An Adaptive Approach for Solving Dynamic Scheduling with Time-varying Number of Tasks – Part II

Manuel Blanco ABELLO*, Lam Thu BUI†, Zbignew MICHALEWICZ‡
*School of Computer Science, University of Adelaide, Adelaide, South Australia
Email: manuel.abello@adelaide.edu.au
†Le Quy Don Technical University, Viet Nam
Email: lam.bui07@gmail.com
‡School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia;
also at the Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland;
and at the Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland,
Email: zbyszek@cs.adelaide.edu.au

***Abstract*—Changes in environment are common in daily activities and can introduce new problems. To be adaptive to these changes, new solutions are to be found every time change occur. This two-part paper employs a technique called Centroid-Based Adaptation (CBA) which utilize centroid of non-dominated solutions found through Multi-objective Optimization with Evolutionary Algorithm (MOEA) from previous environmental change. This centroid will become part of MOEA's initial population to find the solutions for the current change.**

The first part of our paper deals mainly on the extension of CBA, called Mapping Task IDs for CBA (McBA), to solve problems resulting from time-varying number of tasks. This second part will show the versatility of McBA over a portfolio of algorithms with respect to the degree of changes in environment.

This demonstration was accomplished by finding a model relating the degree of changes to the performance of McBA using Nonlinear Principal Component Analysis. From this model, the degree of change at which McBA's performance becomes unacceptable can be found. Results showed that McBA, and its variant called Random McBA, can withstand larger environmental changes than those of other algorithms in the portfolio.

I. Introduction

Real-world problems often contain many uncertain and dynamic factors; i.e., air traffic scheduling is usually affected by unexpected events such as bad weather or emergencies. Therefore, it is unlikely that any solution found for these problems would stay valid for a long time. These factors require an adaptive mechanism to introduce changes to existing solution. This paper is the second part of a two-part paper which deals primarily on adaptation with time-varying number of task as the primary type of environmental change of interest. First and second parts will be called parts I and II, respectively.

The approach taken in our previous publication [1] to search for adaptive solutions to environmental changes is Multi-Objective Optimization with Evolutionary Algorithm (MOEA) [2]. It is possible that the adaptive solution for a current environmental change is strongly related to solutions of some past changes, such that it should be worthwhile to use the latter as initial population of MOEA to speed up the search of the former. In the said publication, a centroid is calculated for a set of non-dominated solutions obtained before a current change. It shows the overall tendency of its corresponding set towards the set of current adaptive solutions. It becomes part of MOEA's initial population to compute for the current adaptive solution. This technique is called Centroid-Based Adaptation (CBA).

The said publication employed Resource-Constrained Project Scheduling (RCPS) [3] as the test environment whereby a solution is represented by a string of scheduled tasks, labelled by ID numbers. It dealt on changes in task duration, task precedence, and availability of resources. Further, it sets the number of components in a solution to be equal to the number of tasks. Part I [4] extends the types of changes to include the change in the number of tasks. Problems arise due to this extension such that innovations are introduced to CBA to solve them. The resulting technique is called, Mapping of Task-IDs for Centroid-Based Adaptation (McBA). Part II will relate the degree of change in environment to the performance of McBA and will look for the limiting degree at which McBA's performance becomes unacceptable. Further, it will compare McBA's limiting degree of change to that of the other techniques. Results show McBA to have larger limiting degree than other techniques, except to its variant called Random McBA.

Part I reviewed literatures on Dynamic Optimization Problems, MOEA, RCPS, and time-varying number of tasks. Part II will review on the area of Algorithm Portfolio and will give a brief introduction to Nonlinear Principal Component Analysis (nlPCA).

A. Algorithm Portfolio

There are several search algorithms in the literature employed for optimization. However, it is desirable to know which among them will yield the best performance on a given

978-1-4244-7835-4/11/$26.00 ©2011 IEEE

problem. The field to investigate this objective is known as Algorithm Portfolio [5]–[8]. It should be noted however that performance of algorithm will vary from one instance of a problem to another [6].

Algorithm performance is related to the factors which measure how difficult for a given algorithm to solve a problem [8], [9]. These factors could be found either experimentally or theoretically. The former is impractical in the case of problems with high number of parameters since it will require an enormous number of sampling points in the set of parameter value combinations; while the latter suffers from degree of problem complexity, such as number of constraints [9]. Further, theoretical model is dependent on type of algorithm, genetic operator employed, fitness function, randomness, and non-linearities. These modelling aspects make theoretical approach difficult [8].

Strongly related to this area is to measure problem difficulty. Fitness Distance Correlation (FDC) in [10] is one such measure. However, this requires optimal solution before being computed and hence is impractical in application. A measure that circumvents this impracticability is Negative Slope Coefficient (NSC) [11] and Fitness Proportional NSC [12]. Both FDC and NSC do not offer a measure on degree of problem difficulty with respect to algorithm performance [8]. Expecting Run Time is another measure and is computed using computational complexity techniques [13]–[15].

