Adaptive Control of the Number of Crossed Genes in Many-Objective Evolutionary Optimization

Hiroyuki Sato[1], Carlos A. Coello Coello[2],
Hernán E. Aguirre[3] and Kiyoshi Tanaka[3]

[1] Faculty of Informatics and Engineering, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585 JAPAN
[2] Departamento de Computación, CINVESTAV-IPN
Av. IPN No. 2508, México, D.F. 07360, México
[3] Faculty of Engineering, Shinshu University
4-17-1 Wakasato, Nagano, 380-8553 JAPAN

Abstract. To realize effective genetic operation in evolutionary many-objective optimization, crossover controlling the number of crossed genes (CCG) has been proposed. CCG controls the number of crossed genes by using an user-defined parameter α. CCG with small α significantly improves the search performance of multi-objective evolutionary algorithm in many-objective optimization by keeping small the number of crossed genes. However, to achieve high search performance by using CCG, we have to find out an appropriate parameter α by conducting many experiments. To improve the usability of CCG, in this work we propose an adaptive CCG which dynamically controls the parameter α during the solutions search in a single run of the algorithm. Simulation results show that the values of α controlled by the proposed method converges to an appropriate value even when the adaptation is started from any initial values. Also we show the adaptive CCG achieves more than 80% with a single run of the algorithm for the maximum search performance obtained by the static CCG using an optimal α^*.

1 Introduction

The research interest of the multi-objective evolutionary algorithm (MOEA) [1] community has rapidly shifted to develop effective algorithms for many-objective optimization problems (MaOPs) because more objective functions should be considered and optimized in recent complex applications. However, in general, MOEAs noticeably deteriorate their search performance as we increase the number of objectives [2], especially Pareto dominance-based MOEAs such as NSGA-II and SPEA2. One reason for this is that these MOEAs face difficulty to rank solutions in the population, i.e., most of the solutions become non-dominated and the same rank is assigned to them, which seriously spoils proper selection pressure required in the evolution process. To overcome this problem in selection, several methods to improve selection pressure have been proposed [2]. Contrary to these studies focusing on selection, to realize effective genetic operation in

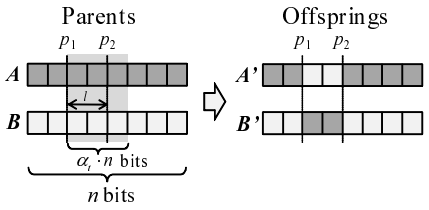


Fig. 1. Controlling crossed genes for two-point crossover ($\mathrm{CCG_{TX}}$)

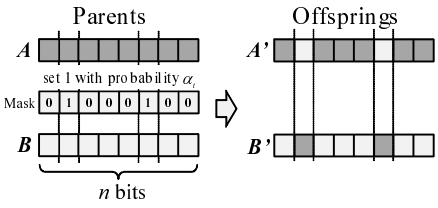


Fig. 2. Controlling crossed genes for uniform crossover ($\mathrm{CCG_{UX}}$)

MaOPs, the variable space of many-objective 0/1 knapsack problem has been analyzed [3]. [3] shows that variables of true Pareto optimal solutions (POS) become noticeably diverse, and true POS becomes distributed almost uniformly in variable space by increasing the number of objectives. Also, [3] shows that offspring created by conventional two-point and uniform crossovers has less chance to be selected as parents because the operators becomes too disruptive and its effectiveness decreases in MaOPs. To overcome this problem and enhance the evolution by crossover operator in MaOPs, controlling the number of crossed genes (CCG) has been proposed [3]. $\mathrm{CCG_{TX}}$, an extension of two-point crossover, controls the maximum length of crossed genes by using an user-defined parameter α_t. Also, $\mathrm{CCG_{UX}}$, an extension of uniform crossover, controls the number of crossed genes by using a parameter α_u. In MaOPs, CCG using a small α remarkably improves the search performance of several MOEAs [3]. However, to achieve high search performance by using CCG, we have to find out an appropriate parameter α by conducting many experiments.

