Genetic and Evolutionary Computation Conference (GECCO'2001)
San Fransisco, July 2001

Priorities in Multi-Objective Optimization for Genetic Programming

Frank Schmiedle **Nicole Drechsler** **Daniel Große** **Rolf Drechsler**
Chair of Computer Architecture (Prof. Bernd Becker)
Institute of Computer Science, University of Freiburg i.Br., Germany
e-mail: {schmiedl,ndrechsl,grosse,drechsle}@informatik.uni-freiburg.de

Abstract

A new technique for multi-objective optimization is presented that allows to include priorities. But in contrast to previous techniques they can be included very easily and do not require much user interaction. The new approach is studied from a theoretical and practical point of view. The main differences to existing methods, like relation dominate and favor, are discussed. An experimental study of applying priorities in heuristics learning based on Genetic Programming (GP) is described. The experiments confirm the advantages presented in comparison to several other techniques.

1 Introduction

When applying optimization techniques, it should be taken into account that many problems in real-world applications depend on several independent components. This is one of the reasons why several approaches to multi-objective optimization have been proposed in the past (see e.g. [13]). They mainly differ in the way the elements are compared and in the granularity of the ranking. One major drawback of most of these methods is that a lot of user interaction is required. (For a more detailed description of the different models and a discussion of the main advantages and disadvantages see Section 3).

With applications becoming more and more complex, the user often does not have enough information and insight to guide the tool. In [5], a new relation has been introduced that allows to rank elements with a finer granularity than [8], keeping the main advantages of the model. Experimental studies have shown that this model is superior to the "classical" approach of relation dominate [9]. Even though, originally developed for *Evolutionary Algorithms* (EAs), recently it has also been applied in the field of *Genetic Programming* (GP) [10]. One of the major drawbacks of the model of [5] is that the handling of priorities is not covered.

In this paper we present an extension of [5] that allows to work with priorities and at the same time keeps all the advantages of the original model. Experimental results in the field of GP-based heuristics learning for minimization of *Binary Decision Diagrams (BDDs)* [2] show that the approach obtains the best result in comparison to previously published methods.

In the next section we first briefly review the application of GP-based heuristics learning. Multi-objective optimization is discussed in detail in Section 3, where we put special emphasis on handling of priorities. In Section 4 the experimental results are described and discussed. Finally, the paper is summarized.

2 Basic Concepts

We assume that the reader is familiar with *GA* and *GP* concepts and refer to [4, 10] for details. A model for heuristics learning with *GAs* has been proposed in [7]. Known basic problem solving strategies are used as heuristics, and ordering and frequency of the strategies is optimized by evolution. Recently, a generalization to the *GP* domain has been reported [6]. The multi-objective optimization approach that is presented in this paper has been used during heuristics learning for BDD minimization and first experiments were given. Therefore, we give a brief review of *GP*-based heuristics learning and the resulting BDD minimization method to make the paper self-contained.

2.1 Heuristics Learning

For learning heuristics in order to find good solutions to an optimization problem, it is necessary that several (non-optimal) strategies solving the problem can be provided. Typically, for different classes of problem instances there are also different strategies that perform best. A strategy that behaves well on one problem class may return poor results when being applied to another problem class. Thus, it is promising to learn how to combine the strategies to heuristics that can be applied successfully to most or even all classes of problems.

The learning process by *GP* and for a better understanding, some fundamental terms are introduced by the following

Definition 1 *Given an optimization problem P and a non-empty set of different non-optimal strategies $B = \{b_1, \ldots, b_{max}\}$ to solve the problem. Then the elements of B are called* BOMs (Basic Optimization Modules)*.*

Moreover, a heuristics *for P is an algorithm that generates a sequence of BOMs.*

During evolution, the strategies are combined to generate heuristics that are the individuals in the population. The fitness of an individual can be evaluated by application of the heuristics to a training set of problem instances. The target is to find heuristics that perform well according to some given optimization criteria. Additionally, a good generalization is important, i.e. a heuristics that returns good results for the training set examples should also performs well on unknown instances. Note that for this, the handling of the different criteria, i.e. the special multi-objective optimization approach, plays a critical role for the success of the evolutionary process.