On the extreme opposite, No Free Lunch (NFL) theorem showed that all problems has similar difficulty when averaged among all algorithm and conversely, all algorithms has similar efficiency when averaged among all problems. However, this theorem can measure algorithm performance only when the problem is closed under permutation [8].

Another approach is to create a model on algorithm's performance derived from its output. This model can offer automated selection or abortion of algorithms and could be use to analyze extreme variability of algorithm's behaviour across its various instances [9]. One such model is called empirical hardness models which dealt on satisfiability problem (SAT) involving binary variables and arbitrary constraints [16]–[18]. Another approach is to run different algorithms and then measure the risk of selecting one of them based on each solution quality obtained [6]. A model with simple approach is base on fitness of different solutions [19]. However, this restricts the construction of training set.

None of the current literatures in Algorithm Portfolio model algorithm's performance with respect to various types and degrees of environmental changes.

B. Nonlinear Principal Component Analysis

Gathered data in real-life observations could have more dimensions than the number of factors that generated them. For instance, increase in crime rate, unemployment, and resource scarcity could be due only to greed. Because of this dimension-factor relationship data will cluster around a curve, or manifold to be general, when graphically presented. It is then of interest to find these factors. We employed Nonlinear Principal Component Analysis (nlPCA) as the search tool.

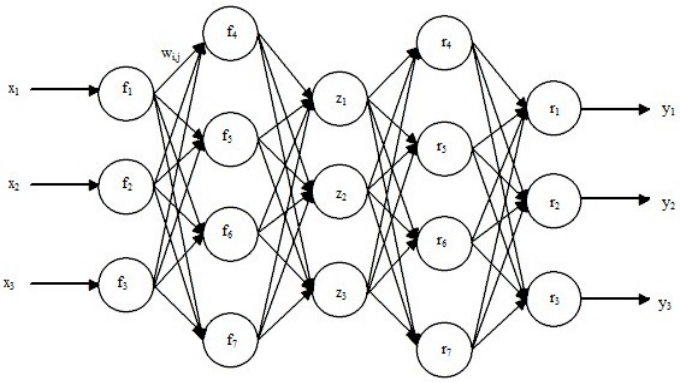

Fig. 1. Multi-layer Perceptron for Nonliner PCA

Multi-layer Perceptron (MLP) [20], shown in Figure 1, is the core structure of nlPCA. Its nodes are denoted by circles and its node index by labels inside the circle. An input data to MLP is a vector $\vec{x} = \{x_1, x_2, \ldots, x_n\}$ in space X; a vector called factor sample point $\vec{z} = \{z_1, z_2, \ldots, z_k\}$ is a point in the manifold Z with $k < n$; $w_{i,j}$ is the weight connecting node i to node j.

Nodes with labels z and r comprise the rear section of the perceptron while nodes with labels f and z the front section. The rear section will map factor sample point to the estimate,

$$\Psi(\vec{z}) \rightarrow \vec{\tilde{x}} \tag{1}$$

and the front section will map the given data to factor sample point,

$$\Theta(\vec{x}) \rightarrow \vec{z} \tag{2}$$

The innermost nodes – three in the figure – represent the factors in nlPCA. The main function of MLP is to find the weights which will minimize the error E^{rr} between the given data $\vec{x}$ at the input and the estimated data $\vec{\tilde{x}}$ at the output,

$$E^{rr} = \frac{1}{2}\left\|\vec{x} - \vec{\tilde{x}}\right\|^2 + \mu \sum_{i,j} w_{i,j}^2 \tag{3}$$

where μ is the weigth decay term to penalize large values of weights, and has a typical value of .001.

1) Hierarchical nlPCA: In linear Principal Component Analysis (PCA), the magnitude of eigenvalue determines the eigenvector hierarchy, i.e., the maximum eigenvalue corresponds to the principal component. The perceptron just described is called standard nlPCA, *s*-nlPCA, in which there is no regard on component's (factor's) hierarchy.

The hierarchical nlPCA, *h*-nlPCA, on the other hand, does. Let $E^{rr}_{1,2,\ldots,n}$ be the estimation error with only n number of middle nodes. Starting from $n = 1$ to $n = k$ the *h*-nlPCA hierarchy is found by minimizing,

$$E^{rr}(k) = E^{rr}_1 + E^{rr}_{1,2} + \ldots + E^{rr}_{1,2,\ldots,k} \tag{4}$$

provided $E^{rr}(k-1)$ is already minimized. The index k corresponds to component's hierarchy; $k = 1$ being the

principal component [21]. It is also the subscript of label "z" in the middle nodes.

The remaining sections of this paper are organized as follow: the problem formulation, schedule generation, and Genetic Algorithm's operators are described in Section II. Section III deals with approaches to model the performance of McBA. This is followed by the description of experimental set-up and the discussion on the experimental results in Section IV. Lastly, the conclusion and future works in Section V.

