05	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %
kb-g06	0.02 %	56.7 %	0.00 %	96.7 %	0.02 %	56.7 %	0.02 %	56.7 %	0.02 %	63.3 %
kb-g07	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %
kb-g08	0.00 %	96.7 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %
kb-g09	0.06 %	30.0 %	0.04 %	10.0 %	0.06 %	46.7 %	0.03 %	60.0 %	0.06 %	43.3 %
kb-g10	0.03 %	30.0 %	0.01 %	80.0 %	0.05 %	16.7 %	0.02 %	53.3 %	0.04 %	33.3 %
all		81.0 %		82.7 %		77.0 %		83.33 %		78.33 %

higher. In both cases, the number of instances always solved to (near-)optimality is five.

To evaluate the performance of the other cMA variants, we conducted several experiments which are also summarized in Table 1. Again, percentage excess (excess) and success rate (success) are provided. In the table, the cMA variant is specified in parentheses in the headline: rec denotes rec=true, and ub denotes ub=true. Clearly, the updating the model with the best solution is not as a good strategy as updating with the best of the two newly generated solutions. Furthermore, using a sophisticated recombination operator as the innovate variation operator improves the standard cMA only slightly. The standard cMA without update with the best and recombination is superior to the MA with population. Therefore, in the case of the BQP instances studied in this work, a sophisticated recombination operator and a population-based MA, are not required to arrive at high quality solutions with high probability.

To investigate the influence the remaining parameter of the cMA – the psize parameter – we performed another experiment on the kb-g10 instance. The results are summarized in Table 2. The experiments have shown that the influence of the parameter on the overall performance is relatively small and a value of 1000 appears to be superior to the other values tested in the experiments.

In a final experiment, we were interested in the question, whether the results are specific to the local search used. Therefore, we tested the same algorithms with 1-opt local search [10]. The results are displayed in Table 3. For each algorithm, the percentage excess (ex.) and the success rate (suc.) in percent are displayed. Here, the cMA variants performing the update with

Table 2: Influence of different population sizes in cMA

		cMA (rec,ub)		
Instance	psize	objective	excess	success
kb-g10	10	274075.3	0.11 %	13.3 %
kb-g10	40	274219.6	0.06 %	23.3 %
kb-g10	100	274185.9	0.07 %	23.3 %
kb-g10	1000	274264.6	0.04 %	33.3 %
kb-g10	10000	274272.1	0.04 %	23.3 %

the best solution found so far, perform better. Moreover, the use of recombination increases the success rate significantly. The MA performs slightly better than the cMA with recombination and 'best update'.

Summarizing, the results show that 'best update' leads to a premature convergence in the case of a strong local search, but can increase the selection pressure for a weak local search. The sophisticated recombination scheme has the higher importance the weaker the local search.

5 Conclusions

New memetic algorithms have been presented that simulate the behavior of previously proposed memetic algorithms without explicitly maintaining a population of solutions. These compact memetic algorithms (cMAs) are easier to implement, require less memory, and in their simplest form require only a single parameter. Focusing on binary coded problems, it is shown for hard instances of the binary quadratic programming problem (BQP) that the simple cMA outperforms the population-based MA if a k-opt lo-

