Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Construction of Image Feature Extractors Based on Multi-objective Genetic Programming with Redundancy Regulations

Ukrit Watchareeruetai, Tetsuya Matsumoto, Yoshinori Takeuchi,
Hiroaki Kudo, and Noboru Ohnishi
Department of Media Science, Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 484-8603, Japan
ukrit@ieee.org, {matumoto,takeuchi,kudo,ohnishi}@is.nagoya-u.ac.jp

***Abstract*—This paper proposes a multi-objective genetic programming (MOGP) for automatic construction of feature extraction programs (FEPs). The proposed method is modified from a well known non-dominated sorting evolutionary algorithm, i.e., NSGA-II. The key differences of the method are related with redundancies in program representation. We apply redundancy regulations in three main processes of the MOGP, i.e., population truncation, sampling, and offspring generation, to improve population diversity. Experimental results exhibit that the proposed MOGP-based FEPs construction system provides obviously better performance than the original non-dominated sorting approach.**

***Index Terms*—Multi-objective optimization, linear genetic programming, image feature extraction, redundancy regulation, non-dominated sorting**

I. Introduction

Recently, many researchers have been interested in automatic program construction for object recognition. Many approaches exploit evolutionary computation techniques [6], [20], especially genetic programming (GP) [13], to search for the optimal programs. Some researches attempt to construct just a part in object recognition, e.g., feature extractor [8], [17] or interest point detector [9], while some researches focus on construction of complete object recognition programs [2], [11], [14]. In this work, we focus on automatic construction of feature extraction programs (FEPs). Our approach considered here can construct FEPs without domain-specific knowledge.

In most approaches, including our previous works, single objective GPs were adopted to optimize performance of constructed programs [2], [8], [11], [14], [17]. However, in practice, we may want to optimize various objectives simultaneously (e.g., to find a program that achieves both true positive and true negative rates), and prefer various alternatives to make a decision. Consequently, we attempt to introduce evolutionary multi-objective optimization (EMO) [1] to the automatic FEP construction system.

Nowadays, Pareto-based EMO, e.g., non-dominated sorting genetic algorithm (NSGA-II) [7], have been exploited to solve many problems successfully, especially in GA domain. We tried to adopt the NSGA-II technique into our GP-based FEP construction, so we called it non-dominated sorting GP (NSGP). However, it appears that the NSGP could not work well with our automatic FEP construction system. The main reason seems to be the high redundancies in GP representations. This quite contrasts with GA representations, which usually contain no or low redundancies. The redundancies in GP representation together with elitist selection, a main aspect of NSGA-II, decrease population diversity rapidly in a few generations, and leads to poor performance in FEP construction.

In this paper, we propose a multi-objective optimization GP (MOGP) techniques, which is modified from the NSGP, for automatic construction of FEPs. Three redundancy-regulated mechanisms are introduced to improve population diversity as well as convergence rate. The first is named semi-elitist truncation that firstly selects the program with better rank but only one copy for each round. The second is a sampling mechanism that chooses different phenotypes uniformly (not just to choose each individual with equal probability), named phenotypic-uniform sampling. The third redundancy regulation is exploited in offspring generation process—the prohibition of offspring that represent programs discovered before. Experimental results suggest that the proposed method significantly outperforms the original NSGP method. We also demonstrate that the proposed MOGP method can even better than a single objective GP-based approach in solving a single objective problem.

The rest of this paper is organized as follows. Section II provides some backgrounds in multi-objective optimization and description of the NSGA-II. Section III explains our GP-based system for FEPs construction. Section IV describes the proposed MOGP techniques. Experimental results and discussion are in section V. Finally, section VI concludes the paper.