II. Methodology

RCPS is the test environment in this paper, which we renamed as *Adaptive Planning Problem* (APP) in [1] for being specifically applied to adaptation. This application requires to consider the following issues: (i) by choosing a reactive approach to adaptation, there could be some components of the solution (plan or schedule) already finished or in-progress when change occurs (ii) not only is execution time of plan (makespan) vital but also the cost, such as money or human, of moving resources from one task to another. These issues require multi-objective optimization (MOO) with makespan and cost as the first and second objectives, respectively.

A. Mathematical formulation of APP

Most parts of the following problem formulation is copied from our previous paper [1] and is described as follows:

- **Inputs:**
  - A set V of non-pre-emptive tasks:

$$V = \{V_0, V_1, V_2, ..., V_N, V_{N+1}\} \tag{5}$$

    where V_0 and V_1 are not tasks but starting and ending points, respectively; and N as the actual number of tasks. In the foregoing discussion, the points are disregarded, except when stated otherwise. Each V_i will have:
    - A durations d_i
    - A vector rr_i of required resources by tasks: $rr_i = \{rr_{ij}\}$ with $j = 1, ..., M$ (M is the number of resource types)
  - A network G of tasks where nodes and arcs represent the tasks and the precedence relations respectively:

$$G = (V, E) \tag{6}$$

    where E is a set of directed arcs. $Pred(j)$ defines a set of direct predecessors, while $Succ(j)$ is the set of direct successors of task j. Each task $V_i \in V$ has a vector $S_i = \{s_{i,j}\}$ where,

$$s_{i,j} = \begin{cases} 1 & \text{task } j \text{ succeeds } i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

  - A matrix c of operational costs $c = \{c_{i,j,k}\}$, $i = 0, .., N$; $j = 0, ..., N$, and $k = 1, ..., M$. Here $c_{i,j,k}$ is the cost of moving resource type k from task i to task j. $c_{i,0,k} = 0 \ \forall \ i, k$ - no cost is imposed on the return of items to base (the starting point V_0)
  - A set R of M resources $R = \{R_1, R_2, ..., R_M\}$ with each resource type R_i composed of items,

$$\{R_{i,1}, R_{i,2}, \ldots, R_{i,r_i}\} \tag{8}$$

    and has r_i as its maximum number of items.
- **Parameters:**
  - A vector of start time ts: $ts = \{ts_i\}$, with $i = 1, ..., N$
  - For each R_i at time t, a vector of locations for each item of a resource type l_{it} is defined to indicate where the item is located (or the task index). A zero value means the item is at the central base V_0): $l_{it} = \{l_{ijt}\}, j = 0, ..., R_i$
  - For each R_i at time t, a vector of previous locations for each item of a capability type lc_{it} is defined to indicate where the item was from (or the task index). A zero value means the item is at the central base): $lc_{it} = \{lc_{ijt}\}, j = 0, ..., R_i$
  - For each R_i at time t, a vector of locations for each item of a capability type m_{it} is defined to indicate if the item was moved or not: $m_{it} = \{m_{ijt}\}, j = 0, ..., R_i$

$$m_{ijt} = \begin{cases} 1 & l_{ijt} \neq lc_{ijt} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

  - A vector $r_t = r_{it}, i = 1, ..., M$ presents the current amount of resources being used at time t
  - Indices of the tasks: $I = \{I_1, I_2, ..., I_N\}$ (a schedule)
- **Constraints:**
  - Time constraint: If task i is predecessor of task j, then it needs to be completed before starting task j.

$$ts_i + d_i \leq ts_j \tag{10}$$

    $\forall \ j$, and $\forall \ i \in Prec(j)$
  - Resource constraint: the amount of being-used resources can not exceed the total amount

$$r_{it} \leq R_i \tag{11}$$

    $\forall \ i$ and t
  - Precedence constraint: A task needs to be scheduled before its successors.

$$I_i \notin Succ(I_j) \ \forall \ i, j \mid I_i \leq I_j \tag{12}$$

- **Objective functions:**
  - Makespan (f_1): Minimization of the start time of the last task to be scheduled)

$$f_1 = ts_{\overline{N}} \tag{13}$$

    where $\overline{N}$ is the last task to be scheduled
  - Cost of resource operations (f_2): cost of moving resources between locations of tasks.

$$f_2 = \sum_{t=1 \rightarrow T} \sum_{j=1 \rightarrow M} \sum_{k=1 \rightarrow R_j} m_{jkt} \times c_{lc_{jkt}, l_{jkt}, j} \tag{14}$$

- **Outputs:**
  A schedule ts based on the obtained index I, $ts = \{ts_1, ts_2, ..., ts_N\}$

