2002

# Optimization under uncertainty with application to data clustering

Jumi Kim
*Iowa State University*

# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

# Optimization under uncertainty with application to data clustering

by

**Jumi Kim**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
Sigurdur Ólafsson, Major Professor
Irvin R. Hentzel
Doug Gemmill
Timothy Van Voorhis
Ananda Weerasinghe

Iowa State University

Ames, Iowa

2002

Graduate College

Iowa State University

This is to certify that the doctoral dissertation of

Jumi Kim

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

# DEDICATION

I would like to dedicate this thesis to my parents, my sister and my husband without whose support I would not have been able to complete this work.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

A new simulation optimization technique that extends the pure nested partition (NP) algorithm is presented in this thesis. This method is called the nested partition with inheritance. Furthermore, a statistical selection method and a random search method are introduced to overcome certain shortcomings of the pure NP (Nested Partitions) algorithm. Finally, the suggested algorithms are applied to a data mining problems, namely data clustering.

The basic idea of a NP algorithm is very simple. At each iteration, the most promising region is partitioned and the performance of the partitioned region is evaluated using sampling. Based on the performance evaluation, the most promising region is chosen for the next iteration. These procedures are repeated until it satisfies the termination condition.

Even though the pure NP method guarantees the convergence to the optimal solution, it has two apparent problems. The first problem is related to the two sources of errors in the estimate of each region during the sampling phase of the NP algorithm: the sampling error due to the use of a limited number of sample points from the region and the estimation error due to the use of optimization under uncertainty. The other problem is that at each iteration, there is no guarantee of whether the correct move is made or not. To handle these shortcomings, two extensions to the pure NP are suggested. To rigorously determine the required sample effort, some statistical selection methods are implemented, which include the Nelson Matejcik procedure, the Rinott procedure, and the Dudewicz and Dalal procedure, as

well as a subset procedure. In addition, Genetic Algorithms (GAs) are used to speed convergence and to overcome the difficulty in the backtracking stage of the NP algorithm. The resulting algorithms are evaluated through problems with two types of noisy performance: a Monte Carlo simulation problem and a discrete event simulation, queuing problem.

As an application of the new methodology, this work also suggests the methods to be applied to a data clustering problem. This is a very hard problem within data mining, with two of the main difficulties being lack of scalability with respect to amount of data and problems with high dimensionality. The new algorithms are found to be effective for solving this problem. Random sampling enhances scalability and the iterative partitioning addresses the dimensionality.

# Chapter 1
# Introduction

Optimization under uncertainty has been found to be useful in areas such as designing manufacturing systems, evaluating the requirements of computer systems, determining policies in inventory systems, designing and operating transportation facilities, evaluating designs for service organizations and analyzing financial systems. Complex and large systems cannot be solved by simple analytical or mathematical methods. For this reason, using simulation is often necessary. Moreover, optimization with uncertainty has become one of the most widely used tools in operations research and management science, especially when large finite feasible region is given.

Evaluating the performance of every feasible point using optimization under uncertainty is very time consuming. Even though many optimizations with uncertainty algorithms have been developed, there are some difficulties when applying these algorithms to real world problems. One of the reasons is that there is no guarantee for convergence to the optimal; therefore, new algorithms are needed to overcome this problem.

Every algorithm that is discussed in this thesis is based on the Nested Partitions (NP) method, an optimization with uncertainty method that is guaranteed to converge to an optimal solution. In this thesis, mainly three works contributions are made. First, a new method that is called the *nested partitions method with inheritance* is developed and evaluated that can be adopted as an optimization with uncertainty method. By introducing this new concept to the pure NP method, it will be shown that the new method improves efficiency of the pure NP method. A detailed description of this new method is presented. To further improve the

efficiency of the pure NP method, combined algorithms with the statistical selection method and a random search method are suggested and validated through the numerical evaluation. Finally, data mining, which is recently becoming a hot issue, will be discussed. Specifically, data clustering will be focused upon. Data clustering has numerous applications, including loan payment prediction and customer credit policy analysis, classification and clustering of customers for target marketing, as well as detection of money laundering and other financial crimes. By applying the suggested algorithms to the data clustering problem it is shown that the NP methodology perfectly fits to solve the data clustering problem.

There are many methods for optimization under uncertainty. Deciding which method to use depends on the problem structure. For example, gradient estimation, stochastic approximation, and sample path optimization are applicable when the feasible input variables are continuous. On the other hand, the random search method and statistical method are applicable for discrete input variables. More detailed discussion of these methods is follows.

## 1.1 Continuous Input Variables

Until recently most techniques were developed for continuous input parameters. There are several common methods used for continuous input variables. Several of them are categorized as gradient-based methods. The classical stochastic optimization methods are based on an iterative search in the direction of time of the gradient. This was originally suggested by Robbins-Monro and Kiefer-Wolfowitz in the 1950s (Robinson *et al.*, 1951). The Robbins-Monro algorithm is simply a root finding procedure for functions whose values are not known but observed with noise. The Robbins-Monro algorithm estimates the gradient

directly; whereas, the Kiefer-Wolfowitz algorithm uses finite differences to the derivative. In both cases, the primary implementation problem determines the step size.

The most straightforward approach for the gradient estimation method is the finite differences method (Glynn, 1989; Andradottir, 1989). This method is simple to implement and generally applicable, but it has several difficulties when applied to practical problems. One of the big difficulties is that too much time is spent calculating the gradient. Forward differences and backward each requires $n+1$, $2n$ iterations for a $n$ dimensional function. The other difficulty is the gradient estimates obtained using finite differences are generally biased. Trying to reduce bias leads to another difficulty: large variance in the estimations. There are several studies describing these problems (Glynn, 1989; L'Ecuyer and Perron, 1994). For reducing the large variance problem, methods such as CRN (Common Random Number) have been suggested (Glasserman and Yao, 1992).

Unlike finite differences, Perturbation Analysis (PA) and Likelihood Ratios (LR) require only a single simulation run to obtain an estimate. Infinitesimal Perturbation Analysis (IPA) is the best know variant of PA. The basic idea of IPA is simply to take the estimator of the expected performance (L'Ecuyer, 1991). The basic idea of LR is that the gradient of the performance is expressed as an expectation with respect to the same distribution as the performance function itself (Glynn, 1989). However, both methods have some limitations. They are not always applicable and more complicated to understand and implement than finite differences. A drawback to using all gradient-based search techniques is that they find only local optima (Fu, 1994).

The effort to solve this problem by converting the original simulation into an approximate deterministic optimization problem is started by Rubinstein and Shapiro (1993). The sample path method involves converting the original simulation into an approximate deterministic optimization problem. This approach is also called the stochastic counterpart method that includes the random search method and statistical method.

## 1.2 Random Search Method

In this section, brief random search methods are reviewed when the feasible region is discrete. These methods also cannot usually guarantee a global optimal, and therefore they are often called heuristics methods. Three common random search methods are mentioned below.

Tabu Search was originally proposed by Glover (1977) for escaping local optimal by using a list of prohibited solutions known as the tabu list. The commonly used diversification method is re-starting from the best solution obtained so far. Another drawback of the tabu search is unless there is a long tabu list, it may reach a previously visited solution.

Simulated Annealing (SA), introduced by Kirkpatrick *et al.* (1983), is a random search method that is able to escape local optima using a probability function. Unlike the tabu search, SA does not evaluate the entire neighborhood in every iteration. Instead, it randomly chooses only one solution from the current neighborhood and evaluates its costs. That means SA tends to need more iterations to find the best solution than the tabu search method. Another disadvantage is that it does not have memory, and hence it may re-visit a recent solution. There is a combination method of tabu and SA.

Genetic Algorithms (GAs) were originally developed by Holland (1975). This is the most widely known evolutionary method, which is both powerful and broadly applicable to stochastic optimization. It mimics the mechanisms of natural selection and natural genetics where stronger individuals are more likely to survive in a competing environment. Thereby, the strongest individual (having the best performance) survives. Commonly used operators include selection, reproduction, crossover, and mutation.

Another class of methods that can be used when the input parameters are discrete are the statistical selection methods.

## 1.3 Statistical Selection Method

Simulations are often performed to compare two or more system designs. The most popular statistical methods are ranking and selection (R&S) and multiple comparison procedures (MCPs), which are applicable when the input parameters are discrete and the number of designs to be compared is both discrete and small (say, 2 to 20). R&S are statistical methods developed to select the best system or a subset that contains the best system design from a set of competing alternatives (Goldman and Nelson, 1994). MCPs use pairwise comparison to derive relationships among all designs. In other words, R&S yields the best system while the MCPs yields information about the relationships among the alternative solutions. Sanchez (1997) gives an overview of R&S with samples. Wen and Chen (1994) present single-stage sampling procedures for different MCPs. Goldsman and Nelson (1994, 1998) provide comprehensive reviews of R&S and MCPs. More detailed discussion is found in Chapter 3.

Optimization method with uncertainty has been briefly surveyed. For the next chapter, the focus is on the NP method, a recently developed optimization with uncertainty method and the extended method.

The remainder of this thesis is organized as follows: in Chapter 2, the pure NP method is described and the concept of inheritance is introduced to improve efficiency of the NP method. As another way to improve the NP method, two different methods are employed: statistical selection methods and random search methods. These methods are discussed in Chapter 3 and Chapter 4. In Chapter 3, the combined NP methods with the statistical selection method are suggested and numerical evaluation is also shown. In Chapter 4, Genetic algorithms (GAs) are combined with the methods which are introduced in Chapter 3. Also, numerical results are shown. In Chapter 5 and Chapter 6, data clustering is introduced as a specific application of new algorithms which are introduced through the whole thesis and finally, in Chapter 7, the conclusion and future research is proposed.

# Chapter 2

# The Nested Partitions Method with Inheritance

In general, the optimization problem considered here minimizes an objective function $f : \Theta \rightarrow R$, over the feasible region $\Theta$; that is

$$\min_{\theta \in \Theta} f(\theta)$$

where $\Theta$ is finite and $f$ is unknown but can be estimated with $\hat{f}(\theta)$ for each. $\theta \in \Theta$.

## 2.1 Nested Partitions (NP) Method

The Nested Partitions (NP) method was originally developed by Shi and Ólafsson (2000a). This method solves both deterministic and stochastic finite optimization problems. The basic idea of this algorithm is to shift the focus from points in the feasible region to a sequence of the feasible region. The Nested Partitions (NP) method is mainly composed of 4 procedures: partitioning, sampling, estimating promising index, and backtracking. In each iteration of the algorithm, it is assumed there is a region, i.e., a subset of $\Theta$, that is considered the *most promising region*. Then this most promising region is *partitioned* into $M$ regions and the entire surrounding region is aggregated into one region. Therefore $M+1$ disjoint subsets of the feasible region $\Theta$ are looked for at each iteration. Each of these $M+1$ regions is *sampled* using some random sampling scheme, and for each region a *promising index* is calculated. These promising indices are then compared to determine which region is the most promising index in the next iteration. If one of the sub-regions is found to be best,

this sub-region becomes the most promising region. However, if the surrounding region is found to be the best, the algorithm *backtrack*s and a larger region containing the current most promising region becomes the new most promising region. The new most promising region is then partitioned and sampled in a similar fashion. This process is repeated until the terminate criteria is satisfied. Generally, simulation is done when the maximum depth is reached. The singleton regions are called regions of *maximum depth*. Since the singleton regions cannot be partitioned further, they are considered regions of maximum depth.

## 2.2 NP with Inheritance

In the pure NP method, independent sampling is performed in each iteration. In other words, after partitioning, given most promising region by throwing away good solutions, information that has been learned is lost. This allows for nice convergence properties as the algorithm generates a Markov chain. However, the basic idea of our new algorithm is to keep good solutions by inheriting these solutions to the next iteration. Every algorithm that will be considered in this thesis focuses on the use of sampling procedures in the pure NP method. The idea of inheritance also focuses on the sampling procedure, especially the part of how to start sampling each sub-region. If there is a minimum probability correct selection ($P^*$), the sample point that is currently chosen satisfies $P^*$ and every iteration will satisfy $P^*$. So, by inheriting the current best solution to the next iteration, the probability of a correct selection continues to increase. Intuitively, better solutions are formed in the next iteration and the computation time is reduced.

The notation used by these algorithms is summarized below.

$\Sigma$ = $\{\sigma \subseteq \Theta \mid \sigma$ is a valid region given a fixed partitioning $\}$

$\Sigma_0 \subset \Sigma$ = $\{\sigma \subseteq \Theta \mid \sigma$ is of maximum depth $\}$

$\sigma(k) \in \Sigma$ = The most promising region in the $k^{th}$ iteration

$N_k(\sigma) \in N$ = The number of times $\sigma \in \Sigma$ has been the most promising region in the first $k^{th}$ iterations

$\hat{\sigma}(k) \in \Sigma_0$ = The maximum depth region that has most frequently been selected as the promising region in the first $k^{th}$ iterations

$d(\sigma)$ = The depth of $\sigma \in \Sigma$

$s(\sigma) \in \Sigma$ = The superregion of $\sigma \in \Sigma$

$d^*$ = Maximum depth

$M_{\sigma(k)}$ = Number of partitions of the $\sigma(k)$

The first variant of the newly proposed method follows.

**Algorithm NP with Inheritance**

Step 1. *Initialization*

Set $k = 0$ and $\sigma(k) = \Theta$.

Step 2. *Partitioning*

If $d(\sigma(k)) \neq d^*$, that is, $\sigma(k) \neq \sum_0$, partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ sub-regions $\sigma_1(k),\ldots,\sigma_{M_{\sigma(k)}}(k)$. If $d(\sigma(k))=d^*$, then let $M_{\sigma(k)} = 1$ and

$\sigma_1(k) = \sigma(k)$. If $d(\sigma(k)) \neq 0$, that is, $\sigma(k) \neq \Theta$, aggregate the surrounding region $\Theta \setminus \sigma(k)$ into one region $\sigma_{M_{\sigma(k)+1}}(k)$.

Step 3. *Random Sampling*

If $k = 0$, then use a random sampling to obtain $N_j$ points from each of the regions $\sigma_j(k)$, $j = 1,2,\ldots, M_{\sigma(k)} + 1$,

$$\theta_l^{j1}, \theta_l^{j2}, \ldots, \theta_l^{jN}, \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

Else if $k > 0$, then use the sample points of $INH_{k-1}$ and fulfill the lack of $N_j$ using random sampling and calculate the corresponding sample performance values $L(\theta)$

$$L(\theta_l^{j1}), L(\theta_l^{j2}), \ldots, L(\theta_l^{jN}), \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

Step 4. *Estimating the Promising Index*

For each region $\sigma_j(k)$, $j = 1,2,\ldots, M_{\sigma(k)} + 1$, define a promising index function, $I(\sigma_j)$, and calculate the *promising index*. For example, define $I(\sigma_j)$ as the best performance value in the region

$$I(\sigma_j) = \min_{\theta \in \sigma_j} L(\theta), \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

and estimate $I(\sigma_j)$ using

$$\hat{I}(\sigma_j) = \min_{i = \{1,2,\ldots,N_j\}} L(\theta^{ji}), \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

Step 5. *Backtracking*

Determine the *most promising region* $\sigma_{j_k}$

$$\hat{j}_k \in \arg \min_{j=1,2,\ldots,M_{\sigma(k)}+1} \hat{I}(\sigma_j)$$

If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a sub-region of $\sigma(k)$, then let this sub-region be the most promising region of the next iteration.

That is $\sigma(k+1) = \sigma_j(k)$, $j < M_{\sigma(k)}$ and keeps the sample points of the promising index

$$INH_k = [\theta^{j1}, \theta^{j2}, \ldots, \theta^{jN}], \quad j = \hat{j}_k$$

Otherwise, if the index corresponds to the surrounding region, backtrack to the region, $s(\sigma(k))$, of the current most promising region. That is, let $\sigma(k+1) = s(\sigma(k))$.

Step 6. *Checking the Stopping Rule*

If $\sigma(k+1) \in \Sigma_0$, stop and let $\sigma_{opt} = \sigma(k+1)$, else $k = k+1$ and go back to Step 2.

# Chapter 3
# Statistical Selection in NP

As observed in Ólafsson (1999), the pure NP method has two apparent shortcomings. First, there are clearly two sources of error in the estimate of each region: the sampling error due to the use of a sample of the points in the region, and the estimation error due to the use of simulation. Secondly, in each iteration, there is no guarantee concerning whether the correct move is made. In Ólafsson (1999), a two-stage NP is proposed to address both of these concerns. By using statistical selection methods to determine a second-stage sample size, it is possible to assure that the correct move is made with a given probability while simultaneously controlling the total error, possibly by using different numbers of sample points in each region.

One key idea of many statistical ranking and selection methods is that the number of sample points obtained for each system should be proportional to the variance of the performance of each system. When incorporated into the NP method, this intuitively suggests that since the sizes of the regions vary greatly, and, in particular, the surrounding region tends to be much larger than the sub-regions, then some regions can be expected to have higher variance and will therefore need a larger sampling size.

To state the two-stage NP approach rigorously, let $D_{ij}(k)$ be the $i^{\text{th}}$ set of random sample points selected from the region $\sigma_j(k)$ in the $k^{\text{th}}$ iteration, where $i \geq 1$ and $j = 1,2,,\ldots,M+1$. Let $N = \left|D_{ij}(k)\right|$ denote the initial number of sample points, which is

assumed to be constant. In addition let $\theta \in D_{ij}(k)$ denote a point in this set and let $L(\theta)$ be a simulation estimate of the performance of this point. Then in the $k^{th}$ iteration, for every $i$,

$$X_{ij}(k) = \min_{\theta \in D_{ij}(k)} L(\theta) \tag{1}$$

is an estimate of the performance of the region $\sigma_j$, which is referred to as the $i^{th}$ system performance for the $j^{th}$ system, $i \geq 1$, $j = 1,2,\ldots,M+1$. The two-stage ranking and selection procedure first obtains $n_0$ such system estimates, and then uses that information to determine the total number of $N_j$ of system estimates needed from the $j^{th}$ system, which is, subregion $\sigma_j(k)$. This number is selected to be sufficiently large so that the correct subregion is selected with probability at least $P^*$, subject to an indifference zone of $\varepsilon > 0$.

