ICHEA for Discrete Constraint Satisfaction Problems

Anurag Sharma and Dharmendra Sharma

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

Abstract. Constraint satisfaction problem (CSP) is a subset of optimization problem where at least one solution is sought that satisfies all the given constraints. Presently, evolutionary algorithms (EAs) have become standard optimization techniques for solving unconstrained optimization problems where the problem is formalized for discrete or continuous domains. However, traditional EAs are considered 'blind' to constraint as they do not extract and exploit information from the constraints. A variation of EA – intelligent constraint handling for EA (ICHEA) proposed earlier models constraints to guide the evolutionary search to get improved and efficient solutions for continuous CSPs. As many real world CSPs have constraints defined in the form of discrete functions, this paper serves as an extension to ICHEA that reports its applicability for solving discrete CSPs. The experiment has been carried on a classic discrete CSP – the N-Queens problem. The experimental results show that extracting information from constraints and exploiting it in the evolutionary search makes the search more efficient. This provision is a problem independent formulation in ICHEA.

Keywords: Constraints, constraint satisfaction problem (CSP), optimization problem, evolutionary algorithm (EA), intelligent constraint handling evolutionary algorithm (ICHEA), N-Queens problem.

1 Introduction

Many engineering problems ranging from resource allocation and scheduling to fault diagnosis and design involve constraint satisfaction as an essential component that require finding solutions to satisfy a set of constraints over real numbers or discrete representation of constraints [4, 5, 21]. EAs are used to solve optimization problem from 1960s. It produces very efficient and robust solutions for unconstrained optimization problems. Even though CSPs are integral part of computer science, little research have been reported on the development of efficient and effective constraint-handling techniques – as compared with a plethora of new methods developed for unconstrained optimization [12]. Traditional EAs are 'blind' towards CSPs as they do not take into account the information from constraints which can reduce the search space; but only heuristically search for the solution in the vast search space. Generally objective functions are designed to use problem dependent penalty functions but some uses error measurements like distance from constraint regions that is applicable to

M. Thielscher and D. Zhang (Eds.): AI 2012, LNCS 7691, pp. 242–253, 2012.

continuous CSPs only. This has been a motivating factor in developing a novel variation of EAs that can extract and exploit information from constraints to produce more efficient solutions for CSPs irrespective of their problem domains. Intelligent constraint handling for EA (ICHEA) has been introduced in [24] to solve continuous CSP that shows promising results when information from constraints are extracted and exploited. In this paper ICHEA is enhanced to solve discrete CSP. Constraint problems are divided into two classes: Constrained Optimizing Problems (COPs) and constraint satisfaction problems (CSPs). The difference between these classes is that in COPs an optimal solution that satisfies all constraints should be found, while for CSPs any solution as long as all the constraints are satisfied is acceptable [9]. This paper has been confined to CSPs only.

Characteristically, CSPs 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 [3]. However, its 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 [16]. 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 [3]. In multi-objective optimization (MOO) constraints are transformed into multiple objectives. Pareto-based selection approaches are currently the most popular multi-objective evolutionary algorithm (MOEA) solution technique. In a typical MOO problem there exists a set of solutions which are superior to the rest of the solution in the search space when all objectives are considered but are inferior to other solutions in the space in one or more objectives. These solutions are known as *pareto-optimal* solutions or non-dominated solutions [25]. (Definition of *pareto* concepts can be found in [26]). There are many established MOO algorithms like MOGA [10], VEGA [20], NSGA and NSGAII [6]. Generally, this type of algorithm requires inequality constraints that can be transformed into many objective functions to be optimized simultaneously. Paredis in [18] 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 main focus of this paper is to enhance ICHEA to solve CSPs for discrete domains by exploiting information from constraints without making the algorithm problem dependent. The paper is organized as follows: Section 2 briefly discusses the EA techniques used in handling constraints and formalization of discrete CSPs. Section 3 describes changes made in ICHEA to make it compatible for discrete CSPs. Section 4 shows experimental results on N-Queens problems followed by discussion on Section 5. Section 6 concludes the paper by summarizing the results and proposing some further extensions to the research.

