Dependency Identification Technique for Large Scale Optimization Problems

Eman Sayed, Daryl Essam, and Ruhul Sarker
School of Engineering and Information Technology
UNSW@ADFA
Canberra, Australia
e.hasan@student.adfa.edu.au, d.essam@adfa.edu.au, r.sarker@adfa.edu.au

Abstract—**Large scale optimization problems are very challenging problems. Most of the recently developed optimization algorithms lose their efficiency when the dimensionality of the problems increases. Decomposing a large scale problem into smaller subproblems overcomes this drawback. However, if the large scale optimization problem contains dependent variables, they should be grouped into one subproblem to avoid a decrease in performance. In this paper, the Dependency Identification with Memetic Algorithm (DIMA) model is proposed for solving large scale optimization problems. The Dependency Identification (DI) technique identifies the best arrangement to group the dependent variables into smaller scale subproblems. These subproblems are then evolved using a Memetic Algorithm (MA) with a proposed self-directed Local Search (LS). As the subproblems of a nonseparable large scale problem may contain interdependent variables, the proposed model, DIMA, uses an Information Exchange Mechanism to maintain one value for all the instances of any independent variable in the different subproblems. A newly designed test suite of problems has been developed to evaluate the performance of DIMA. The first evaluation shows that the DI technique is competitive to other decomposition techniques in the literature in terms of consuming less computational resources and better performance. Another evaluation shows that DI makes the optimization of a decomposed large scale problem using DIMA as powerful as the optimization of a complete large scale problem using MA. This makes DIMA a promising optimization model for optimization problems which can be 10 times larger (or more) than the large scale optimization problems under consideration in this paper.**

Keywords- Memetic Algorithms; large scale problems; dependency identification; problem decomposition; Local Search.

I. INTRODUCTION

Many real life problems (in industry, environment, bio-computing, and operations research, etc.) are large scale optimization problems. Solving large scale optimization problems has become a challenging research topic, mainly due to both the increased need for high quality decision making in practice, and also the increased computational power. Recently developed optimization algorithms use Evolutionary Algorithms (EAs) to solve such large scale optimization problems. However, the performance of EAs eventually decreases with the increased dimensionality of optimization problems [1]. There are two solutions to overcome this drawback. The first solution is to apply a decomposition approach, and the second solution is to use a hybridization approach.

The first attempt for decomposition was the Cooperative Coevolution (CC) approach [2]. This decomposition approach breaks the large scale problem into smaller subproblems. The CC approach is very successful with separable optimization problems where the variables are independent from each other, but loses its efficiency with nonseparable optimization problems [2]. This is because in nonseparable optimization problems at least one variable is optimized in one subproblem and depends on other variables which are optimized in different subproblems. These variables are known as interdependent variables. To improve the performance of the CC approach with nonseparable large scale optimization problems, the interdependent variables should be minimized. This emphasizes the importance of developing a technique to detect the best arrangement of variables that decreases the interdependent variables as much as possible when the large scale problem is decomposed into subproblems. Some such grouping approaches were developed recently, as Random Grouping (RG) technique [3], correlation based Adaptive Variable Partitioning (AVP) [4], Delta Grouping [5], and Variable Interactive Learning (VIL) [6]. However, some of these techniques don't have a systematic way to clearly identify the dependencies amongst the variables before commencing the optimization process, while other techniques have large computational requirements which could be better used for the optimization process. This motivates the development of a new Dependency Identification technique (DI).

Although the decomposition approach helps to overcome the dimensionality problem, the EA needs to be more reliable and powerful when it is applied to nonseparable large scale problems. The hybridization of the EAs with other techniques has been proven to enhance the performance of the optimization algorithms when solving large scale optimization problems [7]. Local Search (LS) is one of these techniques that can be hybridized with EAs. The hybridization of EA and LS are called the Memetic Algorithms (MA) [8]. This paper introduces a new large scale optimization model, Dependency

Identification with Memetic Algorithm (DIMA). DIMA is inspired by the two solutions for the dimensionality problem which were discussed earlier.

A new test suite of 12 unconstrained large scale optimization problems is proposed in this paper. The new test suite is based on the basic problems in CEC2008 [9]. The reason behind developing a new test suite and not using the CEC2010 test suite [10] is that most of the problems in CEC2010 are represented as independent subcomponents of dependent variables. There is no *overlapping* representation in any of these problems. This sort of structure does not help in testing the performance of the decomposition techniques. Although using a newly designed test suite decreases the opportunity of comparing DI to other techniques, the newly proposed test suite adds more challenge to the decomposition techniques by representing *overlapping* and dependent *spliced* variables. The first experiment in this paper is conducted to evaluate the performance of DI and to compare it to one of the successful decomposition techniques in the literature. The results show the competitive performance of the DI technique. Another experiment in this paper shows the contribution of the DI in achieving a performance that makes DIMA as powerful as the optimization models of the large scale problems without decomposition.

This paper is organized as follows: Section II is the literature review of the grouping techniques. Section III discusses the proposed model. The Experiment and results are presented in section IV and V. Finally, the conclusion and future work in section VI.

II. LITERATURE REVIEW

Problem decomposition helps to enhance the performance of the optimization algorithms by decomposing the large scale problems into small subproblems. The first attempt for problem decomposition is the Cooperative Coevolution (CC) approach [2]. The initial strategies that were developed based on CC are the one-dimension based strategy, and the splitting-in-half strategy [11-12]. The one-dimension based strategy decomposed a N dimension large scale problem into N subproblems. The second approach produces two subproblems of size $\frac{N}{2}$which is still a large scale optimization subproblem if N is very large. Different smaller subproblem sizes were used by other researchers [13]. The subproblem of size 100 has been found to be convenient for both the separable and the nonseparable problems [3]. However, the performance of the optimization model still depends on identifying the variables dependency. If the dependent variables in a large scale optimization problem are grouped into different subproblems, the performance of the optimization algorithm will decrease. This motivates the development of decomposition techniques to detect and group the dependent variables into subproblems.