## 3.1 Statistical Selection Methods

Discrete-event stochastic simulation is often used to choose the best system among a set of proposed systems where best is defined by the maximum or minimum expected simulation output. Thus, considerable interests exist for Ranking and Selection (R&S) procedures. The fundamentals of R&S were first proposed by Bechhofer (1954). The original indifference zone R&S procedure proposed by Bechhofer (1954) is single-stage and assumes unknown means and known, common variances for all systems. But indifference zone R&S procedures need not be single-stage. By defining the user-specified number of observations, they can extend to multi-stage procedures (sequential procedures) assuming common, known variances. Paulson (1964) and Bechhofer *et al.* (1968) present such methodologies. Koeing and Law (1985) extend the indifference zone approach for use as a screening procedure.

Unlike the articles discussed, Dudewicz and Teneja (1978) present a multivariate procedure which does not require reduction to a univariate model. If the indifference zone procedures use a least-favorable configuration (LFC) to allocate additional replications, the optimal computing budget allocation (OCBA) (Chen *et al.*, 1996) and Bayesian decision-theoretic methods (e.g., Berger, 1988; Gupta and Miescke, 1996; Chick and Inoue, 1998; Chick and Inoue 1999) use an average case analysis to allocate additional replications (Inoue *et al.*, 1999). All three procedures assume that simulation output is independent and normally distributed having unknown mean and variance and applicable to both two-stage and sequential procedures. Inoue and Chick (2000) show empirically that the two-stage procedure of Rinott (1978) performs competitively with sequential OCBA and Bayesian decision-theoretic methods when the number of systems under consideration is small ($k < 5$). For a large number of systems ($k \geq 5$), or when the difference in the mean output of the best system and other systems varies significantly, the Rinott procedure is less effective at identifying the best system. Among two-stage procedures, the Bayesian decision-theoretic procedures have the best overall performance characteristics.

Recently, many articles have tried to unify the fields of R&S and MCPs. Multiple comparisons with the best (MCB) is one of the most widely used MCPs. To apply MCB in a discrete-event simulation, the simulation runs must be independently seeded and the simulation output must be normally distributed, or averaged so that the estimators used are somewhat normally distributed. There are four R&S-MCB procedures having normally distributed data, but do not require known or equal variance: Rinott's Procedure (Procedure $R$), Dudewicz and Dalal's Procedure (Procedure $DD$), Clark and Yang's Procedure

(Procedure *CY* ), Nelson and Matejcik's Procedure (Procedure *NM* ) (Nelson Matejcik, 1995). Procedure *R* and Procedure *DD* are performed in the same manner with the only difference being in the calculation of the sample means. Both algorithms require independence among all observations. The total sample size depends on the sample variance of the systems. So the larger the sample variance, the more replications are required. Unlike these algorithms, Procedure *CY* and Procedure *NM* requires fewer total observations by employing the CRN. Clark and Yang (1986) use the Bonferroni inequality to account for the dependence induced by CRN. However, Nelson and Matejcik (1995) observed that the benefit gained from using Procedure *CY* is diminished when the number of systems to be compared is large. To overcome this problem, they present Procedure *NM* . Procedure *NM* assumes that the unknown variance–covariance matrix exhibits a structure known as sphericity that implies the variances of all paired differences across systems are equal, even though the marginal variances and covariances may be unequal. The difference between Procedure *CY* and *NM* is the calculation of sample variance. This sample variance affects the total number of sample size for second-stage sampling. Nelson and Matejcik (1995) reported that Procedure *NM* is superior to Procedure *R* , *DD*, and *CY* in terms of the total observations required to obtain the desired confidence level. The only potential drawback with Procedure *NM* 's is that the assumption of sphericity may not be satisfied. The following table summarizes these characteristics.

Table 3.1: Summary of Statistical Methods

( $X_{ij}$ : The output of the $j^{th}$ replication of system $i$ )

| Method | CRN | Single/Two/Sequential Stage | Major Assumption |
|---|---|---|---|
| Procedure $P_B$ Bechhofer (1954) | | Single (Indifference zone) | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ $\sigma^2$ : common, known |
| Paulson (1964), Bechhofer *et al.* (1968) | | Sequential | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ $\sigma^2$ : common, known |
| Procedure $P_{DD}$ Dudewicz and Dalal (1975) | | Two | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ |
| Procedure $P_{DD}$ Dudewicz and Zaino (1976) | | Single (MCP) | $X_{ij} \sim N(\mu,\sigma^2)$ |
| Rinott (1978) | | Two (Indifference zone) | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ |
| Procedure *R* Nelson and Matejcik (1995) | | Two (MCB, Indifference zone) | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ |
| Procedure *DD* Nelson and Matejcik (1995) | | Two (MCB, Indifference zone) | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ |
| Procedure *CY* Nelson and Matejcik (1995) | * | Two (MCB, Indifference zone) | $X_j \sim N(\mu,\Sigma)$ $\mu$ : unknown matrix $\Sigma$ : unknown variance–covariance matrix |
| Procedure *NM* Nelson and Matejcik (1995) | * | Two (MCB, Indifference zone) | $X_j \sim N(\mu,\Sigma)$ $\mu$ : unknown matrix $\Sigma$ : unknown variance–covariance matrix |
| Procedure *OCBA* Chen *et al.* (1999) | | Two/Sequential (Optimal computing budget allocation (OCBA)) | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ |
| Procedure $0-1(B)$ Chick and Inoue (2000) | | Two/Sequential (Bayesian decision-theoretic methods) | $X_{ij} \overset{i.i.d}{\sim} N(\mu,\sigma^2)$ |

Three different methods are used to identify the best systems in terms of sample characteristics. Two of these methods have an assumption of independence of between systems. Generally, independence requires many sampled points. As a result, Nelson and Matejcik (1995) suggest using Common Random Numbers (CRNs) for a small number of alternatives.

## 3.2 Independent Sampling

### 3.2.1 Two-Stage Sampling with Subset Selection

When using statistical selection methods, computation can be made more efficient by filtering inferior systems. A subset selection technique is used for filtering systems. The subset selection technique has been studied by many researchers. In 1965, Gupta proposed a single-stage procedure with the assumption that alternatives are independent equal-sized and normally distributed with the common unknown variance. This procedure produces random size subsets having an optimal system with pre-specified probability $P^*$(Rinott, 1978) without an indifference zone. In 1989, Sullivan and Wilson proposed a general restricted subset selection procedure that allows unknown and unequal variance with an indifference zone having an exact size to be included in a subset. Unlike Gupta's method, the number of systems in the subset can be controlled. In 1993, Gupta and Santer extended the above methods for pre-specifying the maximum size of a subset and showed relationship between indifference zone approaches. It is efficient if the size of a subset is clearly upper-bounding than having the exact size of the subset method. Because exact size of subset method was used, then some inferior system which is already known could be included. The shortcoming

of the subset selection approach is that the best system cannot be found. As an illustration of the two-stage NP approach, Ólafsson (1999) uses the classic Rinott's ranking and selection procedure. An indifference zone $\varepsilon$ is assumed to be given that describes our tolerance for selecting a system that has up to $\varepsilon$ units worse performance than the optimal performance. By using $\varepsilon$, the number of systems being compared, and the desired probability $P^*$ of correct selection, a constant $h$ is calculated. Then $n_0$ initial samples are obtained from each system. After calculating sample variance $S_j^2$ for each system, the second stage sample size for each system is calculated according the following formula.

$$N_j = \max\left\{n_0 + 1, \left\lceil \frac{h^2 S_j^2}{\varepsilon^2} \right\rceil\right\} \tag{2}$$

After finishing second stage sampling, a system with performance within $\varepsilon$ of the optimal performance is selected with probability $P^*$ (Rinott, 1978).

Another incorporation method with NP is Procedure $P_{DD}$. Procedure $P_{DD}$ is originally proposed by Dudewicz and Dalal (1975). This method assumes normality and independence of observations. Procedure $P_{DD}$ is almost the same as Rinott's Procedure but the difference is the selection of the best system is based on weighted averages. They use the weighted average of each stage to find the best system. Weights are calculated according the following formula.

$$W_{j1}(k) = \frac{n_0}{N_j}\left[1 + \left\{\left(1 - \frac{N_j}{n_0}\left(1 - \frac{(N_j - n_0)\varepsilon^2}{h^2 S_j^2(k)}\right)\right)^{1/2}\right\}\right], \tag{3}$$

$$W_{j2}(k) = 1 - W_{j1}(k) .$$

The shortcomings of the Rinott procedure are well documented. Most notably, the derivations of equation (2) assumes the least favorable configuration among the system, which typically leads to a very conservative value for the number of sample points which tends to require too much sampling effort. Also, equation (2) only uses the variance, such that there is no consideration for the mean performance in the first stage. Thus, it may be beneficial in terms of computation time to filter out such inferior systems, which can be accomplished by combining it with a subset selection procedure, resulting in the following algorithm:

**Algorithm NP/Subset/Rinott**

Step 1. *Initialization*

Set $k = 0$ and $\sigma(k) = \Theta$.

Specify the overall desired probability $P^*$ of correct selection and indifference zone $\varepsilon$, the common initial sample size $n_0 \geq 2$, the number of sub-regions $M$. Determine $t$ from the $t$-distribution and $h$ for Rinott's integral. $t$ and $h$ are constants which are determined by $n_0$, the minimum probability $P^*$ of correct selection, and $M$ (See the tables in Bechhofer *et al.*, 1995).

$$t = t_{1-(1-\frac{\alpha}{2})^{\frac{1}{M-1}}, n_0 - 1} .$$

Step 2. *Partitioning*

Given the current most promising region $\sigma(k)$, partition $\sigma(k)$ into $M$ sub-regions $\sigma_1(k), \ldots, \sigma_M(k)$, and aggregate the surrounding region $\Theta \backslash \sigma(k)$ into one region $\sigma_{M+1}(k)$.

Step 3. *First-Stage Sampling*

Step 3-1. Let $i = 1$.

Step 3-2. Use uniform sampling to obtain a set $D_{ij}(k)$ of $N$ sampling points from region $j = 1, 2, \ldots, M+1$.

Step 3-3. Use discrete event simulation of the system to obtain a sample performance $L(\theta)$ for every $\theta \in D_{ij}(k)$ and estimate the performance of the region as

$$X_{ij}(k) = \min_{\theta \in D_{ij}(k)} L(\theta).$$

Step 3-4. If $i = n_0$ continue to Step 4. Otherwise, let $i = i+1$ and go back to Step 3-2.

Step 4. *Estimating Mean and Variance of First-Stage Sampling*

Calculate first-stage sample means and variances

$$\overline{X}_j^{(1)}(k) = \frac{1}{n_0} \sum_{i=1}^{n_0} X_{ij}(k),$$

and

$$S_j^2 = \frac{\sum_{i=1}^{n_0} [X_{ij}(k) - \overline{X}_j^{(1)}(k)]^2}{n_0 - 1},$$

for $j = 1, 2, \ldots, M+1$.

Step 5. *Filtering*

Calculate the quantity

$$W_{ij} = t\left(\frac{S_i^2 + S_j^2}{n_0}\right)^{1/2}$$

for all $i \neq j$.

Include the $i^{th}$ region in the selected subset $I$ if

$$\overline{X}_i \leq \overline{X}_j + (W_{ij} - \varepsilon)^+ \quad \text{for all } i \neq j.$$

Step 6. *Computing Total Sample Size*

If $I$ contains only a single region, $I = \{\sigma_j(k)\}$, then this has the best promising index so update $\sigma(k+1)=\sigma_j(k)$ and go to Step 11. Otherwise, compute the total sample size for all $j \in I$

$$N_j = \max\left\{n_0 + 1, \left\lceil \frac{h^2 S_j^2}{\varepsilon^2} \right\rceil\right\}$$

where $\varepsilon$ is the indifference zone and $h$ is a constant determined by $n_0$ and the minimum probability $P^*$ of correct selection.

Step 7. *Second-Stage Sampling*

Obtain $N_j(k) - n_0$ more simulation estimates of the system performance for all $j \in I$ as in Step 3-1 through Step 3-4 above.

Step 8. *Estimating Mean of Second-Stage Sampling*

Let the overall sample mean be the promising index for all $j \in I$,

$$\hat{I}(\sigma_j(k)) = \bar{X}_j(k) = \frac{\sum_{i=1}^{N_j(k)} X_{ij}(k)}{N_j(k)}.$$

Step 9. *Estimating Promising Index*

Select the index of the region with the best promising index,

$$\hat{j}_k \in \arg\ \min \hat{I}(\sigma_j)\ \text{ for all } j \in I.$$

If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a sub-region, $\sigma(k)$, then let this be the most promising region in the next iterations. Otherwise, if the index corresponds to the surrounding region, backtrack to a larger region containing the current most promising region. That is, let

$$\sigma(k+1) = \begin{cases} \sigma_{\hat{i}_k}(k), & \text{if } \hat{i}_k < M+1 \\ s(\sigma(k)), & \text{otherwise} \end{cases}$$

Step 10. *Checking Stopping Rule*

If $\sigma(k+1) \in \Sigma_0$, stop and let $\sigma_{opt} = \sigma(k+1)$, else $k = k+1$ and go back to Step 2.

**Algorithm NP/Subset/DD**

Step 1. *Initialization*

See Step 1 in Algorithm NP/Subset/Rinott.

Step 2. *Partitioning*

See Step 2 in Algorithm NP/Subset/Rinott

Step 3. *First-Stage Sampling*

See Step 3 in Algorithm NP/Subset/Rinott

Step 4. *Estimating Mean and Variance of First-Stage Sampling*

See Step 4 in Algorithm NP/Subset/Rinott

Step 5. *Filtering Subset*

See Step 5 in Algorithm NP/Subset/Rinott

Step 6. *Computing Total Sample Size for Second-Stage Sampling*

See Step 6 in Algorithm NP/Subset/Rinott

Step 7. *Second-Stage Sampling*

See Step 7 in Algorithm NP/Subset/Rinott

Step 8. *Estimating Mean of Second-Stage Sampling*

Calculate the second-stage sample means based on $N_j - n_0$ replications

$$\overline{X}_j^{(2)}(k) = \frac{1}{N_j - n_0} \sum_{i=1}^{N_j - n_0} X_{ij}(k).$$

Step 9. *Calculating Weights for each Stage Samples*

$$W_{j1}(k) = \frac{n_0}{N_j}\left[1 + \left\{\left(1 - \frac{N_j}{n_0}\left(1 - \frac{(N_j - n_0)\varepsilon^2}{h^2 S_j^2(k)}\right)\right)^{1/2}\right\}\right], \quad W_{j2}(k) = 1 - W_{j1}(k).$$

Step 10.*Calculating Weighted Averages*

Calculate weighted averages for all $j \in I$

$$\overline{X}_j(k) = W_{j1}\overline{X}_j^{(1)}(k) + W_{j2}\overline{X}_j^{(2)}(k).$$

and let these weighted averages be the promising index for all $j \in I$,

$$\hat{I}(\sigma_j(k)) = \bar{X}_j(k)$$

Step 11.*Estimating Promising Index*

See Step 9 in Algorithm NP/Subset/Rinott

Step 12. *Checking Stopping Rule*

See Step 10 in Algorithm NP/Subset/Rinott

### 3.2.2 Numerical Evaluation

To numerically evaluate the performance of Algorithm NP/Subset/Rinott and NP/Subset/DD, consider a production system with a given number of $M$ workstations configured in parallel and jobs that are to be processed by exactly one of the stations. The objective is to find the optimal resource allocation that minimizes the expected makespan having assumption that there are some $R$ resources that can be assigned to perform the necessary work within each station, and those resources can be moved to other workstations upon completion of a job. This is a Monte Carlo simulation where the randomness derives from random processing times, subsequently referred in this thesis as the Monte Carlo problem.

For in all parameters of these experiments. Let $M = 2$, $R = 5$, and the first stage sample points in each region set to $n_0 = 20$. Twenty replications are used for each experiment which were performed with $P^* \in [0.55,0.95]$.

One of the primary benefits of two-stage sampling is that more computational effort is allocated in regions where it is needed. To insure that the two-stage approach indeed makes a substantial difference, the total number of sample points is used at each depth level. The results are shown in Figure 3.1. and Figure 3.2. These figures show that the computational

effort decreases as the depth increases, although there is a peak at depth two because this is the first depth where a surrounding region is considered. Intuitively the reason for this may be that, as the depth increases, the subregions become more and more homogeneous leading to lower variance, and hence less effort is required to evaluate each region. The opposite is true for the surrounding region, but for Algorithms NP/Subset/Rinott and NP/Subset/DD, it may often be possible to filter this region out early, especially when substantial progress has been made and the quality of the subregion is high.

The potential benefit of two-stage methods without subset selection is illustrated in both figures. When the pure NP method is used, the number of sample points is constant. The first plot of Figure 3.1 shows that over 3,000 sample points are needed to guarantee 95% success probability at depth two. Contrast with this, what is needed using the two-stage sampling at depth six is only 2,000 samples. Thus, variable sampling reduces total computational effort by one-third. In addition, the subset selection creates an even greater savings, and for many settings of $P^*$ the effort that would be required for NP without two-stage sampling is three times that of which would be required for the NP/Subset/Rinott Algorithm.

A comparison of the two graphs in Figure 3.1 show that the NP/Subset/Rinott Algorithm requires fewer sample points. In particular, if the $P^*$ is low, the total number of sample points for the NP/Subset/Rinott Algorithm is less than half of that used by the NP/Rinott Algorithm; however, the relative difference between these algorithms is decreased by increasing $P^*$. Similar results are seen from the NP/Subset/DD Algorithm. In conclusion,

at least for this problem the NP/Subset/Rinott Algorithm is less computationally expensive than the NP/Rinott Algorithm and these benefits are higher when $P^*$ is set to a low value.