- **Dynamic factors**
  - **Duration**: Dynamic duration of a task V_i is defined as,
    $$d'_i(t) = N_m(d_i, \delta) + \delta \quad (15)$$
    where N_m is the normal distribution with the mean as the pre-defined duration d_i and δ be the standard deviation. Constraint Eq. 10 is rewritten as follows
    $$ts_i + d'_i(t) <= ts_j \quad (16)$$
  - **Availability of resources**: we use a sign function to indicate the availability of resources:
    $$l'_{ijt} = sign(t) * l_{ijt} \quad (17)$$
    where $sign(t)$ is the sign function returning either +1 or 1, a negative value means unavailable but at the location $|l_{ij}|$. Note that the number of resources that change their status usually follows the Poisson distribution.
  - **Number of Tasks**: The dynamic function representing this change is defined as,
    $$G^{curr} = \Gamma^{N^{curr}}_{N^{prev}}(V^{prev}, E^{prev}) \quad (18)$$
    where V^{prev} and E^{prev} are the sets of tasks and arcs, respectively; and the superscripts $prev$ and $curr$ denotes their associated variable for the previous and current change, respectively. This superscript convention will be applied from here onwards.
- **Parameters after a change**
  Structures of l_{it}, lc_{it}, m_{it} and r_t remain unchanged. The only change is applied to the indices in which $I = I_1, I_2, ..., I_{N'}$ where N' is the number of tasks in $V'(t)$

B. Genetic Algorithm

Given RCPS being an NP-complete problem and the multi-objectivity aforementioned, we propose to employ MOEA. We employ McBA described in part I to generate the initial population for MOEA. This initial population will be evolved over-time where all of its good solutions are preserved in every stage of the evolution process. In part I, a variant of McBA called Random McBA (McBAR) utilizes random initialization to generate its initial population in addition to its associated centroids.

To perform the said preservation, we employ NSGA-II [2]; a non-dominated sorting mechanism where the parent and offspring populations are combined and sorted to generate a population for the next generation. MOEA's selection process that produce offspring is again through NSGA-II wherein crowding tournament selection is used that gives preference to solutions far from other solutions in the associated search space, thereby implementing diversity of solutions and hence better exploration of the associated search space.

1) Genetic operators: MOEA's crossover and mutation operations for NSGA-II are redesigned to be suitable for our system:

Consider two parent solutions P_1 and P_2 for crossover from where two offspring solutions O_1 and O_2 will be generated. Each of the parent solution was broken into three parts on two randomly selected points. Then the first, second, and third parts of O_1 is taken from P_2's first part, P_1's second part, and P_2's third part, respectively. The three parts of O_2 is taken in reverse order, i.e., $P_1 - > P_2 - > P_1$. To satisfy precedence constraint, inheriting of the second and third parent parts will only be for solution components that are absent in the previous parts.

Mutation of solution will swap two consecutive components, at a given probability, provided the resulting sequence of tasks is precedence feasible.

2) Solution Presentation: A solution/schedule S in our problem is represented by a string of tasks I_i endowed with starting time ts_i, duration time d'_i, and ID i,

$$S = (I_1, I_2, ..., I_{N'})$$

Makespan is first objective in our MOEA problem and is found to be the time at which the last task is finished. If the sequence of tasks is precedence-infeasible, this objective is set to infinity such that its associated solution has the least preference during MOEA's selection process. Cost, as the second objective, is determined by simulating the solution and by computing the resources movement cost, taking into account tasks which are either finished or in-progress.

Procedure SSGS
Begin
  Determine a set of eligible tasks ε
  For g=1 to N'
    Select a task $j \in \varepsilon$
    Determine $t = max\{0, max\{st_i + d_i | i \in Pred(j)\}\}$
    Schedule j at the earliest precedence - and resource-feasible start time $t' > t$
    Set $st_j = t'$ and update ε
  End
End

Fig. 2. Serial Schedule Generation Scheme

3) Schedule Generation: Before any change in the environment occurs a baseline solution is created through MOEA whereby the initial population employed is generated through Serial Schedule Generation Scheme (SSGS) [22] as illustrated in Figure 2. This scheme will verify if a scheduled task satisfies precedence relation (sample shown in Figure 4 of part I) and if there is enough resources for this task.

4) Initial Population: In part I of this paper, McBAs (McBA and McBAR) utilized the optimal solution they obtained from their respective previous changes, aside from their centroids, as part of their initial population for the current change. As mentioned there, the previous events in the chain of changes can influence McBAs' search of optimal solution

for the current change. If we are to find the role of degree of change for the current change only, the said influence must be minimize or eradicated.

In part I, $SC(X, Y)$ is called the set coverage of technique X over Y. It is equal to the number of solutions produced by technique Y being dominated [2] by solutions produced by technique X divided by the number of solutions in technique Y. Let technique X be regarded as superior to Y if $SC(X, Y) > SC(Y, X)$. One way to achieve the minimization of influence from past changes is to have all techniques employ the optimal solution of a particular technique, that has the most number of techniques being superior to, as part of their initial population. For example, suppose in the previous change a technique called NDLPOP (defined in Part I) is superior to 4 techniques while the rest of techniques are only superior to lesser than 4 techniques, then the optimal solution found by NDLPOP will be utilized by all techniques as part of their respective intialize populations for the current change. In particular, $P'(t-1)$ for McBA in Equation 3 of part I will not be derived from the optimal solution found by McBA from the previous change but rather from that of NDLPOP. Experimental results in Section IV-A proves this approach to achieve the objective.

