Real-Valued Constraint Optimization with ICHEA

Anurag Sharma and Dharmendra Sharma

Faculty of Information Sciences and Engineering
University of Canberra, ACT, Australia
{Anurag.Sharma,Dharmendra.Sharma}@canberra.edu.au

Abstract. *Intelligent constraint handling evolutionary algorithm* (ICHEA) is a recently proposed variation of evolutionary algorithm (EA) that solves real-valued constraint satisfaction problems (CSPs) efficiently [20]. ICHEA has ability to extract and exploit information from constraints that guides its evolutionary search operators in contrast to traditional EAs that are 'blind' to constraints. Even its efficacy to solve CSPs it was not implemented to handle constraint optimization problems (COPs). This paper proposes an enhancement to ICHEA to solve real-valued COPs. The presented approach demonstrates very competitive results with other state-of-the-art approaches in terms of quality of solutions on well-known benchmark test problems.

Keywords: Intelligent constraint handling evolutionary algorithm (ICHEA), evolutionary algorithm (EA), constraint satisfaction problem (CSP), constraint optimization problem (COP).

1 Introduction

Evolutionary algorithm (EA) has been successful in solving many difficult NP class problems; however, it suffers from some of its inherent approaches to solve constraint problems as it does not make use of information from constraints and blindly search in the solution space using various heuristic search operators [3, 5, 16]. Characteristically, constraint problems solved by EAs use penalty based functions. A penalty function updates the fitness of chromosomes in EA. A penalty term is used in general for reward and punishment for satisfying and/or violating the constraints [4]. Use of penalty functions has been commonly reported in literature for use in constrained optimization. Two basic types of penalty functions exist; exterior penalty functions, which penalize infeasible solutions, and interior penalty functions, which penalize feasible solutions [2]. The most popular method adopted to handle constraints in EAs was taken from the mathematical programming literature – penalty functions (mostly exterior penalty functions) – where the aim is to decrease (*punish*) the fitness of infeasible solutions as to favor those feasible individuals in the selection and replacement processes. The main advantage of the use of penalty functions is their simplicity; however, their main shortcoming is that penalty factors, which determine the severity of the punishment, must be set by the user and their values are problem dependent that requires a careful fine-tuning of parameter to obtain

T. Huang et al. (Eds.): ICONIP 2012, Part III, LNCS 7665, pp. 406–416, 2012.

competitive results [12, 13]. A self-adaptive penalty function based genetic algorithm (SAPF) is proposed in [21] that penalizes individuals based on ratio of total feasible and infeasible individuals present in the population. There are various forms of penalties reported in the literature, like static penalty, dynamic penalty, annealing penalty and death penalty [4].

Some other constraint handling approaches include expensive *repair* algorithms that promote the local search to transform infeasible solutions to feasible solutions because the feasible parents not necessarily produce feasible progenies [4]. In multi-objective optimization (MOO) constraints are transformed into multiple objectives. There are many established MOO algorithms like MOGA [9], VEGA [19], NSGA and NSGAII [6]. Paredis in [17] has used co-evolution strategies that utilizes *predator-prey* model to keep two populations – one population represents solutions that satisfies many constraints while other population represents those individuals whose constraint(s) is violated by lots of individuals in the first population. This strategy requires extra computational effort to find the intersection of a line with the boundary of the feasible region.

The use of domain knowledge within an EA can also be utilized to improve its performance as EAs are 'blind' to constraints. Recently, there have been few algorithms developed that move away from penalty based fitness functions to generic distance function given in Eq. (8). ICHEA [20] uses its *intermarriage* crossover operator to look for overlapping feasible regions through differentiating the boundaries of feasible regions for each constraint. This reduces the search space to obtain the solution efficiently. Cultural algorithms are also used to extract domain knowledge for its evolutionary search by using two subpopulations – population space and the belief space. Ricardo and Carlos in [18] proposed cultured differential evolution (CDE) that uses differential evolution (DE) as the population space and belief space as the information repository to store experiences of individuals for other individuals to learn. Amirjanov in [1] proposed changing domain range based genetic algorithm (CRGA) that adaptively shifts and shrinks the size of search space of the feasible region by employing feasible and infeasible solution in the population to reach the global optimum. Mezura-Montes et. al. in [14] proposed simple multi-membered evolution strategy (SMES) that uses a simple diversity mechanism by allowing infeasible solutions to remain in the population. A simple feasibility-based comparison mechanism is used to guide the process toward the feasible region of the search space. The idea is to allow the individual with the lowest amount of constraint violation and the best value of the objective function to be selected for the next population. PSO-DE proposed by [12] is another algorithm that integrates particle swarm optimization (PSO) and DE to solve real-valued COPs.