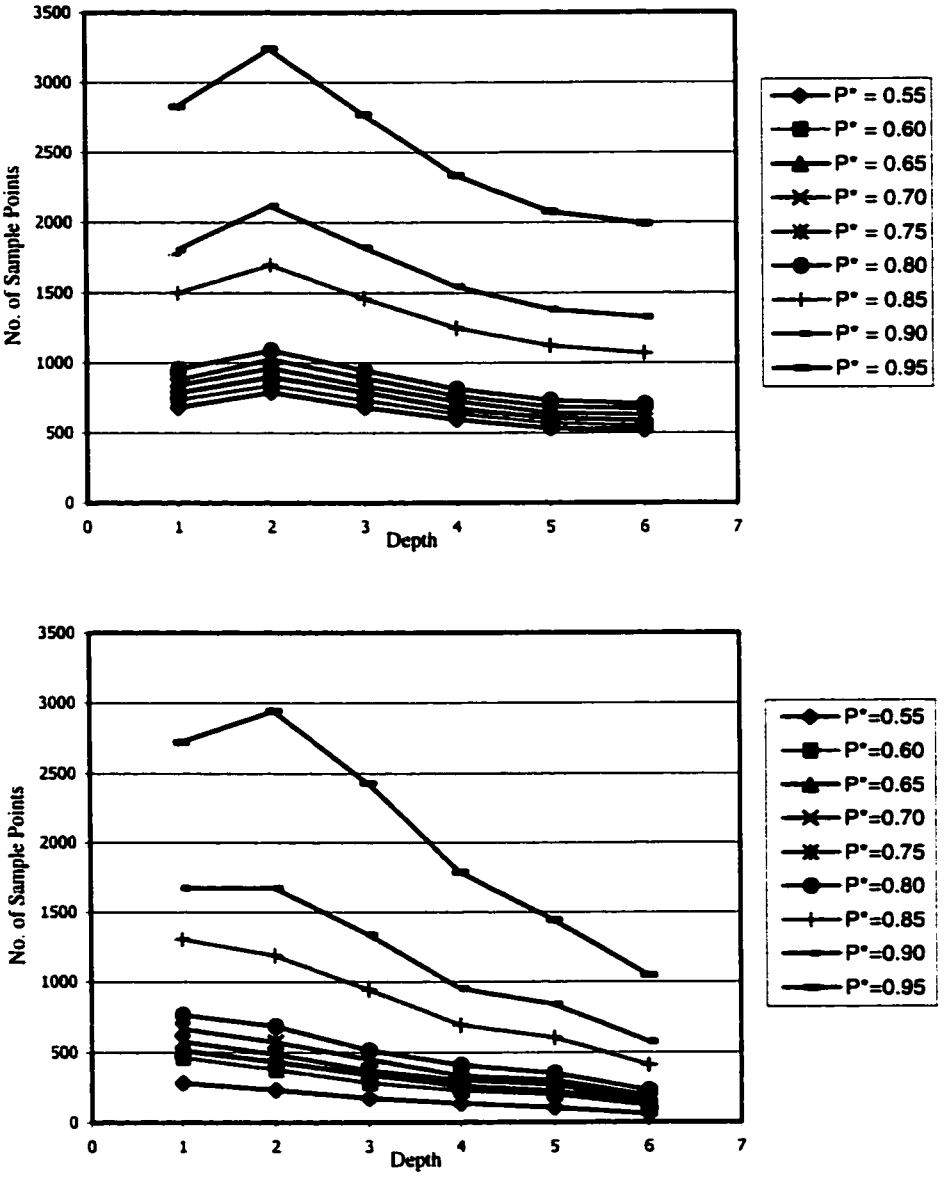



Figure 3.1: Total Number of Sample Points of Each Depth for NP/Rinott (above) and NP/Subset/Rinott (below)

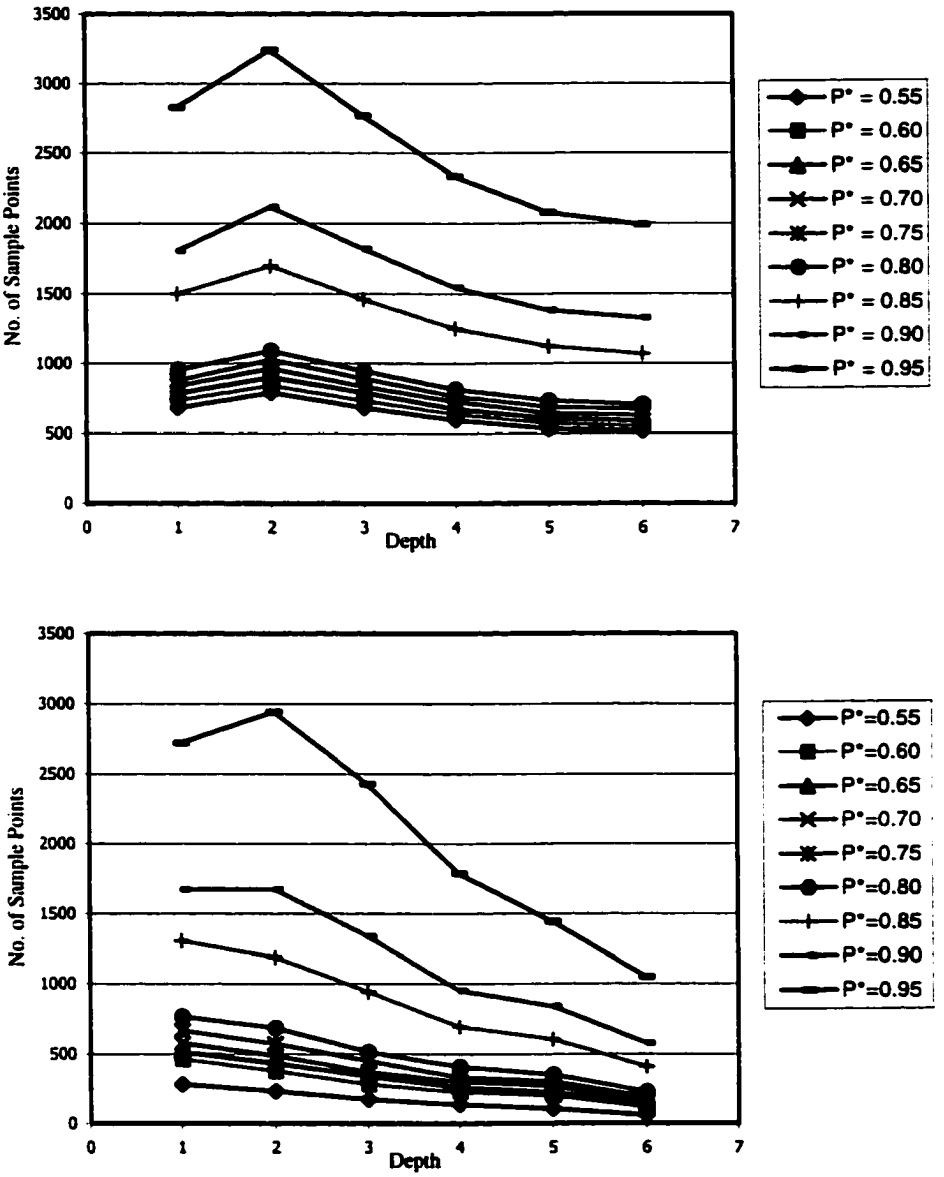



Figure 3.2: Total Number of Sample Points of Each Depth for NP/DD (above) and NP/Subset/DD (below)

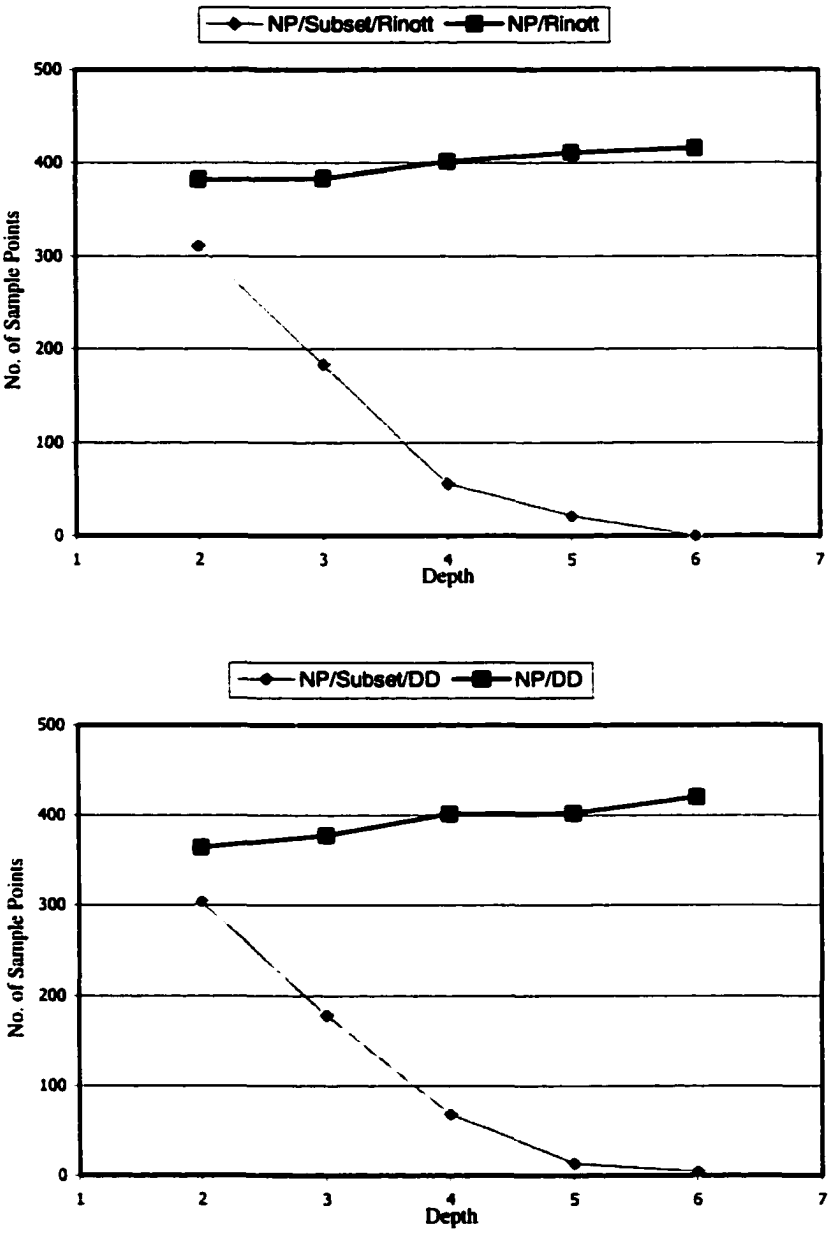



Figure 3.3: Computation effort in the Surrounding Region for NP/subset/Rinott and NP/Rinott (above) and NP/DD and NP/Subset/DD (below)

The main benefit of subset selection is the improvement of the computational efficiency by eliminating inferior systems in the first stage. This to be particularly effective as the depth increases such that the surrounding region becomes larger, thus usually increasing the variance, which in turn dictates more computational effort. However, if the search is identifies a very good region, a thorough search of the surrounding region may become wasted effort and it would be beneficial to filter this region out early. Figure 3.3 shows the results for $P^* = 0.90$. As the depth increases, the NP/Subset/Rinott Algorithm filters out the surrounding region more and more frequently, resulting in lower average effort in the region. On the other hand, the NP/Rinott Algorithm uses more effort in the surrounding region, which is reasonable due to its high variance. Thus, the NP/Subset/Rinott Algorithm can realize substantial benefits over NP/Subset.

## 3.3 Correlated Sampling

### 3.3.1 Two Stage Sampling with Nelson-Matejcik

One assumption in the statistical selection procedure used by the NP/Subset/Rinott Algorithm is that each system is independent, which in simulation optimization implies the simulation samples for comparing the regions must also be independent. However, when comparing simulated systems, researchers prefer to use common random numbers (CRNs), thus making the systems independent. Hence it is important to consider statistical selection methods that allow for correlated systems. One such method is proposed by Nelson and Matejcik (1995) which will now be incorporated into the NP framework. Given a fixed first-stage sample size $n_0$, first-stage samples are randomly obtained from each region by using

the same stream of random numbers for each region. Using these samples, sample variance $S$ of the difference of the sample means is determined, then use this to compute the final sample size given indifference zone $\varepsilon$.

$$N = \max\left\{n_0, \left\lceil \left(\frac{gS}{\varepsilon}\right)^2 \right\rceil\right\} \tag{3}$$

Note that this requires computing the constant $g$ which depends on the initial sample size $n_0$ and the number of regions $M$ that are compared (Nelson and Matejcik, 1995). Furthermore, note that unlike Rinott's two-stage sampling, the sample size for each system is identical in the second stage.

**Algorithm NP/ NM**

Step 1. *Initialization*

Set $k = 0$ and $\sigma(k)=\Theta$.

Specify the constants $\varepsilon$, $\alpha$, and $n_0$. Let $g = T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer *et al.* (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

Given the current most promising region $\sigma(k)$, partition $\sigma(k)$ into $M$ sub-regions $\sigma_1(k), \ldots, \sigma_M(k)$, and aggregate the surrounding region $\Theta\backslash\sigma(k)$ into one region $\sigma_{M+1}(k)$.

Step 3. *First-Stage Sampling*

Step 3-1. Let $i = 1$.

Step 3-2. Use uniform sampling to obtain a set $D_{ij}(k)$ of $N$ sampling points from region $j = 1, 2, \ldots, M+1$ using CRN across regions.

Step 3-3. Use discrete event simulation of the system to obtain a sample performance $L(\theta)$ for every $\theta \in D_{ij}(k)$ and estimate the performance of the region as

$$X_{ij}(k) = \min_{\theta \in D_{ij}(k)} L(\theta).$$

Step 3-4. If $i = n_0$, continue to Step 4. Otherwise, let $i = i+1$ and go back to Step 3-1.

Step 4. *Estimating the Variance of First-Stage Sampling*

Compute the approximate sample variance of the difference of the sample means

$$S^2 = \frac{2\sum_{j=1}^{k}\sum_{i=1}^{n_0}(X_{ij} - \bar{X}_{i\cdot} - \bar{X}_{\cdot j} + \bar{X}_{\cdot\cdot})^2}{(k-1)(n_0-1)}.$$

Where $\bar{X}_{i\cdot} = \sum_{j=1}^{k} X_{ij}/k$, $\bar{X}_{\cdot j} = \sum_{i=1}^{n_0} X_{ij}/n_0$ and $\bar{X}_{\cdot\cdot} = \sum_{i=1}^{n_0}\sum_{j=1}^{k} X_{ij}/kn_0$

Step 5. *Computing Total Sample Size*

Compute the total sample size $N(k) = \max\left\{n_0, \left\lceil \left(\frac{gS}{\varepsilon}\right)^2 \right\rceil\right\}$.

Step 6. *Second-Stage Sampling*

See Step 7 in the NP/Subset/Rinott Algorithm

Step 7. *Estimating Mean of Second-Stage Sampling*

See Step 8 in the NP/Subset/Rinott Algorithm

Step 8. *Estimating Promising Index*

See Step 9 in the NP/Subset/Rinott Algorithm

Step 9. *Checking Stopping Rule*

See Step 10 in the NP/Subset/Rinott Algorithm

Three new algorithms for incorporating statistical selection into the NP framework has been described. In the next chapter, these algorithms are combined with a specific method for implementing the inheritance introduced in Chapter 2.

### 3.3.2 Numerical Evaluation

Since common random numbers are used in the NP/NM Algorithm, less sampling should be required. However, this algorithm will use the same amount of computational effort in each region; whereas, our numerical results from Section 3.2.2 indicate that substantial benefits could be obtained by using a variable sampling effort. Thus, since there are competing benefits to the two approaches, it is not clear which algorithm will perform better, the NP/NM Algorithm, the NP/Subset/DD Algorithm, or the NP/Subset/Rinott Algorithm; therefore it is necessary to evaluate this numerically.

The Monte Carlo problem is previously considered in Section 3.2. In this section a new queuing problem with very different structured is also considered. In this problem, each server represents a user and each buffer slot represents a resource that is to be allocated to a user. Jobs arrive at this system at a rate of $\lambda$. Each user is processing jobs at a rate

$\mu_i$, $i = 1,2,\ldots,N$, and if a job is routed to a user with a full queue, the job is lost. Let $L_i(n_i)$ be the probability of the $i^{th}$ server losing a job ($n_i$ is the number of buffers allocated to the $i^{th}$ server). The goal is to allocate all $K$ available buffer slots to the users in order to minimize job loss. Let $\lambda = 10$, $\mu = 10$, $N = 6$, $K = 18$ and $n_0 = 10$. Twenty replications are used for each experiment with $P^* \in [0.55,0.95]$.

The performance of the three algorithms are compared along two dimensions: speed as measured by the total number of simulation runs, and quality as measured by the average performance of the final solution obtained. Table 3.2 shows the total number of sample points for the three algorithms and both problems. For both problems, notice that the NP/NM Algorithm requires substantially fewer sample points than the NP/Subset/Rinott and NP/Subset/DD. With respect to solution quality, Figure 3.4 shows the performance results for both problems. For the Monte Carlo problem, the NP/NM Algorithm performs better; however, in the queuing problem, the NP/Subset/Rinott and NP/Subset/DD Algorithms show better performance. Moreover, there was little difference between NP/Subset/Rinott and NP/Subset/DD Algorithms. Thus, which algorithm performs better depends on the problem structure, but these results indicate that the NP/NM Algorithm is faster and thus better if the simulation budget is very limited.