To improve the usability of CCG, in this work we propose an adaptive CCG which dynamically controls the parameter α during the solutions search in a single run of the algorithm. In this work, we analyze the adaptation process of α and verify the effectiveness of adaptive $\mathrm{CCG_{TX}}$ and $\mathrm{CCG_{UX}}$ on many-objective 0/1 knapsack problems with $m = \{4, 6, 8, 10\}$ objectives.

2 Controlling the Number of Crossed Genes

2.1 CCG for Two-point Crossover ($\mathbf{CCG_{TX}}$)

$\mathrm{CCG_{TX}}$ controls the length of crossed genes by using a parameter α_t. **Fig.1** shows the conceptual diagram of $\mathrm{CCG_{TX}}$. First we select parents $\boldsymbol{A}$ and $\boldsymbol{B}$ from the parent population $\mathcal{P}$, and randomly choose the 1st crossover point p_1. Then, we randomly determine the length of the crossed genes ℓ in the range $[0, \alpha_t \cdot n]$. The 2nd crossover point is set to $p_2 = (p_1 + \ell) \mod n$. The possible range of the parameter α_t is $[0.0, 1.0]$. $\alpha_t = 1.0$ indicates the conventional two-point crossover, and the length of crossed genes becomes short by decreasing α_t.

2.2 CCG for Uniform Crossover ($\mathbf{CCG_{UX}}$)

$\mathrm{CCG_{UX}}$ controls the probability of 1 in the mask bits by using a parameter α_u. According to the generated mask bits, we perform uniform crossover as shown in

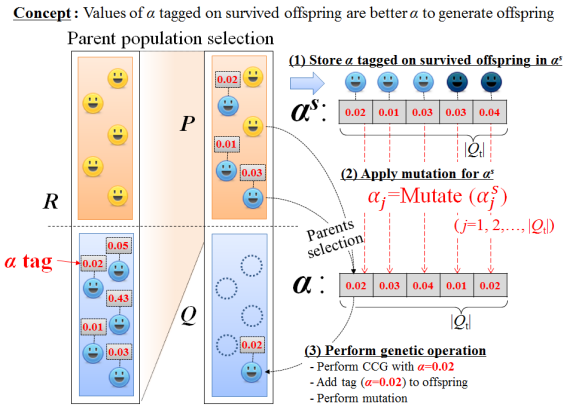


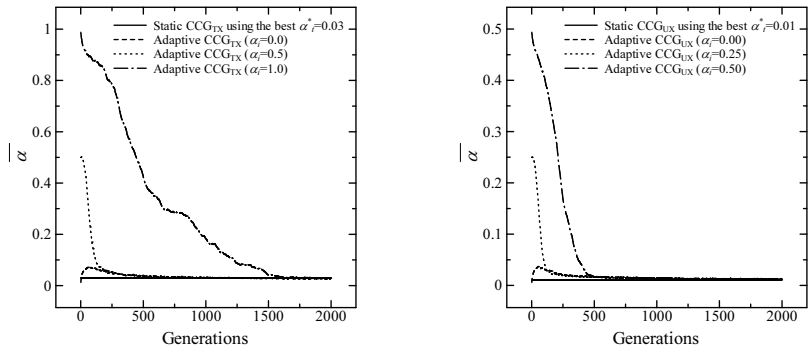
Fig. 3. Conceptual figure of the proposed adaptive CCG

Fig.2. The possible range of α_u is $[0, 0.5]$. $\alpha_u = 0.5$ indicates the typical uniform crossover, and the number of crossed genes becomes small by decreasing α_u.

3 Adaptive Control of the Number of Crossed Genes

CCG using a small α remarkably improves the search performance of several MOEAs in MaOPs [3]. However, to achieve high search performance by the conventional CCG using a static parameter α for the entire solutions search [3], we have to find out an appropriate α by conducting many experiments. To improve the usability of CCG while achieving high search performance in a single run of the algorithm, in this work we propose an adaptive CCG which dynamically controls α so that the parameter is automatically guided to an appropriate value during the solutions search in a single run of the algorithm.

