CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

The *Differential Evolution* algorithm (Storn and Price, 1995, 1996, Storn, 1996a, Price, 1997, Price and Storn, 1997) has emerged as one of the powerful tools for real parameter optimization, under the roof of evolutionary algorithms, more than a decade ago. After the first technical report on $DE$ by Storn and Price (1995), the competitive performance of $DE$ was demonstrated in the first International Contest on Evolutionary Optimization (ICEO) in 1996 (Storn and Price, 1996). $DE$ got third position in the ICEO contest, but the first two positions were bagged by non-evolutionary algorithms. Interestingly in the 2$^{nd}$ ICEO competition, $DE$ turned out to be the top performing algorithm among the other competing algorithms (Price, 1997). The algorithmic description of $DE$ was presented with sufficient details in (Price and Storn, 1997, Storn and Price, 1997).

Since 1999, $DE$ has been used widely for real world optimization problems from diverse fields of science and engineering. $DE$ became an attractive optimization tool owing to its algorithmic features as compared to other EAs. $DE$ presents a simple concept of employing difference vectors, as search direction, to perturb population elements. The implementation of $DE$ algorithm demands less coding effort in any of the programming languages. Even though Particle Swam Optimization ($PSO$) is also easy to implement, $DE$ has been shown to display competitive performance against $PSO$ (Vesterstrom and Thomsen, 2004). $DE$ has lesser number of parameters viz. $C_r$ (crossover rate), $F$ (scaling factor) and $NP$ (population size). The efficiency of $DE$ has shown to be improved by proper tuning of the parameters $F$ and $C_r$. Various techniques for self-adaptation of $DE$ parameters are demonstrated in (Brest et al., 2006a, Qin et al., 2009, Zhang and Sanderson, 2009).

The conceptual and algorithmic simplicity of $DE$ has attracted many researchers who are actively working on its various aspects, resulting in many variants of $DE$ algorithm viz. SaDE (Qin and Suganthan, 2005), $jDE$ (Brest et al., 2006a), JADE (Zhang and Sanderson, 2007b, 2009), ODE (Rahnamayan et al., 2008), DEGL (Das et al., 2009) etc. Detailed reviews of active research in $DE$ has been presented in (Das and Suganthan, 2011, Neri and Tirronen, 2010).
Diversity enhancement, high dimensional optimization, self adaptive mixing of trial vector generation techniques, multi-objective optimization (Yang et al., 2007b, Pant et al., 2009a, 2009b, Qin et al., 2009), to cite but a few examples, are some of the research works being carried out in DE (Chakraborty, 2008). The survey of DE in this thesis has been organized in the following nine major groups:

1. Empirical analysis of the existing DE variants
2. DE algorithms with additional component integrated to its structure
3. DE algorithms with modification to its structure
4. Control parameter analysis of DE
5. Mutation and crossover operators of DE
6. Exploration and exploitation analysis of DE
7. DE algorithm for multi-objective/constrained objective function optimization
8. DE algorithm for high dimensional/noisy function optimization
9. Parallel Differential Evolution

### 2.2 Empirical Analysis of the Existing DE Variants

With various mutation and crossover operators available for DE, there exist many variants of DE. Since the choice of mutation/crossover operators for trial vector generation largely affects the performance of classical DE, a thorough understanding of the performance of different DE variants is crucial. However, little research effort has been devoted to understand and compare the efficacy of existing DE variants to solve problems with different features.

The most common variant of DE, in the literature, is *DE/rand/1/bin*. The mutation schemes *DE/best/1, DE/best/2, DE/rand/2* and *DE/target-to-best* were suggested in (Price et al., 2005, Price 1999). By considering the 5 mutation schemes (*DE/best/1, DE/best/2, DE/rand/1, DE/rand/2* and *DE/target-to-best*) and 2 most commonly used crossover schemes (exponential and binomial (Price et al., 2005)), ten different trial vector generation strategies (*variants*) were suggested in (Storn and Price, 1997, Price et al., 2005). The *DE/current-to-rand/1* trial vector generation strategy was proposed in (Price, 1999), which performs rotationally invariant arithmetic recombination. Price et al. (2005) proposed a strategy called *DE/rand/1/either-or*. A rotation invariant mutation scheme was presented by Lampinen and Zelinka (1999a). Das et al. (2009) empirically showed that *DE/rand/1/either-or* variant displayed competitive performance...
over the variants $DE/rand/1/bin$ and $DE/target-to-best/1/bin$. Based on the probabilistic distribution of the trial vectors generated in $DE$, a new mutation scheme with a probabilistic scheme was proposed in (Ali and Fatti, 2006).

Consequently, $DE$ has a number of trial vector generation strategies, each one of them best suited only for certain specific problems (Feoktistov and Janaqi, 2004). Thus, irrespective of the availability of many $DE$ variants, none turned out to be the best for a large number of, if not for all the type of, problems. This necessitates the need for thorough investigation on the performance of all the $DE$ variants. Mezura-Montes et al., (2006b) empirically compared the performance of eight $DE$ variants, involving arithmetic recombination along with binomial ($bin$) and exponential ($exp$) crossover, on unconstrained optimization problems. In their study $DE/rand/1/bin$, $DE/best/1/bin$, $DE/current-to-rand/1/bin$ and $DE/rand/2/dir$ were identified the most competitive variants. However, the potential variants like $DE/best/2/*$, $DE/rand-to-best/1/*$ and $DE/rand/2/*$ were not considered in their study (* represents both binomial and exponential crossovers).

Babu and Munawar (2001) compared the performance of ten variants of $DE$ to solve the optimal design problem of shell-and-tube heat exchangers. They concluded $DE/best/*/*$ strategies to be better than $DE/rand/*/*$ strategies (the first * represents number of pairs involved in finding the difference vector and the second * presents the crossover used).

An implementation of $DE/rand/1/bin$ variant on CEC05 test bed (Suganthan et al., 2005) is reported in (Ronkkonen et al., 2005). The test bed covers functions with different features. The study covers the performance analysis of the variant $DE/rand/1/bin$ for 10 and 30 dimensions. The population sizes for the experiments were decided in the range of $2D$ to $10D$. It was found in the result that due to the limit set by the competition on number of function evaluations (300,000), $DE/rand/1/bin$’s success performance was very poor.

Mezura-Montes et al. (2010) presented an extensive analysis of different variants of $DE$ on 24 benchmark problems. Even though the study focused only on constrained numerical optimization, the insight from the study is worth considering. Their first experiment (out of 3 experiments) concluded that $DE/rand/1/bin$ shows better performance by its percentage of successful runs, $DE/target-to-best/1$ was very competitive for high-dimensional problems and $DE/best/1/bin$ takes lesser number of function evaluations.

Their second experiment focused on performing parameter study for the two variants
DE/rand/1/bin and DE/best/1/bin. The parameters viz. F (scaling factor) and NP (population size) were taken into consideration for the experiment. The simulation results concluded that DE/rand/1/bin was highly competitive for high-dimensional test problems and was less sensitive to F and NP. DE/best/1/bin was less reliable than DE/rand/1/bin and DE/rand/1/bin needed smaller population because it generates search direction from different random vectors whereas DE/best/1/bin needed larger population since the search directions are based on the best solution so far.