Table 3.2: Increase of Sample Points for Different $P^*$ and Algorithms

| | Monte Carlo Problem | | Queuing Problem | |
|---|---|---|---|---|
| | $P^* = 0.55$ | $P^* = 0.95$ | $P^* = 0.55$ | $P^* = 0.95$ |
| NP/NM | 14000 | 29785 | 6783 | 153865 |
| NP/Subset/Rinott | 59585 | 249261 | 27623 | 201568 |
| NP/Subset/DD | 76638 | 302855 | 27360 | 200858 |

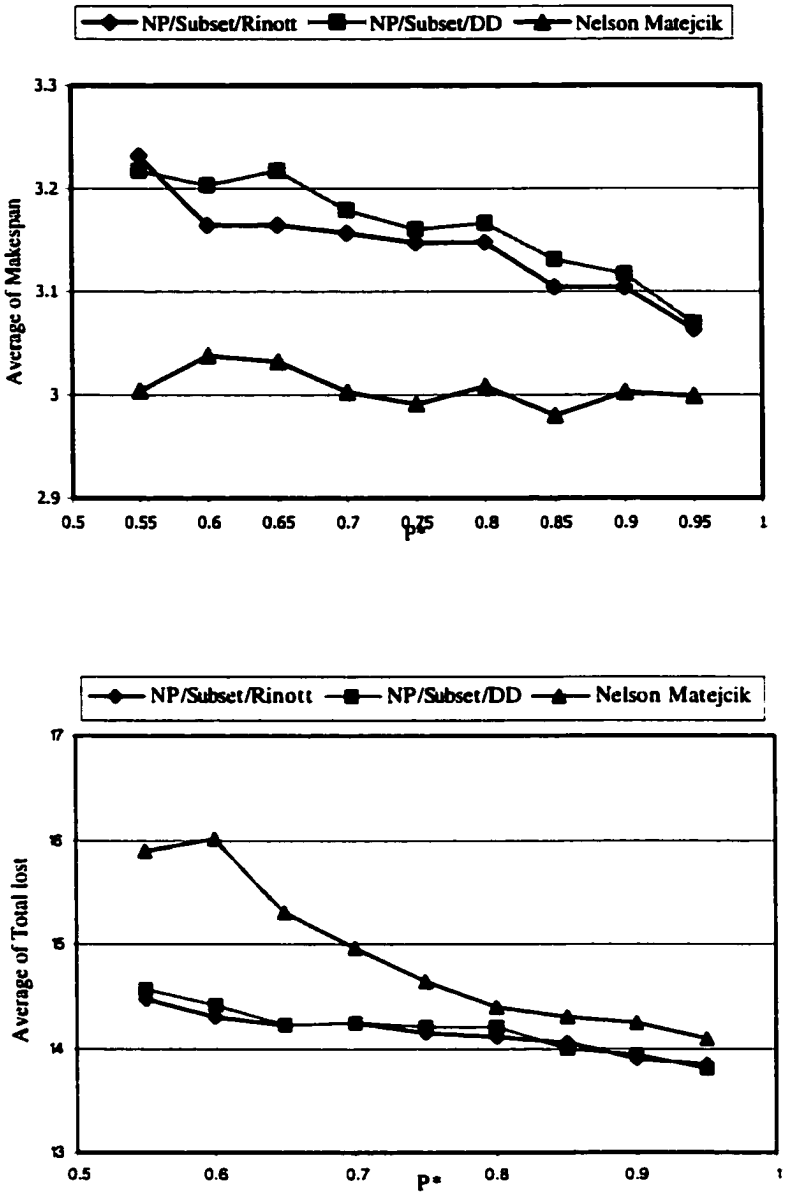



Figure 3.4: Average Performance of Final Solution for the Monte Carlo Problem (above) and the Queuing problem (below)

## 3.4 Best Probability of Correct Selection

One of the key parameters that must be carefully chosen for the two-stage NP method is the probability of correct selection ($P^*$) in each iteration. If computation time is not an issue, it can be set using some equation of Ólafsson (1999) to set it according to the desired probability of terminating correctly. However, with limited computing budget it is of interest to empirically determine its best value.

This section does that for both the Monte Carlo problem and queuing problem. Thus for this section, the algorithm is terminated only after a fixed number of simulation evaluations have been conducted. For the Monte Carlo problem, the simulation was run for 6 different sets of simulation estimates: 20,000, 30,000, 40,000, 50,000, 65,000, or unlimited for the NP/NM Algorithm 9 different sets of simulation estimates: 75,000, 100,000, 125,000, 150,000, 175,000, 200,000, 225,000, 250,000, or unlimited for the NP/Subset/Rinott Algorithm, and 10 different sets of simulation estimates: 100,000, 125,000, 150,000, 175,000, 200,000, 225,000, 250,000, 275,000, 300,000, or unlimited for the NP/Subset/DD Algorithm. For the queuing problem, the simulation was run for 6 different sets of simulation estimates: 50,000, 75,000, 100,000, 125,000, 150,000, or unlimited for the NP/NM algorithm, 9 different sets of simulation estimates: 75,000, 100,000, 125,000, 150,000, 175,000, 200,000, 225,000, 250,000, or unlimited for the NP/Subset/Rinott algorithm, and 10 different sets of simulation estimates: 100,000, 125,000, 150,000, 175,000, 200,000, 225,000, 250,000, 275,000, 300,000, or unlimited for the NP/Subset/DD Algorithm. Figure 3.5 shows the result of the NP/NM Algorithm. For the range of $P^* \in (0.55,0.60,0.65,0.70)$, there is no significant difference of average performance and sample points even if the simulation

estimates are increased. Similar results are obtained for the NP/Subset/Rinott Algorithm, as illustrated by Figure 3.6. Those results show that it is possible to improve the performance found by increasing the $P^*$ value to the range of $P^* \in [0.7,0.8]$, but with limited computation budgets the performance degenerates very quickly for high $P^*$ values. Furthermore, relatively low $P^*$ values are recommended because the amount of computational effort increases exponentially with $P^*$.

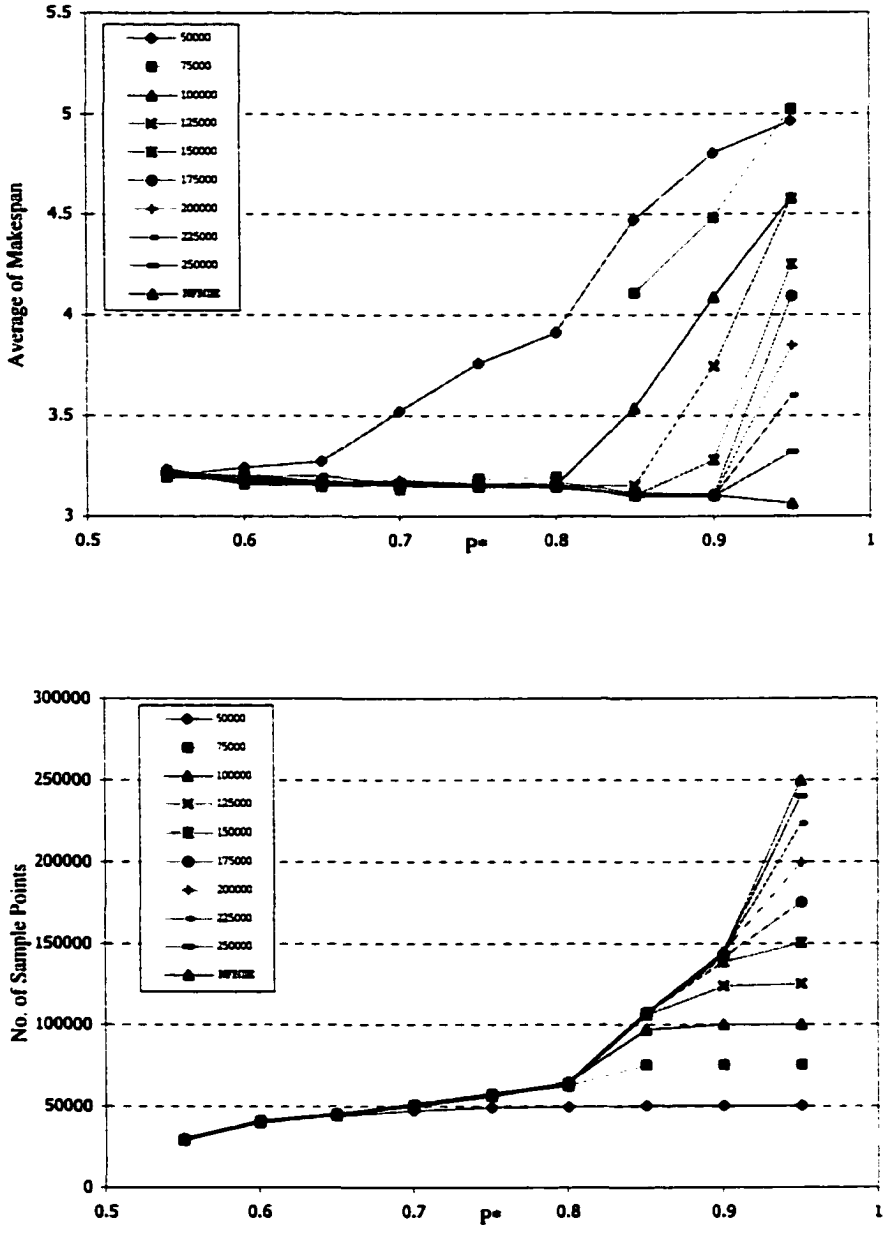



Figure 3.5: Performance for the Monte Carlo problem using NP/Subset/Rinott Algorithm

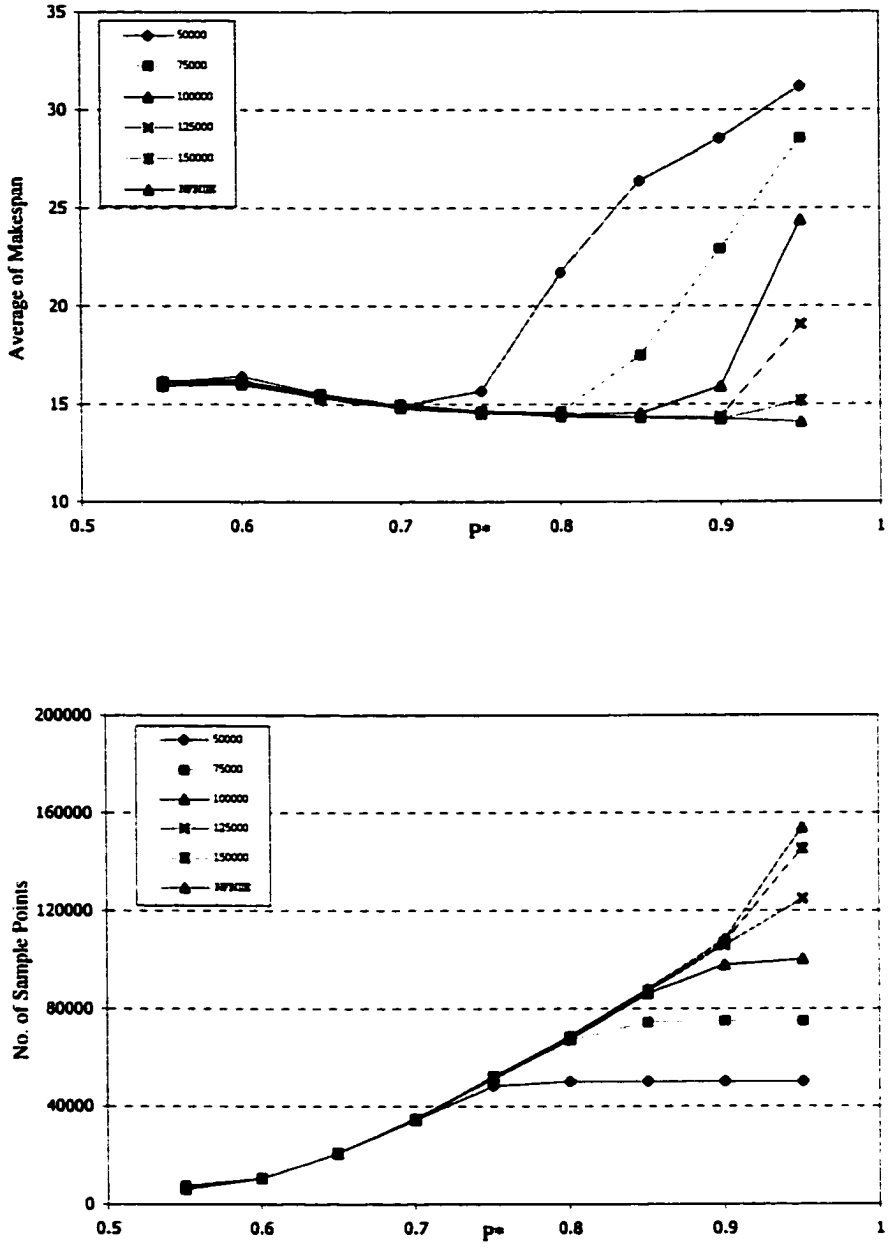



Figure 3.6: Performance for the Queuing Problem using NP/NM Algorithm

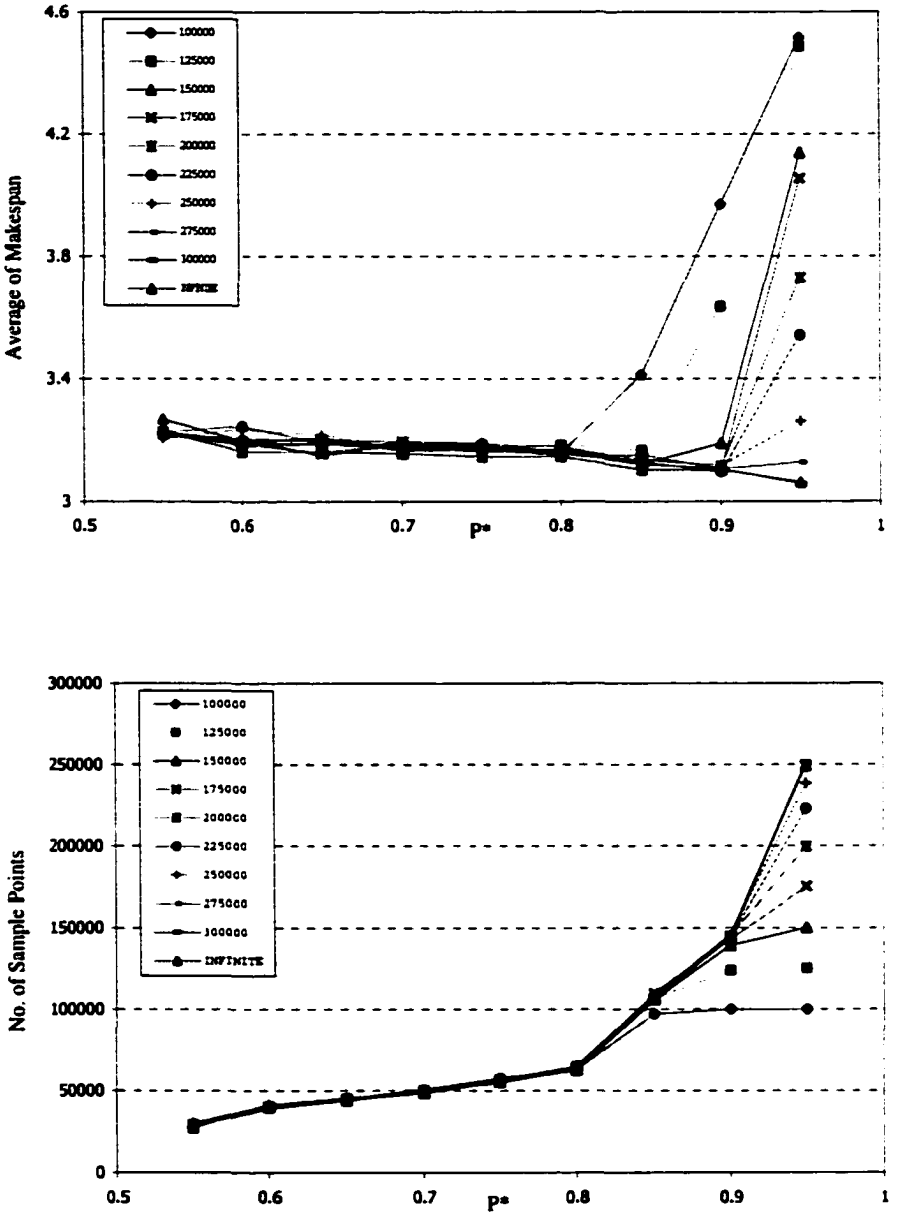



Figure 3.7: Performance for the Monte Carlo problem using NP/Subset/DD Algorithm

## 3.5 Conclusions

The above three methods take advantage of the statistical selection procedure, namely Rinott's with subset procedure, *DD*'s with subset procedure, and Nelson Matejcik's procedure. The results show that the NP/NM Algorithm needs relatively less computational effort which is good for optimization with budget limits. When restricting computation time, the performance degenerates very quickly for high $P^*$ values. Also, relatively low $P^*$ values are advisable because the amount of computational effort increases exponentially with respect to $P^*$. Using the subset selection procedure, inferior systems can be deleted in advance. As a results, the NP/Subset/Rinott Algorithm is preferred over the NP/Rinott Algorithm and the NP/Subset/DD Algorithm is preferred over the NP/DD Algorithm. It has been shown that which algorithm is best depends on the problem to solve. The NP/NM Algorithm is good for Monte Carlo problems. By using a two-stage sampling computation effort can be saved over pure NP. For the queuing problem, the NP/Subset/Rinott Algorithm and the NP/Subset/DD Algorithm give better results. And in both problems, the NP/Subset/Rinott Algorithm and the NP/Subset/DD Algorithm show nearly the same results.

# Chapter 4
# NP/GA

In this chapter, one particular method that increases the effectiveness of the inheritance is presented. The potential drawbacks of the pure NP method are that it can be stuck in a local optimum by the uneven sampling error, and it has some difficulty in backtracking. In Chapter 3, the statistical selection method was used to address these problems. In this chapter, NP is combined with statistical selection, Genetic Algorithm (GA) and inheritance.