This paper is organized as follows: Section 2 describes formalization of CSPs and COPs. Section 3 revisits ICHEA introduced in [20]. Section 4 describes enhanced ICHEA that can solve COPs. Section 5 shows experimental results of ICHEA with other state-of-the-art algorithm to solve number of benchmark COPs. Section 6 discusses the outcomes of the experiments performed and section 7 concludes the paper by summarizing the results and proposing some further extensions to the research.

2 Formalization of CSPs and COPs

Constraint problems can be divided into two classes: Constrained Optimizing Problems (COPs) and constraint satisfaction problems (CSPs).The difference between these classes is that in the first an optimal solution that satisfies all the constraints should be found, while in the second class any solution as long as all the constraints are satisfied is acceptable [8]. It has been shown in [20] that ICHEA is very effective in solving real-valued CSPs, however, its ability to solve COPs was not investigated. This current work is an enhancement of ICHEA to solve real-valued COPs.

A solution to real-valued COP has two folds – search for an optimum solution that also must satisfy all the constraints. Real-valued COP can be formulated as:

$$\text{optimize } f(\vec{x}) \tag{1}$$

where COP's objective function $f(\vec{x})$ has an n-dimensional input vector $\vec{x} = \{x_1, x_2, \ldots x_n\}$ that is defined in a search space S. More specifically, $\vec{x} \in \mathcal{F} \subseteq S$, where $\mathcal{F}$ being the feasible region on the search space $S \subseteq \mathbb{R}^n$. Usually, the search space S is defined as a n-dimensional rectangle in $\mathbb{R}^n$. The domain of variables is defined by their lower bounds l_i and upper bounds u_i:

$$l_i \leq x_i \leq u_i, \quad 1 \leq i \leq n \tag{2}$$

The feasible region $\mathcal{F}$ with bounds on each dimension is further restricted by a set of m additional constraints that can be given in two relational forms – equality and inequality [6, 12, 21].

$$g_i(\vec{x}) \geq 0 \qquad i = 1, \ldots, k \tag{3}$$

$$h_j(\vec{x}) = 0 \quad j = k+1, \ldots, m \tag{4}$$

The equality constraints $h_j(\vec{x})$ cannot be solved directly using EAs so it is converted into inequality constraints by introducing a positive tolerance value δ.

$$g_j(\vec{x}) = \delta - \left|h_j(\vec{x})\right| \geq 0 \tag{5}$$

A set of individual feasible regions $\{\mathcal{F}_1, \mathcal{F}_2, .. \mathcal{F}_m\}$ for each constraint can also be defined as:

$$\mathcal{F}_i = \{\vec{x} \in \mathcal{F} \mid g_i(\vec{x}) \geq 0, 1 \leq i \leq m, i \in Z\} \tag{6}$$

where Z is the set of integers. Many EAs uses a distance function as their fitness function to rank individuals. The distance function indicates how far a chromosome is from the feasible regions [15]. This fitness function tries to bring the chromosomes closer to the feasible region using the following function for $\forall i : \{1 \leq i \leq m\}$:

$$fitness_i(\vec{x}) = \begin{cases} g_i(\vec{x}), & if\ g_i(\vec{x}) < 0 \\ 0, & if\ g_i(\vec{x}) \geq 0 \end{cases} \tag{7}$$

$$e = \sum_{i=1}^{m} |fitness_i(\vec{x})| \tag{8}$$

The fitness function fitness_i is a measurement of *euclidean* distance of a vector $\vec{x}$ from a feasible region $\mathcal{F}_i$. The error function e is the summation of all the fitness functions. Minimizing the error value e leads toward a CSP solution where the objective function $f(\vec{x})$ is not needed. A solution to CSP is found when $e = 0$. To get a COP solution, CSP solutions are further processed to get optimum value of $\vec{x}$ that optimizes the objective function $f(\vec{x})$.