In the third part of the experiment the variants DE/rand/1/bin and DE/best/1/bin were combined to form a single new variant called DECV (differential evolution combined variants) to have the combined advantage of both DE/rand/1/bin and DE/best/1/bin. DE/rand/1/bin was able to generate more diverse set of search directions and DE/best/1/bin took lesser number of function evaluations to reach the vicinity of the feasible region. Even though DECV did not show any competitive performance than the other state-of-the-art DE variants, this experiment proposed the idea of combining different DE variants.

Neri and Tirronen (2010) presented a survey on DE and its recent advances. The various improved DE algorithms in the literature were clustered into two groups in (Neri and Tirronen, 2010). The first group comprised the algorithms which integrate additional component to DE structure and the second group involved the algorithms which employ a modified DE structure. Four algorithms under each group are considered for their study. The DE algorithms considered under the group 1 are (i) DE with Trigonometric Mutation (Fan and Lampinen, 2002, 2003b, Hu et al., 2005, Angira and Santosh, 2007, 2008) (ii) DE with Simplex Crossover Local Search (Noman and Iba, 2005, 2008) (iii) DE with Population Size Reduction (Brest and Maucec, 2008, Brest et al, 2008) and (iv) DE with Scale Factor Local Search (Neri and Tirronen, 2009, Tirronen et al., 2009). The DE algorithms considered under the group 2 are (i) Self Adaptive Control Parameters DE (Brest et al, 2006a, 2006b, 2007, 2008, Zamuda et al., 2007) (ii) Opposition Based DE (Rahnamayan et al., 2006, 2007b, 2008, Rahnamayan and Wang, 2008) (iii) Global-Local Search DE (Chakraborty et al., 2006, Das et al., 2009) and (iv) Self Adaptive Coordination of Multiple Mutation Rules (Qin and Suganthan, 2005, Yang et al., 2008b, Qin et al., 2009). A representative algorithm from each group has been implemented and tested on a broad set of various test problems. A detailed review of the basic concepts of DE and a survey of its major variants are presented in (Das and Suganthan, 2011) too.
2.3 Differential Evolution with Added Component

In this group the algorithms which use \textit{DE} as an evolutionary framework and assisted by additional algorithmic components (Neri and Tirronen, 2010) are considered for discussion. These additional components are local search algorithms (Chen et al., 2008, Tirronen et al., 2007, 2008) and surrogated assisted models (Zhang and Sanderson, 2007a). There is always a clear distinction between \textit{DE} and the added component.

In \textit{DE} with trigonometric mutation, \textit{TDE}, (Fan and Lampinen, 2002, 2003b, Hu et al., 2005, Angira and Santosh, 2007, 2008) the mutation operation of classical \textit{DE} is replaced by the trigonometric mutation. The trigonometric mutation makes \textit{DE} more exploitative, by generating the offsprings in most promising directions (Neri and Tironen, 2010). By combining the mutation operation of the classical \textit{DE} and trigonometric mutation of \textit{TDE} a directed mutation is proposed in (Fan and Lampinen, 2003a).

In general, the local search algorithms are to explore a smaller search space to find a local optimal point. There are many variants of \textit{DE} proposed in literature by blending the classical \textit{DE}'s performance with some local search algorithm. \textit{DE} with simplex crossover local search proposed in (Noman and Iba, 2005, 2008), uses an adaptive hill climbing simplex crossover to improve the performance of \textit{DE}. The proposed \textit{DE} variant was named as \textit{DEahcSPX}. The objective of this variant is to balance the explorative ability of \textit{DE} by the exploitative ability of the local search algorithm. Thus, \textit{DEahcSPX} hybridizes an evolutionary algorithm (\textit{DE}) and a local search algorithm (\textit{SPX}).

Yang et al. (2007a) proposed a variant of \textit{DE} called \textit{NSDE}, with the objective of hybridizing \textit{DE} and the neighborhood search (\textit{NS}). In \textit{NSDE}, each target vector is added with a normally distributed random value, during mutation. An enhanced \textit{NSDE} called self adaptive \textit{NSDE} was used in the co-operative co-evolution framework for high dimensional function optimization, in (Yang et al., 2008a). Similar to this, \textit{SaNSDE} (self adaptive differential evolution with neighborhood search) was reported in (Yang et al., 2008b), which uses \textit{SaDE} proposed in (Qin et al., 2009) for its self adaptation.

\textit{DE} with population size reduction (\textit{DEPSR}) (Brest and Maucec, 2008, Brest et al., 2008), employs the process of reducing the population size during the search process. The \textit{DEPSR} adds a new variable to \textit{DE}'s structure for population size. During the initial stage of evolution \textit{DE}
needs larger search space (i.e. larger population size) in order to explore more promising spaces, and as evolution proceeds the search space is to be reduced (i.e. smaller population size) in order to make DE to exploit the promising spaces. The objective of this reduction of population size is to focus the search in smaller spaces, in order to reduce the occurrence of stagnation especially in high dimensional function optimization.

DE with scale factor local search (DESFLS) was first proposed in (Tirronen et al., 2009), and it was extended by Neri et al. (2009) to employ two local search algorithms for self-adaptive DE schemes. An improved version of DESFLS is presented in (Neri and Tirronen, 2009). The improved DESFLS applies local search to scale factor \((F)\), with a certain probability. In (Tirronen et al., 2009), this local search was designated as a minimization function over the variable \(F\). Two local search algorithms viz. scale factor golden section search and scale factor hill climb have been considered for minimization in (Tirronen et al., 2009) and (Neri et al., 2009). The cooperative performance of these two local search algorithms was experimented and reported in (Neri and Tirronen, 2009). Caponio et al. (2009) proposed to add PSO and two other local search algorithms to classical DE. Two additional operations (at acceleration phase and at migration phase) are embedded to classical DE, and this hybrid method is used for optimal parameter selection of bioprocess system in (Chiou and Wang, 1998, 1999).

Other than adding local search algorithms as additional component to the classical DE, researchers have attempted to hybridize DE with other global search algorithms like particle swarm optimization (PSO) (Kennedy et al., 2001), ant colony systems (Dorigo and Gambardella, 1997), artificial immune systems (AIS) (Dasgupta, 1999), bacterial foraging optimization (BFOA) (Passino, 2002) and simulated annealing (SA) (Kirkpatrik et al., 1983).