Even though there is no guarantee of global convergence, GA is well known and effective for heuristic algorithms. Therefore, by applying a GA search to each sub-region, sample points that better represent the best performance in their region can be obtained. Based on these sample points, the next promising region can be more precisely determined. This is because GA is used for finding local optimums from whole region by finding the best solution of each region. As a result, a combined algorithm retains the benefits of both methods. GA is important in terms of the more work we put into obtaining good points, the more valuable it is to keep them using inheritance. In addition, by adding inheritance to this algorithm, the time to find optimums can be accelerated and good starting points for sampling in a GA procedure can be obtained.

## 4.1 Genetic Algorithm

Genetic Algorithm (GA), which was originally developed by Holland (1975), is a search algorithm based on the mechanisms of natural selection and genetics. This algorithm states that stronger individuals are most likely to survive in a competing environment.

GA uses strings of characters (elements) to represent solutions called chromosomes. And each character in such a string is a gene representing a certain choice of gene. In biology, genes are passed to the next generations through reproduction. The reproduction process converges to optimal very much like an optimization paradigm. GA uses a positive value, known as a fitness value to reflect the degree of "goodness" of the chromosome which coincides with the value of the object for the problem (Man, 1999). Recently, the GA has received recognition as a tool for solving optimization problems in the industrial engineering area. GA is the most widely known evolutionary computation method which is both powerful and broadly applicable to stochastic optimization. The basic premise behind GA is that better solutions have better gene combinations (schema). Therefore, by carrying over this good schema during many generations, a population containing the optimal solutions can be obtained. Various hybrid genetic algorithms that outperformed pure genetic algorithms have been proposed (Gong *et al.*, 1997; Shi *et al.*, 1999). Further discussion can be found in Man (1999).

### 4.1.1 Conventional Genetic Algorithm

GA starts by generating a test population. Based on a fitness evaluation of these population members, two parents are selected. After applying crossover and mutation using

these parents, a new generation is formed. The fitness test is then applied to this new generation and this evolutionary procedure is repeated as necessary.

Step 1. *Initial populations*

Randomly generate an initial population

Step 2. *Fitness test*

Evaluate the fitness of population members

Step 3. *Selection*

Choose two parents from population based on the fitness test

Step 4. *Mutation and crossover*

Apply crossover and mutation to selected parents results to get the new generation

Step 5. *Fitness test*

Apply the fitness test against the new generation

Step 6. *Stopping criteria*

Check stopping criteria. If it satisfies stopping criteria, then stop. Otherwise, go to step 3

### 4.1.2 Parent Selection

There are several methods for selecting parents. The most common types are the roulette wheel, $(\mu + \lambda)$, tournament, and ranking-ordered selection.

- *Roulette wheel selection* (Holland, 1975)

Roulette wheel is the best known selection method. The sum of the fitness measures for all members is first calculated and then the fitness for each member is normalized against of this sum. All members are put on a roulette wheel so the larger the fitness, the larger the space on the wheel for the member, and the larger the probability of being chosen for mating.

- $(\mu + \lambda)$-*selection* (Bäck, 1994)

  Many researchers prefer this method when dealing with combinatorial optimization because it prohibits selection of duplicate chromosomes from the population.

- *Tournament selection* (Goldberg *et al.*, 1989)

  This method contains random and deterministic features. A set of chromosomes is first randomly chosen, then a chromosome is picked out of this set for production. Tournament size is the number of chromosomes in the set, which is generally 2.

- *Rank-ordered selection*

  A prescribed number of parents are taken for mating from the top of a rank-ordered list according to its fitness value.

### 4.1.3 Crossover and Mutation

After parents have been chosen, they mate to produce two children. This is done by crossing (mixing) over the genes of one parent with the other parent's. Crossover is used to facilitate the mixing of good genes within a population. There are several methods for crossover.

- *Point crossover*

A point is randomly chosen in the chromosomes of the parents then the left side of parent 1 is combined with the right side of the other parent and the right side of parent 1 is combined with the left side of the other parent. This makes another combinations.

- *Multiple-point crossover*

  Multiple points are used to make new combinations.

- *Uniform crossover*

  It's analogous to flipping a coin. For example, if head comes up, a gene of parent 1 is given to child 1.

Mutation is the random change of once or more of the gene value and generally occurs with a very small probability. Mutation is used to introduce genetic variation.

## 4.2 Algorithm

Like the development seen in Chapter 3, the main framework of Algorithm NP/GA is the NP method. In this method, GA procedures are incorporated into the NP method in the sampling step, and make the inheritance record after calculating the promising index.

**Algorithm NP/GA with Inheritance**

Step 1. *Initialization*

Set $k = 0$ and $\sigma(k)=\Theta$.

Step 2. *Partitioning*

If $d(\sigma(k)) \neq d^*$, that is, $\sigma(k) \neq \sum_0$, partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ sub-regions $\sigma_1(k),...,\sigma_{M_{\sigma(k)}}(k)$.

If $d(\sigma(k))=d^*$, then let $M_{\sigma(k)}=1$ and $\sigma_1(k)=\sigma(k)$.

If $d(\sigma(k))\neq 0$, that is $\sigma(k)\neq\Theta$, aggregate the surrounding region $\Theta\backslash\sigma(k)$ into one region $\sigma_{M_{\sigma(k)+1}}(k)$.

Step 3. *Initial Population*

If $k=0$ and $d(\sigma(k))\neq d^*$, use random sampling to obtain an initial population $N$ strings from each of the regions $\sigma_j(k)$, $j=1,2,...,M_{\sigma(k)}+1$,

$$POP_I^j=[\theta_I^{j1},\theta_I^{j2},\ldots,\theta_I^{jN}],\ j=1,2,\ldots,M_{\sigma(k)}+1$$

else use the population $INH_{k-1}=[\theta^{j1},\theta^{j2},\ldots,\theta^{jN}]$, $j=\hat{j}_k$ as the initial population. Part of the lack should be fulfilled using uniform sampling.

Step 4. *GA Search*

Apply the GA procedure to each initial population $POP_I^j$ to obtain a final population for each region $\sigma_{lj}(k)$, $j=1,2,...,M_{\sigma(k)}+1$

$$POP_F^j=[\theta_F^{j1},\theta_F^{j2},\ldots,\theta_F^{jN}],\ j=1,2,\ldots,M_{\sigma(k)}+1$$

Step 5. *Calculating Promising Index (Overall Fitness)*

Estimate the overall fitness of the region by the performance of the fittest chromosome in the final population. That is, the overall fitness of each region is estimated by

$$\hat{L}(\sigma_j)=\min_{i\in\{1,2,\ldots,N_j\}} L(\theta_F^{ji}),\ j=1,2,\ldots,M_{\sigma(k)}+1$$

Step 6. *Backtracking*

Calculate the index of the region with the best overall fitness (most promising region).

$$\hat{j}_k \in \arg \min_{j \in \{1,2,\ldots,M_{\sigma(k)}+1\}} \hat{L}(\sigma_j)$$

If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a sub-region of $\sigma(k)$, then let this sub-region be the most promising region of next iteration. That is let $\sigma(k+1) = \sigma_j(k)$, $j < M_{\sigma(k)}$ and keep the population of this promising index

$$INH_k = [\theta^{j1}, \theta^{j2}, \ldots, \theta^{jN}], \; j = \hat{j}_k$$

Otherwise, if the index corresponds to the surrounding region, backtrack to the region, $s(\sigma(k))$, of the current most promising region. That is, let $\sigma(k+1) = s(\sigma(k))$.

Step 6. *Checking Stopping Rule*

If $\sigma(k+1) \in \Sigma_0$, stop and let $\sigma_{opt} = \sigma(k+1)$ else let $k = k+1$ and go back to Step 2.

The next algorithm illustrates how the NP method is combined with GA and statistical selection. By combining GA and statistical selection in the sampling procedure of the NP method as well as inheriting the sample points to the next iteration, this hybrid algorithm should converge more quickly with better solutions.

**Algorithm NP/GA/Statistical Selection with Inheritance**

Step 1. *Initialization*

Set $k = 0$ and $\sigma(k) = \Theta$.

Step 2. *Partitioning*

If $d(\sigma(k)) \neq d^*$, that is, $\sigma(k) \neq \sum_0$, partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ sub-regions $\sigma_1(k),...,\sigma_{M_{\sigma(k)}}(k)$

If $d(\sigma(k))=d^*$, then let $M_{\sigma(k)} = 1$ and $\sigma_1(k) = \sigma(k)$.

If $d(\sigma(k)) \neq 0$, that is $\sigma(k) \neq \Theta$, aggregate the surrounding region $\Theta \setminus \sigma(k)$ into one region $\sigma_{M_{\sigma(k)-1}}(k)$.

Stage I Sampling

Step 3. $i = 1$

Step 4. *Initial Population*

If $k = 0$ and $d(\sigma(k)) \neq d^*$, use random sampling to obtain an initial population $N$ strings from each of the regions $\sigma_j(k)$, $j = 1,2,...,M_{\sigma(k)} + 1$,

$$POP_I^j = [\theta_I^{j1}, \theta_I^{j2}, \ldots, \theta_I^{jN}], \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

else use the population $INH_{k-1} = [\theta^{j1}, \theta^{j2}, \ldots, \theta^{jN}]$, $j = \hat{j}_{k-1}$ as the initial population. Part of the lack should be fulfilled using uniform sampling.

Step 5. *GA Search*

Apply the GA to each initial population $POP_I^j$ to obtain a final population for each region $\sigma_{Ij}(k)$, $j = 1,2,...,M_{\sigma(k)} + 1$

$$POP_F^j = [\theta_I^{j1}, \theta_I^{j2}, \ldots, \theta_I^{jN}], \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

Step 6. *Calculating Overall Fitness*

Estimate the overall fitness of the region based upon the performance of the fittest chromosome in the final population. That is, the overall fitness of each region for a given final population is estimated by

$$\hat{L}_i(\sigma_j) = \min_{k \in \{1,2,\ldots,N\}} L(\theta_F^{jk}), \quad j = 1, 2, \ldots, M_{\sigma(k)} + 1$$

Step 7. If $i = n_0$, continue to Step 8. Otherwise let $i = i + 1$ and go back to Step 3.

Stage II Sampling

Step 8. *Estimating Mean and Variance of First-Stage Sampling*

Calculate the first-stage sample means and variance

$$\bar{L}_j(k) = \frac{1}{n_0} \sum_{i=1}^{n_0} \hat{L}_i(\sigma_j),$$

and

$$S_j^2 = \frac{\sum_{i=1}^{n_0} [\hat{L}_j(\sigma_j) - \bar{L}_j(k)]^2}{n_0 - 1},$$

for $j = 1, 2, \ldots, M_{\sigma(k)} + 1$

Step 9. *Computing Total Sample Size for Second-Stage Sampling*

Compute the total sample size for all $j$

$$N_j(k) = \max\{n_0 + 1, \left\lceil \frac{h^2 S_j^2(k)}{\varepsilon^2} \right\rceil\},$$

where $\varepsilon$ is the indifference zone and $h$ is a constant that is determined by $n_0$ and the minimum probability $P^*$ of the correct selection.

Step 10.*Second-Stage Sampling*

Obtain $N_j(k)-n_0$ more simulation estimates of the system performance for all $j$ as the same way of Step 3 thru Step 7 above.

Step 11.*Estimating Mean of Second-Stage Sampling*

Calculate the second-stage sample means of the chromosome

$$\hat{L}(\sigma_j(k)) = \overline{L}_j(k) = \frac{1}{N_j}\sum_{i=1}^{N_j}\hat{L}(\sigma_j), \quad j = 1, 2, \ldots, M_{\sigma(k)}+1$$

Step 12.*Calculating Promising Index (Overall Fitness)*

Calculate the index of the region with the best overall fitness (most promising region).

$$\hat{j}_k \in \arg\min_{j \in \{1,2,\ldots,M_{\sigma(k)}+1\}} \hat{L}(\sigma_j)$$

If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a sub-region of $\sigma(k)$, then let this sub-region be the most promising region of the next iteration. That is let $\sigma(k+1) = \sigma_j(k)$, $j < M_{\sigma(k)}$ and keep the population of this promising index

$$INH_k = [\theta^{j1}, \theta^{j2}, \ldots, \theta^{jn_0}], \; j = \hat{j}_k$$

Otherwise, if the index corresponds to the surrounding region, backtrack to the region, $s(\sigma(k))$, of the current most promising region. That is, let $\sigma(k+1) = s(\sigma(k))$.

Step 13.*Checking the Stopping Rule*

If $\sigma(k+1) \in \Sigma_0$, stop and let $\sigma_{opt} = \sigma(k+1)$ else $k = k+1$ and go back to Step 2.

## 4.3 Numerical Evaluation

The Monte Carlo problem considered in Chapter 3 is used to numerically evaluate the performance of these algorithms. In all of these experiments, parameters are set as $M = 4$, $R = 5$, and the size of first-stage sample points in each region are set as $n_0 = 10$. Twenty replications are used for each experiment with $P^* \in [0.55, 0.95]$.

### 4.3.1 Probability of Finding Exact Solution

Intuitively, one would expect using the inheritance has a high probability of finding the exact solution. The following sets of experiments support this. Figure 4.1 indicates that the probability of finding the correct selection can be increased significantly with the number of the GA iterations used. Also, the difference between algorithms are increasing with the number of the GA iterations used, which means the more work we put into getting good point, the more valuable it is to keep them using inheritance.

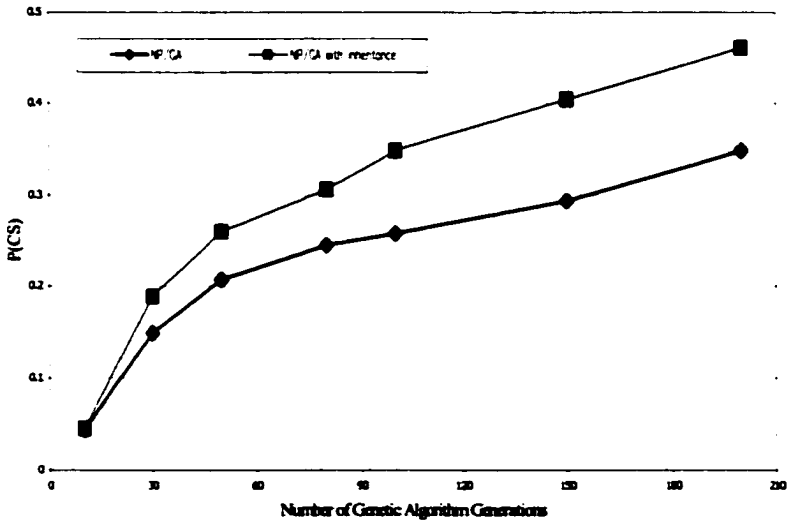



Figure 4.1: Probability of Finding Exact Optimal Solution

### 4.3.2 Probability of Finding Solutions Using Indifference Zone

Section 4.3.1 showed the results without statistical selection method. In this section, results with statistical selection method are shown. The indifference zone is set as $\varepsilon = 0.5$. Figure 4.2 shows similar results with Figure 4.1. Inheritance gives significantly better results than without inheritance. Also it can be seen that the P(CS) with statistical selection method is better than without statistical selection method whether the inheritance is hired or not. Let's see when the 80 GA iterations are used. When inheritance is not used the value of P(CS) of NP/GA is 0.25 in Figure 4.1. On the contrary, the value of P(CS) of NP/GA/Statistical Selection is 0.7 in Figure 4.2. When inheritance is used the value of P(CS) of NP/GA is 0.3 in Figure 4.1. On the contrary, the value of P(CS) of NP/GA/Statistical Selection is 0.75 in Figure 4.2. Also, it can be seen that the difference between with inheritance and without inheritance of P(CS) with statistical selection is smaller than without statistical selection.

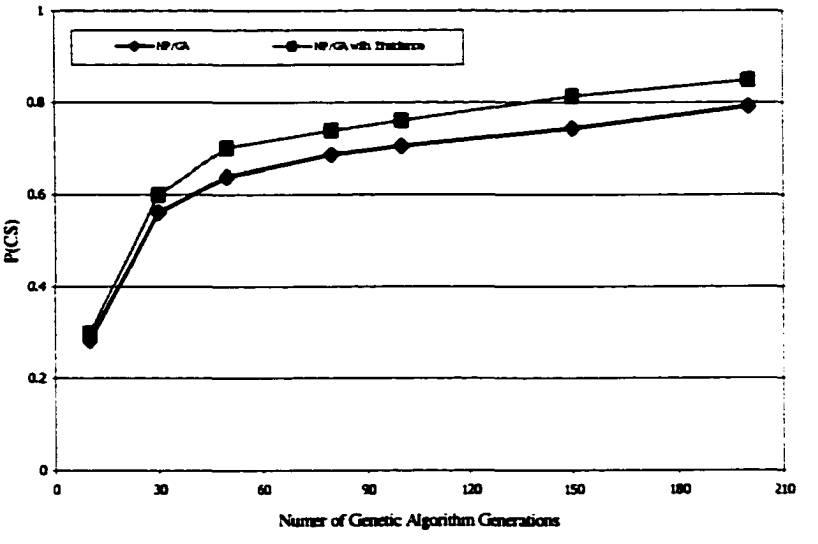



Figure 4.2: Probability of Finding Optimal Solution with Indifference Zone

### 4.3.3 Average of Makespan

The objective of the Monte Carlo Problem is to find the resource allocation that minimizes total completion time. Figure 4.3 shows the average of 60 simulations which are similar to previous results. By using inheritance the total completion time is reduced. This implies that even when an optimal solution cannot be found, inheritance still yields a better solution than without inheritance.