2 Constraint Handling through EAs

Traditional EAs are 'blind' to constraint as they do not extract the information from the constraints but search the solution through random heuristic greedy approach [4, 9]. This causes the search engine to spent extra computational effort in searching for the solution into the wider search space without only concentrating in the restricted smaller feasible search space. Constraints can reduce the search space and it can make the heuristic search more efficient by harnessing information from constraint to guide the search engine, search in feasible search space only.

Generally violation count is used as a fitness function for CSPs for discrete domains. Depending on the strengths of constraints, individual weights can be assigned to constraints in a penalty function to calculate the fitness value. To avoid problem dependent penalty functions and to utilize some information from constraints to guide the evolutionary search a distance function is used instead of violation count to indicate how far an individual is from the feasible regions [17]. However this is generally limited to continuous domain only. The main motivation behind developing a novel variation of EA is to avoid using problem dependent penalty based functions for CSPs that can be used for both continuous and discrete domains utilizing only the information from constraints.

ICHEA attempts to solve CSPs by utilizing information from constraints to guide the evolutionary search whether the constraints are given in the form of continuous or discrete functions. It does not use penalty functions, problem dependent formulations or error functions which all of the current EAs do. It does not disregard the information from constraints to produce more efficient results. ICHEA also does not require initial feasible solution and it is also not restricted to produce feasible progenies from feasible parents.

CSP is defined by an input vector $\vec{x} = \{x_1, x_2, \ldots x_n\}$ of size n in a finite space S where each variable x_i has a finite domain D_i. A set of m constraints $\{c_1, c_2, \ldots c_m\}$ is defined in the form of functions:

$$c_i(x_1, x_2, \ldots x_n) = \begin{cases} 1 & ,if\ satisfied \\ 0 & ,if\ violated \end{cases} \tag{1}$$

Constraint satisfaction sets or feasible regions $\{S_1, S_2, .. S_m\}$ can also be defined where:

$$S_i = \{\, \vec{x} \in S \mid c_i(\vec{x}) = 1, 1 \leq i \leq m, i \in Z(integer\ set)\} \tag{2}$$

The fitness function of any CSP can be given as:

$$f(\vec{x}) = \sum_{i=1}^{m} c_i(\vec{x}) \tag{3}$$

To incorporate the weighted penalty function Eq. (3) can be redefined as:

$$f(\vec{x}) = \sum_{i=1}^{m} w_i c_i(\vec{x}) \tag{4}$$

where $w_i \geq 0$ are the weighted coefficients representing the relative importance of the constraints. Its main weakness is the difficulty to determine the appropriate weights when there is not enough information about the problem [2]. The solution of a CSP is $s \in S$ when all the constraints c_i are satisfied.

3 ICHEA for Discrete Data

ICHEA is a variation of EA that solves CSPs by extracting information from constraints as described in [24] for continuous domain by realizing *intermarriage* crossover. *Intermarriage* crossover selects two parents from different *constraint satisfaction sets* to make them come closer iteratively towards their corresponding feasible boundary because the CSP solutions lie in the overlapping boundary region of feasible regions that satisfy different constraints. The iterative move for parent P_i and P_j to produce offspring O_i is given as:

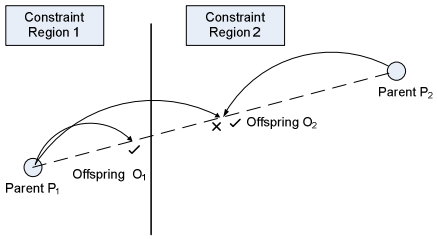

Fig. 1. Intermarriage crossover for continuous CSPs

$$O_i = r^i(P_j - P_i) \tag{5}$$

where r is a coefficient in the range $(0,1)$ which is generally 0.5. Variable i gets incremented from 1 to a threshold value T in the sequence $\langle 1, 2, \dots, T \rangle$. The intermarriage crossover process is shown in the Fig. 1 where ✓ mark indicates possible placement for an offspring and × mark indicate the offspring vector is unacceptable in that particular position. An offspring is accepted if it satisfies equal or more constraints than its corresponding parent. Corresponding parent for offspring O_1 is P_1. So using the Eq. (5) the next i value is used until the offspring finds an acceptable place or a threshold value T is reached. This *greedy* approach of crossover might result in generating no offspring at all.