The hybridization of DE with PSO, swarm differential evolution algorithm, was first proposed in (Hendtlass, 2001). Zhang and Xie (2003) proposed DEPSO as another hybrid algorithm comprising DE and PSO. In DEPSO the PSO and DE algorithms are alternatively used to search in odd and even generations. Das et al. (2005a) proposed PSO-DV called as particle swarm optimization with differentially perturbed velocity. In PSO-DV, the velocity-update operation of PSO is done by the DE operators. Continuing this trend various hybridization algorithms have been proposed in literature (Moore and Venayagamoorthy, 2006, Liu et al., 2005, Kannan et al., 2004, Hao et al., 2007, Omran et al., 2009, Kennedy, 2003, Xu et al., 2008). The integration of fuzzy logic with DE for DE’s parameter setting was attempted in (Liu and Lampinen, 2002a,
Hybridizing SA with DE, to replace the selection logic of DE using SA, was attempted in (Das et al., 2007). Another hybrid version of DE and SA, called SaDESA, was proposed in (Hu et al., 2008). Biswas et al. (2007) proposed hybridization of BFOA with DE. Hybridization of ant colony optimizer with DE was studied in (Chiou et al., 2004, Zhang and Duan et al., 2008). He and Han (2007) proposed to mix AIS with DE.

### 2.4 Differential Evolution with Modified Logic

The high convergence characteristics and robustness of DE has made it one of the popular techniques for real-valued parameter optimization. Many researchers are working towards enhancing the performance of classical DE, further. In this group the algorithms which made considerable modification to the structure of the classical DE, primarily in vector generation strategy or selection has been considered.

In the classical DE algorithm, repeated cycles of differential mutation and crossover generate a single trial vector. Then a one-to-one tournament selection between the parent and the trial vector is carried out and the winner is placed in the new population. The multiple trial vectors generations scheme was proposed in (Storn, 1999, Mezura-Montes et al., 2007), with the objective of increasing the probability of generating fitter trial vector for each parent. Storn (1999) used the multiple trial vectors generation scheme to generate the trial vectors one after another until finding a better trial vector than the parent vector. Mezura-Montes et al. (2007) used multiple trial vectors generation scheme in DE to solve constrained optimization problems in engineering design. It produced five trial vectors for each parent using DE/rand/1/bin variant. The best of the trial vectors was made to compete with the parent vector.

To alleviate the problem of static population update mechanism of the classical DE, Qing proposed Dynamic Differential Evolution (DDE) in (Qing, 2006). He analyzed the performance of the variant DDE/best/1 variant on a benchmark electromagnetic inverse scattering problem. The study identified the significant competitive performance of DDE over classical DE. A comparative study on the performance of DDE/rand/1/bin and DDE/best/1/bin variants and their classical counterpart variants was reported in (Qing, 2008). The study presented representative results of the experiments done using a test bed with 37 test functions and three application problems, and concluded that DDE/best/1/bin outperforms other variants considered.
Qin and Suganthan (2005) proposed a variant to DE called Self Adaptive DE (SaDE) in which the parameter settings are gradually self-adapted during the evolution, according to the learning experience. Five mutation strategies viz. \texttt{DE/rand/1}, \texttt{DE/best/1}, \texttt{DE/current-to-best/1}, \texttt{DE/best/2} and \texttt{DE/rand/2} are considered in the study. The SaDE demonstrated good performance on problem with different properties. The SaDE also integrates Quasi-Newton method as an additional component to DE to do local search in order to speed up its convergence. Another self adaptive DE (SaDE) is proposed in (Qin et al., 2009), in which a self-adaptation scheme which learns from its past experience was used to adapt both the trial vector generation strategies and their associated control parameters. Four variants viz. \texttt{DE/rand/1/bin}, \texttt{DE/rand-to-best/2/bin}, \texttt{DE/rand/2/bin} and \texttt{DE/current-to-rand/1/bin} were considered in the strategy candidate pool.

Das et al. (2007) proposed a variant of DE called Annealed DE (AnDE) which replaces the deterministic selection mechanism of classical DE by a stochastic selection mechanism. The deterministic selection mechanism selects the better candidates for the next generation, which may cause the algorithm to trap in the local optimum. A conditional acceptance function (motivated by that of SA) is used for the stochastic selection mechanism. The conditional acceptance function not only accepts the better solution that decreases the objective function value (in case of a minimization problem), but also accepts some inferior solution that increases it. A new mutation operator called center of mass based (CM-based) mutation is also proposed, where the mean of all the vectors in the current generation is considered as best vector. The AnDE was considered to be a hybridization of DE and SA, which performed statistically better than its constituents DE and SA.

A variant of DE called EPSDE was proposed in (Mallipeddi and Suganthan, 2010, Mallipeddi et al., 2011). EPSDE is a DE with a pool of mutation and crossover strategies along with pool of values for their associated control parameters. This strategy pool and value pool are made available during the evolution process. The mutation strategies and the parameters in the pools were chosen such a way that they exhibit distinct performance during different stages of evolution. The mutation strategy and the parameter values which produced a better trial vector than the parent vector are retained with the trial vector itself, for use in the next generation.

Bi and Xiao (2010, 2011) proposed a new DE variant called \texttt{p-ADE} with a new mutation strategy. \texttt{DE} has many mutation operators which use difference between two or more randomly selected candidates’ information in the population. Because of the randomness involved in the
above operators, DE may deviate from the search direction and hence may become vulnerable to premature convergence or stagnation. So, it is desirable to use some directional information during the search to avoid the problem of slow and premature convergence of the DE. The p-ADE uses a new mutation strategy DE/rand-to-best/pbest, which uses the best-so-far and best previous candidates (as directional information) instead of two random candidates for mutation. The new mutation strategy also uses a classification mechanism to balance the exploration and exploitation during the search.

Another variant of DE called SspDE was proposed in (Pan et al., 2011). SspDE is a DE with self-adaptive trial vector generation strategy and control parameters. SspDE maintains a list of strategy (SL), scaling factor (FL) and crossover rate (CRL). For generating a trial vector a strategy, a scaling factor and a crossover rate is chosen from their respective lists. If the generated trial vector is better than the target vector the chosen values are added to winning lists (wSL, wFL and wCRL). After certain number of generation the SL, FL and CRL are updated by wSL, wFL and wCRL, respectively. Thus the trial vector generation strategy and its associated parameters are self-adapted from their previous successful experience.

A new differential evolution algorithm (JADE) is proposed in (Zhang and Sanderson, 2007b, 2009) with a new mutation strategy DE/Current-to-pbest, and adaptive selection of control parameters. Two variants of JADE viz. rand-JADE and nona-JADE are experimented and compared with other DE variants. (Pant et. al., 2009a) proposed MDE, which uses a new mutant vector with self adaptive F (scaling factor). Since F is used in generating mutant vector, the presence of good scaling factor helps to preserve the diversity. The MDE was stated as semi adaptive DE, by the authors. The results showed that the use of random variable, having Laplace distribution as a scaling factor F, improves the performance of classical DE significantly.

Thangaraj et al. (2005) performed a study by proposing five new mutation schemes to DE, named as MDE1, MDE2, MDE3, MDE4 and MDE5. Each MDEi is differentiated from the classical DE by its mutation logic. The MDE1 uses only two vectors to generate a mutant vector, MDE2 (like current-to-best of classical DE) uses the best vector as the base vector. The MDE3 generates two mutant vectors using MDE1 scheme and uses the one with good fitness function values as the mutant vector. In MDE4, the MDE1 scheme and the classical DE schemes are used stochastically. The fifth scheme, MDE5 generates mutant vector by adding a random vector with a weighted
difference between the best vector and another random vector. The proposed algorithms are validated with a directional over-current relay setting optimization problem.