II. Background and Related Work

A. Multi-objective Optimization and Concept of Domination

The multi-objective optimization (MO) problem considered here is a maximization problem which is generally described as follows:

$$Maximize\ f(x) = (f_1(x), f_2(x), f_3(x), ..., f_m(x)),$$

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

where x is a solution and $f_1(\cdot), f_2(\cdot), ..., f_m(\cdot)$ are m objectives to be maximized. A solution x is said to dominate the other solution y (denoted by $x \succ y$) if the following conditions are satisfied:

$$\begin{aligned} &\forall i \in \{1, 2, ..., m\} : f_i(x) \geqslant f_i(y) \quad \wedge \\ &\exists i \in \{1, 2, ..., m\} : f_i(x) > f_i(y). \end{aligned} \tag{1}$$

The ideal goal of multi-objective optimization is to find so called Pareto-optimal front—the set of all possible solutions that are not dominated by the other possible solutions. However, Pareto-optimal front is unknown for most problems, and it may be very difficult to achieve all solutions in the fronts. In practice, we may just prefer a non-dominated front that aligns closely to the Pareto-optimal front. Also a well distribution of the solutions in the front is preferred; more uniform distributions provide wide-range choices to be chosen, compared with compact ones.

B. Non-dominate Sorting Genetic Algorithm: NSGA-II

Srinivas et al. [12] proposed non-dominated sorting genetic algorithm (NSGA) in 1995, and its improved version called NSGA-II was proposed in 2000 by Deb et al. [7]. NSGA-II was mainly designed to resolve three criticisms of NSGA, i.e., computational complexity, lack of elitism, and the need for specifying a sharing parameter. NSGA-II has been adopted to solve many problems successfully, and becomes a popular EMO method over the years.

Pseudo code in Fig. 1 describes how NSGA-II works. The main concept of NSGA-II is based on non-dominated sorting (step 2). As shown in Fig. 2, a set of all solutions that do not dominate each other are grouped together, and a rank is assigned to each solution in the front. All solutions in the first front are not dominated by any solutions in the other fronts (so called non-dominated front). In a front, each solution will be assigned a relative distance measurement called crowded distance, which is the average distance (in objective space) from itself to the adjacent solutions. To create new parent populations, elitist truncation (steps 3–6) is adopted. Specifically, the truncation is firstly based on the rank (step 4), and then based on the crowded distance for the last front to be selected (steps 5–6). The rank and crowded distance are also considered in selection process (step 7). Binary tournament selection (pool size of two) is adopted. The solutions with better rank are selected in the first priority, and if two solutions have the same rank, the solution with higher crowded distance is preferred.

1) $R_t \leftarrow P_t \cup Q_t$ (P_t: parent population, Q_t: offspring population, t: generation index)
2) Apply non-dominated sorting to R_t to obtain non-dominated fronts $F = \{F_1, F_2, ..., F_{n_t}\}$, where n_t is the number of fronts
3) $P_{t+1} \leftarrow \phi$ and $i \leftarrow 1$
4) while $|P_{t+1}| + |F_i| \leq N$
   $P_{t+1} \leftarrow P_{t+1} \cup F_i$
   $i \leftarrow i + 1$
5) Sorting $|F_i|$ based on crowded distance, $j \leftarrow 1$
6) while $|P_{t+1}| < N$
   $P_{t+1} \leftarrow P_{t+1} \cup s_j$, where $s_j \in F_i$
   $j \leftarrow j + 1$
7) Create new offspring Q_{t+1} from P_{t+1}

Fig. 1. Pseudo code of NSGA-II

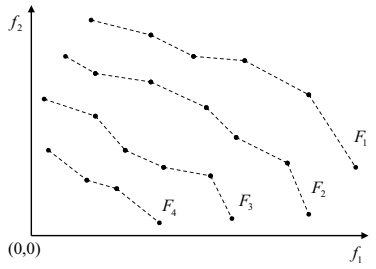

Fig. 2. Example of rank assignment by non-dominate sorting

III. Evolutionary Construction of Feature Extraction Programs

A. System Overview

Figure 3 exhibits the overview of evolutionary system for construction of FEPs. In this system, inputs needed from user are just image processing library, training images, and objective function(s). Image processing library consists of basic image processing operations, e.g., edge detection, lowpass filtering, image thresholding. These operations are used as primitive operations (POs) for FEP construction. Firstly, the system randomly generates an initial population of individuals (or chromosomes), which encode feature extraction programs. These individuals are then interpreted into FEPs and are evaluated. In the evaluation process, the defined objective function(s) is used to compute fitness (which describes performance of the program). The individuals with the higher fitness will have higher chance to survive and be evolved by recombination operators. After evolution process finished, the program that gives the best fitness is considered as the output of the system.