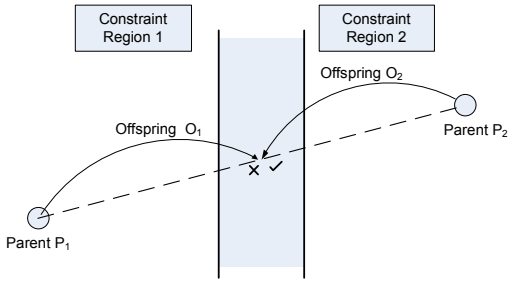

Fig. 2. Intermarriage crossover for discrete CSPs

Favouring individuals that satisfy higher number of constraints and the use of feasible regions in *intermarriage* crossover guides the evolutionary search in finding the solution space quickly [24]. When constraint regions are discrete then the *intermarriage* crossover for continuous CSP cannot be used directly to generate progenies as its formulation is based on real numbers for continuous domain. The concept of *intermarriage* crossover is to fuse feasible solutions from two different constraint satisfaction sets together that makes the offspring "generic" that satisfy more constraints because its parents are from two different constraint satisfaction sets. If the fusion of two discrete feasible solutions is represented by $\oplus$ then the *intermarriage* crossover of two parents for discrete CSP transformed from Eq. (5) can be given as:

$$O_i = r^i\big(P_j \oplus P_i\big) = (P_j \oplus P_i) \tag{6}$$

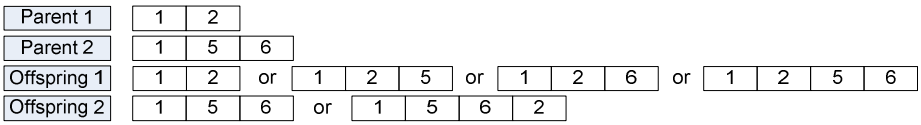

Fig. 3. Variable length intermarriage crossover

For discrete *intermarriage* crossover value of r and i is 1 because fusion is non-iterative as shown in Fig. 2 where offspring are accepted if fusion results better chromosome(s). ICHEA uses variable length chromosomes (partial solutions) to accomplish discrete valued *intermarriage* crossover where genotype is used as phenotypes. Variable length chromosome has been used in many applications like [1, 23, 27]. Partial solutions $\vec{p} = \{x_1, x_2, \ldots x_k\}$ where $k \leq n$ are chromosomes that satisfy all the constraints partially i.e. $\sum_{i=1}^{m} c_i(\vec{p}) = m$. Its fitness can be given as:

$$fitness(\vec{p}) = n - |\vec{p}| \tag{7}$$

The partial solutions are fused incrementally considering all constraints at once. For example a CSP problem of size n has parents P_1 and P_2 with partial solutions $\{1, 2\}$ and $\{1, 5, 6\}$ respectively. The generated offspring from these parents either satisfy equal or more constraints as shown in Fig. 3. Each offspring has traits from both parents. The *intermarriage* crossover only tries to append the allele values of other chromosome as shown in Fig. 3. All the allele values that violate the constraints are dropped so the offspring are also feasible chromosomes. An advantage of using variable length chromosome in this manner is reduction in computational time. *Intermarriage* crossover avoids recalculation of objective function because it only requires allele values to be appended. For example an N-Queens problem on chess board of size N requires $N(N+1)/2$ operations on a single function call and for one complete crossover it requires $N(N+1)$ operations every time. Its time complexity of *Big-O* order is $O(N^2)$. On the other hand, the *intermarriage* crossover only checks the violation of appended allele value with all other existing feasible values that requires $(l_1 + l_2 + N) + 2l'_1 l'_2$ operations where l_1 and l_2 are the lengths of partial solutions of the parents and l'_1 and l'_2 are length of their non-duplicate allele values. The first expression of time complexity $(l_1 + l_2 + N)$ indicates number of operations required to find the duplicate values. The second expression $2l'_1 l'_2$ indicates the operations required to append the non-duplicate allele values to each other parents. The best time complexity is $2 + N + 2 = N + 4$ operations when lengths of both parents are 1 and the worst time complexity is $(N/2 + N/2 + N) + 2(N/2)(N/2) = 0.5N^2 + 2N$. It is observed that ICHEA's partial solutions quickly attain the size of $N - 1$ or close to it. So two parents with chromosomes of length $N - 1$ is taken for average time complexity which can be calculated as $((N-1) + (N-1) + N) + 2(1 \times 1) = 3N$. Hence the average time complexity has the *Big-O* order of *O(N).* The algorithmic description of ICHEA is given in Appendix (A).