One of the recently developed decomposition techniques is Random Grouping (RG) [3]. In it, variables of a large scale problem are grouped randomly into smaller subproblems. RG has achieved good performance in the literature [1, 3], but it does not have a systematic way to detect the dependencies among the variables. Following afterwards, the grouping technique of correlation based Adaptive Variable Partitioning [4] was developed. This technique creates a correlation matrix at each generation during the evolution process. Creating a correlation matrix for a large scale problem consumes computational resources. Moreover, this technique does not detect nonlinear dependencies amongst the variables, but only detects the linear dependencies of the variables. The Delta Grouping technique [5] was developed later to group the dependent variables into subproblems. This technique calculates the magnitude of change that happens through one generation during the evolution of the subproblems. The grouping is updated every generation after recalculating the magnitude of change. Although this technique seems to be reasonable for large scale nonseparable optimization problems, it is less efficient when the problem has more than one nonseparable group [5], as in some problems in the CEC2010 test suite [10]. The Variable Interactive Learning (VIL) [6] technique was developed recently to decompose a large scale optimization problem into subproblems. VIL is used in the learning stage of the optimization model CCVIL [6]. In that model, a large scale optimization problem is decomposed into one-dimension subproblems and a learning stage starts. The current and previous populations are tested after optimizing each subproblem. The subproblems which affect each other are merged and then the newly formed subproblems are optimized. The initial one-dimension based decomposition in this technique is not suitable for large scale problems of dimension N because the initial subproblems will be as many as the dimensions of the large scale problem (N subproblems). Also, the merge of the subproblems will probably reform large scale subproblems and there is no size limit to the subproblems. Another critical disadvantage of the VIL technique is that it consumes up to 60% of the total evaluations at the learning stage [6], which leaves at least 40% for the optimization stage in CCVIL. Some of the decomposition techniques in the literature are not systematic. Other techniques consume many computational resources in the decomposition process, which could be better used for the optimization process. These drawbacks motivate the need for developing a dependency identification technique that is capable of detecting the dependencies of the variables in a systematic way, and that uses computational resources efficiently.

III. THE PROPOSED MODEL

The proposed model in this paper is Dependency Identification with Memetic Algorithm (DIMA) for solving large scale optimization problems. DIMA is a new Dependency Identification (DI) technique that is developed and integrated with a Memetic Algorithm (MA). The MA of the proposed model uses a self-directed Local Search (LS) with adaptive search steps. DIMA consists of two main stages. The first stage is dependency identification and decomposition. The second stage is optimization and information exchange. A framework for the proposed model is presented in Figure 1.

A. Dependency Identification (DI) and Problem Decomposition

The first stage of DIMA applies the proposed DI technique. The DI technique is inspired from the definitions of the problem separability in the literature [6, 10, 14]. DI identifies the best arrangement for the variables based on their

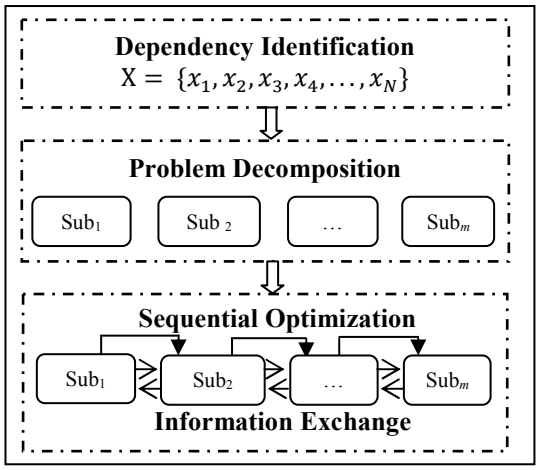

Figure 1: DIMA

dependencies. The following section describes in detail the mechanism of the proposed DI technique. The detailed pseudo code of the DI is shown in Figure 2.

The problem separability definition in [14] defines a fully separable problem as a problem that can be written in the form of linear combinations of subproblems of individual variables where $F(x) = \sum_{i=1}^{N} f(x_i)$. A partially separable problem is defined in [4] as $F(x) = \sum_{k=1}^{m} f_k(x_v),\ v = [1, V], N = m * V$. Where N is the total number of variables of the large scale optimization problem $F(x)$ which is decomposed into m equal size subproblems. Each subproblem has V dependent variables and no variable is represented in more than one subproblem. A fully nonseparable problem that is optimized without decomposition is defined as a special case of the partially separable problem with one subproblem (m=1) and the variables in the subproblem when $V = N$.

From these definitions it is obvious that the best grouping for the variables to decompose a large scale optimization problem into subproblems should minimize the number of interdependent variables, where interdependent variables are variables that have more than one instance in two or more subproblems. This can be achieved by minimizing the least square difference of $F(x)$ and the summation of all $f_k(x_v)$, $v = [1, V]$, as defined in equation (1). Inspired by that, the proposed DI technique finds the arrangement of variables with the minimum least square difference as in equation (2). Finding the least square difference (sq_diff) is an internal minimization optimization problem, which is solved during the external optimization of the large scale problem itself. The evaluation of this internal optimization problem is implemented as follows. The complete solution of the large scale optimization problem is solved for all the variables $x_i = c_1, \forall\ i = [1, N], and\ c_1 \neq 0$, to get $F\left(x_{all_{c_1}}\right)$ and all the variables $x_i = c_2, \forall\ i = [1, N], and\ c_2 \neq 0$, to get $F\left(x_{all_{c_2}}\right)$. The summation of these two solutions is $total_c$ as in equation (3). The N variables are arranged and decomposed into m subproblems. Each subproblem consists of V variables. The V variables of one subproblem k are set to a certain value c_1, where $c_1 \neq 0$, and the rest of the $(N - V)$ variables are set to another value c_2, where $c_2 \neq 0$. $f\left(x_{c_{1_{sub_k}}}\right)$ is calculated for this subproblem k. The same is done to calculate $f\left(x_{c_{1_{sub_k}}}\right)$ for all the m subproblems. The same procedure is applied with the variables V of one subproblem k being set to c_2 and the rest of the variables $(N - V)$ are set to the value c_1 to calculate $f\left(x_{c_{2_{sub_k}}}\right)$ for all the m subproblems. The summation of $f\left(x_{c_{1_{sub_k}}}\right)$ and $f\left(x_{c_{2_{sub_k}}}\right)$ for all the subproblems m is $subs_c$ as shown in equation (4). Note that $c_1 \neq 0$ and $c_2 \neq 0$ to avoid mathematical errors when using DI. This will also generalize the DI technique to be applicable to variant large scale optimization problems. The squared difference is calculated as in equation (2) where $total_c$ is multiplied by m to make it equivalent to when $f\left(x_{c_{1_{sub_k}}}\right)$ and $f\left(x_{c_{2_{sub_k}}}\right)$ were calculated m times.