III. Performance Model of McBA

The objective of part I is to show the superiority of McBA over other methods. This second part will deal on the limitation of McBA with respect to the degree of environmental change and compare this limitation to those of other methods. We will begin by defining the degree of environmental change.

A. Environmental Change

As mentioned in part I, RCPS is applied to mission planning where there are three types of changes: task duration, task number, and amount of resources. The degree of change in task duration is defined as,

$$dT = \sum_{t=1}^{N^{prev}} \|d_t^{prev} - d_t^{curr}\| \tag{19}$$

where t is the task index in V^{prev}. Note that by the convention in Section II, V^{prev} is the set of tasks (defined in Equation 5) in the previous change. Current change in number of task is made to be positive always so that if there is a simultaneous change in task duration and number N^{prev} is always lesser than number of task in the current change. Hence, Equation 19 measures duration changes only for tasks present in both current and previous changes.

The change in number of tasks will also change the precedence relationship. To model the latter type of change, we define the degree of change in task precedence relationship (DCTPR) as,

$$dN = \sum_{i=1,j=1}^{N^{prev}} \|s_{i,j}^{prev} - s_{i,j}^{curr}\| + \sum_{i,j=N^{prev}+1}^{N^{curr}} s_{i,j}^{curr} \tag{20}$$

where $s_{i,j}$ is defined in Equation 7. Note that the summation indices of Equation 20 exclude the starting and ending tasks indices (defined through Equation 5) of current and previous changes. Further, the amount of change in task number may not be identical to the DCTPR. One could simply reverse the precedence order of tasks and DCTPR may not be zero while change in task number is.

A resource item $R_{i,j}$ defined in Set 8 has a status $S_{i,j}^t$ of being available or occupied which are assigned with values one and zero, respectively. The degree of change in resource availability is defined as,

$$dR = \sum_{i=1}^{M} \sum_{j=1}^{r_i} \|S_{i,j}^{t,prev} - S_{i,j}^{t,curr}\| \tag{21}$$

where index i is for resource type; j for resource item; and r_i and M are inter-related through Set 8.

B. Model in Algorithm Portfolio

There are many ways to compare various algorithms to solve similar problems, as discussed in Section I-A. However, none of the literatures in Algorithm Portfolio model algorithm's performance with respect to degree of change in environmental parameters. Due to this unique challenge we have chosen a model-based approach employing *h*-nlPCA, described in Section I-B.

Let the difference in set coverage be,

$$dSC(X, RI) = SC(X, RI) - SC(RI, X) \tag{22}$$

where RI is Random Initialization algorithm, and $X \neq RI$ is the algorithm being considered. Both RI and the various techniques represented by X are defined in part I. $dSC(X, RI)$ is positive when X has more number of solutions dominating that of RI which implies X performing better than RI.

The input vector to *h*-nlPCA is composed of degree of change for various types of changes occurring simultaneously, and $dSC(X, RI)$. For example, to model the performance of McBA with respect to simultaneous changes in task duration dT and resources dR, the input vector to *h*-nlPCA will be,

$$\vec{x} = \{dT, dR, dSC(McBA, RI)\}$$

Let the set of vectors isomorphic to $\vec{x}$ produced from all experimental runs and all orders of change be,

$$X_n = \{\vec{x}_k\} \tag{23}$$

Further, let the last component of $\vec{x}_k$ corresponds to dSC. Using X_n to Mapping 2, *h*-nlPCA with only one middle node yields the principal component,

$$P = \{p_k\} = \Theta(X_n) \tag{24}$$

which is a set of scalar values. Then using P to Mapping 1 yields the set of estimates $\widetilde{X_n} = \{\vec{\tilde{x}}_k\}$.

Let S_e be a curve that best fits the set of points,

$$\left\{\left(p_k, \widetilde{dSC}_k(X, RI)\right)\right\} \longrightarrow S_e \tag{25}$$

where $\widetilde{dSC}_k(X, RI)$ is the last component of the vector $\vec{\tilde{x}}_k$. Our objective is to find the degree of environmental change at

which $dSC(X, RI) = 0$, implying the limit in X's superiority in performance over RI. This is accomplished by finding the root p_{root} of S_e. This root is then feed to $\Psi(p_{root})$ of Mapping 1 to obtain $\vec{\tilde{x}}_{root}$ which, in our example, will be $\{dT_{root}, dR_{root}, 0\}$ where dT_{root} and dR_{root} are the sought-after limiting degrees of change. Comparison of algorithm performance will be based on roots such as these, excluding the dSC component.

To generalize, the performance model P_m is the function that best fits the set of points,

$$\{(\Psi(p_k)|_K, \Psi(p_k)|_{\neq K})\} \longrightarrow P_m \tag{26}$$

where $p_k = \Theta(x_k)$; and the expression $|_K$ and $|_{\neq K}$ meant to include and exclude the last component of their associated vectors, respectively. The root is,

$$x_{root} = \Psi(p_{root})|_{\neq K} \tag{27}$$

where

$$p_{root} = \Xi\{P_m\} \tag{28}$$

and Ξ is the root extraction operation.