Table 3: Comparison of MA and four cMA variants with 1-opt local search

	cMA		cMA (ub)		cMA (rec)		cMA (rec,ub)		MA	
Instance	ex.	suc.	ex.	suc.	ex.	suc.	ex.	suc.	ex.	suc.
kb-g01	0.07 %	0.0 %	0.05 %	23.3 %	0.06 %	13.3 %	0.04 %	20.0 %	0.03 %	16.7 %
kb-g02	0.16 %	0.0 %	0.06 %	10.0 %	0.18 %	0.0 %	0.05 %	30.0 %	0.10 %	3.3 %
kb-g03	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %	0.00 %	100.0 %
kb-g04	0.30 %	0.0 %	0.24 %	10.0 %	0.28 %	13.3 %	0.15 %	40.0 %	0.09 %	63.3 %
kb-g05	0.01 %	83.3 %	0.05 %	76.7 %	0.02 %	90.0 %	0.07 %	73.3 %	0.03 %	86.7 %
kb-g06	0.16 %	3.3 %	0.14 %	10.0 %	0.29 %	6.7 %	0.14 %	20.0 %	0.22 %	0.0 %
kb-g07	0.35 %	0.0 %	0.16 %	36.7 %	0.40 %	0.0 %	0.18 %	16.7 %	0.06 %	70.0 %
kb-g08	0.08 %	6.7 %	0.07 %	33.3 %	0.14 %	20.0 %	0.09 %	30.0 %	0.06 %	36.7 %
kb-g09	0.48 %	0.0 %	0.27 %	0.0 %	0.54 %	0.0 %	0.25 %	10.0 %	0.53 %	0.0 %
kb-g10	0.19 %	0.0 %	0.14 %	13.3 %	0.19 %	3.3 %	0.09 %	3.3 %	0.24 %	3.3 %
all		19.3 %		31.3 %		24.7 %		34.3 %		38.0 %

cal search is used. For MAs utilizing the weaker 1-opt local search, the simple cMA can be enhanced by using a sophisticated recombination operator and a slightly modified update rule. The results demonstrate that maintaining a population of solutions can increase the performance of the MA for the BQP only slightly.

Therefore, the cMA appears to be an alternative for the standard MA or even more complex MAs with spatial or tree structured population structures. If a strong local search such as k-opt local search is used, the cMA may be at least as good as these MAs. Moreover, in applications in which memory requirements or the 'time to market' are a major concern, the cMA is clearly the better choice.

There are several areas for future research. Firstly, increasing the selection pressure by creating more than two solutions per generation may increase the performance of the cMA. Secondly, time-varying/adaptive learning rates may also increase the performance of the cMA. Additional experiments are needed to verify this. Problems which cannot be binary-coded efficiently, are an issue for future work. Finally, the compact representation of the population allows the evolution of probability vectors in a multiple cMA agent framework.

References

[1] Syswerda, G.: Simulated Crossover in Genetic Algorithms. In Whitley, D., ed.: Proceedings of the Second Workshop on Foundations of Genetic Algorithms (FOGA II), Morgan Kauffmann, San Mateo (1993) 239–255

[2] Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CS-94-163, Carnegie Mellon University, School of Computer Science (1994)

[3] Baluja, S., Caruana, R.: Removing the Genetics from the Standard Genetic Algorithm. In Prieditis, A., Russel, S., eds.: Proceedings of the International Conference on Machine Learning, 1995, San Mateo, CA, Morgan Kaufmann Publishers (1995) 38–46

[4] Baluja, S., Davies, S.: Fast Probabilistic Modeling for Combinatorial Optimization. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98), Menlo Park, AAAI Press (1998) 469–476

[5] Pelikan, M., Goldberg, D.E., Cantu-Paz, E.: BOA: The Bayesian Optimization Algorithm. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99). Volume 1., Morgan Kaufmann (1999) 525–532

[6] Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics - Part B **26** (1996) 29–41

[7] Harik, G.R., Lobo, F.G., Goldberg, D.E.: The Compact Genetic Algorithm. IEEE Transactions on Evolutionary Computation **3** (1999) 287–297

[8] Beasley, J.E.: Heuristic Algorithms for the Unconstrained Binary Quadratic Programming

Problem. Technical report, Management School, Imperial College, London, UK (1998)

[9] Merz, P., Katayama, K.: Memetic Algorithms for the Unconstrained Binary Quadratic Programming Problem. Bio Systems (2002) To appear.

[10] Merz, P., Freisleben, B.: Greedy and Local Search Heuristics for Unconstrained Binary Quadratic Programming. Journal of Heuristics **8** (2002) 197–213

[11] Amini, M.M., Alidaee, B., Kochenberger, G.A.: A Scatter Search Approach to Unconstrained Quadratic Binary Programs. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, London (1999) 317–329