Fig.3 shows the conceptual figure of the proposed adaptive CCG. Since this method is designed based on a framework used in NSGA-II and S-CDAS [3], entire population $\mathcal{R}$ consists of parent (elite) population $\mathcal{P}$ and offspring population $\mathcal{Q}$. In the process of adaptive CCG, we use two vectors. The first one is $\boldsymbol{\alpha^s} = \{\alpha^s_1, \alpha^s_2, \cdots, \alpha^s_{|\mathcal{Q}|}\}$, in which effective values of α are kept to create superior offspring. The other one is $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \cdots, \alpha_{|\mathcal{Q}|}\}$, which is used to generate offspring for the next generation. Also, for all solutions in the offsprings population $\mathcal{Q}$, we individually put a tag showing α used to create each solution. Before we start the solutions search, we initialize α^s_j $(j = 1, 2, \cdots, |\mathcal{Q}|)$ by initial α_i. Also, we initialize the counter that measures the number of survived offspring c by 0. For each generation, we update the elements in $\boldsymbol{\alpha^s}$. After selection of new parents population $\mathcal{P}$, we pick up one survived offspring from $\mathcal{P}$. Then, we increment c and replace the element $\alpha^s_{1+c \mod |\mathcal{Q}|}$ with the value of α tagged on the current offspring. We repeat this process for all survived offspring in $\mathcal{P}$. Next, we determine the value of α_j $(j = 1, 2, \cdots, |\mathcal{Q}|)$ in $\boldsymbol{\alpha}$ by applying polynomial mutation [4] to all the elements in $\boldsymbol{\alpha^s}$. Finally, we create offspring by performing CCG using the updated elements in $\boldsymbol{\alpha}$ to fill up new offsprings population $\mathcal{Q}$.



(a) The adaptive $\mathrm{CCG_{TX}}$ (b) The adaptive $\mathrm{CCG_{UX}}$

Fig. 4. Transition of $\overline{\boldsymbol{\alpha}}$ over generation ($n = 500$ and $m = 8$ objectives)

4 Experimental Results and Discussion

4.1 Problems, Parameters and Metrics

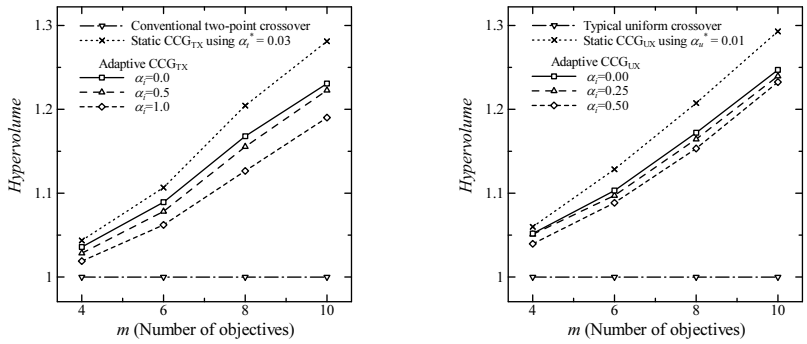
In this work we use many-objective 0/1 knapsack problems [5] with $m = \{4, 6, 8, 10\}$ objectives, $n = 500$ items, and feasibility ratio $\phi = 0.5$. We verify the effects of CCG when it is combined with S-CDAS for parents selection similar to [3]. We adopt crossovers with a crossover rate $p_c = 1.0$, and apply bit-flipping mutation with a mutation rate $p_m = 1/n$. In the following experiments, we show the average performance with 30 runs, each of which spent $T = 2,000$ generations. Population size is set to $N = 200$ ($|\mathcal{P}| = 100$ and $|\mathcal{Q}| = 100$). Also, we employ polynomial mutation [4] with $\eta_m = 40$ to obtain $\boldsymbol{\alpha}$ from $\boldsymbol{\alpha^s}$.

