An Incremental Approach to Solving Dynamic 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 problems (CSPs) underpin many science and engineering applications. Recently introduced *intelligent constraint handling evolutionary algorithm* (ICHEA) in [14] has demonstrated strong potential in solving them through evolutionary algorithms (EAs). ICHEA outperforms many other evolutionary algorithms to solve CSPs with respect to success rate (SR) and efficiency. This paper is an enhancement of ICHEA to improve its efficiency and SR further by an enhancement of the algorithm to deal with local optima obstacles. The enhancement also includes a capability to handle dynamically introduced constraints without restarting the whole algorithm that uses the knowledge from already solved constraints using an incremental approach. Experiments on benchmark CSPs adapted as dynamic CSPs has shown very promising results.

Keywords: Constraint satisfaction problem (CSP), intelligent constraint handling evolutionary algorithm (ICHEA), evolutionary algorithm (EA), local optima, dynamic constraints, incremental approach.

1 Introduction

CSPs are at the core of many real-world applications, including control, and diagnosis of physical systems such as car, planes, and factories. It is also used in modern robotic systems such as control of modular, hyper-redundant robots, which are robots with many more degrees of freedom than required for typical tasks. Sometimes the environment of the CSPs changes along with time as in obstacle avoidance, vehicle routing and reusing previously generated university timetable. Even though CSPs are an important area of research in part of computer science, little has been reported on the development of efficient and effective constraint-handling techniques – relative to the development of new methods for unconstrained optimization using EAs [7]. Recently introduced ICHEA [14] is able to solve real-valued CSPs efficiently with relatively higher success rate (SR) than other tested well known EAs. The strength of ICHEA is that it makes use of knowledge from constraints rather than blindly search in the solution space as done by traditional EAs [2]. ICHEA has been demonstrated to outperform other well regarded EAs like NSGA II [3] and PSO-DE [10]. However it also exhibited drawbacks in solving hard CSPs where it was computationally

T. Huang et al. (Eds.): ICONIP 2012, Part III, LNCS 7665, pp. 445–455, 2012.

expensive as its median solutions to benchmark CSPs generally required more than 200.0 seconds of CPU time to solve on a common standalone machine [14]. Its SR was also very low in the range of 0.0-0.6 for hard problems. Hence we have introduced some new strategies to improve ICHEA to address these drawbacks.

Moreover, ICHEA is only able to handle static CSPs in its current state so we propose an enhancement to the ICHEA to realize *incrementality* in constraint solving. Using this approach the dynamic behavior of CSPs can be handled efficiently as it is quintessential for real-time dynamic CSPs (DCSP) where only little research has been reported using EAs. Furthermore, there are not any established benchmark problems for them. Some benchmark problems are compiled in [12] and [6] but they are used for dynamic COPs and dynamic optimization problems respectively. The difference between COPs and CSPs is that in first an optimal solution that satisfies all the constraints should be found, while in second any solution as long as all the constraints are satisfied is acceptable [4]. Because of the unavailability of benchmark DCSPs we have transformed some existing benchmarks CSPs from [15] to DCSPs.

The main contribution of this paper is to enhance the performance of existing ICHEA (called ICHEA+) to solve CSPs and introduce an incremental approach to solve real-valued DCSPs using ICHEA. The paper is organized as follows: Section 2 describes the formalization of CSPs and DCSPs. Section 3 briefly discusses EA techniques used to handle dynamic behavior of CSPs (called I-ICHEA) and the available benchmark problems. Section 4 describes the enhancement of ICHEA with some new strategies to overcome getting local optimal solutions. Section 5 shows experimental results with discussions in Section 6 about the outcome. Section 7 concludes the paper by summarizing the results and proposing some further possible extensions to the research.