ICHEA has been demonstrated to outperform many well-known EAs to solve CSPs in [20] as it utilizes the information from constraints to guide its evolutionary search operators. The motivation behind this paper is to propose an enhancement of ICHEA to show its efficacy in solving real-valued COPs based on the test results of some benchmark problems.

3 Intelligent Constraint Handling Evolutionary Algorithm

ICHEA uses its novel search operator *intermarriage* crossover that uses information from constraints rather than blindly searching for the solution. In this crossover both parents belong to different feasible regions $\mathcal{F}_i$ and $\mathcal{F}_j$where $i \neq j$. It is also possible that a parent does not belong to any of the feasible regions $S - \mathcal{F}$. The generated offspring contains genes from both parents. The purpose is to make a "generic" offspring that tries to satisfy more than one constraint because its parents are from two different feasible regions. The algorithm favors those offspring which satisfy more constraints by utilizing Deb's ranking scheme based on feasibility [6] to rank the population where the population is first sorted according to number of satisfied constraints in decreasing order then by fitness value given in Eq. (8) in increasing order.

3.1 Intermarriage Crossover for Real-Valued CSPs

In *intermarriage* crossover, two parents generate two offspring. This is a dual process where both parents move closer to each other one at a time and their new positions are considered as two new offspring. An offspring from two parents through intermarriage is defined in a search space as a constant multiple of difference of two parent vectors as shown in Eq. (9). Initially offspring O_1 is placed at position $(P_2 - P_1)/r$ where r is a coefficient in the range within $(0,1)$ which is 0.75 if both parents satisfy at least one different constraint and r is 0.1 if both parents satisfy all same constraints. Then O_1 moves iteratively closer to parent P_1 until it also satisfies the constraint(s) that P_1 satisfies and similarly offspring O_2 is designated. The iterative move can be captured as:

$$O_1 = r^i(P_2 - P_1) \tag{9}$$

Variable i gets incremented from 1 to a threshold value Tin the sequence $\langle 1,2,\ldots,T\rangle$. We have used $T = 5$ for our experiments. So using the Eq. (9) the i value is incremented by 1 until the offspring finds an acceptable place or a threshold value T is reached. This causes two selected vectors (parents) of different constraint satisfaction sets to come closer (offspring) towards constraint boundary because the solution space lies in the overlapping boundary region. Favoring points for *intermarriage* that satisfy more constraints, results in finding solution space quickly [20].

This *intermarriage* crossover tends to converge quickly resulting in low diversity of the population. To avoid this early convergence, the concept of multi-parent crossover has been incorporated where rather than picking most desirable parents from the population, new parents are generate on the vertices of a hyper rectangle that encloses a parent. This hyper rectangle is dynamically created from the locations of two chosen parents P_i and P_j for crossover. To make a hyper rectangle around each parent the following steps are being followed:

- Determine the distance from P_j to P_i $\Delta P_{j,i}$ which is then multiplied by $dimMatrix$. $dimMatrix$ is a square diagonal matrix of size n which is the total dimensions of the search space. The diagonal entries are only ± 1 as shown below. $dimMatrix$ produces 2^n possible combinations of matrices that are used to generate set of all 2^n vertices $P_{dimMatrix}$ of the hyper rectangle where only maximum of up to 2 vertices are chosen randomly. An instance i of $dimMatrix$ namely $dimMatrix_i$ is chosen to create a parent $P_{dimMatrix_i}$ which represents the i^{th} vertex of the hyper rectangle. Matrix multiplication of $dimMatrix_i$ and $\Delta P_{j,i}$ gives the distance from new parent $P_{dimMatrix_i}$ to P_i denoted by $\Delta P_{dimMatrix_i,i}$.

$$dimMatrix = \begin{bmatrix} \pm 1 & 0 & \cdots & 0 & 0 \\ 0 & \pm 1 & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & \pm 1 & 0 \\ 0 & 0 & & 0 & \pm 1 \end{bmatrix} \qquad \begin{aligned} \Delta P_{j,i} &= (P_j - P_i) \\ \Delta P_{dimMatrix_i,i} &= \\ \Delta P_{j,i} &\times dimMatrix_i \end{aligned}$$