To evaluate the search performance of MOEAs, we use *Hypervolume* (HV) [6], which measures the m-dimensional volume of the region enclosed by the obtained non-dominated solutions and a dominated reference point in objective space. Here, we use $\boldsymbol{r} = (0, 0, \cdots, 0)$ as the reference point. Obtained POS showing a higher value of hypervolume can be considered as a better set of non-dominated solutions from both convergence and diversity viewpoints.

4.2 Transition of $\overline{\alpha}$ in the proposed adaptive CCG

First, we observe the adaptation process of $\boldsymbol{\alpha}$ by the adaptive CCG. **Fig.4** shows the transition of average $\overline{\boldsymbol{\alpha}} = \sum_{j=1}^{|\mathcal{Q}|} \alpha_j / |\mathcal{Q}|$ over generations. For the adaptive $\mathrm{CCG_{TX}}$, we plot three different results by using initial $\alpha_i = \{0.0, 0.5, 1.0\}$. For the adaptive $\mathrm{CCG_{UX}}$, we use $\alpha_i = \{0.0, 0.25, 0.5\}$. Also, we plot $\alpha_t^* = 0.03$ and $\alpha_u^* = 0.01$ maximizing HV by the static $\mathrm{CCG_{TX}}$ and $\mathrm{CCG_{UX}}$ as horizontal lines.

From the result for adaptive $\mathrm{CCG_{TX}}$ shown in **Fig.4 (a)**, we can see that $\overline{\boldsymbol{\alpha}}$ converges to a specific value even when we start adaptation from any initial α_i. Also, the converged value of $\overline{\boldsymbol{\alpha}}$ is close to $\alpha_t^* = 0.03$. Convergence of $\overline{\boldsymbol{\alpha}}$ by the adaptive $\mathrm{CCG_{TX}}$ using $\alpha_i = 0.0$ is fastest among three different adaptive $\mathrm{CCG_{TX}}$. Also, from the result for adaptive $\mathrm{CCG_{UX}}$ shown in **Fig.4 (b)**, we can see the similar tendency of the adaptive $\mathrm{CCG_{TX}}$. From these results, the adaptation of $\boldsymbol{\alpha}$ by the proposed adaptive CCG is thought to be working well.



(a) Adaptive $\mathrm{CCG_{TX}}$ (b) Adaptive $\mathrm{CCG_{UX}}$

Fig. 5. HV obtained by the adaptive and static CCG ($n = 500$ bits)

4.3 Performance of the proposed adaptive $\mathrm{CCG_{TX}}$ and $\mathrm{CCG_{UX}}$

Next, we verify the search performance of the adaptive CCG. **Fig.5 (a)** shows results of HV obtained by the conventional two-point crossover, the adaptive $\mathrm{CCG_{TX}}$ with $\alpha_i = \{0.0, 0.5, 1.0\}$ and the static $\mathrm{CCG_{TX}}$ with the optimal $\alpha_t^* = 0.03$ maximizing HV in [3]. All results are normalized by the results of the conventional two-point crossover. Here, in the case of the static $\mathrm{CCG_{TX}}$ with α_t^*, we only show the best result among many experiments varying α_t step by step in the possible range of $\alpha_t \in [0, 1]$. On the other hand, in the cases of the conventional two-point crossover and the proposed adaptive $\mathrm{CCG_{TX}}$, we show results obtained by a single run of the algorithm. As De Jong mentioned in [7], note that performance comparison between EA with static parameter setting and EA with adaptive setting is unfair since it is likely that the static setting is established via preliminary parameter tuning by many experiments conducted in advance, which is not included in the comparison. Therefore, note that comparing the adaptive $\mathrm{CCG_{TX}}$ with the static $\mathrm{CCG_{TX}}$ using α_t^* is not fair comparison. In these graphs, we show the achievement of the search performance by the adaptive $\mathrm{CCG_{TX}}$ between the basic performance by conventional two-point crossover and the maximum performance by the static $\mathrm{CCG_{TX}}$ using α_t^*.