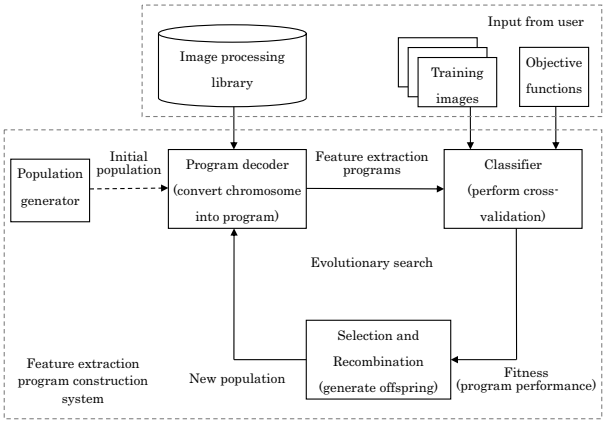

Fig. 3. Overview of an evolutionary system constructing feature extraction program

B. Representation and Decoding

We adopt linear genetic programming (LGP) representation [10], in which a program is represented as a sequence of instructions (fixed- or variable-length), and program execution is based on a set of shared registers. In particular, each instruction is encoded by operation code, which indicates PO to be executed, and arguments that specify input and output registers. Program execution starts from the first operation in the sequence, and move to the next operation sequentially. Because image processing operations are utilized as POs, a special register type, i.e., image register (R_I), is needed to store input and processed images. We also need some numerical registers (R_N) to store some parameters of POs.

Our LGP representation is slightly different from the original representation; we use sub-program structure as shown in Fig. 4. One linear program consists multiple sub-programs; each is executed independently of the others. Each sub-program generates one feature image, which is the content stored in a pre-defined image register after sub-program execution finished. Once all sub-programs are executed, feature images are inputted into a classifier to obtain recognition result. We use Bayesian classifier with histogram approximation [15] as the classifier.

C. Genetic Operators

Genetic operators, i.e., crossover and mutation, are applied to selected solutions (called parents) to produce new solutions (called offspring). Crossover we used is parameterized uniform crossover [19] with 0.2 probability of instruction exchange between two parents. Also a crossover operator that allows swapping of entire sub-programs is adopted. Probability that each crossover type will be used is equal. Mutation operator used here randomly inserts, deletes, or modifies an instruction.

D. Program Evaluation

In our previous work [17], a single objective approach, we used recognition accuracy as fitness value. In the case that the number of object and background pixels are not in balance, e.g., objects are relatively small, GP tried to optimize recognition accuracy by avoiding false positive, resulting in missing inner-boundary of objects and even missing entire small objects.

In this work, we add the other performance measure, i.e., true positive rate, to force GP find more object pixels. In evaluation process, leave-one-out cross validation is adopted. Specifically, validation is done T times, where T is the number of training images. In each time, $T-1$ images are used for classifier training and the remaining one image is for validation. From all validations, we find the numbers of true positive (tp), true negative (tn), false positive (fp), and false negative (fn) pixels. Then recognition accuracy ACC and true positive rate TP are computed as shown in Eqs. 2 and 3, respectively.