2.2 BDD Minimization by Genetic Programming

Binary Decision Diagrams (BDDs) [2] are a state-of-the-art data structure often used in VLSI CAD for efficient representation and manipulation of Boolean functions. BDDs suffer from their size being strongly dependent on the variable ordering used. In fact, BDD sizes may vary from linear to exponential for different orderings. Optimization of variable orderings for BDDs is difficult, but nevertheless, successful strategies for BDD minimization that are based on dynamic variable ordering have been reported, see e.g. *sifting* [11].

For heuristics learning, the strategies sifting (SIFT), group sifting (GROUP), symmetric sifting (SYMM), window permutation of size 3 and 4, respectively, (WIN3, WIN4) are used as BOMs. For all these techniques there is an additional BOM that iterates the method until convergence is reached, and the "empty" operator NOOP completes the set of BOMs B.

The individuals of the *GP* approach for BDD minimization consist of trees with leaf nodes labeled with BOMs and inner operator nodes that belong to different types. During evaluation of the heuristics, the tree is traversed by a *depth-first-search*-based method in order to generate a flat sequence. The types of the inner nodes decide if

- both subtrees are evaluated subsequently (CONCAT),

- according to the truth value of a given condition either the left or the right son is considered (IF) or
- evaluation of the sons is iterated until a truncation condition is fulfilled (WHILE).

For recombination, two crossover operators are provided. While CAT concatenates the two parents, the more sophisticated MERGE does the same for subtrees of the parents and by that, bloating can be prevented. In addition to this, there are four mutation operators that exchange BOMs (BMUT), node types (CIMUT, CWMUT) and modify conditions of IF-nodes (IFMUT), respectively. A probability table determines the frequencies for using the different operators. (For more details see [6].)

3 Multi-Objective Optimization with Priorities

In this section, the multi-objective aspect for solving optimization problems is analyzed. Without loss of generality we consider only *minimization* problems.

For n optimization criteria, an objective vector $(c_1, \ldots, c_n) \in \mathbb{R}_+^n$ of values for these criteria completely characterizes a solution belonging to the search space Π. Thus, solutions can be identified with objective vectors and as a consequence, $\Pi \subset \mathbb{R}_+^n$.

In most cases some or all of the c_i's are mutually dependent, and often conflicts occur, i.e. an improvement in one objective c_i leads to a deterioration of c_j for some $j \neq i$. This must be taken into account during the optimization process. If priorities have to be considered, a good handling of multi-objective optimization becomes even more complex.

3.1 Previous Work

In the past, several techniques for ranking solutions according to multiple optimization criteria have been developed. Some approaches define a *fitness* function $f : \mathbb{R}_+^n \mapsto \mathbb{R}_+$ that maps solutions c to one scalar value $f(c)$. The most commonly used method is linear combination by WEIGHTED SUM*.

Values for the c_i's $(1 \leqslant i \leqslant n)$ are weighted by constant coefficients W_i, and $f(c)$ is given by

$$f(c) = \sum_{i=1}^{n} W_i \cdot c_i.$$

The fitness value is used for comparison with the fitness of other solutions. Obviously, criteria with large weights have more influence on the fitness than those with small coefficients.

There are other methods that compare solutions based on one of the relations which are introduced by

Definition 2 *Let $c = (c_1, \ldots, c_n)$ and $d = (d_1, \ldots, d_n) \in \Pi$ be two solutions. The relations $\prec_d$* (dominate) *and $\prec_f$* (favor) *$\subset \Pi \times \Pi$ are defined by*

$$\begin{aligned} c \prec_d d \quad &:\Leftrightarrow \quad \exists i : c_i < d_i \;\wedge \\ &\qquad\;\; \forall i : c_i \leqslant d_i \quad (1 \leqslant i \leqslant n) \\ c \prec_f d \quad &:\Leftrightarrow \quad |\{c_i < d_i | 1 \leqslant i \leqslant n\}| > \\ &\qquad\;\; |\{c_i > d_i | 1 \leqslant i \leqslant n\}| \end{aligned}$$

We say that c dominates *d if $c \prec_d d$ and $c \prec_f d$ means that c is* favored *to d.*

$\prec_d$ is a partial ordering on any solution set $S \subset \Pi$ and the set $P \subset S$ that contains all non-dominated solutions in S is called *pareto-set*. In [9], the DOMINATE approach that approximates pareto-sets has been proposed.