$$min\ \ [F(x) - \sum_{k=1}^{m} f_k(x_v)]^2, v = [1, V] \tag{1}$$

$$sq\_diff = (m * total_c - subs_c)^2 \tag{2}$$

$$total_c = F\left(x_{all_{c_1}}\right) + F\left(x_{all_{c_2}}\right) \tag{3}$$

$$x_{i_{all_{c_1}}} = c_1\ \forall\ i = [1, N],\ x_{i_{all_{c_2}}} = c_2\ \forall\ i = [1, N]$$

$$subs_c = \sum_{k=1}^{m}\left(f\left(x_{c_{1_{sub_k}}}\right) + f\left(x_{c_{2_{sub_k}}}\right)\right) \tag{4}$$

$$x_{c_{1_{sub_k}}} = \begin{cases} c_1\ \forall\ x = [1, V]\ in\ sub_k \\ c_2\ otherwise \end{cases}, k = \{1,2,3, \dots, m\}$$

$$x_{c_{2_{sub_k}}} = \begin{cases} c_2\ \forall\ x = [1, V]\ \text{in}\ sub_k \\ c_1\ otherwise \end{cases}, k = \{1,2,3, \dots, m\}$$

$total_c$ is calculated once only at the beginning of the optimization process, while $subs_c$ is calculated at the beginning of every cycle. A cycle is the process of optimizing all the m subproblems sequentially for one generation (Gen). After finding the least square difference (sq_diff), the large scale optimization problem is decomposed into equal size subproblems based on the best arrangement from the DI technique, and the optimization stage starts.

The proposed Dependency Identification technique (DI) is competitive to the other decomposition techniques in the literature in terms of using the computational resources and the number of Fitness Evaluations (FE) that are consumed in the decomposition process. This gives the opportunity for more evaluations to be assigned to the optimization process. Moreover, DI follows a systematic way to identify the best arrangement of the variables that should be grouped together in an arrangement that minimizes the interdependent variables among the subproblems. Thus, DI is expected to overcome the drawback of the decomposition techniques in [5] and to perform well on separable and nonseparable large scale optimization problems.

B. Subproblem Optimization and Information Exchange

The second stage after decomposing the large scale optimization problem into equal size smaller scale subproblems has two tasks. The first task is to optimization these subproblems. The second task is to maintain the representation of only one instance for each interdependent variable during the optimization. Although the EAs are very successful optimization techniques, their performance decreases when the dimensionality of a problem increases. This disadvantage is solved by using the CC decomposition approach and by hybridizing the EAs with other techniques as Local Search (LS). It is well known that hybridizing EA with other techniques positively improves the efficiency of the optimization algorithm [7]. The hybridization of EA and LS forms what are called Memetic Algorithms (MAs). MAs are capable of searching noisy search spaces efficiently [15]. This encourages the use of MA in the optimization stage of the proposed model. MA uses GA [16] and a self-directed LS. One GA crossover operator is the Simulated Binary Crossover (SBX) [17], which generates offspring y^1 and y^2 of two parents x^1, x^2 as in equations (5) and (6). η is a constant value (η=*2* is used by most researchers [18]).

$$y_i^1 = \frac{1}{2}\left[\left(1+\overline{\beta}\right)\times x_i^1 + \left(1-\overline{\beta}\right)\times x_i^2\right], i=[1,N] \quad (5)$$

$$y_i^2 = \frac{1}{2}\left[\left(1+\overline{\beta}\right)\times x_i^2 + \left(1-\overline{\beta}\right)\times x_i^1\right], i=[1,N] \quad (6)$$

$$\bar{\beta} = \begin{cases} (2u)^{\frac{1}{1+\eta}}, u \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{1+\eta}}, otherwise \end{cases} \quad (7)$$

The mutation operator $\delta_{i,j}(t)$ is an adaptive nonuniform mutation [19]. It follows a uniform mutation in the initial search space, and then the search step gets smaller as the algorithm proceeds. Offspring $x_i^{'}(t)$ is created as in equation (8).

$$x_i^{'}(t) = \left(x_{i,1}^{'}, x_{i,2}^{'}, \dots, x_{i,n}^{'}\right), x_{i,j}^{'} = x_{i,1}^{'} + \delta_{i,j}(t) \quad (8)$$

$$\delta_{i,j}(t) = \begin{cases} \left(x_{i,upper} - x_{i,j}\right)\times\left(1-[(u(t)]^{\left(1-\frac{t}{T}\right)^b}\right), probability\ 0.5 \\ \left(x_{i,lower} + x_{i,j}\right)\times(1-[(u(t)]^{(1-\frac{t}{T})^b}), probability\ 0.5 \end{cases} \quad (9)$$