$$ACC = \frac{tp + tn}{tp + tn + fp + fn} \tag{2}$$

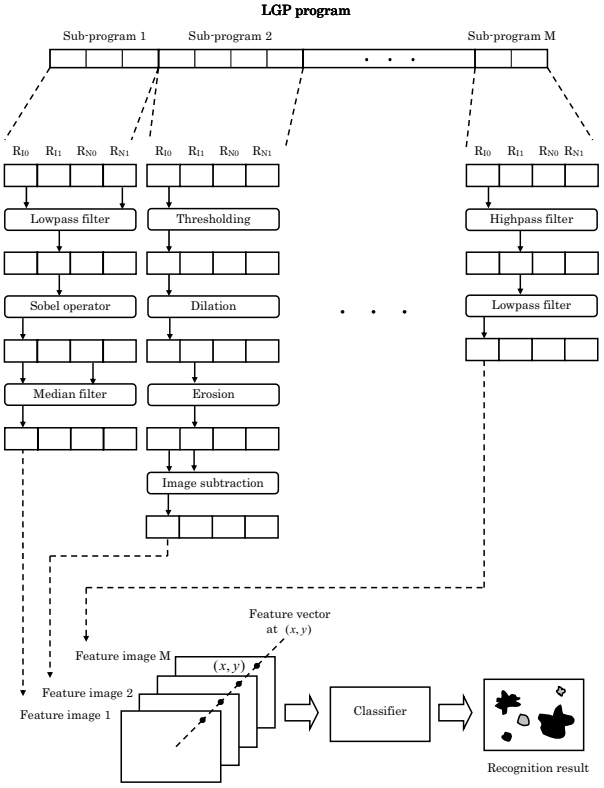

Fig. 4. Example of sub-program structure and its execution

$$TP = \frac{tp}{tp + fn} \tag{3}$$

E. Redundancy and Canonical Transformation

Generally, GP representations, including tree-, linear-, or graph-based GPs, contains a lot of redundancies, i.e., a program may be represented by different chromosomes. There are several causes of redundancies in GP representations, e.g., existence of introns (ineffective instructions), protection mechanism, and program structure. In [17], we have identified the causes of redundancies in LGP-based representations, and proposed a canonical transformation for converting original LGP representation into canonical form in which redundancies are removed. In canonical form, it is easy to verify whether two chromosomes represent the identical program. This enables GP to be improved by various techniques [5], [17], [18].

IV. Proposed Method

A. Problems and Motivation

In this paper, we attempt to implement a MOGP based on the concept of NSGA-II, and we call it non-dominated sorting GP (NSGP). However, the NSGP could not work well in our evolutionary FEP construction system. The reasons

would be related with a difference in characteristics of GA and GP problems—redundancy level. In many GA problems, chromosome representations do not contain any redundancy—genotype-phenotype mapping is one-to-one. On the contrary, chromosome representations in GPs often have high-level redundancies (e.g., as described in section III-E). This means that the mapping from the genotype (representation) space into phenotype (program) space is many-to-one. Moreover, the mapping from phenotype space into objective space is generally many-to-one too, i.e., it is possible that two completely different programs may lead to the same objective values. Consequently, elitism truncation in NSGA-II seems to make population lose diversities rapidly. In addition, it seems that difference in structure difficulties appears, like in tree-based GP [4]. In other words, some programs may be produced easily, whereas the other programs are rarely produced. A solution in the non-dominated front to be easily produced may occupy the most population space within a few generations, resulting in diversity loss. These motivated us to develop the ways to regulate redundancies in MOGP-based FEP construction.

B. Non-dominated Sorting With Redundancy Regulations

We propose a MOGP improved from NSGA-II based GP for automatic construction of FEPs. The proposed method was designed to resolve the redundancies related problems described in the previous section. It is described as the pseudo code in Fig. 5. The key different points of the proposed method are listed as follows:

- *Semi-elitist truncation (steps 3–5)*: we firstly select the solutions in the first front but only one copy for each point, and go to the next front and repeat this process. If all fronts are processed and the next parent population is not full, we start selection process from the first front again. By doing that, we can maintain diversity of population while guarantee that the each elite point still exists in the population (if the number of the elite points are not greater than the population size).
- *Phenotypic-uniform sampling (in steps 6–7)*: instead of uniform sampling, we use a sampling mechanism that is likely to choose the points in objective space with less or no copy point. The selection probability of a point p_i is defined as $1/(N_{diff} * C_i)$, where N_{diff} is the total number of different points in objective space, and C_i is the number of solutions that are located in p_i.
- *Prohibition of redundant individuals (in step 7)*: once an offspring is produced, we transform it into canonical form [17] and verify whether it represents a program discovered before in the evolutionary search. If it is a redundant program (already discovered), we apply mutation operation on that individual until it becomes the new one that has not been discovered before. In [17], we have demonstrated that the use of such constraint can significantly help improve search performance of single objective GP.