IV. Experiments

Types of environmental changes experimented in this second part are listed in Table I. The first column is for labels of change type, the second is for parameters that changes simultaneously. These types of changes are combined to form

TABLE I
TYPES OF ENVIRONMENTAL CHANGES

Type	Variable
0	task duration
1	resource availability
2	task number
3	task duration
	task number
4	task duration
	resource availability
5	task number
	resource availability
6	task duration
	task number
	resource availability

various sequences of changes, labelled by S_i in Table II. For example, sequence S_3 begins with change in task duration (0), followed by change in resource availability (1), then by simultaneous change in task duration, task number, and resource availability (6 in Table I). Within the sequence S_i is a set of task number changes that when sequenced according to its order of occurrence form a sequence, labelled by T_k in Table III . Referring to the example, the first change in task number occurred at the 3^{rd} change in S_3 due to change type 6; wherein, based on Table III is an increase by 6. There are 30 experiments undertaken with attributes shown in Table IV. Experiment numbers on the lower and upper half of the diagonal correspond to $\delta = 3.0$ and $\delta = 6.0$, respectively, where δ is the standard deviation defined in Equation 15.

TABLE II
TYPES OF CHANGE SEQUENCE

Order \ Label	S_1	S_2	S_3
1	0	0	0
2	1	1	1
3	0	2	6
4	2	0	0
5	0	0	4
6	3	3	0
7	0	0	0
8	4	4	5
9	0	6	0
10	5	5	2
11	6	0	0
12	0	0	3

TABLE III
TYPES OF TASK NUMBER INCREASE SEQUENCE (TNIS)

Order \ Label	T_3	T_4	T_5	T_6	T_7
1	3	4	5	6	7
2	2	4	3	2	1
3	2	1	1	1	1
4	3	1	1	1	1

Tables I to IV must be interrelated to find when does task number changes and at what amount. For instance, Experiment 24 belongs to change sequence S_3 and based on the example above has an increase in task number by 6 simultaneously occurring with change in task duration and resource availability at the 3^{rd} change. The task duration has $\delta = 6.0$ since Experiment 24 is in the upper half of the diagonal.

A. Results and Discussion

From all experiments and orders of change, the points $\vec{x}_k$ are gathered together comprising set X_n defined in Equation 23. Set X_n was inputted to Equation 24 from where the principal component P was found. Using the principal component for McBA with $\delta = .3$ and change type 6 to Equation 26 the models were found and shown in Figure 3 indicated by the curves with marks ".", "o", and "x" relating dT, dN, and dR to $dSC(McBA, RI)$, respectively. The values at which the model crosses zero are the limiting degree of change being sought-for. Models for other cases are not shown. The relationship (Equation 1) between factor sample points and

TABLE IV
TYPES OF EXPERIMENTS

S_1	S_2	S_3	δ	TNIS
04 / 01	05 / 02	06 / 03	6.0 / 3.0	T_3
10 / 07	11 / 08	12 / 09	6.0 / 3.0	T_4
16 / 13	17 / 14	18 / 15	6.0 / 3.0	T_5
22 / 19	23 / 20	24 / 21	6.0 / 3.0	T_6
28 / 25	29 / 26	30 / 27	6.0 / 3.0	T_7

estimated data is shown in Figure 4 with input vectors in X_n represented by the scattered points. The figure expresses a large estimation error. However, useful results can be obtained from this as will be shown in the following.

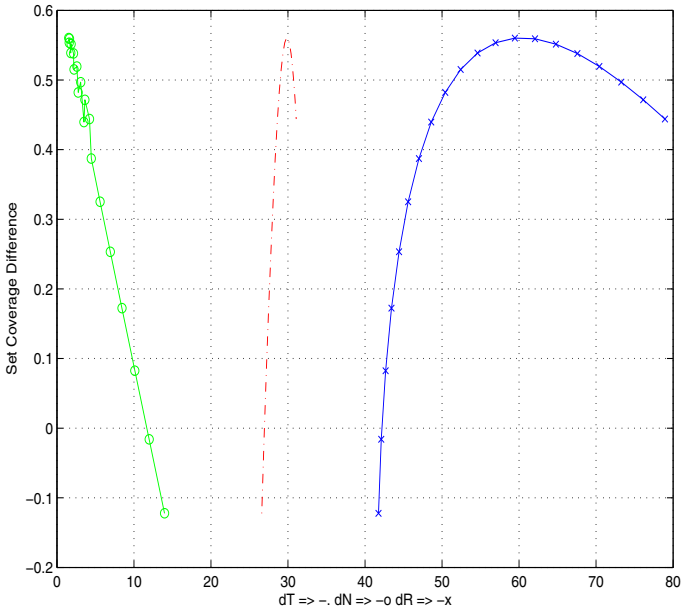

Fig. 3. Model for McBA

The limiting degree of change for various types of changes for an algorithm X was found through Equation 27 and the results are in Table V. Before discussing this table note that in Table II change type 3 occurs at 6^{th} and 12^{th} orders of change, and change type 5 occurs at 8^{th} and 10^{th} orders of change where at these orders McBAs have more centroids employed which made them generally superior to RI technique. This superiority kept McBAs' $dSC(McBAs, RI)$ mostly positive. Consequently, their models (Equation 26) does not manifest any zero-crossing and hence Equation 27 is inapplicable. Similar situation happens to change types 0, 1, and 4. For these reasons, only the model for change types 2 and 6 are created.