2 Formalization of CSPs and DCSPs

A CSP is defined by an n dimensional input vector $X = \{x_1, x_2, \ldots x_n\}$ in a finite space S where each variable x_i has a finite domain D_i. A set of m constraints $\{c_1, c_2, \ldots c_m\}$ are 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 $\{S_1, S_2, \ldots S_m\}$ can also be defined where:

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

where Z is the set of integers. The solution of a CSP is $s \in S$ when all the constraints c_i are satisfied, which can be given as:

$$\sum_{i=1}^{m} c_i(s) = m \tag{3}$$

For real-valued CSPs numerical constraints can be given in two forms – equality and inequality functions [3, 10, 16]:

$$g_i(X) \geq 0 \qquad i = 1, \ldots, k \tag{4}$$

$$h_j(X) = 0 \qquad j = k+1, \ldots, m \tag{5}$$

The equality constraints cannot be solved directly using EAs so they are converted to inequality constraints by introducing a positive tolerance value δ.

$$g_j(X) = \delta - \left|h_j(X)\right| \geq 0 \tag{6}$$

Generally violation count is used as a fitness function for any CSPs. Depending on the strengths of constraints, individual weights is assigned to constraints in a *penalty* function to calculate the fitness value. To avoid using problem dependent *penalty* functions and utilizing some knowledge from constraints to guide the evolutionary search many EAs do not use violation count but use a distance function to indicate how far an individual is from the feasible regions [11]. It transforms the inequality constraint functions to a fitness function to rank individual members in the population generated by ICHEA. This fitness function tries to bring the individuals closer to the feasible space using the following functions for $\forall i : \{1 \leq i \leq m\}$:

$$fitness_i(X) = \begin{cases} \mathrm{g}_i(X), & if\ g_i(X) < 0 \\ 0, & if\ g_i(X) \geq 0 \end{cases} \tag{7}$$

$$e = \sum_{i=1}^{m} \left|fitness_i(X)\right| \tag{8}$$

The fitness function $fitness_i$ is a measurement of *euclidean* distance of vector X from the nearest point of the feasible region where constraint c_i is satisfied. The error function e is the summation of all the fitness functions. The objective is to minimize the error value e where the solution to a CSP is found when $e = 0$.

For DCSPs the total number of constraints m is not know a priori and the solution has to be produced based on constraints that come to hand. A DCSP can be defined as a sequence of static CSPs where each one differs from the previous one by the addition or removal of some constraints. It is indeed easy to see that all the possible changes to a CSP (constraint or domain modifications, variable additions or removals) can be expressed in terms of constraint additions or removals [17]. The same fitness function given in Eq. (7) and Eq. (8) are used for DCSPs. To solve such a sequence of CSPs, it is always possible to solve each constraint from scratch as it has been done for the first one but this naive method, which remembers nothing from the previous reasoning, has two significant drawbacks [17]:

- **Inefficiency:** which may be unacceptable in the framework of real time applications (planning, scheduling etc), where the time allowed for re-planning is limited.
- **Instability:** of the successive solutions, which may be unpleasant in the framework of an interactive design or a planning activity, if some work has been started on the basis of the previous solution.

A DCSP is a sequence of static CSPs that are formed by constraint changes. The notion of DCSP has been introduced to represent such situations by [13]. Some attempt has been made to solve DCSP using EA as [5] uses Multi-objective optimization (MOO) to transform changes in constraints as a new objective function with changes in so called *disruption* function. This function is used to estimate the effect of changing an initial constraint to a new one. The changes are reflected in *pareto set* and program runs again to get the new *pareto* optimal set guided by previous *pareto* front. 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. A local search is another approach to reuse previous solutions for DCSP. The previous solution can simply be used as a starting assignment for the new local search repair-based algorithm. DCSP features employing the previous related CSP to find a minimal change solution to the current CSP but it can be computationally challenging [9, 17].