1) $R_t \leftarrow P_t \cup Q_t$ (P_t: parent population, Q_t: offspring population, t: generation index)
2) Apply non-dominated sorting to R_t to obtain non-dominated fronts $F = \{F_1, F_2, ..., F_{n_t}\}$, where n_t is the number of fronts
3) Subdivide each F_i into groups G_{ij} that contains the solutions with same objectives ($F_i = \bigcup_j^{m_i} G_{ij}$, where m_i is the number of groups).
4) $P_{t+1} \leftarrow \phi$, $i \leftarrow 1$, and $j \leftarrow 1$
5) while $|P_{t+1}| \leq N$
   if $i = n_t$ then $i \leftarrow 1$
   if $j = m_i$ then $j \leftarrow 1$ and $i \leftarrow i + 1$
   if $|G_{ij}| > 0$ then randomly select a solution s from G_{ij}, $G_{ij} \leftarrow G_{ij} - \{s\}$, and $P_{t+1} \leftarrow P_{t+1} \cup \{s\}$
   $j \leftarrow j + 1$
6) Calculate sampling probability and averaged crowded distance for each solution in P_{t+1}
7) Create new offspring Q_{t+1} from P_{t+1} by using phenotypic-uniform sampling, big tournament size, and redundant-offspring prohibition

Fig. 5. Pseudo code of the proposed MOGP

- *Averaged crowding-distance (in steps 6–7)*: the direct use of crowding-distance assignment algorithm in [7] may result in assigning of different distance for the solutions located in the same point (some may have zero distance whereas the others have positive distance). Therefore we find the summation of distance values of all solutions located in the same point (except the extrema), and assign the average value for each point instead.
- *Big tournament size (in step 7)*: the balance between exploration (global search) and exploitation (local search) powers is a crucial issue in EAs [6]. The above redundancy regulation mechanisms greatly increase diversity of population, i.e., exploration power is increased. However, there are only a little copies of each elite solutions (non-dominated solutions) existing in each generation. If we use small tournament size (e.g., two), only a few numbers of elite solutions will be exploited in each generation (i.e., too low exploitation power), and this leads to slow convergence. Therefore big tournament size (in this paper, 10 for population size of 50) is preferred to maintain well balance between exploration and exploitation.

V. Experiments

To evaluate the performance of the proposed method, we have conducted two experiments. The first experiment compares FEP construction performance of the proposed and NSGP methods. In the second experiment, the proposed MOGP-based approach is compared with our previous single objective approach.

A. Test Problem

In this work, we consider lawn weed detection problem [16], [17], which is a two-class segmentation problem. The goal is to segment the area of weeds from lawn background in order to perform precision spraying. The dataset we used is shown in Fig. 6. It consists of five images of size 160×120 pixels.

B. NSGP v.s. NSGP with Redundancy Regulations

In this experiment, we compare performance of the NSGP and the proposed methods in FEP construction. Two objective

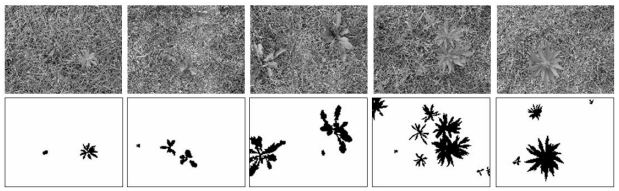

Fig. 6. Lawn weed images and their corresponding ground truths

TABLE I
LIST OF PARAMETERS.

Parameter	Setting
Population size	50
Max. generations	100
Crossover operator	parameterized uniform and sub-program crossovers
Mutation operator	insertion, deletion, modification
# Crossover offspring	24 (48%)
# Mutated offspring	26 (52%)
Tournament size	NGSP: 2
	proposed: 10
Max. chromosome length	20 operations
# image registers	4
# numerical registers	4
# primitive operations	51

functions described in Eqs. 2 and 3 were used. Their parameters were set as shown in Table I. The NSGP and the proposed method were executed 30 times, and their average results were compared.

Performance comparison was done based on two complementary measures used in [3]. The first measure is hypervolume, the summation of block areas under non-dominated front (Fig. 7). An algorithm that provides large hypervolume implies that it found non-dominated front with well distribution and/or good convergence. The second measure is coverage $C(A \succ B)$, which is the fraction of non-dominated solutions found by algorithm B that are dominated by at least one solution found by algorithm A. Note that $C(A \succ B)$ is not usually equal to $1 - C(B \succ A)$; consequently, we need to consider both values. The coverage measure provides relative information on convergence between two algorithms.

The comparison based on the hypervolume measure is shown in Table II. As we expected, the proposed method provides larger hypervolume than the NSGP method because redundancy regulation mechanisms would guide GP to find non-dominated solutions with (at least) better distribution. The t-test confirms these two distributions are significantly different. To assure its superiority in convergence, we have to consider the coverage measure shown in Table III. From the result, most solutions found by the NSGP method are dominated by those of the proposed method. It obviously suggests that the proposed method provides much better convergence than the NSGP method.