An interactive technique for multi-objective optimization that divides Π into three subsets containing solutions of different *satisfiability classes* has been reported in [8]. It was generalized to the use of a variable number of satisfiability classes in [5]. The classes can be represented by the strongly connected components in the relation graph of $\prec_f$ and hence they can be computed by known graph algorithms. By this, it becomes possible to classify

*This is also the name of the method.

solutions $c \in \Pi$. We refer to this technique (introduced in [5]) as PREFERRED in the following. If priorities have to be handled, lexicographic sorting is used instead of $\prec_f$. This method will be called LEXICOGRAPHIC in further sections.

3.2 Drawbacks of Existing Approaches

The WEIGHTED SUM method is most popular for multi-objective optimization since it is easy to implement and allows to scale objectives. However, there are two major drawbacks:

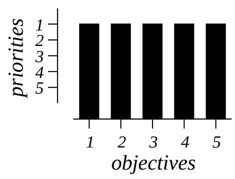
1. Priorities cannot be handled directly but only by huge penalty weights. If there are many different priorities, the fitness function becomes very complex by that.

2. For adjusting the weights, problem specific knowledge is necessary. Usually, good settings for the weights are not known in advance and for finding and tuning them in experiments much effort has to be spent.

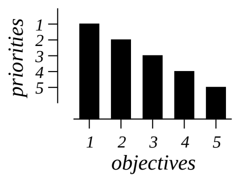
The approach proposed in [8] does not use weights that have to be adjusted, but it is interactive and therefore additional effort by the user is required, too. Moreover, the granularity of the method is very coarse since the solutions are divided in three different classes only. PREFERRED is a generalization of that technique that overcomes this drawback, i.e. an arbitrary number of satisfiability classes can be handled. By that, objectives with nearly the same importance can be optimized in parallel conveniently. However, priorities can not be considered by PREFERRED and in the approach presented in [5], LEXICOGRAPHIC is applied instead of PREFERRED if different priorities occur. By that, the following disadvantages are implied:

1. Instead of the relation $\prec_f$, the less powerful lexicographic sorting is applied for comparison of solutions and hence the results that can be expected are not as good as if $\prec_f$ was used.

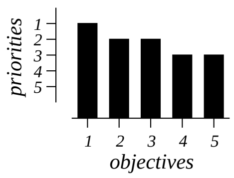
2. Lexicographic sorting does not permit assigning the same priority to more than one optimization criteria. Thus if there are two objectives with a similar impact on the overall quality of solutions, one of them has to be preferred during LEXICOGRAPHIC in comparison to the other one.



(a) PREFERRED



(b) LEXICOGRAPHIC



(c) PRIORITY

Figure 1: Priority schemes for different optimization methods.

Figure 1 (a) and (b) illustrate the priority handling of PREFERRED and LEXICOGRAPHIC, respectively. None of the existing methods can deal with priority schemes like described in Figure 1 (c), where the same priority is assigned to some objectives while some other criteria have lower or higher priorities. In the next section, an approach is presented that fulfills this requirement.

3.3 Multi-Objective Optimization with Priority Handling

The PRIORITY multi-objective optimization method introduced in this section combines properties of PREFERRED and LEXICOGRAPHIC. Thus it is more powerful and arbitrary priority numbers can be assigned to each objective. Without loss of generality we assume that the priorities $1, 2, \ldots, m$ are used in non-descending order for the objectives $1, \ldots, n^{\dagger}$.