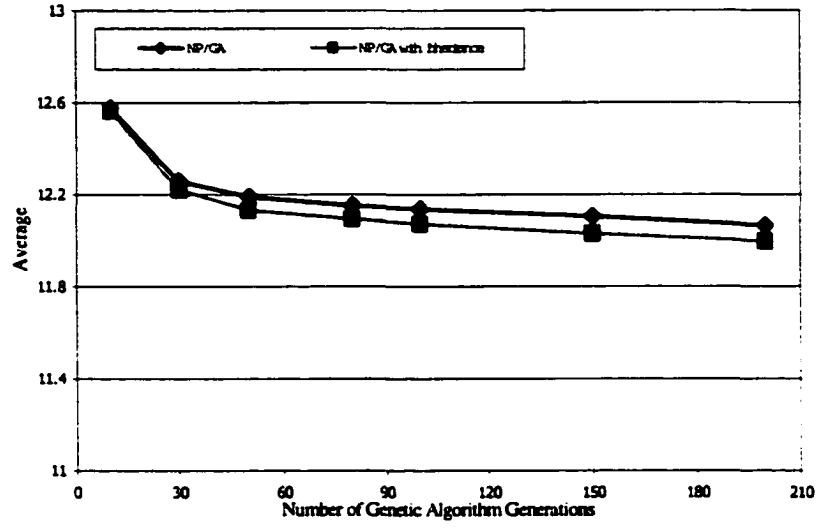



Figure 4.3: Average of MakeSpan

## 4.4 Conclusions

Potential drawback of the pure NP method is that it can be stuck in a local optimum by the uneven sampling error and has some difficulty in the backtracking. With the statistical selection method which is introduced in previous chapter, genetic algorithm (GA) is used to address these problems in this chapter. Additionally, good sample points which are obtained in every iteration, are carried over to the next iteration by employing inheritance concept.

Two algorithms are suggested in this chapter: NP/GA with Inheritance and NP/GA/Statistical Selection with Inheritance. There are three noticeable results. First, inheritance gives better results than without inheritance. Second, the statistical selection method gives better results than without the statistical selection method whether the inheritance is hired or not. Lastly, the difference in finding the probability of correct selection between simulations with inheritance and without inheritance using the statistical selection method is smaller than the simulations without the statistical selection method.

# Chapter 5
# Data Clustering

Today, the amount of data being collected in databases far exceeds our ability to reduce and analyze these data without the use of automated analysis techniques. Many scientific government and corporate information systems are being overwhelmed by the flood of data that are generated and stored routinely in large databases. As a result, a tool is needed to sift out the important data from these databases. Knowledge discovery in databases (KDD) or data mining is the field that is evolving to provide automated analysis solutions. This chapter mainly focuses upon clustering analysis. New algorithms for clustering are suggested.

## 5.1 Knowledge Discovery in Databases and Data Mining

The general goal of data mining is to extract implicit, previously unknown, hidden, and potentially useful information from raw data in an automatic fashion, rather than developing individual applications for each particular need (Deogun *et al.*, 1997). Data mining techniques can be divided into five classes in terms of mathematical formulation: Predictive Modeling, Clustering, Dependency Modeling, Data Summarization, Change and Deviation Detection (Bradley *et al.*, 1998). The goal of predictive modeling is to predict a specific attribute based on the other attributes in the data given in the training set. If the output of the predicted model is numeric or continuous, then this prediction problem can be viewed as a regression problem. Otherwise, it can be categorized as a classification problem. The concern with predictive modeling is how well has it been generalized so that it performs

on data not included in the training set. By using a complex estimate function, training data can be *overfitted* (Shavlik and Dietterich, 1990). *Overtraining* can also lead to poor generalization even when the complexity of the function constructed is optimal (Bös, 1996). Therefore, there is a trade-off situation between generalization and overfitting.

In classification, the goal is to predict the most likely state of a categorical variable given the value of other variables. There are several techniques where projection methods are the most common approach. This method divides the attribute space into decision regions and associates a prediction with each (Hertz *et al.*, 1991). Decision tree or rule-based classification make a piecewise constant approximation of the decision surface (Bennett, 1992; Breiman *et al.*, 1984). The other types of methods are metric-space based methods. This method defines a distance metric based on data points and predicts the class value based on the proximity to these data points in the training set. The best-known method is K-nearest-neighbor method (Duda *et al.*, 1973). Relative to the generalization or how well the estimated classification function performs on new data not included in the training set, there are two approaches with the goal of refining the generalization ability of the resulting classifier. The first is feature selection or computing a separating surface utilizing a minimum number of problem features. The second is the support vector machine (SVM) that attempts to compute a set separation while maximizing the margin of separation between the two subsets.

Clustering or segmentation will group the data records into subsets where items in each subset are more similar to each other than items in other subsets. In this thesis, the main focus is on the data clustering, which is described in detail in Section 5.3.

Dependency Modeling targets modeling that generates a joint probability density function of the process (or processes) that describes the data. Insight into data is often gained

by deriving some causal structure within the data. Models of causality can be derived based on the functional dependencies between fields within the data (Piatetsky-Shapiro *et al.*, 1991). The most well known density estimation method is the Expectation-Maximization (EM) algorithm (Dempster *et al.*, 1977). The EM algorithm iteratively updates the Gaussian parameter and the prior density probability given the posterior density probability. EM algorithm converges to a local minima (Neal and Hinton, 1999).

In some cases, data mining simply needs to extract a compact pattern that describes subsets of the data. Sometimes summarization of the subset or predictions relations between fields is all that is needed. One common method is association rule (Agrawal *et al.*, 1994). Associations are rules that state certain combinations of values occur with other combinations of values with a certain frequency and certainty. Sometimes detecting changes and deviations, like in a time-series or some other ordering, is needed. In this method, the ordering of observations is taken into account.

As discussed above, numerous problems in data mining and knowledge discovery can be formulated as an optimization problem.

## 5.2 Data Clustering

Clustering can be defined as the process of grouping data into classes or clusters so that objects within each cluster have a high similarity in comparison to one another, but are very dissimilar to objects in other clusters (Han and Kamber, 2001). Many applications of data clustering can be found, such as the grouping of genes and proteins that have similar functionality (Harris *et al.*, 1992), grouping of spatial locations prone to earthquakes from seismological data (Byers, 1998), characterization of different customer groups based on

purchasing patterns, and characterization of documents on the World Wide Web (Boley, 1999). Clustering analysis is the unsupervised equivalent of classification where a given collection of unlabeled patterns is grouped into meaningful clusters. In contrast, with supervised classification there is a collection of labeled patterns (Jain *et al*, 1988). Existing clustering algorithms can be classified into the following two categories: hierarchical clustering and non-hierarchical clustering.

### 5.2.1 Hierarchical Clustering

Hierarchical algorithms can be either agglomerative or divisive. In agglomerative methods, the first stage starts with the singleton clusters. The pair of clusters that optimizes some criterion is agglomerated at each stage corresponding to each number of clusters. These steps repeat until the desired number of clusters is obtained. Most agglomerative hierarchical algorithms primarily differ in how they update the similarity between existing clusters and the merged clusters (Jain and Dubse, 1988). Most hierarchical clustering algorithms use criteria such as the distance between clusters (single linkage, average linkage) or sum of squares (Fraley and Raftery, 1998) for optimization.

In the single linkage method, each cluster is represented by all the data points in the cluster. The similarity between two clusters is measured by the similarity of the closest pair of data points belonging to different clusters. This method can find clusters of arbitrary shapes and different sizes; however, it is highly susceptible to noise, outliers, and artifacts. Unlike the other methods, each cluster can be represented by a centroid of the points and the similarity between clusters is measured by the centroids of these clusters. CURE (Sudipto *et al.*, 1998) has been proposed to cover the drawback of both methods. CURE uses a constant

number of representative points to represent a cluster. Similarity between two clusters is measured by the similarity of the closest pair of representative points belonging to different clusters.

In some agglomerative hierarchical algorithms, the similarity between two clusters is captured by the interconnectivity between pairs of items belonging to different clusters. ROCK (Sudipto *et al.*, 1999) is an agglomerative algorithm that operates on a derived similarity graph, which scales the aggregate interconnectivity with respect to a user-specified inter-connectivity model.

Divisive hierarchical clustering algorithms such as Minimal Spanning Tree (MST) or graph-partitioning algorithms that use a sparse similarity matrix can be represented by a sparse graph. Despite this benefit, MST-based methods are highly susceptible to noise. However, graph partitioning based methods are overall much more robust than the other method (Karypis and Kumar, 1998).

Several hierarchical algorithms are designed to find clusters that fit some static models. In this case, the algorithms cannot work if the choice of parameters in the static model is incorrect with respect to the data set being clustered. Karypis *et al.* (1999) suggest CHAMELON that measures the similarity of two clusters based on a dynamic model.

### 5.2.2 Partitional Clustering

Partitional clustering algorithms are more iterative than hierarchical. In a partitioning algorithm, the number of clusters $K$ is usually specified in advance. Partitioning clustering procedures typically start with the patterns partitioning the data set into a number of clusters and divide the patterns by increasing the number of partitions. A popular performance

function for measuring data clustering accuracy is the total within-cluster variance, or the total mean-square error (MSE). The K-means algorithm is a popular algorithm that attempts to find k clusters that minimizes MSE. K-means (Ball and Hall, 1964) is a centroid-based clustering algorithm. Centroid-based algorithms are suitable for data in metric spaces in which it is possible to compute a centroid of a given set of points. The K-means method can get trapped in local optima and is very sensitive to the choice of the initial centroid. The K-means algorithm is sensitive to outliers since the distortion of data can be happened with an object of extremely large value. In K-medoid method, the medoid can be used as the most centrally located object in a cluster, instead of taking the mean value of the objects in a cluster as a reference point (Han and Kamber, 2001).

Another iterative procedure, nearest neighbor clustering, was proposed by Lu and Fu (1978). It assigns each unlabeled datum to the cluster of its nearest labeled neighbor datum, provided the distance to that labeled neighbor is below a threshold. The process continues until all the data are labeled or no additional labeling occurs. Raghavan and Birchand (1979) applied genetic algorithm to clustering to minimize the squared error criteria function. Babu and Murty (1993) studied hybrid algorithm of GA and K-means. GA is only used to find good initial cluster centers. Al-Sultan (1995) applied the tabu search approach to the clustering problem. This method is also combined with the K-means algorithm for cluster generation (Al-Sultan *et al.*, 1996). In Rose *et al.* (1993), a deterministic annealing approach was proposed for clustering. The use of deterministic annealing in proximity-mode clustering was explored in Hofmann and Buhmann (1997). In the paper by Aarts and Korst (1989), they showed that simulated annealing is statistically guaranteed to find the global optimal solution.

## 5.3 Clustering using NP

In data mining there may be millions of data objects or instances that are to be learned. To effectively deal with large number of instances, learning may be based on a randomly selected subset of instances, not on the entire database. However, by using random sampling, there is a risk in introducing a new bias into the learning; therefore, it is essential to explicitly account for the noise introduced by sampling. Most clustering algorithms cannot handle this and the use of sampling introduces the risk of bias. Moreover, most clustering algorithms work well only in a few dimensions (attributes), so better scalability with respect to a large number of instances and high dimensionality is required. NP is of course specifically designed to handle noisy performance such as the estimated performance resulting from using a sample of instances. In particular, backtracking allows the algorithm to make corrections for wrong moves due to a biased sample. Also, by fixing the attributes one at a time, the NP method can effectively deal with high dimensions. In this section, several new algorithm variants are introduced. In particular, the well-known clustering algorithms, K-means algorithm and genetic algorithm are incorporated into the NP framework. The concept of inheritance employed here has been introduced in Chapter 2. In addition, the two-stage sampling procedure explained in Chapter 3 is incorporated here. Thus there are 6 different algorithms that are applied to the clustering problem: NP/NM/K-means, NP/NM/K-means with inheritance, NP/NM/Genetic, NP/NM/Genetic with inheritance, NP/NM/Genetic/K-means, NP/NM/Genetic/K-means with inheritance. Since these variants are quite similar, only details for two of the algorithms are presented. The details for the remaining algorithms can be found in the Appendix. As already explained in Section 5.2,

clustering algorithms can be simply classified into two categories: hierarchical clustering and partitional clustering. The clustering using NP is a partitional clustering where each cluster is defined by its center. This algorithm tries to solve the problem which classifies given instances into fixed number of clusters so as to minimize the distance between points within a cluster. Every algorithm repeats until the stopping criteria is met.

The following notation will be used to describe the algorithms.

$N_C$ : Total number of clusters (given).

$N_F$ : Total number of features (given).

$n_k$ : Stopping parameter for K-means algorithm (given).

$m_i$ : The number of values which are considered as centers at each feature $i = 1,\ldots,N_F$ (given without using inheritance).

$z_j^t$ : The center of the cluster $j = 1,\ldots,N_C$ at time $t$ (decision variable).

$I_j^t$ : Set of indices of instances assigned to cluster $j = 1,\ldots,N_C$ at time $t$ (given).

$C_i$ : Cluster $i = 1,\ldots,N_C$.

NOIN : Total number of centers to inherit next iteration.

$X = (x_1, x_2, \ldots x_{N_F})$ : Instances that consists of a vector of $N_F$ measurements (given).

The squared error criterion function is used as a performance measure. Its calculation is as follows.

$$J(z)=\sum_{i=1}^{N_C}\sum_{x\in I_j^i}|x-z_j^i|,\ \ i=1,\ldots,N,\ \ j=1,2,\ldots,M_{\sigma(k)}+1.$$

Using a sample of instances, the estimate

$$\hat{L}(z)=\sum_{i=1}^{N_C}\sum_{x\in I_j^i}|x-z_j^i|,\ \ i=1,\ldots,N,\ \ j=1,2,\ldots,M_{\sigma(k)}+1$$

is used instead of $J(z)$.

Now there is an optimization problem for finding the cluster centers that minimizes the above objective function. At each depth, the NP method fixes the center of each cluster one at a time. Figure 5.1 shows simple example of clustering using NP methodology. This is nominal problem with two dimensions. The total number of cluster is 2.

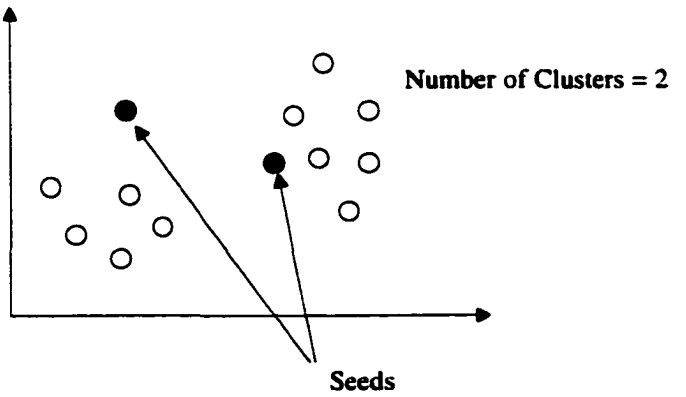
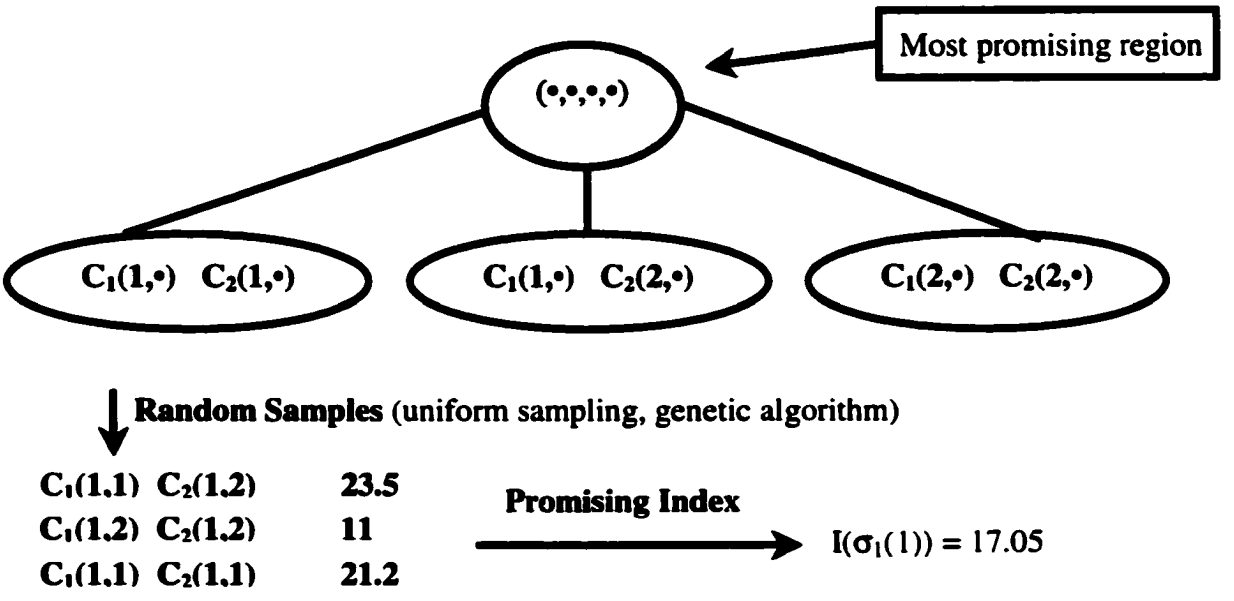



Figure 5.1 Simple Example for Clustering using NP methodology

Each cluster is represented by centers. We need to find out the optimal coordinate that minimizes similarity. To simplify the problem, it is assumed each dimension has only two values. That is, $x_i = \{1,2\}$, $i = 1,2$. Therefore, there can be three subsets using partitioning at the first iteration. In the first subset, 1st dimension of each cluster are set as (1,1). In the second subset, 1st dimension of each cluster are set as (1,2). In the third subset, 1st dimension of each cluster are set as (2,2).