LS is a technique, that guides the search to the most promising solution area. In the proposed DIMA, self-directed LS uses variant search steps, $d^{'}$, based on the improvement of the search process as in equation (10). Small search steps concentrate the search in a smaller area using small search steps if the new solution $f_{d'}(x)$ using $d^{'}$ is better than the previous solutions, while larger search steps allow the exploration of new search areas and avoids local optima. This self-directed local search decreases the greediness in LS that can cause premature convergence [20].

$$d^{'} = \begin{cases} d + e^{\ 1/d} & if\ f_{d'}(x) < f_d(x) \\ d + e^{\ -1/d} & otherwise \end{cases} \quad (10)$$

Although the proposed DI finds the best arrangement for the subproblems, there is always a probability in the large scale nonseparable problems for at least two dependent variables to be grouped or optimized in different subproblems. When this happens, each subproblem will have an instance of the other dependent variable that is grouped in another subproblem. These variables are interdependent variables. An interdependent variable is a variable which has more than one instance in two or more subproblems and that is hence being optimized in another different subproblem. During the optimization of a subproblem, the instances of the interdependent variables are required for the evaluation of the optimization process. When optimizing the subproblems using the proposed DIMA model, the values of the variables will differ from their initial values during the optimization. So there should be a technique to update the value of the instances of the interdependent variable after the optimization of each subproblem. This motivates the need for an Information Exchange Mechanism to maintain one value for all the instances of any interdependent variable.

An Information Exchange Mechanism is used in DIMA. It is performed in the second stage during the optimization of the subproblems as shown in Figure 1. This Information Exchange Mechanism guarantees that each variable in the large scale optimization problem has only one instance during the optimization process. After optimizing any k subproblem, the latest best V variables of this subproblem are updated in the complete solution vector X. When another subproblem with V variables is being optimized and it requires any variable from the other $N-V$ variables (as interdependent variables) for evaluation, there will be only one instance for these variables with the latest best value. The Information Exchange Mechanism is important because if the complete solution vector X is updated only at the end of the cycle, then the optimization process will not be accurate. The subproblems will use the initial value of the interdependent variables which have been changed after the optimization in their original subproblem.

The advantage that DI provides to the DIMA model is that it enhances the performance by minimizing the interdependent variables. To keep one instance for every interdependent variable, there must be communication among the subproblems. This communication will add to the computational cost of DIMA. So minimizing the interdependent variables will minimize the required information to be exchanged, and the computational resources. The saved computational resources will be used for optimization to enhance the performance. The detailed steps of the two stages of DIMA are shown in Figure 2.

IV. Experiments

Two experiments are carried out in this paper to evaluate the performance of the proposed model - DIMA. The first experiment shows the effect of applying the newly developed decomposition technique DI. In the first experiment, DI is compared to the decomposition technique RG which has the least disadvantages and achieved good performance in the literature [1, 3, 21]. The second experiment investigates the performance of DIMA to the performance when optimizing large scale problems without decomposition. This experiment is inspired from the finding that nonseparable problems achieve better performance when they are optimized in large dimensions [21].

1. Initialize population *NP* for *N* variables.
2. Calculate $total_c$
3. Decompose the N variables into m subproblems of fixed size s
4. for $k = 1$ to $k = m$
   a. randomly rearrange 10% of the N variables in S_v to get $S_{\bar{v}}$
      $S_v = \{x_1, x_2, \dots, \dots, x_N\}$,
      $S_{\bar{v}} = \{x_5, x_{N-10}, \dots, \dots, x_6\}$
   b. calculate $f\left(x_{c_{1_{sub\,k}}}\right)$ and $f\left(x_{c_{2_{sub\,k}}}\right)\ \forall\ k$
   c. calculate $subs_c$
   d. get the sq_diff as in equation (5)
   e. choose *S* of the smallest sq_diff
   f. rearrange S into m subproblems
5. Optimise the subproblems
6. Information Exchange among subproblems
7. If *FE*< *max_FE* go to step 4
8. End

Figure 2: Pseudo code of DIMA

These two experiments are conducted on a new test suite of 12 unconstrained large scale optimization problems based on the basic problems of CEC2010 [10]. The CEC2010 test suite has some problems with *D*/subproblem_size nonseparable components but it does not have an *overlapping* representation in any of the 20 benchmark problems. Also the *D*/subproblem_size nonseparable components are independent. These reasons motivate the design of a new test suite of benchmark problems. The proposed test suite is scalable to be used for large dimensions. It also covers different degrees of separability and *overlapping* among the variables to represent real life problems where the variables are rarely independent. The Sphere, Elliptic, Rastrigin's, and Ackley problems are fully separable problems, while the Schwefel's and Rosenbrock's problems are fully nonseparable.

The problems of the proposed test suite are categorized into 3 main categories; A) nonseparable problems, B) *overlapping* nonseparable, C) *overlapping* partially nonseparable. The first category consists of the two basic nonseparable problems Schwefel's and Rosenbrock's where the variables are divided between these two nonseparable problems. The difference between F_1 and F_2 is that in F_1 the variables are sequentially dependent. While in F_2 any two following variables belong to two different basic nonseparable problems. Any problem in the proposed test suite which has the same description as F_2 is called a *spliced* problem. In the second category, some variables are common between the two main nonseparable problems. These variables are called *overlapping* variables (*ov*). The variables in F_3 are sequentially divided and in F_4 are *spliced* between the two nonseparable problems. The third category is a partially separable category where some variables (*sep*) are represented by a separable problem of the basic fully separable problems (Sphere, Elliptic, Rastrigin's, and Ackley's) and the rest of the variables (*D-sep*) are shared between the two fully nonseparable problems sequentially as in problems from F_5 to F_8 or *spliced* as in problems F_9 to F_{12}. *ov* are represented sequentially at the *overlapping* problems (F_3, F_5 to F_8) or shared equally amongst the subproblems of the *spliced overlapping* problems (F_4, F_9 to F_{12}). The proposed test suite of problems is defined as follows:

A. Nonseparable problems

F1: $\frac{D}{2}$ Schwefel's and $\frac{D}{2}$ Rosenbrock's

$$F_1 = \sum_{i=1}^{\frac{D}{2}} \left[F_{schwefel}[x_i] + F_{rosenbrock}[x_{i+\frac{D}{2}}]c\right]$$

F2: *Spliced* $\frac{D}{2}$ Schwefel's and $\frac{D}{2}$ Rosenbrock's

$$F_2 = \sum_{i=1}^{\frac{D}{2}} \left[F_{schwefel}[x_{2i-1}] + F_{rosenbrock}[x_{2i}]\right]$$

B. Overlapping nonseparable problems

F3: $\frac{D}{2}$ Schwefel's and $\frac{D}{2} + ov$ Rosenbrock's

$$F_3 = \sum_{i=1}^{\frac{D}{2}} \left[F_{schwefel}[x_i] + F_{rosenbrock}\left[x_{i+\frac{D}{2}}\right]\right] + \sum_{j=\frac{D}{2}-ov}^{\frac{D}{2}} F_{rosenbrock}[x_j]$$

F4: *Spliced* $\frac{D}{2}$ Schwefel's $\frac{D}{2} + ov$ Rosenbrock's

$$F_4 = \sum_{i=1}^{\frac{D}{2}} \left[F_{schwefel}[x_{2i-1}] + F_{rosenbrock}[x_{2i}]\right] + \sum_{n=1}^{\frac{D}{ov}} \sum_{j=1}^{\frac{D}{ov}} F_{rosenbrock}\left[x_{(2j-1)+(n-1)*s\_size}\right]$$

C. Overlapping Partially nonseparable problems

F5: Sphere $\frac{D-sep}{2}$ Schwefel's $\frac{D-sep}{2} + ov$ Rosenbrock's

$$F_5 = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_i] + F_{rosenbrock}\left[x_{i+\frac{D-sep}{2}}\right]\right] + \sum_{j=\frac{D-sep}{2}-ov}^{\frac{D-sep}{2}} F_{rosenbrock}[x_j] + \sum_{k=D-sep}^{D} F_{sphere}[x_k]$$

F6: Elliptic $\frac{D-sep}{2}$ Schwefel's $\frac{D-sep}{2} + ov$ Rosenbrock's

$$F_6 = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_i] + F_{rosenbrock}\left[x_{i+\frac{D-sep}{2}}\right]\right] + \sum_{j=\frac{D-sep}{2}-ov}^{\frac{D-sep}{2}} F_{rosenbrock}[x_j] + \sum_{k=D-sep}^{D} F_{elliptic}[x_k]$$

F7: Rastrigin's $\frac{D-sep}{2}$ Schwefel's $\frac{D-sep}{2} + ov$ Rosenbrock's

$$F_7 = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_i] + F_{rosenbrock}\left[x_{i+\frac{D-sep}{2}}\right]\right] + \sum_{j=\frac{D-sep}{2}-ov}^{\frac{D-sep}{2}} F_{rosenbrock}[x_j] + \sum_{k=D-sep}^{D} F_{rastrigin}[x_k]$$

F8: Ackley's $\frac{\mathbf{D-sep}}{\mathbf{2}}$ Schwefel's $\frac{\mathbf{D-sep}}{\mathbf{2}} + \boldsymbol{ov}$ Rosenbrock's

$$F_8 = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_i] + F_{rosenbrock}\left[x_{i+\frac{D-sep}{2}}\right] \right] + \sum_{j=\frac{D-sep}{2}-ov}^{\frac{D-sep}{2}} F_{rosenbrock}[x_j] + \sum_{k=D-sep}^{D} F_{ackley}[x_k]$$

F9: *Spliced* Sphere $\frac{\mathbf{D-sep}}{\mathbf{2}}$ Schwefel's and $\frac{\mathbf{D-sep}}{\mathbf{2}} + \boldsymbol{ov}$ Rosenbrock's

$$F_9 = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_{2i-1}] + F_{rosenbrock}[x_{2i}] \right] + \sum_{n=1}^{\frac{D-sep}{ov}} \sum_{j=1}^{\frac{D}{ov}} F_{rosenbrock}[x_{(2j-1)+(n-1)*s\_size}] + \sum_{k=D-sep}^{D} F_{sphere}[x_k]$$

F10: *Spliced* Elliptic $\frac{\mathbf{D-sep}}{\mathbf{2}}$ Schwefel's $\frac{\mathbf{D-sep}}{\mathbf{2}} + \boldsymbol{ov}$ Rosenbrock's

$$F_{10} = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_{2i-1}] + F_{rosenbrock}[x_{2i}] \right] + \sum_{n=1}^{\frac{D-sep}{ov}} \sum_{j=1}^{\frac{D}{ov}} F_{rosenbrock}[x_{(2j-1)+(n-1)*s\_size}] + \sum_{k=D-sep}^{D} F_{elliptic}[x_k]$$

F11: *Spliced* Rastrigin's $\frac{\mathbf{D-sep}}{\mathbf{2}}$ Schwefel's $\frac{\mathbf{D-sep}}{\mathbf{2}} + \boldsymbol{ov}$ Rosenbrock's

$$F_{11} = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_{2i-1}] + F_{rosenbrock}[x_{2i}] \right] + \sum_{n=1}^{\frac{D-sep}{ov}} \sum_{j=1}^{\frac{D}{ov}} F_{rosenbrock}[x_{(2j-1)+(n-1)*s\_size}] + \sum_{k=D-sep}^{D} F_{rastrigin}[x_k]$$