3 Benchmark Problems

As mentioned above there is little research reported on real-valued DCSPs nor there is any benchmark problems available for it. There are some benchmark problems for dynamic optimization problems in [8] and [6] that are without constraints. Some recently developed EAs have performed well on these benchmark problems like self-adaptive differential evolution algorithm (jDE) [1], dynamic hybrid particle swarm optimization (DHPSO) [6] and triggered memory based PSO (TMPSO) [18]. Nguen and Yao in [12] has introduced some benchmark problems for real-valued dynamic COPs and a novel algorithm repair genetic algorithm (RepairGA) to solve these problems efficiently; however, none of benchmarks are for DCSPs. We took some benchmark CSPs from coconut benchmark [15] and converted them to DCSP by taking one constraint at a time that is solved as a sequence of static constraints. In this paper only addition of constraints are considered to make a dynamic environment. Update of constraints or redefinition of feasible regions has not been considered. A new constraint is added into the environment in every 100 generations or else if all the current constraints are satisfied – whichever is earlier.

4 Enhancement to ICHEA

ICHEA is a variation of EA that uses its own crossover operator namely *intermarriage* crossover that selects two parents from different *constraint satisfaction sets* S_i 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. Favoring individuals that satisfy higher number of constraints and the use of feasible regions for *intermarriage* crossover guide the evolutionary search in finding the solution space quickly [14]. This guiding process has helped ICHEA to outperform other well-known EAs to solve CSPs where constraint strengths are very high i.e. the feasible regions are very small compared to the whole search space. Calculation for constraint strengths has been shown in Section 5.

As mentioned in Section 1, even after ICHEA's success in outperforming other well-known EAs to solve CSP, it is still computationally expensive as its median solutions for some benchmark problems generally require more than 200.0 seconds of a CPU time on a common machine to produce a solution [14]. Its SR is also very low in the range of 0.0-0.6 for some hard benchmark problems. Hence we propose the following enhancement that improves its performance in terms of efficiency and SR. The enhanced ICHEA is called ICHEA+ (ICHEA-plus) where the improvement

observed has been as high as 68 times over the previous ICHEA on benchmark problems. ICHEA+ is even able to produce efficient solutions with high SR of up to 1.00 on low positive tolerance value ($\delta = 10^{-3}$) on hard CSP problems where previous ICHEA had low success with SR = 0.00.

4.1 Diversity Management

According to [10] 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 degrees of constraint violating individuals in the population. If the population pop of size $|pop|$ has m constraints in the problem then the whole population is divided into equal sized m slots where slot i is allocated to individuals that violate i constraints. If there are no individuals with i violations then its allocated space is evenly distributed to other slots. Let pop_i indicate the population of individuals that violate i constraints so the total population is:

$$pop = \sum_{i=1}^{m} pop_i$$

Then pop_i is sorted according to the fitness and the best $|pop|/m$ is selected for subpopulation pop_i.

$$\therefore \max(|pop_i|) = |pop|/m$$

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

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

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

4.2 Stalled Local Optimal Solutions Management

The above diversity management is not sufficient to avoid the population getting stuck into local optimal solution for hard CSPs. This is a common problem for EAs when the whole population gets stuck around local optimal solution and lose its diversity. We introduce the concept of *forced constraint violations* to tackle this issue. This works like *tabu* search algorithm where the individuals try to move away from the forcibly introduced new infeasible regions (*tabu* regions). If the population is stagnant for certain number of generations then the current best solution is considered as local optimal solution where some region around it is marked as a new infeasible region to move the population away from it. This region is defined as a hyper-sphere whose centre is the location of the current best (local optimal) solution with the radius

defined as distance from the location of current best individual with the location of the worst individual that has the same degree of violations as the current best individual. If the current best individual belongs to a subpopulation pop_i which is sorted according to the fitness from best to worst where an individual can be described as $X_j \in \{pop_i | 1 \leq j \leq |pop_i|\}$ has best individual X_1 and worst individual $X_{|pop_i|}$. The radius R of the hyper-sphere can be computed as: $R = \left|X_1 - X_{|pop_i|}\right|$ and hence the new forced dynamic constraint is: $g_{m+1}(X) = \sum_{i=1}^{n}(x_i - \mu_i)^2 > R^2$ where $\mu_i \in X_1$ and $x_i \in$ X. Fig. 1 demonstrates the movement of the current best individual that starts from high violation regions towards low violation regions until it is trapped in a stagnant region which is then referred as stalled local optimal solution.