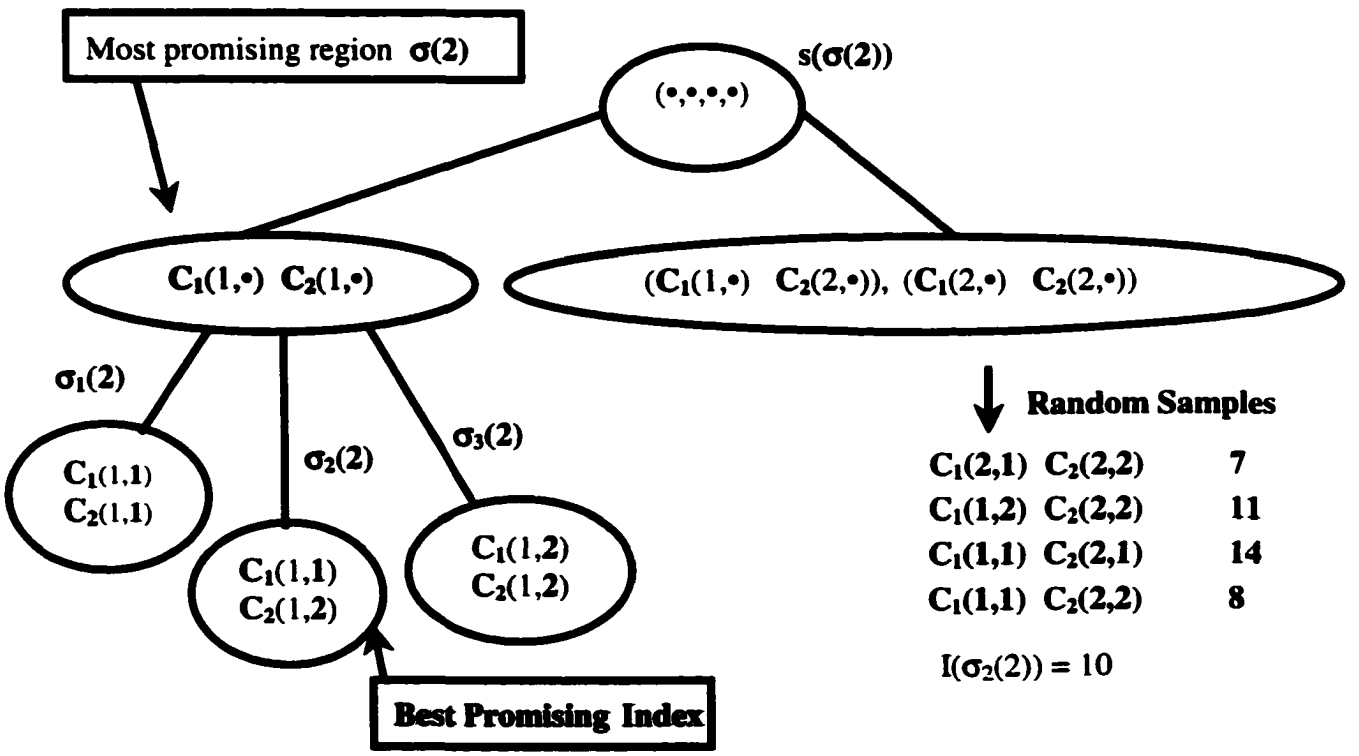
First Iteration




After partitioning three subsets, two-stage sampling is applied. For example, for the first subset, every sample point of this subset has a fixed first dimension; the first cluster and the second cluster are fixed as 1. For the remaining dimension, centers can be randomly assigned from the values {1,2}. Using the sampling from each subset, a similarity value is calculated. Based on these values, the promising index is calculated for each subset. The most promising

region is the first subset because it has the smallest promising index. The partitioning of the second iteration starts from the first subset. Because the 2nd dimension can take 2 different values, three different subsets can be obtained like in the first iteration. Also, the 2nd iteration is the maximum depth because there are two dimensions in this problem. From the 2nd iteration, there is one more subset which is called the surrounding region. After sampling from all subsets, the most promising index is found in the second region, having the 1st cluster's coordinate (1,1) and the 2nd cluster's coordinate (1,2). These coordinates are optimal that minimize the similarity of this problem.

Second Iteration




The following algorithms describe the combined NP method with the Nelson-Matejcik's procedure and the K-means algorithm.

**Algorithm NP/NM/K-means with Inheritance**

Step 1. *Initialization*

Set $k=0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0$, $j=1,2,\ldots,M_{\sigma(k)}$

Specify the constants $\varepsilon$, $\alpha$, $n_k$ and $n_0$. Let $g=T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer *et al.* (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

If $d(\sigma(k))\neq d^*$, that is, $\sigma(k)\neq\sum_0$, then partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ subregions $\sigma_1(k),\ldots,\sigma_{M_{\sigma(k)}}(k)$. If $d(\sigma(k))\neq 0$, that is, $\sigma(k)\neq\Theta$, aggregate the surrounding region $\Theta\setminus\sigma(k)$ into one region $\sigma_{M_{\sigma(k)+1}}(k)$.

Stage I Sampling

Step 3. *First-Stage Sampling*

Step 3-1. Set $t=0$.

Step 3-2. *K-means Algorithm*

Step 3-2-1. $h=1$.

Step 3-2-2. *Randomly Assign Instances to the Clusters*

Use random sampling to obtain $N$ instances and assign them to the center of $INH_{k-1}=[z^{j1}, z^{j2}, \ldots, z^{jN}]$, $j=\hat{j}_{k-1}$ if it matches the current feature values of the subregion. Otherwise,

randomly choose the centers for each of the regions $\sigma_j(k),\ \ j = 1,2,\ldots, M_{\sigma(k)} + 1.$

Step 3-2-3. *Calculate the Squared Error Criterion Function*

Calculate squared error criterion function

$$\hat{L}_h(\sigma_j(k)) = \sum_{i=1}^{N_c} \sum_{x \in I_j^h} | x - z_j^h |,\ \ i = 1,\ldots, N_c,\ \ j = 1,2,\ldots, M_{\sigma(k)} + 1.$$

If $\hat{L}_h(\sigma_j(k)) \ < \ \hat{L}_{h-1}(\sigma_j(k))$, then let $X_{ij}(k) = \hat{L}_h(\sigma_j(k))$

Step 3-2-4. If $h = n_k$, continue to Step 3-3. Otherwise, let $h = h + 1$ and go back to Step 3-2-2.

Step 3-2-4. *Change the Center of each Subregion*

Change the center feature values > $d(\sigma(k))$ for each cluster of each subregion and back to Step 3-2-2.

Step 3-3. If $t = n_0$, continue to Step 4. Otherwise, let $u = u + 1$ and go back to Step 3-2-2.

Stage II Sampling

Step 4. *Estimating Mean and Variance of First-Stage Sampling*

Compute the approximate sample variance for the difference of the sample means

$$S^2 = \frac{2\sum_{j=1}^{k} \sum_{t=1}^{n_0} (X_{ij} - \bar{X}_{t\cdot} - \bar{X}_{\cdot j} + \bar{X}_{\cdot\cdot})^2}{(k-1)(n_0 - 1)}.$$

Where $\bar{X}_{i\cdot} = \sum_{j=1}^{k} X_{ij}/k,\ \ \bar{X}_{\cdot j} = \sum_{u=1}^{n_0} X_{ij}/n_0,$ and $\bar{X}_{\cdot\cdot} = \sum_{t=1}^{n_0} \sum_{j=1}^{k} X_{ij}/kn_0$

Step 5. *Computing the Total Sample Size for Second-Stage Sampling*

Compute the total sample size for all $j$

Compute the total sample size $N(k) = \max\left\{n_0, \left\lceil \left(\frac{gS}{\varepsilon}\right)^2 \right\rceil\right\}$.

for $j = 1, 2, \ldots, M_{\sigma(k)} + 1$

Step 6. *Second-Stage Sampling*

Obtain $N(k) - n_0$ more sample estimates of the system performance for all $j$ as in Step 3 above.

Step 7. *Estimating the Mean of Second-Stage Sampling*

Let the overall sample mean be the promising index for all $j \in I$,

$$\hat{I}(\sigma_j(k)) = \bar{X}_j(k) = \frac{\sum_{i=1}^{N(k)} X_{ij}(k)}{N_j(k)}$$

Step 8. *Calculating the Promising Index*

Calculate the index of the region with the smallest squared error criterion function (most promising region);

$$\hat{j}_k \in \arg\ \min \hat{I}(\sigma_j) \quad \text{for all } j \in I.$$

If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a subregion of $\sigma(k)$, then let this subregion be the most promising region of the next iteration.

That is, let $\sigma(k+1) = \sigma_j(k)$, $j < M_{\sigma(k)}$ and keep the centers of this promising index

$$INH_k = [z^{j1}, z^{j2}, \ldots, z^{jNOIN}], \quad j = \hat{j}_k.$$

Otherwise, if the index corresponds to the surrounding region, backtrack to the region, $s(\sigma(k))$ of the current most promising region. That is, let $\sigma(k+1) = s(\sigma(k))$.

Step 9. *Checking the Stopping Rule*

If $\sigma(k+1) \in \Sigma_0$, stop and let $\sigma_{opt} = \sigma(k+1)$, else $k = k+1$ and go back to Step 2.

The following algorithms are the combined NP method with the Nelson-Matejcik's procedure and Genetic algorithms.

**Algorithm NP/NM/Genetic with Inheritance**

Step 1. *Initialization*

Set $k = 0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0$, $j = 1, 2, \ldots, M_{\sigma(k)}$.

Specify the constants $\varepsilon$, $\alpha$, and $n_0$. Let $g = T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer *et al.* (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

If $d(\sigma(k)) \neq d^*$, that is, $\sigma(k) \neq \sum_0$, partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ subregions $\sigma_1(k),\ldots,\sigma_{M_{\sigma(k)}}(k)$. If $d(\sigma(k)) \neq 0$, that is, $\sigma(k) \neq \Theta$, aggregate the surrounding region $\Theta \setminus \sigma(k)$ into one region $\sigma_{M_{\sigma(k)+1}}(k)$.

Step 3. *Initial Population*

If $k = 0$ and $d(\sigma(k)) \neq d^*$, use random sampling to obtain an initial center population of $N$ strings from each of the regions $\sigma_j(k)$, $j = 1,2,...,M_{\sigma(k)}+1$,

$$POP_I^j = [z_I^{j1}, z_I^{j2}, \ldots, z_I^{jN}], \quad j = 1,2, \ldots, M_{\sigma(k)}+1$$

else use the population $INH_{k-1} = [z^{j1}, z^{j2}, \ldots, z^{jN}]$, $j = \hat{j}_k$ as the initial population. Part of the lacks should be fulfilled using uniform sampling.

Stage I Sampling

Step 4. *First-Stage Sampling*

Step 4-1. Set $h = 1$.

Step 4-2. *GA Search*

Apply the GA to each initial population $POP_I^j$ individually, obtaining a final population for each region $\sigma_j(k)$, $j = 1,2,...,M_{\sigma(k)}+1$

$$POP_F^j = [z_F^{j1}, z_F^{j2}, \ldots, z_F^{jN}], \quad j = 1,2, \ldots, M_{\sigma(k)}+1.$$

Step 4-3. *Calculate the Squared Error Criterion Function (Overall Fitness)*

Randomly assign instances to the best final populations to calculate the squared error criterion function

$$\hat{L}_h(\sigma_j(k)) = \sum_{i=1}^{N_c} \sum_{x \in I_j^h} | x - z_j^{jBest} |, \quad i = 1,\ldots,N_c, \quad j = 1,2,\ldots,M_{\sigma(k)}+1.$$

If $\hat{L}_h(\sigma_j(k)) < \hat{L}_{h-1}(\sigma_j(k))$, then let $X_{ij}(k) = \hat{L}_h(\sigma_j(k))$

Step 4-4. If $h = n_0$, continue to Step 5. Otherwise, let $h = h + 1$ and go back to Step 4-2.

Stage II Sampling

Step 5. *Estimating Mean and Variance of First-Stage Sampling*

See Step 4 in Algorithm NP/NM/K-means with Inheritance

Step 6. *Computing the Total Sample Size for Second-Stage Sampling*

See Step 5 in Algorithm NP/NM/K-means with Inheritance

Step 7. *Second-Stage Sampling*

Obtain $N_j(k)-n_0$ more sample estimates of the system performance for all $j$ as in Step 4 above.

Step 8. *Estimating the Mean of Second-Stage Sampling*

See Step 7 in Algorithm NP/NM/K-means with Inheritance

Step 9. *Calculating the Promising Index*

See Step 8 in Algorithm NP/NM/K-means with Inheritance

Step10.*Checking Stopping Rule*

See Step 9 in Algorithm NP/NM/K-means with Inheritance

The following algorithms are the combined NP method with the Nelson-Matejcik's procedure, Genetic algorithm and K-means algorithm.

**Algorithm NP/NM/K-means/Genetic with Inheritance**

Step 1. *Initialization*

Set $k=0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0$, $j=1,2,\ldots,M_{\sigma(k)}$

Specify the constants $\varepsilon$, $\alpha$, $n_k$, and $n_0$. Let $g=T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can

be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer *et al.* (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

See Step 2 in Algorithm NP/NM/Genetic with Inheritance

Step 3. *Initial Population*

See Step 3 in Algorithm NP/NM/Genetic with Inheritance

Step 4.*GA Search*

Apply the GA to each initial population $POP_I^j$ individually, obtaining a final population for each region $\sigma_j(k)$, $j = 1,2,...,M_{\sigma(k)}+1$

$$POP_F^j = [z_F^{j1}, z_F^{j2}, \ldots, z_F^{jN}], \quad j = 1,2, \ldots, M_{\sigma(k)}+1$$

Step 5. *First-Stage Sampling*

Step 5-1. Set $t = 0$.

Step 5-2. *K-means Algorithm*

See Step 3-2 in Algorithm NP/NM/K-means with Inheritance

Step 5-3. See Step 3-3 in Algorithm NP/NM/K-means with Inheritance

Stage II Sampling

Step 6. *Estimating the Mean and Variance of First-Stage Sampling*

See Step 4 in Algorithm NP/NM/K-means with Inheritance

Step 7. *Computing the Total Sample Size for Second-Stage Sampling*

See Step 5 in Algorithm NP/NM/K-means with Inheritance

Step 8. *Second-Stage Sampling*

See Step 6 in Algorithm NP/NM/K-means with Inheritance

Step 9. *Estimating the Mean of Second-Stage Sampling*

See Step 7 in Algorithm NP/NM/K-means with Inheritance

Step 10.*Calculating the Promising Index*

See Step 8 in Algorithm NP/NM/K-means with Inheritance

Step 11.*Checking the Stopping Rule*

See Step 9 in Algorithm NP/NM/K-means with Inheritance

## 5.4 Numerical Evaluation

To numerically evaluate the performance of these algorithms, two different sizes (large and small) of cancer data are considered (Blake and Merz, 1998). The large data set has 9 features and 699 instances; whereas, the small data set has 9 features and 286 instances. In this section, by varying the number of instances, the aim is to (i) show that NP can use a random sample of instances without sacrificing solution quality and (ii) determine appropriate guidelines for how many instances are needed. Comparison results between different amounts of sampling and between different amounts of inheritance sample points are shown. Numerical design is varied three different ways. First, the number of instances that are considered in every replication are varied: 3, 5, 10, 30, 50, 100, 200, 350, and 699 sample points for the large data set; 1, 2, 4, 12, 20, 41, 82, 143, and 286 sample points for the small data set (0.5, 0.7, 1.5, 4.5, 7, 15, 28, 50, and 100% of data set, respectively). Second, the number of inherited sample points is varied: 0, 3, 6, 10, and 20 sample points for the large data set; 0, 1, 2, 4, and 8 sample points for the small data set (0, 0.5, 0.85, 1.5, and 3% of the size of data, respectively). Finally, the sampling method is varied by using two different methods: Rinott and Nelson- Matejcik.

In general, to determine which method is better than the other, the most straightforward method is using the difference in performance for both methods. If the difference value is greater than 0, then the later is better than the former (in the case of the minimization problem). The same method is used in this section for comparing different parameter values. However, sometimes there is an ambiguous situation with determining which method is better. The *t*-test is then used to resolve these ambiguous situations as usual. If the confidence interval does contain zero, it's hard to say which method is better, so the *t*-test is used to determine if performance differences are significant. Following is the brief summary of the *t*-test based on the difference value.

For $i = 1,\ldots,5$, let $X_{i1}, X_{i2}, \ldots, X_{in}$ be a sample of $n$ IID observations from system $i$ and $\mu_i = E(X_{ij})$: Interval $\delta = \mu_i - \mu_j$ is constructed for $i \neq j$. The $X_{1j}$ with $X_{2j}$ is paired to define $Z_j = X_{1j} - X_{2j}$, for $j = 1,2,\ldots,n$. Then $Z_j$'s are IID random variables and $E(Z_j) = \delta$, the quantity which is to be constructed for a confidence interval. Let

$$\overline{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}$$

and

$$Var(\overline{Z}(n)) = \frac{\sum_{j=1}^{n} [Z_j - \overline{Z}(n)]^2}{n(n-1)}$$

and form the $100(1-\alpha)$ percent interval

$$\overline{Z}(n) \pm t_{n-1,1-\alpha/2} \sqrt{Var[\overline{Z}(n)]}.$$

If the confidence interval does not contain zero, then one method is significantly better than the other $\alpha$ -level.

The NP method is attractive for clustering because it handles noisy performance very well, which makes it possible to use a small sample rather than all of the instances. The number of instances to be considered in each sampling is very important in terms of computation time. If the same or better results can be obtained even if small numbers of instances are used, this is the most desirable result. Because of this reason, a guideline for choosing the number of instances is needed. In this section, this guideline is suggested based on the different sizes of the problems. The results are shown by varying the number of instances: 3, 5, 10, 30, 50, 100, 200, 350, and 699 sample points for the large data set; 1, 2, 4, 12, 20, 41, 82, 143, and 286 sample points for the small data set (0.5, 0.7, 1.5, 4.5, 7, 15, 28, 50, and 100% of the data set, respectively). To clearly state the results, a $t$-test is used based on the difference of the performance measures.