Mininno et al. (2011) proposed a variant of DE, called Compact DE (cDE), which does not use a population of individuals rather uses a statistical representation of the population. The cDE solves complex optimization problems and is more applicable for devices with small computational power such as micro-controllers and commercial robots. A new version of the DE, called jDE, is proposed in (Brest et al., 2006a). The jDE enhances the properties of standard DE by self-adaptive setting of the control parameters $F$ and $C_r$. The jDE has shown good performance for large scale optimization (Brest et al., 2008), constrained optimization (Brest et al., 2006b) and multi-objective optimization (Zamuda et al., 2007). An improved version of jDE is presented in (Brest et al., 2007).

Das et al. (2009) described a set of trial vector generation schemes which uses the concept of neighborhood in the population. The proposed variances are the improved versions of the DE/target-to-best/1/bin scheme, and they were able to balance the exploration and exploitation abilities of DE. An attempt to self-adapt the population size in addition to self-adapting crossover and mutation rates is proposed in (Teo, 2006), based on the self-adaptive Pareto DE proposed in (Abbas, 2002). The mutation operation of classical DE was modified using the attraction-repulsion concept of electromagnetism-like algorithm in (Kaelo and Ali, 2007).

There are other works that primarily compare a particular DE variant with its improved or enhanced version like in (Bui et al., 2005), which compares classical DE and a new DE variant proposed by the authors called Guided-DE on 25 benchmark functions. Variants of DE are compared with Particle Swarm Optimization (PSO) algorithm in (Vesterstrom and Thomsen, 2004, Das et al., 2005a).

### 2.5 Control Parameter Analysis of Differential Evolution

There are many works available in the literature to study the suitable setting of the control parameters viz. $F$, $C_r$, and $NP$ (Zaharie, 2002a, 2002b, Gaemperle et al., 2002, Kukkonen and Lampinen, 2005a). Storn and Price (1995) have suggested the setting of $NP$ as $5.D$ and $10.D$ where $D$ is the problem dimension, and $F = 0.5$. Gaemperle et al. (2002) presented a parametric study on DE on a set of benchmark functions. The study revealed that the capability of a variant to reach the vicinity of the global solution and its convergence speed fully rely upon choosing
suitable values for $F$, $NP$ and $C_r$. They suggested reasonable values for the parameters as: $NP = 3.D$ to $8.D$, $F=0.6$ and $C_r = 0.3$ to $0.9$. Ronkkonen et al. (2005) suggested $F$ to be in $(0.4, 0.95)$ and $C_r$ to be in $(0, 0.2)$ for separable functions and in $(0.9, 1)$ for nonseparable functions. As suggested in (Price et al., 2005) $F \in [0, 1 + ]$, such that $F$ is a positive value and not much greater than 1.

The effect of $NP$ on the quality of the solution is investigated in (Mallipeddi and Suganthan, 2008). They (Mallipeddi and Suganthan (2009)) proposed a strategy to choose population size adaptively to match with different phases of the search, with an ensemble of parallel populations. Self adaptive scheme for population size is first proposed in (Teo, 2005, 2006) and recently improved in (Teng et al., 2009). In the similar line, 3 variants of $DE$ with self adaptation of population size were proposed in (Sing et al., 2007). An integration of fitness diversity adaptation technique with $DE$ to adaptively control the population size was presented in (Tirronen and Neri, 2009).

The terms \textit{jitter} and \textit{dither} have been introduced in (Price et al., 2005), with respect to randomization of $F$ for improving the performance of $DE$. During each generation, before mutation operation, generating $F$ anew for every parameter is called \textit{jitter}. On the other hand generating $F$ anew only for every candidate solution is \textit{dither}. Das et al. (2005b) used dither, with $F$ in $(0.5, 1)$.

Deciding upon the parameters for $DE$ during its evolution is a critical task since the suitable setting of the control parameters will change according to the optimization problem considered and also it changes during the evolution of the optimization. Even though, in general, trial-and-error methods are applied for deciding the parameters, it is advantageous to go for automatic parameter tuning. Zielinski and Laur (2006a) suggest an idea to choose $F$ and $C_r$, based on the feedback given by the previous run. Since the evolution pattern of any $EA$ changes in different stages during its run, it is essential to have different parameter setup at different stages of evolution, thus the parameter setting will not remain fixed during an optimization run (Zielinski and Laur, 2006a). By following this similar idea, there are many self-adaptation techniques for the control parameters proposed in the literature (Ali and Torn, 2004, Brest et al., 2006a, Liu and Lampinen, 2005, Qin et al., 2009, Ronkkonen and Lampinen, 2003, Omran et al., 2005, Teo 2006, Salman et al., 2007, Soliman et al., 2007, Soliman and Bui, 2008, Zhenyu et al., 2006). A comparative analysis of various such adaptive schemes is presented in (Zielinski et al., 2008).
2.6 Mutation and Crossover Operators of Differential Evolution

The differential mutation strategy of DE, most often, selects a random vector or the best vector from the current population as the base vector. In general, differential mutation can be regarded as a local search around the differential mutation base (Lin et al., 2011). The role of the mutation base in the differential mutation search is to provide guidance and diversity. In differential mutation strategy with best vector as the base vector, the search moves to the vicinity of the best vector, which increase the efficiency but the diversity is lost. On the other hand, with random vector as the base vector the differential mutation strategy gets diversity but loses the efficiency. In (Lin et al., 2011), a new base vector selection strategy to select the best of randomly chosen vectors is proposed to balance the diversity and efficiency. The best vector helps to achieve faster convergence (exploitation) and the random vectors helps to increase the diversity (exploitation), and hence the balance between exploration and exploitation is achieved in the differential mutation local search. The numerical results have proved that the new base vector selection strategy is more reliable and efficient for high-dimensional functions.

The crossover operator of DE mixes the parent vector with the newly generated mutant vector to produce a trial vector. There are many ways to do crossover, two most often used crossover operators are binomial crossover and exponential crossover. Zaharie (2007) analyzed the similarities and differences between binomial and exponential crossover. The binomial and exponential crossover operators are similar to the uniform crossover and two-point crossover, respectively, used in the evolutionary algorithms. The crossover operator has a parameter, the crossover rate ($C_r$), which decides the contribution of components by both the parent vector and mutant vector into the trial vector. The behavior of both binomial and exponential crossover is influenced by the parameter ($C_r$) (Zaharie, 2009). In both the crossovers, $C_r$ states the probability for a component of a trial vector to be selected from the mutant vector. However, the same value of $C_r$ gives different distribution of the number of mutated components in binomial and exponential crossover (Zaharie, 2007).