Now Table II shows types 2 and 6 occurring at 3^{rd} change. Results (not shown) manifest that McBAs' performances becomes inferior to RI at this order of change, on the average. In part I, this inferiority is not found at the said order of change which could be due to the support from previous changes. In this second part, initial population employed by McBAs are not necessarily theirs, as discussed in Section II-B4. This implies that initial population scheme in the said section can minimize or even eradicate the influence of the previous events and hence the results here could be due only to the degree of change in environmental parameters; a situation that suites the objective of this second part.

In Table V the column labelled "Score" are the principal component roots, p_{root} in Equation 28, i.e., the values of factor sample point at which $dSC(X, RI)$ is zero. Further, dT_{lim}, dN_{lim}, and dR_{lim} are the degree of change in task

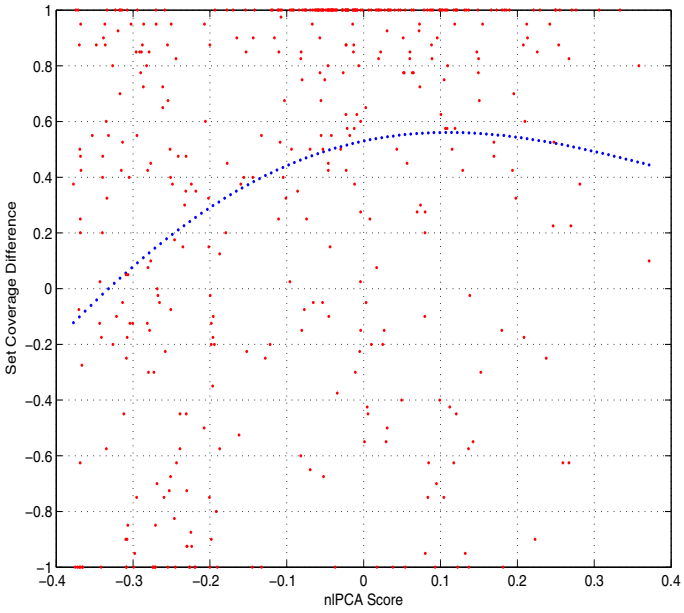

Fig. 4. Sample Estimation and Score Relationship

duration, task precedence relationship, and resource availability, respectively corresponding to p_{root}. It is evident in this table that for change type 2, McBA has the highest dN_{lim} followed by McBAR while the rest of techniques have lesser dN_{lim} than them. This implies that it will take larger degree of change in task precedence relations before McBAs will be equalled or exceeded by RI in performance than other techniques. Comparison of cases with δ equal to 3.0 and 6.0 showed that dN_{lim} is larger in the former than in latter implying that standard deviation in task duration change does influence McBAs performances.

TABLE V
LIMITING ROOTS OF TECHNIQUES

Type	Technique	δ	Score	dT_{lim}	dN_{lim}	dR_{lim}
2	CBA	3	-0.064		10.9337	
		6	-0.1		12.9405	
	McBA	3	-0.12		14.0216	
		6	-0.08		11.7912	
	McBAR	3	-0.11		13.4262	
		6	-0.09		12.3336	
	LPOP	3	-0.035		9.4647	
		6	0.15		-0.3144	
	NDLPOP	3	0		7.5724	
		6	-0.01		8.0977	
6	CBA	3	0.1655	27.1023	10.714	42.4572
		6	0.08	47.7905	8.0014	43.8426
	McBA	3	0.116	27.1619	10.3327	42.6053
		6	-0.0855	48.9566	7.6607	44.7282
	McBAR	3	0.23	26.7592	12.8945	41.8764
		6	0.12	42.9337	9.6183	40.1545
	LPOP	3	0.32	30.3943	2.1408	67.4346
		6	0.385	60.6309	3.8643	53.6017
	NDLPOP	3	-0.225	29.7643	1.5006	58.9874
		6	-0.035	54.5813	5.8291	49.0101

Now in change type 6, McBAs and CBA has better dN_{lim} but not in dT_{lim} and dR_{lim} which implies that McBAs are

effective only in coping with changes in number of tasks; an expected result considering that McBAs were designed for this type of situation.

Comparison of dN_{lim}s between change type 2 and 6 showed a decrease in limiting dN_{lim} which implies that simultaneity has strong effect on the performance of McBAs. With only task number changing (type 2), the degree of environmental change from previous to the current change could be lesser than in the case of simultaneous change (type 6). Consequently, the separation in the search space of previous and current solutions in type 2 could be shorter than in type 6. The large separation in Type 6 requires more evolution cycles and with a fixed number of the latter, McBAs can hardly reach this large separation; whereas RI, which has initial population well-scattered in the search space, is in a better position to find the optimal solution. This could explain the result.