- Add vector P_i to the distance vector $\Delta P_{dimMatrix_i,i}$ to get parent $P_{dimMatrix_i}$:

$$P_{dimMatrix_i} = P_{\mathrm{i}} + \Delta P_{dimMatrix_i,i} \tag{10}$$

Parent P_{i} goes through intermarriage crossover with each of these parents and then only the best offspring is selected to go into the offspring pool. This same process is repeated for other parent P_{j}.

3.2 ICHEA Algorithm

ICHEA is a variation of EA introduced in that adds constraint handling features to the standard GAs. The pseudocode can be given as:

```
chromosomes  = initializeChromosomes();
for each generation
  parents = NoveltyTournamentSelection();
  offspring = interMarriageCrossover(parents);
  Mutation(offspring);
  chromosomes = chromosomes + offspring;
  SortAndReplace();
  CheckTerminationCriteria();
End for loop;
```

The detailed description of the algorithm can be found in [20]:

4 ICHEA for Constraint Optimization Problems

ICHEA introduced in [20] is limited to works for CSPs only. We have enhanced the algorithm as below to improve the solutions of the COPs as well.

4.1 Parallel Processing for CSP and COP

The foundation of ICHEA lies in acknowledging the information from the set of feasible regions $\mathcal{F}$ that guides its evolutionary search to solve CSPs effectively. To enhance its capability in solving COPs a formative approach is taken where ICHEA runs two processes in parallel – one to solve CSP and another to optimize CSP solutions. The parallel process starts by dividing the whole population pop into 2 parts. First part pop_{COP} keeps the CSP solutions that are required for optimization and the second part pop_{CSP} keeps the *good* infeasible solutions that are processed to get CSP solutions. The ratio of pop_{COP}: pop_{CSP} is fine-tuned to 1:4 for our experiments.

pop_{CSP} is divided into equal sized m slots where slot i is allocated for individuals that violate i constraints. If there are no individuals with i violations then its allocated space is evenly distributed to other slots. This is done to keep diverse population of partially feasible solutions as [12] have observed that only keeping individuals with lower degree of constraint violations might cause the population to be trapped in a local optimum. Let pop_{CSP_i} indicate the population of individuals that violate i constraints so the total population pop_{CSP} is:

$$pop_{CSP} = \sum_{i=1}^{m} pop_{CSP_i}$$

Then pop_{CSP_i} is sorted according to the fitness and the best $|pop_{CSP}|/m$ is selected for subpopulation pop_{CSP_i}.

$$\therefore \max\left(\left|pop_{CSP_i}\right|\right) = |pop_{CSP}|/m$$

If after allocation, k slots have $\left|pop_{CSP_i}\right| < |pop_{CSP}|/m$, then unallocated population of individuals $pop_{unalloc}$ is:

$$pop_{unalloc} = \sum_{i=1}^{m} \begin{cases} |pop_{CSP}|/m - \left|pop_{CSP_i}\right|, & if\ \left|pop_{CSP_i}\right| < |pop_{CSP}|/m \\ 0 \quad , & otherwise \end{cases}$$

This unallocated population $pop_{unalloc}$ needs to be allocated evenly in the slots that have $\left|pop_{CSP_i}\right| > |pop_{CSP}|/m$.

4.2 Search Focus towards Best So Far Individual

Intermarriage crossover guides the evolutionary search to focus on feasible regions. In addition to normal *intermarriage* crossover the same parents undergo intermarriage crossover with a neighbor of current best solution to guide the search focus towards best so far individual. This is similar to PSO approach [7] where all swarm particles tend to move towards better positions nearby the best position that leads to optimum

solution [7, 10]. This helps in exploring promising solution in a nearby region of the current best solution. If the *intermarriage* crossover operator is denoted by $\otimes$ then the *intermarriage* crossover initiated by parents P_i and P_j involves the following steps:

1. $P_i \otimes P_j$
2. $P_i \otimes P_{dimMatrix_i}\ where\ \{P_{dimMatrix_i} \in P_{dimMatrix} | i \in randSet \wedge |randSet| = 2\}\ where\ randSet = \{\exists i: 1 \leq i \leq |P_{dimMatrix}|\}$
3. $P_{neighbor_j} = \sigma(P_j + P_{best})$ where $\sigma \in (0.0,1.0)$
4. $P_i \otimes P_{neighbor_j}$

The step (1) is just a normal intermarriage crossover between P_i and P_j followed by step (2) that is an intermarriage crossover between a parent P_i and aforementioned newly created parents on the vertices of the hyper-rectangle $\forall P_{dimMatrix_i}$ (see Section 3.1) so that exploration is not limited to the selected population only. Step (3) determines the common neighbor $P_{neighbor_j}$ of parent P_j and the current best chromosome P_{best} using a randomly generated coefficient σ in the range of (0.0, 1.0). Finally intermarriage crossover happens between P_i and $P_{neighbor_j}$ in step (4) which is inspired from PSO to search near by the current best solution. These four steps are specifically used to find the COP solution.

5 Experiment

To validate the efficacy of ICHEA, 11 benchmark problems from COP domain [11, 12, 15] have been selected. All test problems are mathematical functions of various types like quadratic, linear, nonlinear and trigonometric. ICHEA has been compared against five state-of-the-art approaches briefly mentioned in the section 1: CRGA [1], SAPF [21], PSO-DE [12], CDE [18] and SMES [14]. No parallel processing or distributed environment is used for the experiments.

Table 1. Parameter Settings

Parameters	ICHEA
Population size	100
Maximum generations	1.0E3
Maximum evaluations	1.0E6
Mutation rate	0.1
Crossover rate	1.0

An average of 10 successive runs for ICHEA is taken into account to demonstrate its solution quality against published results of other algorithms mentioned above. Table 1 shows the parameter settings used for all test problems. Generally, ICHEA is able to find a solution close to optimal solution in a few generations but it is allowed to run full 1.0E3 generations to try to obtain best possible solutions. For example best solutions for problem G12, G08, G11 and G01 are obtained in 10, 12, 28, 234 generations with 9.1E3, 1.1E4, 2.4E4, 2.4E5 evaluations respectively. The positive tolerance value δ for problem G03 and G11 is 1.0E-3 and 1.0E-5 respectively.

Table 2 shows the statistical summary of the results for all the tested problems showing best, median, mean and worst solutions obtained with their corresponding standard deviations (SD). Table 3 – Table 5 show the same results compared with

Table 2. Experimental results of ICHEA on 11 benchmark functions

Functions	Best	Median	Mean	Worst	SD
G01	**-15.00000**	-15.00000	**-15.00000**	**-15.00000**	5.4E-07
G02	-0.803036	-0.784636	**-0.768525**	-0.743884	2.3E-02
G03	-1.00497	-1.00483	**-1.00476**	**-1.00483**	1.1E-04
G04	**-30665.539**	-30665.539	-30665.537	-30665.530	3.2E-03
G06	**-6961.814**	-6961.813	**-6961.814**	**-6961.814**	1.85E-05
G07	24.6149	24.9502	25.7139	27.2705	1.0E+00
G08	**-0.095825**	-0.095825	**-0.095825**	**-0.095825**	2.3E-07
G09	680.645	680.742	680.774	680.995	1.1E-01
G10	7128.097	7165.736	7196.508	7297.964	5.8E+01
G11	**0.7500**	0.7500	**0.7500**	**0.7500**	3.2E-05
G12	**-1.00000**	-1.00000	**-1.00000**	**-1.00000**	1.2E-06

Table 3. Comparison of best solutions of ICHEA with five other state-of-the-art algorithms