Definition 3 *Given an optimization problem with search space $\Pi \subset \mathbb{R}_+^n$ and a priority vector $p = (p_1, \ldots, p_m) \in \mathbb{N}_+^m$ such that p_i determines for how many objectives the priority i occurs. According to this, the priority of an objective can be calculated by the function*

$$pr : \{1, \ldots, n\} \mapsto \{1, \ldots, m\},$$
$$pr(i) = k \quad \text{where } \sum_{j=1}^{k-1} p_j \leqslant i < \sum_{j=1}^{k} p_j$$

The projection *of $c \in \Pi$ on a priority i is given by*

$$c|_i \in \mathbb{R}^{p_i}, \quad c|_i = (c_l, \ldots, c_h)$$
$$\text{where } l = \sum_{j=1}^{i-1} p_j + 1 \ \wedge \ h = \sum_{j=1}^{i} p_j$$

Finally, for $c, d \in \Pi$ the relation $\prec_{pf} \subset \Pi \times \Pi$ (priority favor) *is defined by*

$$c \prec_{pf} d \quad :\Leftrightarrow \ \exists j \in \{1, \ldots, m\} : c|_j \prec_f d|_j \ \wedge$$
$$(\forall k < j : c|_k \not\prec_f d|_k \wedge d|_k \not\prec_f c|_k)$$

"c is priority-favored to d" also means $c \prec_{pf} d$.

The priority favor relation is used to compare solutions, but a complete ranking cannot be generated by $\prec_{pf}$ as can be seen in the following

Example 1 *For a problem with $n = 4$ optimization objectives and $m = 2$ different priorities, the search space and the priority vector are given as follows:*

$$\begin{aligned} \Pi_{ex} &= \{(2,8,8,8), (5,6,0,8), (5,7,5,1), \\ &\quad (2,6,0,8), (5,2,7,5), (2,7,8,9)\}, \\ p &= (1,3) \end{aligned}$$

$\dagger$Otherwise the objectives have to be re-ordered.

The relation graph for $\prec_{pf}$ is illustrated in Figure 2. There are three solutions with value 2 and as well three solutions with value 5 for the objective with priority 1. Obviously the solutions with the lower value are priority-favoured to the other ones due to the value of objective 1 and regardless of the values of the other objectives. Among these priority-favored solutions, (2,6,0,8) is priority-favored to the others:

$$(6,0,8) \prec_f (8,8,8) \Rightarrow (2,6,0,8) \prec_{pf} (2,8,8,8)$$
$$(6,0,8) \prec_f (7,8,9) \Rightarrow (2,6,0,8) \prec_{pf} (2,7,8,9)$$

(2,8,8,8) and (2,7,8,9) can not be compared by $\prec_{pf}$ and for the remaining solutions, ranking is not possible since the graph for $\prec_{pf}$ contains a cycle.

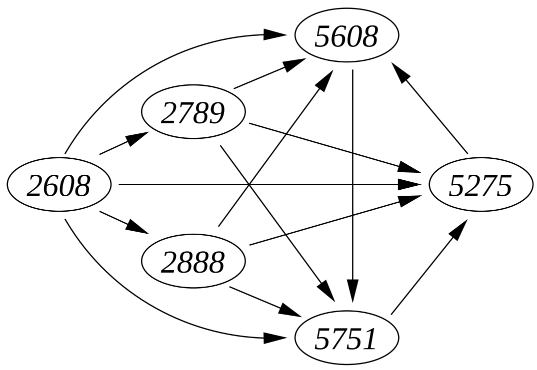


Figure 2: Relation graph $G = (\Pi_{ex}, \prec_{pf})$

The reason why cycles can occur is —as can easily be seen— that $\prec_{pf}$ is not transitive. This is not surprising since $\prec_{pf}$ is based on $\prec_f$ that is not transitive either [5]. To overcome this problem, analogously to the PREFERRED approach, solutions that belong to a cycle in the relation graph $G = (\Pi, \prec_{pf})$ are considered to be equal and merged to one single meta-node. This is done by generation of a meta-graph $G_{m,\Pi}$ by a linear time graph algorithm [3] that finds the set of strongly connected components SCC in G. We have

$$G_{m,\Pi} = (SCC, E) \text{ where}$$
$$E = \{(q_1, q_2) \in \prec_{pf} | SCC(q_1) \neq SCC(q_2)\}$$

Since $G_{m,\Pi}$ by construction is free of cycles, there has to be at least one root node with indegree 0 and by that, it is possible to rank the set of solutions according to

Definition 4 *Given a set of solutions* $\Pi \subset \mathbb{R}^n_+$*, the relation graph* $G = (\Pi, \prec_{pf})$*, the meta-graph* $G_{m,\Pi} = (SCC, E)$ *and the set of its root nodes* $G_0 = \{q | indeg(q) = 0\}$.