V. Conclusion and Future Work

In this second part, we introduced the implementation of Algorithm Portfolio and Nonliner Principal Component Analysis to model the performance, and search for the limiting case, of Centroid-Based Adaptation and its variants, McBA and McBAR. Experiments showed that performance of McBAs is strongly influenced by the amplitude of change in task duration and the simultaneity of change in task duration, task number and resource availability.

For future work, we plan to investigate proactive approach and to use more complex simulation that allows us to present limited human factor in the loop such as ontological negotiation similar to what had been done in [23]. This human factor could be simulated by holons; defined as autonomous and self-reliant agents in [24] where battlefield scenario was modelled. Application of McBA and McBAR to the evolutionary process involved in the task and resource assignment of these agents could enhance computational speed.

Acknowledgements

This work is supported by Australian Research Council (ARC) Grant DP0985723 and by grants N 516 384734 and N 519 578038 from the Polish Ministry of Science and Higher Education (MNiSW). The second author acknowledged the funding from Vietnam's National Foundation for Science and Technology Development (NAFOSTED), Grant No 102.01-2010.12.

References

[1] L. T. Bui, Z. Michalewicz, E. Parkinson, and M. B. Abello, "Adaptation in Dynamic Environments: A Case Study in Mission Planning," *IEEE Transactions on Evolutionary Computation*, 2011.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[3] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[4] M. B. Abello, L. T. Bui, and Z. Michalewicz, "An Adaptive Approach for Solving Dynamic Scheduling with Time-varying Number of Tasks – Part I," in *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*, 2011.

[5] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown, "Satzilla: Portfolio-based algorithm selection for sat," *Journal of Artificial Intelligence Research*, vol. 32, pp. 565–606, 2008.

[6] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, 2010.

[7] J. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[8] M. Graff and R. Poli, "Practical performance models of algorithms in evolutionary program induction and other domains," *Artificial Intelligence*, vol. 174, pp. 1254–1276, 2010.

[9] K. Leyton-Brown, E. Nudelman, and Y. Shoham, "Empirical hardness models: Methodology and a case study on combinatorial auctions," *Journal of the ACM*, vol. 56, no. 4, 2009.

[10] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *L.J. Eshelman (Ed.), ICGA*. Morgan Kaufmann, 1995, pp. 184–192.

[11] L. Vanneschi, M. Clergue, P. Collard, M. Tomassini, and S. Vrel, "Fitness clouds and problem hardness in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)in: Lecture Notes in Computer Science. K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E.K. Burke, P.J. Darwen, D. Dasgupta, D. Floreano, J.A. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A.M. Tyrrell (Eds.)*, vol. 3103. Springer, Seattle, WA, USA, 2004, pp. 690–701.

[12] R. Poli and L. Vanneschi, "Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007). H. Lipson (Ed.* ACM, London, England, UK, 2007, p. 13351342.

[13] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276 (12), pp. 51–81, 2002.

[14] T. Jansen, K. Jong, and I. Wegener, "On the choice of the offspring population size in evolutionary algorithms," *Evolutionary Computation*, vol. 13, no. 4, pp. 413–440, 2005.

[15] C. Witt, "Runtime analysis of the (mu + 1) ea on simple pseudo-boolean functions," *Evolutionary Computation*, vol. 14, no. 1, pp. 65–86, 2006.

[16] E. Nudelman, K. Leyton-Brown, A. Devkar, Y. Shoham, and H. Hoos, "Satzilla: An algorithm portfolio for sat," in *In Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, 2004, pp. 13–14.

[17] F. Hutter, Y. Hamadi, H. Hoos, and K. Leyton-Brown, "Performance prediction and automated tuning of randomized and parametric algorithms," in *Proceedings of the International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science*, vol. 4204. Springer-Verlag, Berlin, Germany, 2006, pp. 213–228.

[18] L. Xu, H. Hoos, and K. Leyton-Brown, "Hierarchical hardness models for sat," in *Proceedings of the International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science*, vol. 4741. Springer-Verlag, Berlin, Germany, 2007, pp. 696–711.

[19] Y. Borenstein and R. Poli, "Information landscapes," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005). H.-G. Beyer, U.-M. OReilly (Eds.)*. ACM, Washington DC, USA, 2005, pp. 1515–1522.

[20] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[21] M. Scholz and R. Vigario, "Nonlinear pca: a new hierarchical approach. in: Verleysen, m., ed." in *Proceedings ESANN*, 2002, pp. 439–444.

[22] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 127, no. 2, pp. 394–407, 2000.

[23] P. Vrba, M. Radakovič, M. Obitko, and V. Mařík, "Semantic extension of agent-based control: The packing cell case study," in *HoloMAS '09: Proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 47–60.

[24] T. Yu, F. Tu, and K. R. Pattipati, "Integration of a holonic organizational control architecture and multiobjective evolutionary algorithm for flexible distributed scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 38, no. 5, pp. 1001–1017, 2008.