Since $C_r$ controls the number of components inherited from the mutant vector, the $C_r$ influences the probability that a component is mutated (mutation probability, $P_m$) (Zaharie, 2009). In general $C_r$ can be interpreted as $P_m$. The relationship between the $C_r$ and $P_m$ is analyzed in (Zaharie, 2009) for the binomial and exponential crossover. The relationship between $C_r$ and
\( P_m \) for binomial and exponential crossover is governed by the equations
\[ P_m = C_r \left( 1 - \frac{1}{n} \right) + \frac{1/n}{n} \]
and \[ P_m = \frac{1-C_r^n}{n(1-C_r)} \], respectively, where \( n \) is the population size. The relation between \( C_r \) and \( P_m \) is linear in the case of binomial crossover and nonlinear in the case of exponential crossover. The major difference between the binomial and exponential crossover operators are (Zaharie, 2009) (i) different values of mutation probability are attained by the same value of crossover rate (ii) the mutated components in the trial vector generated by the exponential crossover are in consecutive position, but they are at random position in the trial vector generated by the binomial crossover.

### 2.7 Exploration and Exploitation Analysis of Differential Evolution

Beyer (1998) analyzed the evolutionary search strategy of the ES/EP-like algorithms in the real-valued problems. The evolutionary progress of an EA has been stated as the result of two opposite forces producing gain and loss.

\[
\text{Evolutionary Progress} = \text{Progress Gain} - \text{Progress Loss} \tag{2.1}
\]

The mutation, crossover and selection operators are responsible for the gain and loss during the search. A close relationship between the selection, mutation and crossover operations is revealed in Beyer (1998) as: “The selection operation influences the gain, loss can be reduced by the crossover operation, but both depend on the strength of the mutation process”. The search behavior of an EA is the antagonism of exploitation and exploration. The “exploitation” process can be interpreted as the ability of an EA to step into the local gradient direction (to step into the direction of desired improvement) and the “exploration” as the ability to leave the gradient path (drives away from the desired direction). Thus, this decomposition (exploitation and exploitation) sheds a new light on the question of how the EAs in the real-valued spaces perform the evolutionary search (Beyer, 1998). As stated in (Beyer, 1998, Feoktistov, 2006) “the ability of an EA to find a global optimal solution depends on its ability to find the right relation between exploitation of the elements found so far and exploration of the search space”.

Thus, EA which fails to balance the exploration and exploitation process during its evolution will fall either in premature convergence or in stagnation (Zaharie, 2003, Zaharie and Zamfirache, 2006, Angela et al., 2008, Zaharie, 2001b). An algorithm is said to be prematurely converged when it experiences that (Lampinen and Zelinka 2000) (1) the population has
converged to local optima or (2) the population has lost its diversity or (3) the search algorithm proceeds slowly or does not proceed at all.

An algorithm suffer with stagnation when it fails to progress even if it has not still converged to any local optimum and the population is still diverse. As the number of dimensions of the problem increases the effect of stagnation also increases (Zamuda et al., 2008, Olorunda and Engelbrecht, 2007, Yang et al., 2007b, 2008a, Noman and Iba, 2005, Gao and Wang, 2007, Brest et al., 2008).

Both premature convergence and stagnation makes the algorithm lose its search capability in the search space. Empirical results on a set of test functions (Zaharie, 2001a, 2002a, 2002b) suggest that rapid and slow decreasing of population diversity induces premature convergence and slow convergence, respectively.

In **DE** algorithms the selection operator is commonly seen as the source of exploitation while exploration is attributed to the variation operators viz. mutation and crossover (Eiben and Schippers, 1998, Beyer and Deb, 1999). The selection step may increase or decrease the population variance. But to avoid any premature convergence or stagnation, the variation operators must adjust the population variance such that it has a reasonable value from one generation to other. Thus, if the selection decreases the population variance then it is necessary that the variation operators increase it.

On the other hand, a different view on the role of mutation and crossover operators in exploration and exploitation is revealed by Eiben and Schippers (1998). Eiben and Schippers (1998) stated that, the mutation operators can be viewed as both explorative and exploitative. It is an explorative operator because it introduces new materials and it can also be seen as an exploitative operator because it preserves most of the information and makes only small changes (Eiben and Schippers, 1998). Similar to that, the crossover operator can be seen as explorative operator because it creates new candidate from the parents and it is exploitative operator because it uses the old material.

During the evolution of an **EA**, from generation to generation, the explorative operator increases the population variability and the exploitative operator decreases the population variability. A mathematical measure of the population variability is its variance. Thus population variance is used (Zaharie, 2008) as an indicator to state how diversified the population is. The
population with diversified candidates is to have higher values of population variance and the population with almost identical candidates is to have lower value of population variance.

Thus, retaining reasonable diversity in population is essential for an EA (indeed critical) during its evolution. The population diversity could be achieved if the exploration and exploitation process are balanced. Various attempts have been proposed in the literature for balancing the exploration and exploration processes, and to enhance the population diversity. Zaharie (2003) suggested that balancing the exploration and exploitation processes in EAs could be achieved by controlling the population diversity through parameter adaptation. (Angela et al., 2008) proposed a modification to the selection operator of classical DE algorithm to reinstates the balance between exploration and exploitation processes. The proposed scheme also increases the probability of escaping from the local optima (Angela et al., 2008).

Zaharie and Zamfirache (2006) implemented a simple population diversity enhancing mechanism to control the population diversity by parameter adaptation to DE and PSO. Another diversity enhancing mechanism for DE in static optimization, called multi-population approach, is proposed. In multi-population approach, in each subpopulation adaptive DE algorithm is applied and a random migration process takes place between the populations.

A significant insight on the influence of the variation operators on the expected population variance was first reported in (Zaharie, 2001a) and then extended in (Zaharie, 2002a, 2002b). Zaharie (2001a) derived a theoretical relationship between the expected population variance after mutation and crossover and the initial population variance of DE/rand/1/bin. The author derived an expression as a measure of the explorative power of population-based optimization methods. This in turn provides theoretical insights on the explorative power of DE. The author also analyzed the evolution of population variance for DE/rand/1/bin variant for two test functions.

Zaharie (2001b) also analyzed the influence of crossover operators on the convergence properties and exploitative power of DE algorithm. An idea of hybridizing different crossover operators at different stages of evolution has been proposed. This study compares the behavior of five crossover operators, among the five operators the fifth one was the hybrid operator which select one of the other four operators based on the current behavior of the algorithm. It was shown experimentally that the hybrid crossover operator outperforms its constituent crossover operators.

Zaharie (2002a) presents the relationship between the control parameters and the evolution of
population variance of $DE$. It was shown that with proper selection of control parameters, the
diversity in the population could be maintained. Zaharie (2008) analyzed the impact of various
mutation and crossover operators used in $DE$ on its expected population variance. The
application of mutation and crossover operators changes the initial distribution of candidate
solution in the current generation. Mathematical expressions to measure the population variance
after the mutation and crossover are derived. The mean and variance measured using the derived
expressions are used to analyze the population distribution. The impact of the mutation and
crossover operators on the population variance could be controlled by changing the parameters
associated with them. A new mutation operator which works based on the population variance
and does not use the differences as the classical $DE$ mutation operator is proposed. The proposed
operator was found behaving similar to $DE/rand/1/^*$

2.8 Differential Evolution for Multi-Objective / Constrained
Objective Function Optimization

$DE$ algorithm was successfully extended to solve multi-objective, noisy and high dimensional
functions. Chang et al. (1999) first extended $DE$ to optimize multi-objective functions, with an
idea of Pareto dominance. Abbass and Sarker (2002) proposed the Pareto differential evolution
($PDE$) for multi-objective problem and compared the performance of $PDE$ with other
evolutionary algorithms for multi-objective problems. Abbass enhanced $PDE$’s convergence
speed by using the back-propagation local search algorithm in (Abbass, 2001).