5 Experiments

ICHEA is a problem independent tool to solve any given n dimensional CSP so we use the following parameters to solve all the problems:
Stall threshold = 12 generations, crossover rate = 1.0, mutation rate = 0.1, maximum generation = 1000 and $|pop| = \begin{cases} 25, & n > 6 \\ 100, & 1 \leq n \leq 6 \end{cases}$

Table 1. Benchmark Quadratic Problem Chem

δ		ICHEA+	I-ICHEA	ICHEA	imp
10^{-1}	SR	1.00	1.00	1.00	0.0
	Best	11 gens at 1.8s	19 gens at 2.7s	54 gens at 0.83s	0.5
	Median	26 gens at 4.03s	25 gens at 3.5s	238 gens at 4.66s	1.2
	Worst	37 gens at 6.7s	33 gens at 4.8s	559 gens at 11.1s	1.7
10^{-3}	SR	1.00	1.00	0.30	0.7
	Best	75 gens at 7.3s	34 gens at 5.4s	5900 gens at 196.4s	26.9
	Median	621 gens at 108.3s	267 gens at 45.7s	-	3.1
	Worst	740 gens at 122.9s	291 gens at 49.6s	-	2.7

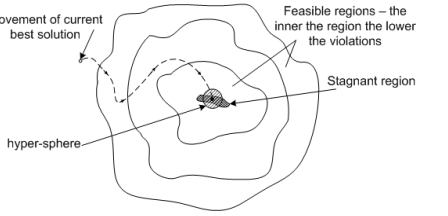

Fig. 1. Making hyper-sphere around stalled local optimal solution

Table 2. COP Benchmark problem G05

δ		ICHEA+	I-CHEA	ICHEA	imp
10^{-5}	SR	1.00	1.00	1.00	0.0
	Best	26 gens at 0.52s	29 gens at 0.53s	18 gens at 0.40s	0.77
	Median	30 gens at 0.57s	34 gens at 0.62s	19 gens at 0.41s	0.72
	Worst	39 gens at 0.72s	36 gens at 0.64s	21 gens at 0.46s	0.64

As one constraint is considered at a time for DCSP, we would also like to see if the constraint strengths of individual constraint matters in finding an efficient solution. Hence two different sequences of static CSPs are used where each constraint is added into the environment – from lowest to highest strength and vice versa. As described in [10] constraint strength (ρ) are computed *offline* by using the formula $= |\Omega|/|pop|$, where $|pop|$ is the number of solutions randomly generated from pop,

Table 3. Benchmark Trigonometric Problem HS109

δ		ICHEA+	I-ICHEA ($\rho\uparrow$)	I-ICHEA ($\rho\downarrow$)	ICHEA	imp
10^{-1}	SR	0.70	0.70	0.70	0.70	0.0
	Best	54 gens at 81.1s	57 gens at 87.7s	59 gens at 79.8s	53 gens at 71.0s	0.9
	Median	131 gens at 205.0s	113 gens at 183.0s	66 gens at 92.1s	70 gens at 239s	1.2
	Worst	192 gens at 361.3s	186 gens at 283.6s	150 gens at 208.6s	-	2.8
10^{-3}	SR	0.10	0.80	0.80	0.0	0.8
	Best	133 gens at 204.7s	100 gens at 155.0s	89 gens at 128.4s	-	4.9
	Median	-	122 gens at 186.0s	125 gens at 183.2s	-	0.0
	Worst	-	156 gens at 250.5s	151 gens at 230.6s	-	0.0