4 Experiments

The motivation behind this research is to show the information extraction and exploitation from constraints can produce the evolutionary solutions more efficiently. We used a toy problem namely N-Queens problem that serves as a classic CSP. Basically, the N-Queens problem can be expressed as placing N queens on N x N chessboard such that no queen is attacked by one another [13]. The first part of the experiment tries to solve N-Queens problem without using any sort of constraints related information/heuristics from the problem. The second part of the experiment does the preprocessing of the chromosomes to work on unique allele values only because same allele value refers to the queens that are in the same row which is a violation of one of the constraints. The idea is to provide as much information about the constraints as possible to the evolutionary search.

4.1 No Exploitation of Information from the Problem

For this test case we compared ICHEA with Differential Evolution (DE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), standard Genetic Algorithms (GAs) and Non-dominated Sorting GA-II (NSGA-II) [6] that do not use any sort of information from the problem domain. GA is taken from Genetic Algorithms toolbox

Table 1. Comparative test results on no problem specific information extraction

N	**CMA-ES** [25]	**DE** [25]	GA	NSGA II	ICHEA
4	456 NFC (SR = 1.00)	134 NFC (SR = 1.00)	367 NFC (SR = 1.00)	93 NFC (SR= 1.00)	39 NFC (SR = 1.00)
5	656 NFC (SR = 1.00)	254 NFC (SR = 1.00)	750 NFC (SR = 1.00)	217 NFC (SR = 1.00)	37 NFC (SR = 1.00)
6	22,013 NFC (SR = 1.00)	1,11,136 NFC (SR = 0.65)	30,086 NFC (SR = 0.75)	694 NFC (SR = 1.00)	51 NFC (SR = 1.00)
7	9,964 NFC (SR = 1.00)	24,338 NFC (SR = 0.95)	1,400 NFC (SR = 1.00)	2631 NFC (SR = 1.00)	34 NFC (SR = 1.00)
8	84,962 NFC (SR = 1.00)	7,576 NFC (SR = 0.75)	3,786 NFC (SR = 0.80)	1273 NFC (SR = 1.00)	41 NFC (SR = 1.00)
9	133,628 NFC (SR = 1.00)	19,296 NFC (SR = 0.50)	18,333 NFC (SR = 0.80)	27,852 NFC (SR = 1.00)	72 NFC (SR = 1.00)
10	263,572 NFC (SR = 0.95)	286,208 NFC (SR = 0.30)	3,300 NFC (SR = 0.30)	1,737 NFC (SR = 1.00)	83 NFC (SR = 1.00)
11	284,382 NFC (SR = 0.95)	68,255 NFC (SR = 0.10)	15,550 NFC (SR = 0.40)	SR = 0.00	132 NFC (SR = 1.00)
12	295,740 NFC (SR = 0.75)	99,120 NFC (SR = 0.25)	23,000 NFC (SR = 0.70)	SR = 0.00	122 NFC (SR = 1.00)
13	376,631 NFC (SR = 0.85)	95,485 NFC (SR = 0.15)	3,400 NFC (SR = 0.10)	SR = 0.00	293 NFC (SR = 1.00)
14	450,654 NFC (SR = 0.85)	160,475 NFC (SR = 0.10)	47,350 NFC (SR = 0.40)	SR = 0.00	308 NFC (SR = 1.00)
15	627,391 NFC (SR = 0.50)	223,425 NFC (SR = 0.10)	95,625 NFC (SR = 0.40)	SR = 0.00	381 NFC (SR = 1.00)

Revision: 1.1.4.2, 2004 available in Matlab 7.0.1 and NSGA II written in C language is taken from [7]. ICHEA has been developed in Java language. The test results for DE and CMA-ES have been taken from [19] where 20 trials for each problem have been taken into account. The test results are based on number of function calls (NFC) and success rate (SR). If the NFC $\geq 2\text{x}10^6$ then it is considered that the solution is not found. The experimental set up is discussed below.