Lampinen (2001) proposed generalized differential evolution ($GDE$), which is $DE/rand/1/bin$
extended to multi-objective problems. The $GDE$ uses Pareto dominance as selection condition
for the classical $DE$. In $GDE2$ (Kukkonen and Lampinen, 2004), the best solution was selected
using a crowding distance measure. Later, $GDE2$ was extended as $GDE3$ in (Kukkonen and
Lampinen, 2005b). The $GDE3$ combines $GDE$ with Pareto-based differential evolution algorithm
of Madavan (2002). The elitist sorting used in (Madavan, 2002) is defined in (Deb et al., 2002).
An algorithm called $\varepsilon$-$MyDE$ is proposed in (Santana-Quintero and Coello Coello, 2005), as an
enhanced version of $DE$ to solve multi-objective functions. The $\varepsilon$-$MyDE$ uses the concept of $\varepsilon$ –
dominance from (Laumanns et al., 2002).
Another multi-objective DE called MODE was proposed in (Xue et al., 2003), which uses a Pareto-based approach to select the best individuals. The Pareto-based ranking and crowding distance sorting mechanisms are incorporated to DE and proposed as DE for multi-objective functions (DEMO) in (Robic and Filipic, 2005). The concept of crowding distance was adapted from NSGA-II in (Laumanns et al., 2002). In fact NSGA-II is an improved version of the NSGA (nondominated sorting genetic algorithm) proposed in (Srinivas and Deb, 1995). A simple modified version of NSGA-II was proposed as NSDE (non-dominated sorting DE) in (Iorio and Li, 2004). NSDE outperformed NSGA-II for multi-objective functions.

Some more research attempts to make DE to solve multi-objective optimization problems also have been carried out in the literature as in (Babu and Jehan, 2003, Li and Zhang, 2006, 2009, Parsopoulos et al., 2004, Becerra and Coello, 2004). DE with self-adaptive parameter selection scheme also has been extended to solve multi-objective functions in (Huang and Qin et al., 2007, Huang et al., 2009, Zamuda et al., 2007). An extensive survey of different DE variants to solve multi-objective problems is presented in (Mezura-Montes et al., 2008).

Even though DE was formulated to solve the unconstrained objective functions, the robustness of DE has made the researchers to extend it further to solve the constrained objective functions also. In a general formulation of a constrained objective function, there exist three kinds of constraints viz. boundary constraints, inequality constraints and equality constraints. The boundary constraints, which are very common in real-world problems, are tackled by penalty methods. There are four penalty methods, used by DE, to handle the boundary constraints: (1) Brick wall penalty (Price et al., 2005) (2) Adaptive penalty (Storn, 1996a, 1996b) (3) Random re-initialization (Lampinen and Zelinka, 1999b, Price et al., 2005) and (4) Bounce-back (Price et al., 2005).

DE was first extended to solve inequality constraints by Storn (1999), in his CADE (constraint adaptation with DE). The CADE was also named as multimember DE. The CADE generates multiple trial vectors for each individual in the population. Mezura-Montes et al. (2005) used this concept for constrained optimization and engineering design (Mezura-Montes et al., 2007). The CADE added with dynamic stochastic ranking was proven to display promising results on CEC 2006 (Liang et al., 2006) competition for constrained optimization in (Zhang and Luo et al., 2008). Lampinen (2002) used DE to handle constrained problems. Mezura-Montes et al. (2004), Zielinski and Laur (2006b) used the feasibility rules given by Deb (2000) with DE to solve
constrained problems. A generalized approach based on DE to solve constrained problems was proposed in (Kukkonen and Lampinen, 2006).

There are many other research attempts which employ DE for solving constrained objective functions viz. (Takahama and Sakai, 2006, Muñoz-Zavala et al., 2006, Tasgetiren and Suganthan, 2006, Mezura-Montes et al., 2006a, Huang et al., 2006, Mezura-Montes and Palomeque-Ortiz, 2009, Ali and Kajee-Bagdadi, 2009, Santana-Quintero et al., 2010).

2.9 Differential Evolution for High Dimensional / Noisy Objective Function Optimization

Despite the fact that many improvements to DE have been attempted resulting multitude of DE variants, most of them have been used for solving low-dimensional problems. Classical DE and its variants have shown excellent search abilities in 30-100 dimensional problems. As the dimension increases beyond 500, their performance deteriorates. The reason for this performance decay is the exponential increase of the search space complexity with the dimension of the problem.

In literature, only limited studies have reported the scalability of DE derivative algorithms. In contrast, other evolutionary algorithms such as evolutionary programming (EP) have been tested on high-dimensional problems up to 1000 dimensions (Liu et al., 2001). A new local search method called fittest individual refinement (FIR) was added to DE by Noman and Iba (2005). The modified, memetic, version of DE showed improved performance than classical DE by solving the higher dimensional problems with lower number of function evaluations. Another memetic DE algorithm, which combines stochastic properties of chaotic system, simplex search method and DE operators, was presented in (Gao and Wang, 2007) for high-dimensional optimization. New variants of DE to solve high dimensional problems using co-operative co-evolution frame work (Potter and DeJong, 1994) have been proposed in (Yang et al., 2007b, Olorunda and Engelbrecht, 2007, Parsopoulos, 2009, Zamuda et al., 2008). Su (2008) proposed a new approach to DE, based on Gaussian process, for solving high dimensional problems. The new variant of DE algorithm \( jDEdyNP-F \), proposed in (Brest et al., 2008) secured third position in CEC2008 (Tang et al., 2007) special session and competition on large scale global optimization.

Apart from high dimensionality, the real-world problems suffer from a range of uncertainties.
In a computing environment, there are four possible cases of uncertainties (Yaochu and Branke, 2005): Noisy fitness function, Changes in the design variables and/or environmental problems, Approximation error in the fitness functions and the global optimum point may get changed over time.

The deterministic selection of scale factor and the greedy selection method of DE make it inefficient to solve noisy optimization problems. Since DE is deterministic by its search logic, it tends to stagnate for noisy optimization problems. This performance of DE on noisy functions was shown experimentally in (Krink et al., 2004). The results showed that for noisy optimization problems the conventional DE performs worst compared to other EAs.

An improvement to DE on noisy problems was done by Das and Konar (2005) and Das et al. (2005c), by choosing scale factor randomly in (0.5, 1) and two new selection mechanisms. The DEOSA, which is the hybrid version of DE, optimal computing budget allocation technique and SA proposed in (Liu et al., 2008) worked well for noisy problems.