Then the satisfiability class *or* fitness $f(c)$ *of a solution* $c \in \Pi$ *can be determined by*

$$\begin{aligned} & f : \Pi \mapsto \mathbb{N}_+, \\ & f(c) = max\{r | \exists (q_1, \ldots, q_r) \in SCC^r : \\ & \qquad q_1 \in G_0 \ \wedge \ c \in q_r \ \wedge \\ & \qquad \forall 1 \leqslant i < r : (q_i, q_{i+1}) \in E\} \end{aligned}$$

The solutions $c \in \Pi$ can now be ranked according to their fitness that by Definition 4 is the increment of the length of the longest path in $G_{t,\Pi}$ from a root node to c . For computation of the ranking, well-known graph algorithms are used.

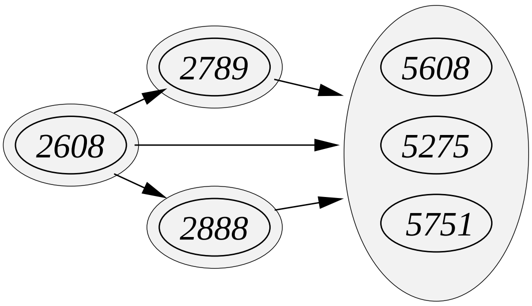


Figure 3: Meta-graph $G_{m,\Pi_{ex}}$

Example 2 *Consider again* Π_{ex} *from Example 1. Figure 3 includes the meta-graph* $G_{m,\Pi_{ex}}$ *with* $G_0 = \{(2, 6, 0, 8)\}$ *with nodes representing SCCs in G. The fitness values for the solutions can easily be derived from* $G_{m,\Pi_{ex}}$*, e.g.* $f((2, 8, 8, 8)) = 2$ *and* $f(5, 6, 0, 8) = 3$.

4 Experimental Results

We implemented the PRIORITY multi-objective optimization approach described in Section 3 in the programming language C++ and embedded it in the software for BDD minimization by *GP*-based heuristics learning (see Section 2). In our experiments‡, examples of Boolean functions that are taken from LGSYNTH91 [12] are used. The corresponding BDDs are minimized by WEIGHTED SUM as well as by relation based methods, i.e. the techniques PREFERRED, LEXICOGRAPHIC and PRIORITY. The objectives are the reduced BDD sizes for the single benchmarks. Notice that the discussion of the experiments in our approach can also be seen in the context of *design of experiments* (for more details see [1]).

For setting the weights in WEIGHTED SUM, several different approaches have been tried and we report two of them here. In EQUAL, weights are adjusted according to the initial BDD sizes of the benchmarks in a way that each example has the same impact on the fitness function. The idea behind this method is to favor the generalization ability of the generated heuristics to their optimized performance on the training set. In other words, by using EQUAL intuitively heuristics can be expected that perform better on unknown examples while slightly weaker results on the training set are tolerated.

For the technique REDR, the reduction rates that are obtained when applying the strategy SIFT to the single benchmarks are calculated. Weights are chosen indirectly proportional to the reduction rates, i.e. large weights are assigned to examples for which a large reduction is observed. Here, the intuition is that learning is focussed on benchmarks with a large potential for reduction in order to generate heuristics that exploit this potential well on unknown functions.

It is obvious that for weight setting, much effort has to be spent on experiments and computation of e.g. the reduction rates for the training set. In comparison to this, PREFERRED needs no preprocessing at all while for LEXICOGRAPHIC, only the objectives have

‡All experiments have been carried out on SUN ULTRA 1 workstations.

Table 1: Results for application on the training set

Name of circuit	I/O in	I/O out	WEIGHTED SUM EQUAL	WEIGHTED SUM REDR	RELATION BASED PREF	RELATION BASED LEXIC	RELATION BASED PRIOR
bc0	26	11	522	522	522	522	522
ex7	16	5	71	71	71	71	71
frg1	28	3	80	82	80	80	80
ibm	48	17	206	207	206	206	206
in2	19	10	233	233	233	233	233
s1196	32	32	597	597	597	597	597
ts10	22	16	145	145	145	145	145
x6dn	39	5	239	237	239	239	239
average	–	–	261.6	262.0	261.6	261.6	261.6

to be ordered (in our experiments according to initial BDD sizes). The same is done for PRIORITY — the only difference is that the same priority is assigned to benchmarks with a similar initial BDD size.