Table 4. Benchmark Polynomial Problem Broyden10

δ		ICHEA+	I-ICHEA	ICHEA	imp
10^{-1}	SR	1.00	1.00	0.80	0.2
	Best	22 gens at 39.6s	31 gens at 45.0s	116 gens at 189.1s	4.8
	Median	29 gens at 55.8s	49 gens at 81.7	248 gens at 235.1s	4.2
	Worst	53 gens at 182.1s	158 gens at 300.0s	-	5.5
10^{-3}	SR	1.00	1.00	-	1.0
	Best	28 gens at 51.8s	36 gens at 59.4s	-	19.3
	Median	42 gens at 85.3s	38 gens at 74.0s	-	11.7
	Worst	79 gens at 269.3s	174 gens at 385.3s	-	3.7

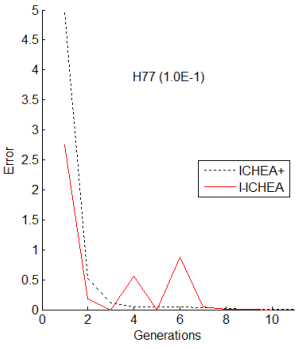

Fig. 2. I-ICHEA and ICHEA+ comparison for H77 ($\delta = 10^{-1}$)

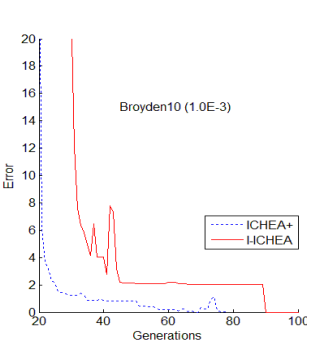

Fig. 3. I-ICHEA and ICHEA+ comparison for Broyden ($\delta = 10^{-3}$)

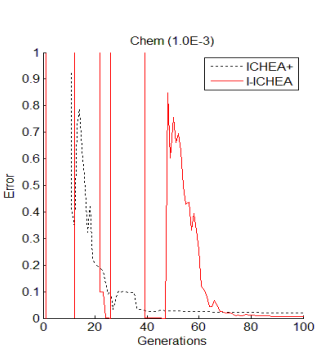

Fig. 4. I-ICHEA and ICHEA+ comparison for Chem ($\delta = 10^{-3}$)

$|\Omega|$ is the number of feasible solutions out of these $|pop|$ solutions. In the experimental setup, $|pop|$=10,000 and ρ value is computed as the average of five successive runs.

It has been demonstrated in [14] that ICHEA outperforms all other tested EAs where other tested EAs have very low SR. Hence we are only providing the results of ICHEA+ with previously introduced ICHEA. ICHEA has been developed in Java language and the tests have been carried out on the same Windows XP machine with Pentium (R) i5 CPU 2.52 GHz and 3.24 GB RAM. No parallel processing or distributed environment is used for the experiment. An average of 10 successive runs is taken into account to test the algorithms based on SR and generation count to reach to

the solution. SR is the rate of successful trials for each problem i.e. $SR = successful\ trials\ /total\ trials$.

Nine test cases have been created using the benchmark problems from CSP domain [15] and COP domain [10, 11]. Tables 1, 3, 4, 5 show CSP test results for problems *Chem, HS109, Broyden10 and H77*, and Table 2 shows test results for a COP – *G05*. Each benchmark problem has been tested on two different δ values $\{10^{-1}, 10^{-3}\}$

Table 5. Benchmark trigonometry problem H77

δ		ICHEA+	I-ICHEA ($\rho \uparrow$)	I-ICHEA ($\rho \downarrow$)	ICHEA	imp
10^{-1}	SR	1.00	1.00	1.00	1.00	0.0
	Best	5 gens at 0.6s	6 gens at 0.7s	9 gens at 1.1s	8 gens at 0.3s	0.5
	Median	8 gens at 1.2s	6 gens at 0.7s	11 gens at 1.5s	22 gens at 0.64s	0.5
	Worst	13 gens at 2.0s	8 gens at 0.9s	13 gens at 2.0s	48 gens at 1.53s	0.8
10^{-3}	SR	1.00	1.00	1.00	1.00	0.0
	Best	7 gens at 0.91s	8 gens at 1.0s	20 gens at 3.4s	447 gens at 19.0s	20.9
	Median	16 gens at 2.3s	28 gens at 4.4s	41 gens at 6.8s	3250 gens at 113.7s	49.4
	Worst	21 gens at 3.1s	37 gens at 5.8s	66 gens at 11.3s	6297 gens at 211.4s	68.2