Rahnamayan et al. (2006) proposed an improved DE algorithm to deal with noisy optimization problems. This algorithm involved opposition-based learning for population initialization, generation jumping and improving population’s best member. The improved version of DE outperformed the classical DE in terms of convergence speed. Subsequently, Rahnamayan et al. (2008) proposed opposition-based DE (ODE) for faster global search and optimization (Tizhoosh (2005) introduced the concept of opposition-based learning). Due to the enhancement achieved by opposition based learning in DE, the ODE was found to outperform the classical DE. Rahnamayan et al. (2007a) proposed a time varying jumping rate (TVJR) for ODE. The results obtained based on TVJR showed that during exploration it is desirable to have higher jumping rate than during exploitation. The ODE has shown promising results for large scale optimization (Rahnamayan and Wang, 2008). In similar line, a new DE algorithm called Quasi ODE (QODE) was presented in (Rahnamayan et al., 2007b).

The DynDE proposed in (Mendes and Mohais, 2005) solved the slowly time-varying objective function. The DynDE uses several populations in parallel. The authors have shown experimentally that the DynDE solves the moving peak benchmark (Branke, 1999), efficiently. Brest et al. (2009) handled dynamic fitness landscape using DE with self-adaption of F and C, and multi population method. Their algorithm named as jDE secured top position in CEC2009.
(Li et al., 2008). In (Angira and Santosh, 2007), the trigonometric mutation logic was used with $DE$ for solving dynamic optimization problems in chemical engineering.

### 2.10 Parallel Evolutionary Algorithms

To solve more complex problems and relatively high-dimensional problems advanced models are added to the realm of evolutionary computation. One such model is island model, which is still being researched up on. The effectiveness of implementing the island model depends on various parameters involved in it. The influence of migration size (how many individuals migrate) and frequency (how frequently migration occurs) on island model is analyzed in (Skolicki and Jong, 2005). The results indeed identifies the strong influence of the migration interval, on the other hand the migration size plays minor role in the performance of the algorithm. The influence of the migration interval on the convergence nature of the algorithm is analyzed by measuring the population diversity. The results show that frequent migration makes the population lose the diversity soon, because individuals in one island dominate the individuals in another island. With large migration intervals the algorithm stops prematurely. The best performance was achieved with moderate migration intervals and small migration sizes (Skolicki and Jong, 2005).

Cantu-Paz (2001) investigated the impact of selecting migrants and replacements on selection pressure of parallel evolutionary algorithms. Four combinations of random and fitness-based emigration and replacement of existing individuals are considered with each combination causing a different selection pressure. The selection pressure was examined by measuring the take over time and the selection intensity. It was found in the experiments that choosing migrants or replacements according to their fitness (best migrants replace worst individuals) increases the selection pressure and makes the algorithm to converge significantly faster.

Skolicki and Jong (2004) proposed an idea to have multiple representations for the individuals in the populations of the islands of an island model to improve the performance of the evolutionary algorithm. During the migration process the individuals get transformed from one representation to another, helping the $EA$ under evolution to escape from the local optima. An island model with two islands is implemented, in which one island used binary encoding and the other island used standard reflective gray code. Spatial separation of the individuals in a population brings a qualitative change in the convergence behavior of the evolutionary algorithms. Two general approaches for this spatial separation are (i) island model: group the
individuals into subpopulations (ii) neighborhood model: the individuals are placed on a grid and are allowed to interact with some neighborhood.

Even though the outcome of this spatial separation is slowing down the information flow between the individuals the result of this is two-fold. In the negative side, it is undesirable because it can prevent successful mixing which may help to produce good solutions. On the other hand, in the positive side, it stops certain individuals dominating the population and helps to discover different feasible regions in the solution space. The migration process helps the populations in the island to exchange their information. The migration process is attributed to various parameters viz. size, frequency and migration policy (Cantu-Paz, 2001, Skolicki and Jong, 2004). It is possible to make the search more effective by having different representations inside the populations. With different representations the algorithm exhibits different evolving characteristics (Skolicki and Jong, 2004). The key reason for such behavior is the local optima of one representation may not necessarily be local optima of the other one, which in turn help the algorithm to escape from the basin of local optima.

An extensive review on parallelization techniques used in evolutionary algorithms is presented in (Alba and Tomassini, 2002). Their study focused on one of the algorithmic issues on designing efficient EA, called parallel models. The parallel implementation of Evolutionary Algorithms (PEAs) shows faster convergence and superior performance, due to the presence of structured population. In structuring the population the individuals in the population are spatially distributed either in the form of a set of islands (distributed EA - dEA) or a diffusion grid (cellular EA – cEA). The structured population is responsible for the benefits of PEAs. The essential features of the tools for parallelizing an EA have also been discussed in the work. A heuristic conclusion made about the empirical behavior of the parallel EA was that using the best individual for the migration process in the island models is advisable.

2.10.1 Parallel Differential Evolution Algorithm

Since the crossover and selection operations in DE are local in nature, parallel DE algorithm is a natural extension to sequential DE (unlike the other EAs). There has been several research works in the field of parallel DE. The investigation on DE parallelization has been carried out way back in 1999 in (Lampinen, 1999), as a first attempt to distribute DE across a cluster of computers connected in LAN. The whole population is retained in the master computer, and the selected
candidates are distributed to the other computers for performing the mutation and crossover operations. Tasoulis et al. (2004), proposed a new scheme for parallel DE where each processor in the cluster is assigned with a subpopulation of the whole population. Now, each subpopulation evolves independently and exchanges their best solution to replace a random solution (best solution of one subpopulation emigrating to random solution of neighbor subpopulation – known as migration process). This exchange is done at a certain frequency of generation (migration frequency), with ring topology (migration topology). It has been experimentally shown in (Tasoulis et al., 2004) that the performance of distributed DE is largely impacted by the information exchange among the subpopulations. Zaharie and Petcu (2003) introduced a coarse-gained parallelization of an adaptive differential evolution algorithm, based on multi-population model (Zaharie, 2004). A random connection topology is used to speedup the execution time and increase probability of convergence. A new migration scheme for parallel differential evolution to replace the oldest member of the population than the random one, in the migration process, is proposed in (Kozlov and Samsonov, 2006). The proposed scheme was observed to achieve high speed of convergence.

Ntipteni et al. (2006) implemented a parallel asynchronous differential evolution algorithm, where a unique population is distributed among the processors in a cluster, with master-slave architecture. The master processor assigns each candidate to each of the slave processor but the last candidate. Thus there are “population size – 1” numbers of slave processors in the cluster. The “slave.exe” program placed in the slave processors performs the evolution of a single candidate in the population. The exchange of information between the candidates in the slaves is done using shared text files.

Pavlidis et al. (2005) employed the distributed DE proposed in (Tasoulis et al., 2004) to train an artificial neural network. Similarly, a distributed DE with ring topology has been employed in (Kwedlo and Bandurski, 2006) to train a neural network. Continuing this application trend, Salomon et al. (2005) attempted parallelizing DE for solving a medical imaging problem. In similar lines Falco et al. (2007a, 2007b and 2007c) employed a distributed DE to solve image registration problem.