F12: *Spliced* Ackley's $\frac{\mathbf{D-sep}}{\mathbf{2}}$ Schwefel's $\frac{\mathbf{D-sep}}{\mathbf{2}} + \boldsymbol{ov}$ Rosenbrock's

$$F_{12} = \sum_{i=1}^{\frac{D-sep}{2}} \left[F_{schwefel}[x_{2i-1}] + F_{rosenbrock}[x_{2i}] \right] + \sum_{n=1}^{\frac{D-sep}{ov}} \sum_{j=1}^{\frac{D}{ov}} F_{rosenbrock}[x_{(2j-1)+(n-1)*s\_size}] + \sum_{k=D-sep}^{D} F_{ackley}[x_k]$$

$$ov = 0.1 * (D - sep), sep = \begin{cases} 0 \; for \; F_1, F_2, F_3, and \; F_4 \\ 0.1 * D \; otherwise \end{cases}$$

$$x \in [-100, 100]^D \quad , s\_size = 100$$

The two experiments have been conducted using the same parameter settings as in the literature [5-6, 21]: problem dimension $D = 1000$, population size $NP = 50$, subproblem size (s_size) =100, maximum number of Fitness Evaluations (Max_FE) = 3E+6, $c_1 = 1$ and $c_2 = 2$. The proposed algorithm is executed for 25 independent runs for each problem in the proposed test suite. The details of the implementation of the two experiments and the results are discussed and analysed in the following section.

V. RESULTS

The first experiment in this paper investigates the performance of the proposed decomposition technique, DI, against the RG over the proposed test suite problems. The MA with self-directed LS is used as the subproblems' optimizer with both DI and RG in the DIMA and RGMA models. The results from this experiment are represented in Table 1. When analyzing the results of Table 1, it is found that DIMA achieved higher performance in 8 out of 12 problems ($F_1, F_2, F_4, F_7, F_9, F_{10}, F_{11}, and\ F_{12}$) and achieved similar results to RGMA in the other 4 problems($F_3, F_5, F_6, and\ F_8$). DIMA was better than RGMA in the two fully nonseparable problems $F_1$and F_2. DIMA performed as good as RGMA at F_3 and outperformed RGMA when the nonseparability of the problem was complex and the dependent variables are *spliced* as in F_4. The average value of DIMA is better than RGMA in the category of the *overlapping* partially nonseparable problems ($F_5$to F_8). Separability is represented by Ackley's multimodal separable problems in the *overlapping* partially nonseparable problem F_8 and the dependent variables are not *spliced*. Although RG is known to be good at the Ackley's problems [20], DIMA achieved better average and very close performance to RGMA in F_8. Also when *spliced* variables were added to the complexity of *overlapping* and nonseparability, and even though RG is good at solving Ackley's which is represented in F_{12}, DI achieved better performance in all the *spliced overlapping* partially nonseparable problems ($F_9$to F_{12}). This indicates the ability of DIMA to outperform RGMA in almost all forms of nonseparable, partially nonseparable, *spliced* and *overlapping* problems. DI is more robust than RG as the standard deviation value of DIMA is smaller than that of RGMA for all the problems of the proposed test suite.

Another advantage of DI over the other decomposition techniques in the literature [6] is that the percentage of evaluations used by DI (FE_{DI}) is approximately 1.16% of the overall evaluations in comparison to the 60% used in the VIL decomposition technique [6]. This leaves 98.84% of the evaluations (not 40% as VIL) to be used for the optimization process when solving a large scale optimization problem. This may be calculated because $max\_FE \approx FE_{init} + FE_{DI} + FE_{opt}$, where $FE_{init} = NP + 2 \times m$, $FE_{DI} = Generations \times 2 \times m$, $FE_{opt} = Generations \times m \times (NP + m \times FE_{LS})$, $FE_{LS} = 2 + m$, $max\_FE =$ 3E+6, and $m = 10$. This is performed during the evolutions process in Figure 2 in step 4.b.

The convergence of DIMA and RGMA is recorded in Table 2 for certain Fitness Evaluations (FEs). One main observation from these results is that DIMA converges faster towards a reasonable solution after 5E+5 and 1E+6 FEs, which is approximately a third of the number of FEs that RGMA requires to reach the same solution. This observation shows that DIMA saves time and the resources to the one third mark in comparison to RGMA. Although LS is shown to be useful, its greedy nature might result in an early premature convergence. This premature convergence can be justified

when the computational resources are very limited, however, DIMA can use these limited resources efficiently to get a fairly reasonable solution.

The second experiment in this paper investigates the performance of DIMA against the same optimizer MA with self-directed LS without using any CC decomposition approach. These two models are tested on the proposed test suite problems with the same parameter setting of the first experiment. The results are presented in Table 1. MA is expected to achieve a performance which is higher than any optimization model that uses the CC approach when solving nonseparable problems [21]. However, DIMA can cope with MA as it achieved similar performance to MA at 8 of the problems in terms of the best value (F_1 to F_3, F_5 to F_8, and F_{12}) and achieved better performance at 3 *overlapping* problems in terms of the mean value (F_3, F_5, and F_7). DIMA achieved similar performance at the *spliced* nonseparable problem F_2, but its performance was slightly lower than MA for the *spliced overlapping* nonseparable problem F_4, and the *spliced overlapping* partially nonseparable problems F_9 to F_{11}. This shows that the performance of DIMA is affected when the problem structure is both *spliced* and has *overlapping* variables. On the other hand, RGMA was only as good as MA in 4 of the problems (F_3, F_5, and F_8) in terms of the best value and 1 problem in terms of the mean value (F_3). This shows that DIMA can achieve performance as good as MA for large scale problems by optimizing smaller scale subproblems.