Table 6. Constraint Strengths for H77

Constraints	ρ
1	6.33E-01
2	6.14E-01
3	6.33E-04

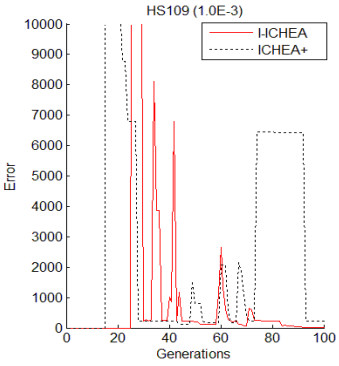

Fig. 5. I-ICHEA and ICHEA+ comparison for HS109 ($\delta = 10^{-3}$)

Table 7. Constraint Strengths for HS109

Constraints	1	2	3	4	5	6	7	8	9	10	11
ρ	0	0.87536	0.87662	0	0	0	0.00004	0.9241	0.44	0.41442	0.41874

except for problem *G05* which only uses $\delta = 10^{-5}$. Table 6 and Table 7 show constraint strengths for problems *H77* and *HS109* respectively. Other problems have same constraint strengths for all the constraints with $\rho = 0$. The ρ values are sorted in both ascending and descending order for separate tests where constraints are incrementally added to the search space in that order. As described earlier DCSPs are basically sequence of static constraints that are incrementally added to the search space. Table 1 – Table 5 show test results of ICHEA+ with previous ICHEA to compare their performances on different benchmark problems. The results of I-ICHEA (both ascending($\uparrow$) and descending($\downarrow$) order of ρ) have also been shown on the same tables to compare the results of ICHEA solving both static CSPs and dynamic CSPs as it is important to show whether knowledge from already solved constraints has been utilized or not. The test results are shown with best, median and worst solutions for each problem in terms of SR and efficiency. Columns are left blank with "-" if either it is not applicable or no good results have been obtained. The last column

named *imp* shows the improvement of the ICHEA+ over ICHEA where the values for best, median and worst indicate how many times the ICHEA+ is better than ICHEA and the values for SR indicate the increase in SRs from ICHEA to ICHEA+.

Fig.2 – Fig. 5 depict the average of all test runs to compare performances of ICHEA+ and I-ICHEA. The y- axis shows the error value given in Eq. (8) and x-axis shows the number of generations. The graphical image shows the progress of ICHEA in solving CSP and DC SP. The spikes in the graphs for I-ICHEA indicate that a new constraint has been introduced into the search space and spikes for ICHEA+ indicate the current best individual at that generation has been improved by solving some additional constraints. This causes the fitness value of current best individual to increase as ICHEA+ favors individuals with less constraint violations which results in new individual (generally with high error value) to be added into the population [14].

6 Discussion

The experimental setup in Section 5 has dual objectives. Firstly, it demonstrates the comparative study of previously published ICHEA in [14] with an upgraded ICHEA that applies some new strategies proposed in Section 4.1 and Section 4.2 and secondly, whether ICHEA is able to handle dynamic constraints in an incremental manner by reusing knowledge from previous increments. The experimental results show that the addition of diversity management and stalled local optimal solutions management has improved the performance of ICHEA to solve CSPs. Previously introduced ICHEA has very low SR for many benchmark problems when δ is 10^{-3} because of local optimal solutions that makes the whole population become stagnant that has been massively improved for problems – *HS109*, *Broyden10* and *Chem*. ICHEA's efficiency has also been improved massively at different rates for all the hard problems except *G05* which is a simple problem in the perspective of CSPs. The second objective of the experiment is to show if I-ICHEA can perform similar to ICHEA+ where the experimental results show that I-ICHEA has not only performed similar to ICHEA+ but has outperformed it for problems – *HS109* and *Chem*. This demonstrates that ICHEA makes full use of constraints solved in previous increments that are transpired to new increments and it is capable of handling dynamic constraints. Constraints can be added dynamically to ICHEA and it can still give the solution with same efficiency and success as of solving all the constraints concurrently. The experimental results on the order of constraint strength did not produce any conclusive results about the performance of ICHEA as shown in problems – *H77* and *HS109* where results with mixed success have been observed.