- **Fitness Function:** the fitness function is the total violation count and the chromosomes are ranked based on this fitness function. The solution for CSP is to find at least one chromosome with no violation i.e. $fitness(\vec{x}) = 0$.
- **Allele Values:** DE, CMA-ES and GA generate real numbers for allele values but in case of N-queens problem the real numbers are converted into integer values by taking the round off value to calculate the fitness. NSGA II uses binary string representation and ICHEA uses integer values. Candidates can have duplicate allele values.
- **Efficiency Measures:** NFC and SR are used to compare the performance of different algorithms. NFC is simply the total count of objective function invoked by the algorithm. SR is the rate of successful trials for each problem i.e. $SR = successful\ trials/total\ trials$.
- **Parameters:** for all the algorithms population size of 100 is used. Scattered crossover is used for GAs. Mutation rate of 0.1 is used for ICHEA and GAs. All default parametric values are used for NSGA-II.

Table 1 shows the comparative results based on NFC and SR to solve N-queens problem. N denotes the size of the chessboard. It can be observed as the problem size increases the solution quality decreases for all the algorithms except ICHEA. The outcome of the test results clearly shows that ICHEA produces consistent results and dominates other EAs. ICHEA is the most efficient algorithm by getting the lowest NFC and highest success rate (SR = 1.00) for all the problems. GA shows unpredictable results where it usually finds the solution in very few evaluations but if it is stuck in local minima then it is generally not able to find the optimum solution.

Table 2. Comparative test results on after information extraction from the problem

N	SA [19]	TS [19]	GA [19]	GA [10]	PSO [14]	ICHEA (best)	ICHEA (median)	ICHEA (mean)
8	493	182	400	**100**	-	84	119	115
10	948	472	4910	266	-	97	162	**176**
20	-	-	-	2000	5669.7	279	698	**898**
30	2160	4655	91790	2300	-	301	538	**970**
50	2849	22663	1759230	5660	14991.4	729	1190	**2257**
75	6091	81030	571170	6300	-	380	2568	**3393**
100	7873	206910	887770	15600	36199.4	1977	3702	**7595**
200	21708	2399940	2287960	460475	934399	7360	14533	**15489**
300	**24636**	9382620	2774820	-	-	6767	34043	37730

4.2 Information Extraction and Exploitation

The second test case involves utilization of information extraction and exploitation from the N-Queens problem in evolutionary search. Here problem specific chromosomes have been used where only unique integer values are taken into account for chromosomes' allele values. Unique integers ensure that queens are at least in different rows which satisfy a part of constraint for this problem. All the parameter remains same as of the previous experiment. We used Simulated Annealing (SA), Tabu Search (TS), Particle Swarm Optimization (PSO) and GA along with ICHEA for the experiment. The test results of SA, TS and GA is taken from [15] and test results for PSO is taken from [11]. ICHEA does not need to be modified as it has problem independent formulation for its *intermarriage* crossover. Appended allele values are not necessarily unique.

Table 2 shows the comparative test results based on NFC only when some problem specific information has been extracted from the problem. The best, median and mean results for ICHEA have also been shown. There is no changes done in ICHEA and it still performs best in most of the problems (shown in bold). The test results obtained by [8] is also impressive where the authors uses partially matched crossover (PMX) and an unusual selection process where only top two candidates are selected for mating in each generation and rest of the population is replaced by making duplicates of this pair.