Functions	ICHEA	CRGA	SAPF	PSO-DE	CDE	SMES
G01	**-15.00000**	-14.9977	-15.000	-15.000000	-15.000000	-15.000
G02	-0.803036	-0.802959	-0.803202	-0.8036145	-0.803619	-0.803601
G03	-1.00497	-0.9997	-1.000	-1.0050100	-0.995413	-1.000
G04	**-30665.539**	-30665.520	-30665.401	-30665.539	-30665.539	-30665.539
G06	**-6961.814**	-6956.251	-6961.046	-6961.8139	-6961.8139	-6961.814
G07	24.6149	24.882	24.838	24.30621	24.30621	24.327
G08	**-0.095825**	-0.095825	-0.095825	-0.095826	-0.095825	-0.095825
G09	680.645	680.726	680.773	680.6301	680.6301	680.632
G10	7128.097	7114.743	7069.981	7049.248	7049.248	7051.903
G11	**0.7500**	0.750	0.749	0.749999	0.7499	0.75
G12	**-1.00000**	-1.000000	-1.000000	-1.000000	-1.000000	-1.000
Top ranked	7/11	3/11	4/11	**11/11**	**9/11**	**11/11**

other algorithms based on best, mean and worst solutions respectively. The results in bold indicate the optimum solutions or one of the best amongst all the algorithms. ICHEA is able to reach global optimum for problems G01, G04, G06, G08, G11 and G12 while problems solutions for G02, G03, G09 is very close to optimum solutions. For problems G10 very good solutions are not observed within the limited generations. This demonstrates the competitiveness of ICHEA with other algorithms.

We have also taken the count of final results that are ranked in top half, achieved by all the algorithms. The last rows of Table 3 – Table 5 shows the count of top ranked final results where PSO-DE, SMES and ICHEA are found to be best 3 out of 6 algorithms for getting good mean and worst solutions and PSO-DE, SMES, CDE and ICHEA are best 4 out of 6 algorithms for reaching towards optimum solution; however, according to "no-free-lunch" theorem no algorithm is the best for all classes of problems [22]. PSO-DE is able to demonstrate very impressive results for benchmark COPs but it is not able to perform well for CSPs as demonstrated in [20] where ICHEA outperforms it in terms of *success rate* and efficiency.

Table 4. Comparison of mean solutions of ICHEA with five other state-of-the-art algorithms

Functions	ICHEA	CRGA	SAPF	PSO-DE	CDE	SMES
G01	**-15.00000**	-14.9850	-14.552	-15.000000	-14.999996	-15.000
G02	**-0.768525**	-0.764494	-0.755798	-0.756678	-0.724886	-0.785238
G03	**-1.00476**	-0.9972	-0.964	-1.0050100	-0.788635	-1.000
G04	-30665.537	-30664.398	-30665.922	-30665.539	-30665.539	-30665.539
G06	**-6961.814**	-6740.288	-6953.061	-6961.8139	-6961.8139	-6961.284
G07	25.7139	25.746	27.328	24.30621	24.30621	24.475
G08	**-0.095825**	-0.095819	-0.095635	-0.0958259	-0.095825	-0.095825
G09	680.774	681.347	681.246	680.6301	680.6301	680.643
G10	7196.508	8785.149	7238.964	7049.248	7049.248	7253.047
G11	**0.7500**	0.752	0.751	0.749999	0.757995	0.75
G12	**-1.00000**	-1.000000	-0.99994	-1.000000	-1.000000	-1.000
Top ranked	**8/11**	2/11	1/11	**10/11**	6/11	**9/11**

Table 5. Comparison of worst solutions of ICHEA with five other state-of-the-art algorithms

Functions	ICHEA	CRGA	SAPF	PSO-DE	CDE	SMES
G01	**-15.00000**	-14.9467	-13.097	-15.000000	-14.999993	-15.000
G02	-0.743884	-0.722109	-0.745712	-0.6367995	-0.590908	-0.751322
G03	**-1.00483**	-0.9931	-0.887	-1.0050100	-0.639920	-1.000
G04	-30665.530	-30660.313	-30656.471	-30665.539	-30665.539	-30665.539
G06	**-6961.814**	-6077.123	-6943.304	-6961.8139	-6961.8139	-6952.482
G07	27.2705	27.381	33.095	24.3062	24.3062	24.843
G08	**-0.095825**	-0.095808	-0.092697	-0.0958259	-0.095825	-0.095825
G09	680.995	682.965	682.081	680.6301	680.6301	680.719
G10	7297.964	10826.09	7489.406	7049.248	7049.249	7638.366
G11	**0.7500**	0.757	0.757	0.750001	0.796455	0.75
G12	**-1.00000**	-1.000000	-0.999548	-1.000000	-1.000000	-1.000
Top ranked	**8/11**	1/11	1/11	**10/11**	7/11	**9/11**

6 Discussion

ICHEA was initially introduced to solve real-valued CSP solutions only where it was able to outperform many other EAs in terms of success rate and efficiency [20]. In this paper ICHEA has been enhanced to solve COPs as well. The comparative test results on benchmark COPs are very promising and competitive with other state-of-the-art algorithms. ICHEA is a problem independent formulation where consistent results have been observed for all the test problems using common parameters.