From the results of **Fig.5 (a)**, we can see that the adaptive $\mathrm{CCG_{TX}}$ using any α_i achieves higher HV than the conventional two-point crossover. Also, the adaptive $\mathrm{CCG_{TX}}$ using smaller initial adaptation range α_i shows higher HV, and the adaptive $\mathrm{CCG_{TX}}$ with $\alpha_i = 0.0$ achieves the highest HV among the adaptive $\mathrm{CCG_{TX}}$ using three different initial α_i. This is because the adaptive $\mathrm{CCG_{TX}}$ with $\alpha_i = 0.0$ realizes the fastest convergence of α to the optimal value as shown in **Fig.4 (a)**. Next, by comparing results of the adaptive $\mathrm{CCG_{TX}}$ with the static $\mathrm{CCG_{TX}}$ using α^*, we can see that the adaptive $\mathrm{CCG_{TX}}$ using $\alpha_i = 0.0$ achieves $\{82.1, 83.7, 82.1, 82.1\}\%$ of HV for the maximum HV obtained by the static $\mathrm{CCG_{TX}}$ with α_t^* for $m = \{4, 6, 8, 10\}$ objectives, respectively.

Next, **Fig.5 (b)** shows results of HV obtained by the typical uniform crossover, the adaptive $\mathrm{CCG_{UX}}$ with $\alpha_i = \{0.0, 0.25, 0.5\}$ and the static $\mathrm{CCG_{UX}}$ with the

optimal $\alpha_u^* = 0.01$ maximizing HV. All results are normalized by the results of the typical uniform crossover. From the results of **Fig.5 (b)**, we can see the similar tendency obtained by the adaptive $\mathrm{CCG_{TX}}$. The adaptive $\mathrm{CCG_{UX}}$ with $\alpha_i = 0.0$ achieves $\{86.6, 80.3, 83.0, 84.3\}\%$ of HV for the maximum HV obtained by the static $\mathrm{CCG_{UX}}$ with α_t^* for each $m = \{4, 6, 8, 10\}$ objectives problem.

5 Conclusions

To improve the usability of CCG and find out an appropriate parameter α in a single run of the algorithm, we have proposed the adaptive CCG which dynamically controls the parameter α during the solutions search. Also, we have analyzed the adaptation process of α, and have verified the effectiveness of the adaptive CCG in the search performance on many-objective 0/1 knapsack problems with $m = \{4, 6, 8, 10\}$ objectives. Simulation results showed that average $\overline{\boldsymbol{\alpha}}$ controlled by the adaptive CCG converges to an appropriate value even when the adaptation is started from any initial values. Also, we showed that the converged value of $\overline{\boldsymbol{\alpha}}$ is close to α^* maximizing HV by the static CCG. Through performance verification, we showed the adaptive $\mathrm{CCG_{TX}}$ and $\mathrm{CCG_{UX}}$ achieve higher HV than the conventional two-point crossover and the typical uniform crossover. Also, we showed that the adaptive CCG using small initial α_i achieves more than 80% with a single run of the algorithm for the maximum HV obtained by the static CCG with α^* found through many experiments.

As future works, we want to further improve the search performance of the proposed algorithm by refining the adaptation mechanism. Also, we are planning to study on the effective crossover operators for many-objective continuous optimization problems.

References

1. C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Boston, Kluwer Academic Publishers, 2002.
2. H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review", *Proc. of 2008 IEEE Congress on Evolutionary Computation (CEC2008)*, pp. 2424-2431, 2008.
3. H. Sato, H. Aguirre and K. Tanaka, "Improved S-CDAS using Crossover Controlling the Number of Crossed Genes for Many-objective Optimization", *Proc. GECCO 2011*, pp.753-760, 2011.
4. K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design", *Computer Science and Informatics*, 26(4), 30-45, 1996.
5. E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms – a comparative case study", *Proc. 5th Intl. Conf. on Parallel Problem Solving from Nature (PPSN-V)*, LNCS Vol.1498, pp.292-304, 1998.
6. E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999.
7. K. De Jong, "Parameter Setting in EAs: a 30 Year Perspective", in *Parameter Setting in Evolutionary Algorithms*, Springer, pp.1–18, 2007.