For the *GP*, the same settings as in [6] are used. The population consists of 20 individuals and in each generation 10 offsprings are generated. The evolutionary process is terminated after 100 generations. For more details about the experimental setup like e.g. the method for generating the initial population, we refer to [6]. In the final population, one of the individuals with the best fitness value is chosen. The results for minimization of the training set examples are given in Table 1.

In the first three columns, the names and the input and output sizes, respectively, of the benchmarks are given. Columns 4 and 5 include the results for the WEIGHTED SUM methods EQUAL and REDR while in the last three columns, final BDD sizes of the heuristics generated by the methods LEXICOGRAPHIC, PREFERRED and PRIORITY are given. It can be seen that nearly all methods perform identically with respect to the behavior of the best individuals on the training set examples. Only The REDR approach slightly differs for three benchmarks.

The situation changes when the heuristics are applied to unknown benchmarks. The results are given in Table 2.

Except for *chkn* where REDR performs slightly better, PRIORITY achieves the best results for all benchmarks. It clearly outperforms the other relation based methods as well as EQUAL on average while being still slightly better than REDR. As a result, it can be seen that setting weights for a fitness function by intuition is not always successful. Although the ideas for both approaches EQUAL and REDR sound sensible, only the latter achieves good results. Thus many experiments have to be conducted for tuning weights if WEIGHTED SUM is used while this is not needed when applying PRIORITY.

5 Conclusions

A new technique for handling priorities in multi-objective optimization has been presented. Application in GP-based heuristics learning has clearly demonstrated that the new approach outperforms existing methods, while at the same time the user interaction is reduced.

It is focus of current work to further study the relation between GA-based and GP-based heuristics learning using multi-objective optimization techniques.

Table 2: Application to new benchmarks

Name of circuit	WEIGHTED SUM		RELATION BASED		
	EQUAL	REDR	PREF	LEXIC	PRIOR
apex2	601	349	601	601	320
apex7	291	288	291	291	288
bcd	568	573	568	568	568
chkn	266	261	266	264	264
cps	975	975	975	975	970
in7	76	78	76	76	76
pdc	793	792	793	792	792
s1494	386	386	386	386	386
t1	112	112	112	113	112
vg2	79	79	79	79	79
average	414.7	389.3	414.7	414.5	385.5

References

[1] F. Brglez and R. Drechsler. Design of experiments in CAD: Context and new data sets for ISCAS'99. In *Int'l Symp. Circ. and Systems*, pages VI:424–VI:427, 1999.

[2] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[3] T.H. Cormen, C.E. Leierson, and R.C. Rivest. *Introduction to Algorithms.* MIT Press, McGraw-Hill Book Company, 1990.

[4] L. Davis. *Handbook of Genetic Algorithms.* van Nostrand Reinhold, New York, 1991.

[5] N. Drechsler, R. Drechsler, and B. Becker. A new model for multi-objective optimization in evolutionary algorithms, LNCS 1625. In *Int'l Conference on Computational Intelligence (Fuzzy Days)*, pages 108–117, 1999.

[6] N. Drechsler, F. Schmiedle, D. Große, and R. Drechsler. Heuristic learning based on genetic programming. In *EuroGP*, 2001.

[7] R. Drechsler and B. Becker. Learning heuristics by genetic algorithms. In *ASP Design Automation Conf.*, pages 349–352, 1995.

[8] H. Esbensen and E.S. Kuh. EXPLORER: an interactive floorplaner for design space exploration. In *European Design Automation Conf.*, pages 356–361, 1996.

[9] D.E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addision-Wesley Publisher Company, Inc., 1989.

[10] J. Koza. *Genetic Programming - On the Programming of Computers by means of Natural Selection.* MIT Press, 1992.

[11] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[12] S. Yang. Logic synthesis and optimization benchmarks user guide. Technical Report 1/95, Microelectronic Center of North Carolina, 1991.

[13] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Trans. on Evolutionary Comp.*, 3(4):257–271, 1999.