7 Conclusion

This paper has proposed an improvement on ICHEA to solve CSPs together with an enhancement of its capacity to handle DCSPs effectively. It has been shown through benchmarks problems that the new strategies applied to ICHEA helps in maintaining the diversity of the populations and dealing with local optimal solutions

by dynamically creating new constraints. This has helped massively in getting higher SR for most of the test problems. ICHEA has also been tested to handle DCSPs on benchmark CSPs that have been transformed to DCSPs. It has been shown that constraints can be added dynamically to ICHEA without restarting the algorithm and it can still give the solution with similar efficiency and SR as of solving all the constraints concurrently because ICHEA utilizes the knowledge from constraints of previous increments. The experimental results on the order of constraint strengths have been inconclusive in finding a CSP solution in an incremental approach. For future work efficiency of ICHEA can be tested on dynamic constraints where previous constraints can be removed or updated that distort the previous feasible regions. ICHEA has been able to solve CSPs and DCSPs. It has potential to be extended to work for discrete data as well because it extracts knowledge from constraints for its evolutionary search. ICHEA+ and I-ICHEA are currently further developed to solve real valued COPs and dynamic COPs respectively.

References

1. Brest, J., et al.: Dynamic optimization using Self-Adaptive Differential Evolution. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 415–422 (2009)
2. Craenen, B.G.W., et al.: Comparing evolutionary algorithms on binary constraint satisfaction problems. IEEE Transactions on Evolutionary Computation 7, 424–444 (2003)
3. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm. NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Eiben, A.E.: Evolutionary Algorithms and Constraint Satisfaction: Definitions, Survey, Methodology, and Research Directions. In: Theoretical Aspects of Evolutionary Computing, pp. 13–58 (2001)
5. El Rhalibi, A., Kelleher, G.: An approach to dynamic vehicle routing, rescheduling and disruption metrics. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 3613–3618 (2003)
6. Karimi, J., et al.: A new hybrid approach for dynamic continuous optimization problems. Applied Soft Computing 12, 1158–1167 (2012)
7. Kramer, O.: A Review of Constraint-Handling Techniques for Evolution Strategies. Applied Computational Intelligence and Soft Computing, 1–11 (2010)
8. Li, C., et al.: Benchmark Generator for CEC'2009 Competition on Dynamic Optimization (2008)
9. Li, T., et al.: Dynamic Constraint Satisfaction Approach to Hybrid Flowshop Rescheduling. In: 2007 IEEE International Conference on Automation and Logistics, pp. 818–823 (2007)
10. Liu, H., et al.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Soft Comput., 629–640 (2010)
11. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4, 1–32 (1996)
12. Nguyen, T., Yao, X.: Continuous Dynamic Constrained Optimisation - The Challenges. IEEE Transactions on Evolutionary Computation 99, 1 (2012)
13. Dechter, R.: Constraint networks. In: Encyclopedia of Artificial Intelligence, pp. 276–285. John Wiley & Sons, Ltd., New York (1992)

14. 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)
15. The COCONUT Benchmark, http://www.mat.univie.ac.at/~neum/glopt/coconut/Bench-mark/Benchmark.html
16. Tessema, B., Yen, G.G.: A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization. In: IEEE Congress on Evolutionary Computation, pp. 246–253 (2006)
17. Verfaillie, G., Jussien, N.: Constraint Solving in Uncertain and Dynamic Environments: A Survey. Constraints, 253–281 (2005)
18. Wang, H., Wang, D.-W., Yang, S.: Triggered Memory-Based Swarm Optimization in Dynamic Environments. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 637–646. Springer, Heidelberg (2007)