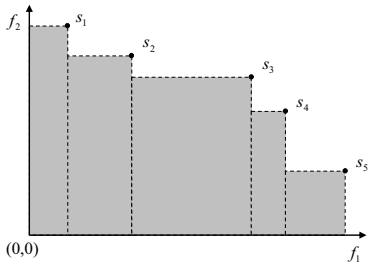

Fig. 7. Hypervolume of two-objective case (origin is the reference point)

TABLE II
COMPARISON OF HYPERVOLUME.

Method	NSGP	proposed
Average	0.5543	0.6843
STD	0.0854	0.0470
t-test	3.29×10^{-7}	

Table IV shows the average size of non-dominated front and average number of different points in the front. In the case of the NSGP method, the non-dominated front size often reached population size but most solutions in the front were redundant. The result of a trial shown in Fig. 8 demonstrates that redundant solutions did not distribute uniformly (see the nearby numbers in Fig. 8). Instead, population seems to converge to a solution that is easy to be produced (different structure difficulties [4]). These reveal an adverse effect of the elitist truncation in high-redundancy EMO. In the case of the proposed method, the ratio between these two values is nearly one, i.e., most copy points were removed. This would be mainly caused by the use of semi-elitist truncation. Also the average size of non-dominated front is far smaller than the population size. This implies that many fronts were preserved and rank-based selection would still properly function.

TABLE III
COMPARISON OF COVERAGES.

A: NSGP, B: proposed	
$C(A \succ B)$	$C(B \succ A)$
0.0394	0.9196

TABLE IV
COMPARISON OF NON-DOMINATED FRONT SIZE.

Method	NSGP	proposed
Average 1^{st} front size	45.87	14.23
Average no. of different points	5.60	13.63
Ratio	0.14	0.96

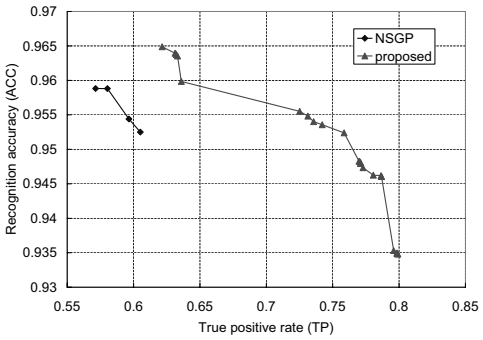

Fig. 8. Example of non-dominated fronts obtained from the NSGP and proposed methods. The numbers in the graph indicate the number of copies of points (if no number, there is only one copy).

TABLE V
COMPARISON OF SEGMENTATION ACCURACY (ACC) BETWEEN SINGLE- AND MULTI-OBJECTIVE APPROACHES.

Method	single objective	multi-objective (proposed)
Average	0.9608	0.9623
STD	0.0034	0.0015
t-test	4.03×10^{-3}	

C. Single Objective v.s. Multi-objective

In the previous section, we have demonstrated that the proposed MOGP method is better than the NSGP method. Here we compare the proposed MOGP-based approach with single objective based approach for solving a single objective problem. Comparison was done based on one objective, i.e., recognition accuracy (Eq. 2). The single objective GP with prohibition of producing discovered offspring described in [17] was experimented. The same parameters as in the previous section were used. The average segmentation accuracy (over 30 independent runs) of the best individual from each run is shown in Table V. The result indicates that the proposed MOGP-based approach outperforms the single-objective approach. Again the t-test indicates that these two distributions are significantly different. This result suggests that the use of the proposed MOGP could help encourage automatic FEP construction system in solving even a single objective problem.

VI. CONCLUSION

We have proposed a MOGP technique for automatic construction of feature extraction programs. The proposed method was modified from the NSGA-II based MOGP (NSGP), by including redundancy regulation mechanisms, i.e., semi-elitist truncation, phenotypic-uniform sampling, and prohibition of redundant offspring, to improve population diversity. The experimental results indicate that the proposed MOGP could effectively control redundancies in the evolutionary process, resulting in better performance (measured by hypervolume and coverage) compared with the NSGP. Also the use of proposed MOGP could even find better programs in a single objective problem, compared with our previous single objective FEP construction system.