Subsequently, the research work in distributed DE have focused more towards proposing novel parametric schemes. Two new enhancements to parallel differential evolution are proposed in (Weber et al., 2011a). Apolloni et al. (2008) proposed migration based on probabilistic criterion.
A distributed evolution employing scale factor inheritance, a novel self-adaptive scheme has been proposed in (Weber et al., 2010). Weber et al. (2011c) studied the combination of structured populations and variable scale factors on three different variants of distributed DE proposed in (Apolloni et al., 2008, Salomon et al., 2005, Tasoulis et al., 2004). The same algorithm has been chosen, but modified to employ a DE with exponential crossover in (Weber et al., 2011b), to analyze the effect of combining a varying scale factor and the exponential crossover on large scale problems.

Rucinski et al. (2010) analyzed the impact of migration topology on island model based parallel differential evolution and simulated annealing. The two effects caused by the migration topology on the underlying optimization process are 1) speed of the information exchange 2) CPU overhead for additional information flow. The migration process could be synchronous or asynchronous. In synchronous migration the exchange of individual takes place in all the islands at the same time, by stopping the other computations. But in asynchronous migration each island exchange the individuals as soon as it is ready, without waiting for the other islands to get ready. The study considered various number of islands viz. 128, 256, 512 and 1024, for comparison. The study also considers a wide range of migration topologies : chain topology (unidirectional ring topology, bidirectional ring topology, Ring+1+2 topology and Ring+1+2+3 topology), torus topology, ladder topology, cart wheel topology, lattice topology, hypercube topology with ring, hypercube topology with torus, broadcast topology and fully connected topology. The migration rate was decided as 10% of population size in case of differential evolution and it is 1 individual for simulated annealing. The migration frequency was set as 100 generations for differential evolution and the end of each annealing cycle for simulated annealing. Asynchronous migration algorithm was taken into account for their study. Their experimental setup includes 14 different topologies, 4 distinct numbers of island, 3 benchmark problems and 3 base algorithms (DE, SA with tuned and untuned parameters). It brings forth 36 different set-ups, the parallel algorithms were ranked based on their performance in 36 different set-ups. The comparative results show that the migration topology is an important parameter for the parallel global optimization algorithms which uses the island model. The quality of the solution and the speed of convergence are largely affected by the migration topology. Based on the ranking done for the different migration topologies, the suggested choice of topology for DE is Ring+1+2+3 or Ring+1+2. For SA, the hypercube topology was found as good choice.
2.10.2 Co-operative Co-evolution with Differential Evolution

“As evolutionary algorithms are applied to the solution of increasingly complex systems, explicit notions of modularity must be introduced to provide reasonable opportunities for solutions to evolve in the form of interacting co-adapted subcomponents” (Yang et al., 2008a, Potter, 1997).

Most of the evolutionary algorithms find it difficult to solve the problems with high dimensions. The convergence speed of the algorithms decreases as the dimensions increase. Consequently, the EAs that perform well on low-dimensions fail in high-dimensional problems. To alleviate this, co-operative co-evolution architecture was first proposed by Potter for genetic algorithm, called CCGA (Potter and DeJong, 1994), and had been successfully applied to other evolutionary algorithms (Liu et al., 2001, Sofge et al., 2002, Bergh and Engelbrecht, 2004). The co-operative co-evolution, which implements the divide-and-conquer strategy, helps to solve large scale and complex problems through problem decomposition (Yang et al., 2008a). The general structure of co-operative co-evolution architecture (Liu et al., 2001) consists of three components viz. Problem decomposition (to divide the high-dimensional problem into number of smaller subcomponents), Subcomponent optimization (to optimize each subcomponent using a certain EA) and Subcomponent co-adaptation (to capture the interdependencies between the subcomponents, during the optimization).

In the context of DE, the co-operative co-evolution has been introduced, and was proposed as CCDE (Shi et al., 2005). However, CCDE only extended the problem domain up to 100 dimensions, which are relatively small for many real-world problems. Yang et al. (2007b) proposed two new DE variants with co-operative co-evolution framework (DECC-I and DECC-II), which use SaNSDE, for high dimensional optimization up to 1000 dimensions. A co-operative micro-DE, which uses small co-operative subpopulations, was proposed in (Parsopoulos, 2009). A new variant of DE which combines the co-operative co-evolution with log-normal self adaptation of control parameters was proposed in (Zamuda et al., 2008). It performed dimension decomposition mechanism for large scale optimization problems. Huang and Wang et al. (2007) proposed to use co-operative co-evolutionary approach with DE to solve constrained optimization problems.

The critical step in the co-operative co-evolution architecture is the problem decomposition, because of the interdependencies existing between the subcomponents. The two simple problem
decomposition methods are (Yang et al., 2008a): i) one dimensional based strategy and ii) splitting-in-half strategy. The one-dimensional based method divides the $n$-dimensional problem into $n$ one-dimensional problems. The splitting-in-half strategy divides the $n$-dimensional problem into two $n/2$-dimensional problems. But performance of both the methods for the nonseparable functions was not satisfactory. A new decomposition method (particularly for nonseparable functions) was proposed in (Yang et al., 2008a), named as grouping based strategy. An adaptive weighting strategy was also introduced in (Yang et al., 2008a), in order to strengthen the co-adaptation among the decomposed subcomponents (if they are independent). The SaNSDE (proposed in (Yang et al., 2008b)) was used as optimizer for each subcomponent. The authors proposed a co-operative co-evolution based SaNSDE, called as DECC-G, and experimentally showed that DECC-G works efficient for large optimization problem up to 1000 dimensions. In the similar line JACC-G, an improved version of the DECC-G, was proposed in (Yang et al., 2009). The JACC-G uses JADE (proposed in (Zhang and Sanderson, 2007b, 2009)) as EA for the subcomponent optimizer. The adaptive weighting strategy of DECC-G was also modified.

2.11 Conclusion

Almost all of the works described above concern research issues regarding the performance enhancement of DE. When it comes to distributed DE almost all of the existing work implements the distributed version with same variants on every island. Although the very idea of co-operative evolution by employing genetic algorithms with different configurations in each island has been attempted in (Herrera and Lozano, 2000). The possibility of mixing various DE variants in island based distributed framework has not yet been addressed, in literature. However there has been very few research works attempted in this direction. A variant of DE called SaDE has been proposed in (Qin et, 2009), which is essentially a serial DE which adapts different DE strategies in a pool of predetermined strategies at different phases of DE evolution. Mallipeddi et al. (2011) proposed a serial DE with ensemble of parameters and mutation strategies. Interestingly, Mallipeddi and Suganthan (2009) have proposed a DE with ensemble of population where fitness evaluation to each population is self-adapted. Weber et al. (2009) proposed a Distributed DE with explorative-exploitative populations (DDE-EEP) where a DE variant has been employed in one family for exploration and in other family of subpopulation for exploitation of the decision space.
This thesis attempts to mix DE variants with different characteristics to enhance the efficacy as a whole. In this thesis two novel classes of algorithms viz. distributed mixed variant DE (dmvDE) and distributed mixed variant DE and DDE (dmvD²E) have been proposed. Towards this, an extensive empirical comparative performance of the DE variants has been carried out in the next chapter.