Table 1: DIMA, RGMA, and MA for D=1000

Model		F1	F2	F3	F4
DIMA	Best	**4.91E+02**	**4.93E+02**	**9.92E+02**	5.89E+02
	Med.	5.78E+02	1.76E+03	9.93E+02	8.06E+02
	Worst	2.52E+05	2.92E+05	1.42E+03	1.36E+05
	Mean	2.04E+04	1.11E+05	**1.07E+03**	4.67E+04
	Std	5.95E+04	1.31E+05	1.65E+02	6.21E+04
RGMA	Best	4.93E+02	5.01E+02	9.92E+02	5.91E+02
	Med.	2.92E+05	2.93E+05	9.93E+02	1.38E+05
	Worst	4.82E+05	4.31E+05	2.58E+03	1.92E+05
	Mean	2.33E+05	2.40E+05	1.27E+03	9.09E+04
	Std	1.65E+05	1.49E+05	3.78E+02	7.99E+04
MA	Best	**4.91E+02**	**4.93E+02**	**9.92E+02**	**5.88E+02**
	Med.	1.85E+04	2.19E+04	1.41E+03	1.36E+04
	Worst	3.72E+04	3.29E+04	2.54E+03	1.52E+04
	Mean	**1.53E+04**	**1.99E+04**	1.34E+03	**1.01E+04**
	Std	1.19E+04	9.88E+03	3.54E+02	6.13E+03
		F5	**F6**	**F7**	**F8**
DIMA	Best	**5.45E+02**	**5.45E+02**	**5.45E+02**	**5.47E+02**
	Med.	5.45E+02	5.46E+02	5.45E+02	7.04E+02
	Worst	1.55E+04	1.18E+05	1.02E+05	1.35E+05
	Mean	**2.37E+03**	9.64E+03	**8.70E+03**	3.27E+04
	Std	4.15E+03	2.96E+04	2.42E+04	5.32E+04
RGMA	Best	5.45E+02	5.45E+02	5.50E+02	5.47E+02
	Med.	1.42E+05	1.27E+05	1.30E+05	1.32E+05
	Worst	2.36E+05	2.04E+05	1.77E+05	1.77E+05
	Mean	1.16E+05	9.51E+04	9.53E+04	8.53E+04
	Std	7.47E+04	7.28E+04	7.05E+04	6.92E+04
MA	Best	**5.45E+02**	**5.45E+02**	**5.45E+02**	**5.47E+02**
	Med.	9.73E+03	9.69E+03	9.68E+03	1.06E+04
	Worst	1.20E+04	1.44E+04	1.40E+04	1.31E+04
	Mean	7.83E+03	**7.38E+03**	8.91E+03	**8.42E+03**
	Std	4.25E+03	5.39E+03	3.94E+03	4.63E+03
		F9	**F10**	**F11**	**F12**
DIMA	Best	5.30E+02	5.25E+02	5.25E+02	**5.25E+02**
	Med.	6.13E+02	5.89E+02	7.27E+03	6.63E+03
	Worst	8.65E+04	8.13E+04	9.68E+04	8.37E+04
	Mean	2.36E+04	1.34E+04	3.79E+04	3.32E+04
	Std	3.62E+04	2.82E+04	4.20E+04	3.66E+04
RGMA	Best	5.32E+02	5.27E+02	5.29E+02	5.27E+02
	Med.	9.21E+04	7.50E+04	8.47E+04	8.39E+04
	Worst	1.37E+05	1.03E+05	1.20E+05	1.37E+05
	Mean	7.16E+04	5.72E+04	6.76E+04	6.82E+04
	Std	4.43E+04	3.83E+04	4.08E+04	5.08E+04
MA	Best	**5.29E+02**	**5.23E+02**	**5.23E+02**	**5.25E+02**
	Med.	7.82E+03	9.08E+03	9.07E+03	8.91E+03
	Worst	1.04E+04	1.16E+04	1.19E+04	1.09E+04
	Mean	**5.87E+03**	**7.74E+03**	**6.34E+03**	**7.38E+03**
	Std	4.15E+03	3.78E+03	4.50E+03	3.53E+03

I. Conclusion and Future work

This paper presents a newly developed Dependency Identification (DI) technique to detect the best arrangement for decomposing the variables of a large scale problem into smaller subproblems. A Memetic Algorithm (MA) with a self-directed Local Search (LS) is the subproblem's optimizer. The proposed DIMA model for large scale optimization is composed from DI and MA. Due to the lack of real life aspects such as *spliced* and *overlapping* variables in the previously designed test suites in the literature [10], a new test suite has been designed to represent unconstrained large scale optimization problems.

The evaluation of DIMA over this test suite of problems showed that the DI technique has higher performance when it is compared to RG [3] at 8 out of 12 problems. DI is also competitive to the VIL technique [6] because the percentage of evaluations that is used for the decomposition identification is approximately 1.16% in comparison to the 60% of VIL. Another advantage of DIMA is that it can reach a solution three times earlier than the other algorithms in literature. This makes DIMA a good optimization model when the available resources are limited. However, in the case of complex problems and unlimited resources, other algorithms with better exploration may have a higher opportunity to explore more solutions. This calls for a separate analysis to investigate the hybridization of DI with other optimization algorithms to combine the advantages of DI and the refining abilities of other optimization algorithms. Further experiments can also be conducted on the CEC2010 problems to investigate the performance of DI against other decomposition techniques in literature.

Another important conclusion is that DIMA has been found to be as powerful as MA which does not use CC approach. It is very promising to achieve very similar performance by optimizing smaller dimension subproblems and avoiding the drawbacks of large scale dimensionality. This encourages the testing of DIMA on very large scale problems which face a compromise between slightly lower performance and the ability to solve very large scale problems. This performance of DIMA will open the door to reconsider hybridizing other EAs with the proposed DI technique, to overcome their previous dimensionality issues. Optimizing the subproblems of a decomposed large scale problem using DIMA in a parallel computing environment is another interesting experiment whereby DIMA could further outperform non-decomposition approaches.