ACKNOWLEDGMENT

The authors thank the Hori Information Science Promotion Foundation for a research grant.

REFERENCES

[1] A. Abraham, L. Jain, and R. Goldberg (Eds.), Evolutionary Multiobjective Optimization: Theoretical Advances and Application, Springer-Verlag, USA, 2005.
[2] A. Teller and M. Veloso, "PADO: a new learning architecture for object recognition," in Symbolic Visual Learning, K. Ikeuchi and M. Veloso, Eds, Oxford, U.K.: Oxford Univ. Press, pp. 77–112, 1997.
[3] E. Zizler and L. Thile, "Multiobjective optimization using evolutionary algorithms: a comparative study," in Parallel Problem Solving from Nature V, A.E. Eiben, Ed, Amsterdam, pp. 292–301, 1998.
[4] J.M. Daida, H. Li, R. Tang, and A.M. Hilss, "What makes a problem GP-hard? validating a hypothesis of structural causes", in Proc. of GECCO-2003, LNCS 2724, Springer-Verlag, pp.1665–16677, 2003.
[5] J. Niehaus, C. Igel, and W. Banzhaf, "Reducing the number of fitness evaluations in graph genetic programming using a canonical graph indexed database," Evolutionary Computation, vol. 15, no. 2, pp. 199–221, 2007.
[6] K.A. De Jong, Evolutionary Computation: A Unified Approach, The MIT Press, 2006.
[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Trans. Evolutionary Computation, vol.6, no.2, pp.182–197, April 2002.
[8] K. Krawiec and B. Bhanu, "Visual learning by evolutionary and coevolutionary feature synthesis," IEEE Transactions on Evolutionary Computation, vol. 11, no. 5, pp. 635–650, 2007.
[9] L. Trujillo and G. Olague, "Automated design of image operators that detect interest points," Evolutionary Computation, vol. 16, no. 4, MIT Press, pp. 483–507, 2008.
[10] M. Brameier and W. Banzhaf, Linear Genetic Programming, New York, Springer-Verlag, 2007.
[11] M. Zhang, V. Ciesielski, and P. Andreae, "A domain-independent window approach to multiclass object detection using genetic programming," EURASIP Journal on Applied Signal Processing, vol.8, pp. 841–859, 2003.
[12] N. Srinivas, and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms", Evolutionary Computation, vol.2, no.3, pp.221–248, MIT Press, 1995.
[13] R. Poli, W.B. Langdon, and N.F. McPhee, A Filed Guide to Genetic Programming, Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008)
[14] S. Shirakawa and T. Nagao, "Genetic image network (GIN): automatically construction of image processing," in Proc. IWAIT-2007, Bangkok, Thailand, pp. 643–648, 2007.
[15] S. Theodoridis and K. Koutroumbas, Pattern Recognition 2^{nd} edition, Academic Press, 2003.
[16] U. Watchareeruetai, Y. Takeuchi, T. Matsumoto, H. Kudo, and N. Ohnishi, "Computer vision based method for detecting weeds in lawns", Machine Vision and Applications, vol.17, no.5, pp.287–296, 2006.
[17] U. Watchareeruetai, Y. Takeuchi, T. Matsumoto, H. Kudo, and N. Ohnishi, "Transformation of redundant representations of linear genetic programming into canonical forms for efficient extraction of image features", in Proc. IEEE Congress on Evolutionary Computation (CEC-08), Hong Kong, China, pp.1996–2003, 2008.
[18] U. Watchareeruetai, Y. Takeuchi, T. Matsumoto, H. Kudo, and N. Ohnishi, "Efficient construction of feature extraction programs by using linear genetic programming with fitness retrieval and intermediate-result caching", in Foundation of Computational Intelligence Volume 4: Bio-inspired Data Mining, A. Abraham et al., Eds, Springer-Verlag, pp.355–375, 2009.
[19] W.M. Spears and K.A. De Jong, "On the virtues of parameterized uniform crossover," in Proc. ICGA-91, R.K. Belew and L.B. Booker, Eds, Morgan Kaufmann, pp. 230–236, 1991.
[20] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Berlin, Germany, Springer-Verlag, 1996.