Introduction of ICHEA in [20] demonstrated that extracting information from constraints can produce very good solutions efficiently. Hence the basic structure of ICHEA has been kept intact while enhancing it to employ constraint optimization

tasks. The current form of ICHEA is still problem independent where addition of parallel processing simultaneously deals with constraint satisfaction and optimization tasks. Intermarriage crossover has been adjusted to search for an optimum solution that still utilizes information from the constraints.

7 Conclusion

ICHEA introduced in [20] has been demonstrated to outperform many well-known EAs including PSO-DE to solve benchmark CSPs. ICHEA has been enhanced in this paper without losing its integrity to solve real-valued COPs which has shown very competitive results. This new ICHEA runs in two parallel processes – one for CSP and another for COP. The CSP process searches feasible regions to make a population of feasible solutions while COP process tries to optimize the solutions using the whole population. The main idea remains the information extraction from constraints that reduces the search space to promising regions only. Currently ICHEA is restricted to solve only real-valued CSP and COP but it has all the potential to be extended to work for discrete constraints problems as it relies on extracting information from constraints. The future work also involves applying ICHEA for dynamic CSPs and COPs.

References

1. Amirjanov, A.: The development of a changing range genetic algorithm. Computer Methods in Applied Mechanics and Engineering 195(19-22), 2495–2508 (2006)
2. Back, T., et al. (eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd. (1997)
3. Brailsford, S.: Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research 119(3), 557–581 (1999)
4. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
5. Craenen, B.G.W.: Solving constraint satisfaction problems with evolutionary algorithms. Phd Dissertation, Vrije Universiteit (2005)
6. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
7. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS 1995, pp. 39–43 (1995)
8. Eiben, A.E.: Evolutionary algorithms and constraint satisfaction: definitions, survey, methodology, and research directions. Presented at the (2001)
9. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416–423. Morgan Kaufmann Publishers Inc., San Francisco (1993)
10. Onwubolu, G.C., Sharma, A.: Particle Swarm Optimization for the assignment of facilities to locations. In: New Optimization Techniques in Engineering. Springer (2004)

11. Liang, J.J., et al.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-parameter Optimization. Nanyang Technological University, Singapore (2006)
12. Liu, H., et al.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Soft Comput., 629–640 (2010)
13. Mezura-montes, E., Coello, C.A.C.: A Survey of Constraint-Handling Techniques Based on Evolutionary Multiobjective Optimization. Departamento de Computación, Evolutionary Computation Group at CINVESTAV (2006)
14. Mezura-Montes, E., Coello, C.A.C.: A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation 9(1), 1–17 (2005)
15. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1), 1–32 (1996)
16. Müller, T.: Constraint-based Timetabling. PhD Dissertation, Charles University (2005)
17. Paredis, J.: Co-evolutionary Constraint Satisfaction. In: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, pp. 46–55. Springer (1994)
18. Becerra, R.L., Coello Coello, C.A.: Cultured differential evolution for constrained optimization. In: Computer Methods in Applied Mechanics and Engineering, vol. 195(33-36), pp. 4303–4322 (2006)
19. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 93–100. Erlbaum Associates Inc. (1985)
20. Sharma, A., Sharma, D.: ICHEA – A Constraint Guided Search for Improving Evolutionary Algorithms. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012, Part I. LNCS, vol. 7663, pp. 269–279. Springer, Heidelberg (2012)
21. Tessema, B., Yen, G.G.: A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization. In: IEEE Congress on Evolutionary Computation, CEC 2006, pp. 246–253. IEEE (2006)
22. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)