Table 2: Fitness Evaluation (FE) Comparisons

FE		F1	F2	F3	F4
5E+5	DIMA	**7.08E+02**	**4.91E+05**	1.02E+03	**5.94E+02**
	RGMA	1.29E+06	4.97E+06	**1.01E+03**	2.38E+06
1E+6	DIMA	**4.92E+02**	**1.79E+05**	**9.93E+02**	**5.91E+02**
	RGMA	4.58E+05	2.45E+06	9.93E+02	1.15E+06
2E+6	DIMA	**4.92E+02**	4.03E+04	9.93E+02	**5.90E+02**
	RGMA	1.05E+05	**5.05E+02**	**9.92E+02**	5.96E+02
3E+6	DIMA	**4.91E+02**	**4.93E+02**	9.92E+02	**5.89E+02**
	RGMA	4.93E+02	5.01E+02	**9.92E+02**	5.91E+02
		F5	F6	F7	F8
5E+5	DIMA	**5.45E+02**	**5.45E+02**	**9.67E+02**	**3.07E+06**
	RGMA	5.94E+06	2.21E+06	1.96E+05	3.56E+06
1E+6	DIMA	**5.45E+02**	**5.45E+02**	**5.45E+02**	**5.47E+02**
	RGMA	5.46E+02	1.02E+06	5.58E+02	1.47E+04
2E+6	DIMA	**5.45E+02**	**5.45E+02**	**5.45E+02**	**5.47E+02**
	RGMA	5.46E+02	3.26E+05	5.51E+02	5.60E+02
3E+6	DIMA	**5.45E+02**	**5.45E+02**	**5.45E+02**	**5.47E+02**
	RGMA	5.45E+02	5.45E+02	5.50E+02	5.47E+02
		F9	F10	F11	F12
5E+5	DIMA	**5.33E+02**	**1.51E+03**	**6.82E+02**	**5.31E+02**
	RGMA	3.81E+05	1.52E+06	1.24E+05	3.05E+06
1E+6	DIMA	**5.31E+02**	**5.31E+02**	**6.04E+02**	**5.28E+02**
	RGMA	1.53E+05	6.54E+05	4.35E+04	1.33E+06
2E+6	DIMA	**5.30E+02**	**5.27E+02**	**5.27E+02**	**5.26E+02**
	RGMA	5.37E+02	5.29E+02	1.11E+03	5.29E+02
3E+6	DIMA	**5.30E+02**	**5.25E+02**	**5.25E+02**	**5.25E+02**
	RGMA	5.32E+02	5.27E+02	5.29E+02	5.27E+02

REFERENCES

[1] M. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," 2010.

[2] M. Potter and K. D. Jong, "A Cooperative Coevolutionary Approach to Function Optimization," in in the proceedings of The Third conference on Parallel Problem Solving from Nature — PPSN III, 1994, pp. 249–257.

[3] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," Information Sciences, vol. 178, pp. 2986-2999, 2008.

[4] T. Ray and X. Yao, "A Cooperative Coevolutionary Algorithm with Correlation Based Adaptive Variable Partitioning," in IEEE Congress on Evolutionary Computation, 2009, pp. 983-989.

[5] M. Omidvar, X. Li, and X. Yao, "Cooperative Co-evolution with Delta Grouping for Large Scale Non-separable Function Optimization," in 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, 2010, pp. 18-23.

[6] C. Wenxiang, W. Thomas, Y. Zhenyu, and T. Ke, "Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning," in Parallel Problem Solving from Nature – PPSN XI. vol. 6239, R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, Eds., ed: Springer Berlin / Heidelberg, 2010, pp. 300-309.

[7] D. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in Proceedings of the genetic and evolutionary computation conference, San Mateo, California, 1999, pp. 220–228.

[8] P. Merz, "Memetic Algorithms for Combinational Optimization Problems: Fitness Landscapes and Effective Search Strategies," PhD, Gesamthochschule Siegen, University of Siegen, Germany, 2000.

[9] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and a. Z. Yang, "Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization," University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL), He´fe´i, A¯ nhu¯ı, China2007.

[10] K. Tang, L. Xiaodong, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark Functions for the CEC 2010 Special Session and Competition on Large Scale Global Optimization. Technical report," presented at the Nature Inspired Computation and Applications Laboratory, USTC, China, 2009.

[11] Y. Liu, X. Yao, Q. Zaho, and T. Higuchi, "Scaling Up Fast Evolutionary Programming with Cooperative Coevolution," in Comgress on Evolutionary computation, 2001, pp. 1101-1108.

[12] M. Potter and K. D. Jone, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," Evolutionary Computation, vol. 8, pp. 1-29, 2000.

[13] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in IEEE Congress on Evolutionary Computation (CEC) 2009, Trondheim, Norway 2009, pp. 1546-1553.

[14] D. Mosk-Aoyama and D. Shah, "Fast Distributed Algorithms for Computing Separable Functions," IEEE Transactions on Information Theory, vol. 54, pp. 2997-3007, 2008.

[15] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera, "Memetic Algorithms for Continuous Optimisation Based on Local Search Chains," Evolutionary Computation, vol. 18, pp. 27-63, 2010.

[16] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning: Addison-Wesley Longman Publishing Co., Inc., 1989.

[17] K. Deb and R. Agrawal, "Simulated Binary Crossover for Continuous Search Space," Complex Systems, vol. 9, pp. 115 - 148, 1995.

[18] E. Talbi, "Metaheuristics: from design to implementation," ed Hopoken, New Jersey, USA: John Wiley & Sons, 2009, pp. 124-127.

[19] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. New York: Springer-Verlag, 1992.

[20] D. Molina, M. Lozano, A. Sánchez, and F. Herrera, "Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SW-Chains," Soft Computing - A Fusion of Foundations, Methodologies and Applications, pp. 1-20, 2010.

[21] Z. Yang, K. Tang, and X. Yao, "Multilevel Cooperative Coevolution for Large Scale Optimization," in IEEE Congress on Evolutionary Computation, Los Alamitos, 2008, pp. 1663-1670.