5 Discussion

The test results show that EAs can be significantly improved if the chromosomes are designed to be problem specific. The experiment in Section 4.2 shows if only the unique integer value is taken into account for allele values then the solution is converged much earlier for N-Queens problem. Considerable improvement has been seen in GA. The objective here is not to get the best results for N-Queens problem but to show how intelligent an algorithm is. The results in Section 4.1 shows that tested optimization algorithm (DE, CMA-ES, GA and NSGA II) blindly searches for the optimum solution through greedy heuristic search manner without extracting the information from constraints while ICHEA utilizes the information from constraints through its *intermarriage* crossover operator and gets the best results. Test results in Section 4.2 again favor ICHEA. The advantage of ICHEA is that its formulation is problem independent which still extracts enough information from the constraint to solve the problems efficiently. It can be argued that ICHEA also uses problem dependent integer values for N-Queens problem. The novel formulation of ICHEA does not require the generated integer values to be unique. ICHEA works with allele coupling only. So only the definition of constraints and the rules for coupling of two allele values to partially satisfy the constraints need to be provided. It only maintains the population of feasible solutions that drastically reduces the size of the search space.

6 Conclusion

This paper has modeled ICHEA to handle discrete CSPs. It has been demonstrated through N-Queens problem that it outperforms other EAs because it makes use of information from the problems and constraints. The search mechanism of ICHEA is guided by constraints where it concentrates in the feasible regions of constraint satisfaction sets to get the solution without putting extra computational effort in searching through the whole search space. N-Queens is a toy problem that does not have complex constraints structure as in some real world problems like university time tabling, vehicle routing etc. Future work consists of modeling ICHEA to provide problem independent solution for problems that have different constraint strengths. ICHEA will be further tested on mixed CSP where problem domain constitutes both continuous and discrete constraints.

Acknowledgment. We would like to thank Dr. Cecil Schmidt of Washburn University, Topeka for providing the code for GA to solve N-Queens problem from his work in [8].

References

1. Bandyopadhyay, S., Pal, S.K.: Pixel classification using variable string genetic algorithms with chromosome differentiation. IEEE Transactions on Geoscience and Remote Sensing 39(2), 303–308 (2001)
2. Coello, C.A.C.: A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. Knowledge and Information Systems 1, 269–308 (1998)
3. 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)
4. Craenen, B.G.W., et al.: Comparing evolutionary algorithms on binary constraint satisfaction problems. IEEE Transactions on Evolutionary Computation 7(5), 424–444 (2003)
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. Deb, K.: Kanpur Genetic Algorithms Laboratory, `http://www.iitk.ac.in/kangal/codes.shtml`
8. Eastridge, R., Schmidt, C.: Solving n-queens with a genetic algorithm and its usefulness in a computational intelligence course. J. Comput. Sci. Coll. 23(4), 223–230 (2008)
9. Eiben, A.E., et al.: Solving Binary Constraint Satisfaction Problems Using Evolutionary Algorithms with an Adaptive Fitness Function. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 201–210. Springer, Heidelberg (1998)
10. 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)

11. Hu, X., et al.: Swarm intelligence for permutation optimization: a case study of n-queens problem. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS 2003, pp. 243–246 (2003)
12. Kramer, O.: A Review of Constraint-Handling Techniques for Evolution Strategies. In: Applied Computational Intelligence and Soft Computing 2010, pp. 1–11 (2010)
13. Letavec, R.: The Queens Problem - Delta. ITE 2(3), 101–103 (2002)
14. Liu, H., et al.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Soft Comput., 629–640 (2010)
15. Martinjak, I., Golub, M.: Comparison of Heuristic Algorithms for the N-Queen Problem. In: 29th International Conference on Information Technology Interfaces, ITI 2007, pp. 759–764 (2007)
16. 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)
17. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1), 1–32 (1996)
18. Paredis, J.: Co-evolutionary Constraint Satisfaction. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 46–55. Springer, Heidelberg (1994)
19. Rahnamayan, S., Dieras, P.: Efficiency competition on N-queen problem: DE vs. CMA-ES. In: Canadian Conference on Electrical and Computer Engineering, CCECE 2008, pp. 000033–000036. IEEE (2008)
20. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 93–100. L. Erlbaum Associates Inc. (1985)
21. Shang, Y., Fromherz, M.P.J.: Experimental complexity analysis of continuous constraint satisfaction problems. Information Sciences 153, 1–36 (2003)
22. Sharma, A.: A new optimizing algorithm using reincarnation concept. In: 2010 11th IEEE International Symposium on Computational Intelligence and Informatics (CINTI), pp. 281–288. Springer, Heidelberg (2010)
23. Sharma, A., Omlin, C.W.: Performance Comparison of Particle Swarm Optimization with Traditional Clustering Algorithms used in Self-Organizing Map. International Journal of Computational Intelligence 5(1), 1–12 (2009)
24. 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)
25. Srinivas, N., Deb, K.: Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2, 221–248 (1994)
26. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. Evolutionary Computation 8, 125–147 (2000)
27. De Weck, O., Kim, I.Y.: Variable Chromosome Length Genetic Algorithm for Structural Topology Design Optimization. Strain. In: AIAA 2004, pp. 1–12 (April 2004)

Appendix

A) ICHEA Algorithm

ICHEA is another variation of EA that has been introduced in [24] adds constraint handling features for continuous CSPs to the standard GAs. Because of space limitations we are only describing the changes in ICHEA accommodated to make it compatible for discrete CSPs. The pseudocode can be given as:

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

The description of changed subroutines is given below:

InitializeChromosomes: The population of chromosomes is generated using sequence of integer values $\langle 1, 2, \ldots, |pop| \rangle$ with the modulus operator (mod) as shown below:

$$\vec{x}_i = mod(counter, |pop|) + 1 \tag{8}$$

where $counter$ is initialized with 0 that is always incremented by 1 for each chromosome and $|pop|$ is the population size. The length of initialized chromosome is 1.

TournamentSelection: Here novelty selection is incorporated along with fitness based selection as described in [24] but the selection of chromosomes is based on the following order of preference:

1. Its fitness in the search space
2. If fitness is same then higher novelty value
3. If novelty is same then a chromosome is picked randomly

InterMarriageCrossover: The crossover techniques have been described in Section 3.

Mutation: ICHEA uses *swap* mutation for permutations.

SortAndReplace: According to [14] the lower the individuals' degree of constraint violation, the higher the probability that it clusters together around the current best solution and individuals with lower degrees of constraint violations are very difficult to jump out of current best individual's adjacent region. This may cause the current best individual to stay on the same position for a long time leading to loss of diversity in the population. To avoid this scenario the ICHEA keeps the fair share of all levels of fitness in the population. If the population pop of size $|pop|$ has m constraints in the problem of size n then the whole population is divided into equal sized $\lfloor \sigma n \rfloor$ slots where σ is in the range of (0.1 1.0). σ = 0.1 is used in the experiments. Slot i

is allocated to individuals based on its fitness value $fitness(\vec{p}) = n - |\vec{p}|$ from Eq. (7) where $i = \lfloor \sigma(n - |\vec{p}|) \rfloor + 1$. If slot i remains empty then its allocated space is evenly distributed to other slots. Let pop_i indicate the population of individuals that belong to slot i so the total population is:

$$pop = \sum_{i=1}^{\lfloor \sigma n \rfloor} pop_i$$

Then pop_i is sorted according to the fitness and the best $|pop|/\lfloor \sigma n \rfloor$ is selected for subpopulation pop_i.

$$\therefore \max(|pop_i|) = |pop|/\lfloor \sigma n \rfloor$$

If after allocation, k slots have $|pop_i| < |pop|/\lfloor \sigma n \rfloor$, then unallocated population of individuals $pop_{unalloc}$ is:

$$pop_{unalloc} = \sum_{i=1}^{m} \begin{cases} |pop|/\lfloor \sigma n \rfloor - |pop_i|, & if\ |pop_i| < |pop|/\lfloor \sigma n \rfloor \\ 0, & otherwise \end{cases}$$

This unallocated population $pop_{unalloc}$ needs to be allocated evenly in the slots that have $|pop_i| > |pop|/\lfloor \sigma n \rfloor$.

Once the chromosomes are sorted a *random death* concept is used defined in [22] to delete some predefined number of chromosomes randomly from the population. The certain top percentage of the population is spared to control the search focus.

CheckTerminationCriteria: The iteration is stopped when:

1. The maximum number of generation is reached or
2. The CSP solution is found or
3. The process is stalled by no improvement in the solution for some generations.