### 5.4.1 Nelson-Matejcik Sampling

Figure 5.2, Figure 5.3, Table 5.1, and Table 5.2, Table 5.3, and Table 5.4 show the results of a large data set when Nelson-Matejcik sampling method is used. Here $\mu(i)$ is defined as the expected similarity value of each system and $\tau(i)$ is defined as the expected computation time when the number of instances are $i \in \{3, 30, 200, 350, 699\}$.

In NP/NM/K-means algorithm with no inheritance, the difference value of 3 instances with 30 instances (see the confidence interval of $\mu(3) - \mu(30)$ in Table 5.3) included zero in the confidence intervals, which means it cannot be verified if the similarity value of

considering 3 instances is higher than 30 instances until 200 instances is reached (see the confidence interval of $\mu(3)-\mu(200)$ in Table 5.3). Intuitively, computation time should decrease as fewer instances are used to calculate the performance. For example, when looking at the NP/NM/K-means algorithm in Table 5.1, the similarity value and computation time are 4,259 and 394,780 each when all instances are used and 4,208 and 101,670 each when 50% of instances are used. Thus, the computation time is cut almost 75% with no reduction in quality. However, the results show that 30 instances required the least time, not 3 instances (see the confidence interval of $\tau(3)-\tau(30)$ and $\tau(200)-\tau(30)$ in Table 5.4). This can be explained by considering the backtracking step in the nested partitions method. Too small number of instances decreased the probability of correctly choosing the most promising region. In the nested partitions method, a backtracking step is used to correct this kind of error and repeated backtracks increase computation time.

For the NP/NM/K-means with inheritance method, it should be noticed that using 350 instances, the best similarity value is obtained and there is no difference in similarity value even if more than 350 instances are used (see the confidence interval of $\mu(350)-\mu(699)$ in Table 5.3). However, 30 instances still require the smallest computation time (see the confidence interval of $\tau(3)-\tau(30)$ and $\tau(200)-\tau(30)$ in Table 5.4).

NP/NM/Genetic algorithm results show that to get the minimum similarity value at least 350 instances are needed and there is no difference in the similarity value even if 350 instances are used (see the confidence interval of $\mu(350)-\mu(699)$ in Table 5.3). Until 200 instances there is no difference in computation time (see the confidence interval of $\tau(200)-\tau(30)$ in Table 5.4). However, above 200 instances, computation time increased

when inheritance is not used. When inheritance is used, less sample instances are needed with 200 instances, to get a minimum similarity value than without inheritance and there is no difference greater than 200 instances (see the confidence interval of $\tau(200) - \tau(350)$ in Table 5.4). Computation time is minimal between $30 \sim 200$ instances.

NP/NM/Genetic/K-means algorithm results show that to get the minimum similarity value at least 200 instances are needed with no difference above 200 instances (see the confidence interval of $\tau(200) - \tau(350)$ in Table 5.4). Runs using between 30~350 instances require the least computation time when inheritance is not used. When inheritance is used, there is no difference in computation time until 200 instances. However above 200 instances, computation time increased (see the confidence interval of $\tau(350) - \tau(200)$ and $\tau(200) - \tau(30)$ in Table 5.4).

Table 5.1 and Table 5.2 show that the mean and the variance of accuracy, computation speed, and the amount of backtracking for a large data set when the Nelson-Matejcik method is used. When using half of the instances instead of using all instances, the computation time and variance are decreased without changing the solution quality. Also, when using inheritance, better accuracy, better speed and lesser amounts of backtrackings are obtained compared to when not using inheritance. This is noticeable in the case of NP/NM/K-means algorithm.

Table 5.1: Effect of Using Fraction of Instance Space for the Large Data Set Without Inheritance when the Nelson-Matejcik Sampling Method is Used

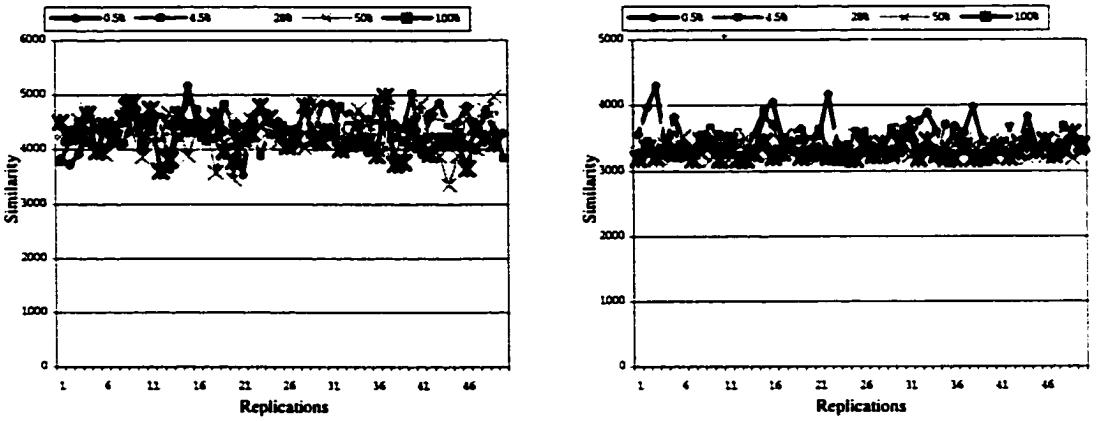
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NM/K-means | 100% | 4259.0 | 46 | 394777 | 6668 | 0.14 | 0.049 |
| | 50% | 4207.7 | 53 | 101666 | 1352 | 0.08 | 0.039 |
| | 28% | 4264.7 | 51 | 43794 | 498 | 0.08 | 0.039 |
| | 4.5% | 4363.4 | 41 | 38966 | 590 | 0.10 | 0.052 |
| | 0.5% | 4401.1 | 49 | 44065 | 775 | 0.14 | 0.057 |
| NP/NM/Genetic | 100% | 4203.7 | 40 | 404534 | 23118 | 0.82 | 0.168 |
| | 50% | 4198.7 | 41 | 152181 | 7237 | 1.14 | 0.187 |
| | 28% | 4369.5 | 43 | 120173 | 5521 | 1.02 | 0.182 |
| | 4.5% | 4362.2 | 44 | 129202 | 7907 | 1.24 | 0.243 |
| | 0.5% | 4606.7 | 51 | 115850 | 5352 | 0.80 | 0.164 |
| NP/NM/Genetic/K-means | 100% | 4444.2 | 39 | 231200 | 10487 | 0.90 | 0.149 |
| | 50% | 4418.7 | 37 | 84920 | 5610 | 0.78 | 0.243 |
| | 28% | 4401.2 | 38 | 80074 | 527 | 1.02 | 0.175 |
| | 4.5% | 4652.4 | 39 | 80971 | 4968 | 1.02 | 0.247 |
| | 0.5% | 4661.6 | 49 | 63127 | 1185 | 0.16 | 0.060 |

Table 5.2: Effect of Using Fraction of Instance Space for the Large Data Set with Inheritance when the Nelson-Matejcik Sampling Method is Used

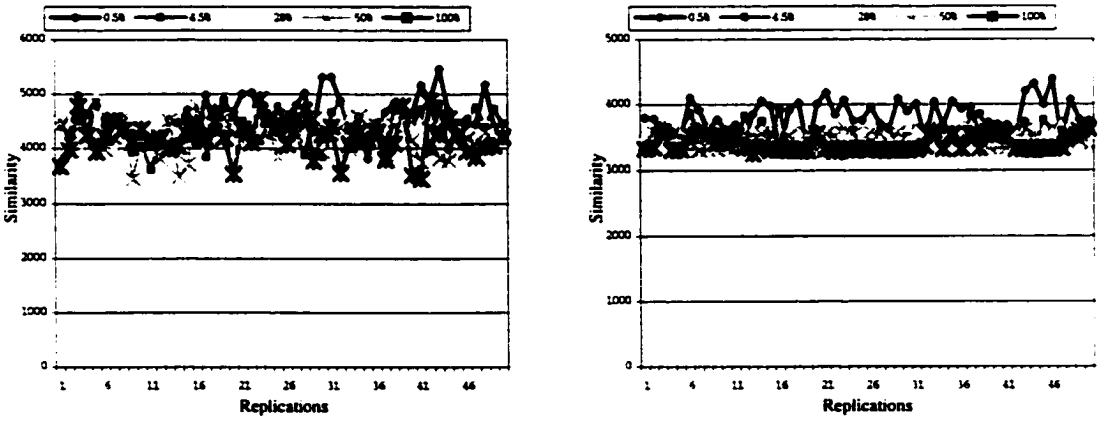
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NM/K-means | 100% | 3292.1 | 16 | 373743 | 2692 | 0.00 | 0.00 |
| | 50% | 3304.1 | 17 | 97891 | 631 | 0.00 | 0.00 |
| | 28% | 3412.1 | 29 | 42493 | 179 | 0.00 | 0.00 |
| | 4.5% | 3423.1 | 23 | 37898 | 40 | 0.00 | 0.00 |
| | 0.5% | 3369.5 | 34 | 41804 | 57 | 0.00 | 0.00 |
| NP/NM/Genetic | 100% | 3388.3 | 17 | 186965 | 6782 | 0.08 | 0.039 |
| | 50% | 3436.1 | 18 | 71249 | 1698 | 0.02 | 0.020 |
| | 28% | 3453.4 | 21 | 58647 | 1699 | 0.10 | 0.059 |
| | 4.5% | 3567.3 | 23 | 573332 | 1324 | 0.00 | 0.028 |
| | 0.5% | 3819.7 | 35 | 62028 | 1196 | 0.02 | 0.020 |
| NP/NM/Genetic/K-means | 100% | 3811.9 | 33 | 101676 | 6001 | 0.08 | 0.039 |
| | 50% | 3809.3 | 33 | 32070 | 1160 | 0.14 | 0.057 |
| | 28% | 3841.9 | 28 | 25364 | 891 | 0.08 | 0.039 |
| | 4.5% | 4033.3 | 38 | 24791 | 1061 | 0.30 | 0.010 |
| | 0.5% | 4394.6 | 44 | 23973 | 978 | 0.30 | 0.082 |

Table 5.3: Similarity Results of $t$-test with 95% Confidence Interval of the Large Data Set when the Nelson-Matejcik Sampling Method is Used

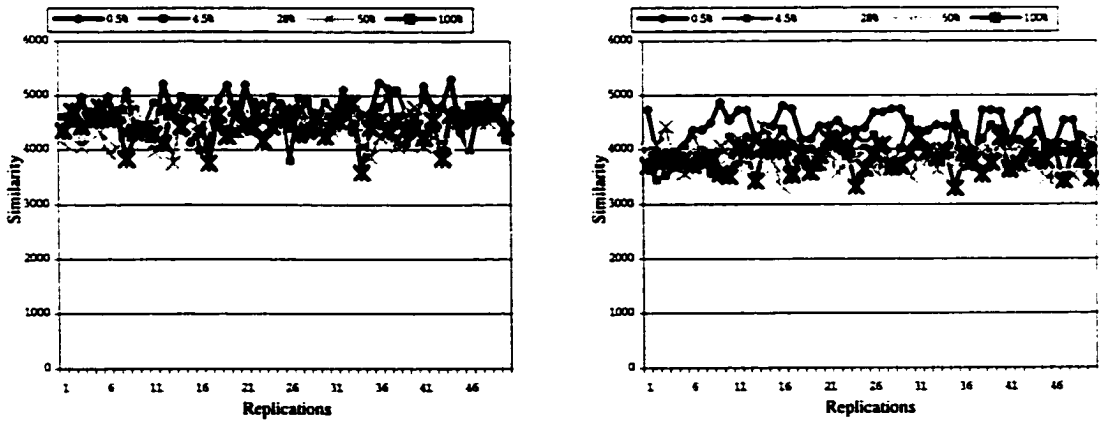
| Algorithm | Inheritance | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | No | $\mu(3)-\mu(30)$ | 37.7 | 3256.32 | [-76.9, 152.3] |
| | | **$\mu(3)-\mu(200)$** | **136.3** | **4353.61** | **[3.7, 268.8]** |
| | | $\mu(200)-\mu(350)$ | 57.1 | 5899.13 | [-97.2, 211.3] |
| | | $\mu(200)-\mu(699)$ | 5.8 | 5412.31 | [-142.0, 153.5] |
| | Yes | $\mu(3)-\mu(30)$ | 146.4 | 1534.71 | [67.7, 225.1] |
| | | $\mu(30)-\mu(200)$ | 11.0 | 1318.71 | [-61.9, 83.9] |
| | | **$\mu(30)-\mu(350)$** | **118.9** | **922.07** | **[57.9, 179.9]** |
| | | $\mu(350)-\mu(699)$ | 12.0 | 647.78 | [-39.1, 63.1] |
| NP/NM/Genetic | No | $\mu(3)-\mu(30)$ | 244.5 | 3753.09 | [121.4, 367.5] |
| | | $\mu(30)-\mu(200)$ | -7.2 | 3714.45 | [-129.6, 115.2] |
| | | **$\mu(30)-\mu(350)$** | **163.6** | **3626.09** | **[42.5, 284.5]** |
| | | $\mu(350)-\mu(699)$ | -5.0 | 2872.49 | [-112.6, 102.6] |
| | Yes | $\mu(3)-\mu(30)$ | 252.4 | 2162.61 | [158.9, 345.8] |
| | | **$\mu(30)-\mu(200)$** | **113.9** | **995.72** | **[50.5, 177.3]** |
| | | $\mu(200)-\mu(350)$ | 17.3 | 1030.99 | [-47.1, 81.8] |
| | | $\mu(200)-\mu(699)$ | 65.1 | 819.11 | [7.5, 122.5] |
| NP/NM/Genetic/K-means | No | $\mu(3)-\mu(30)$ | 9.3 | 3477.89 | [-109.2, 127.7] |
| | | **$\mu(3)-\mu(200)$** | **260.4** | **4572.93** | **[124.5, 396.2]** |
| | | $\mu(200)-\mu(350)$ | -17.5 | 2324.49 | [-114.3, 79.3] |
| | | $\mu(200)-\mu(699)$ | -42.9 | 2997.95 | [-152.9, 67.0] |
| | Yes | $\mu(3)-\mu(30)$ | 361.2 | 2769.88 | [255.4, 466.9] |
| | | **$\mu(30)-\mu(200)$** | **191.4** | **2280.21** | **[95.4, 287.2]** |
| | | $\mu(200)-\mu(350)$ | 32.7 | 1595.84 | [-47.5, 112.9] |
| | | $\mu(200)-\mu(699)$ | 30.1 | 2287.95 | [-66.0, 126.1] |




NP/NM/K-means algorithm




NP/NM/Genetic algorithm

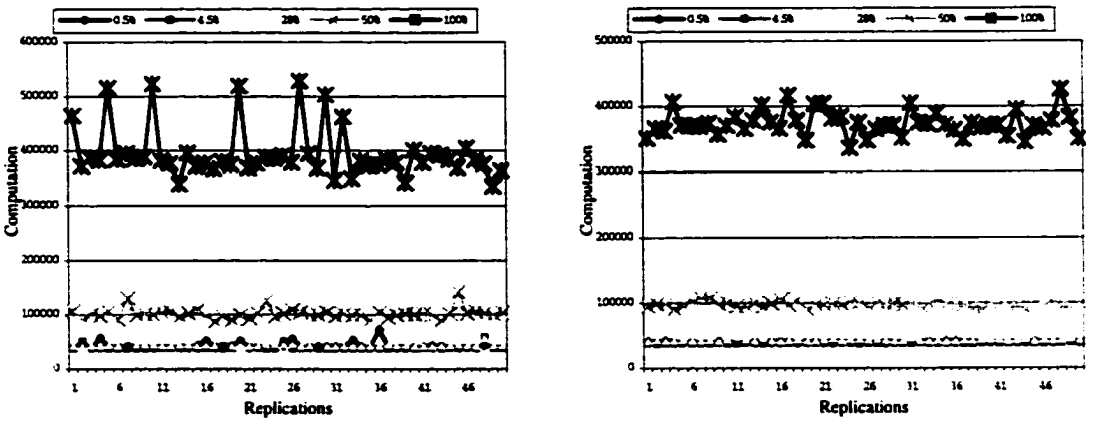



NP/NM/Genetic/K-means algorithm

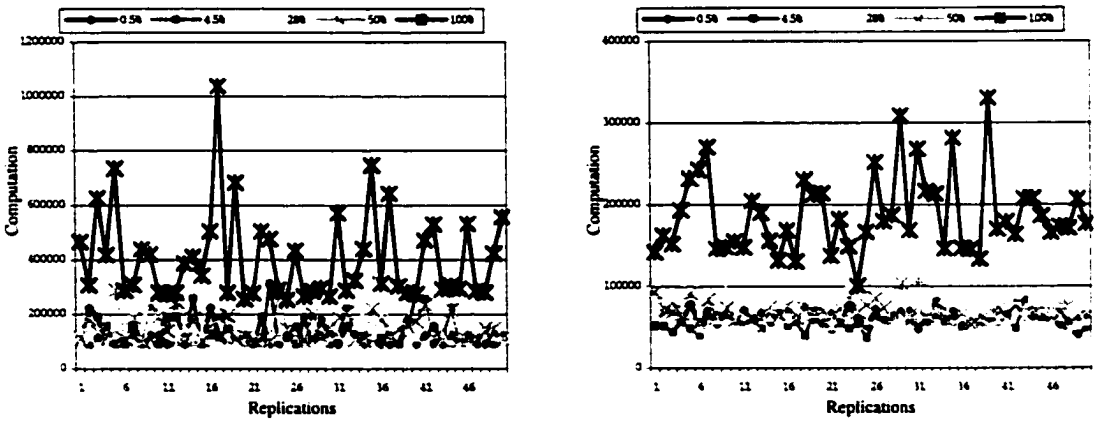
Figure 5.2: Similarity Value of Each Algorithm with No Inheritance (Left) and Inheritance (Right) for the Large Data Set when the Nelson-Matejcik Sampling Method is Used

Table 5.4: Computation Time Results of $t$-test with 95% Confidence Interval for the Large Data Set when the Nelson-Matejcik Sampling Method is Used

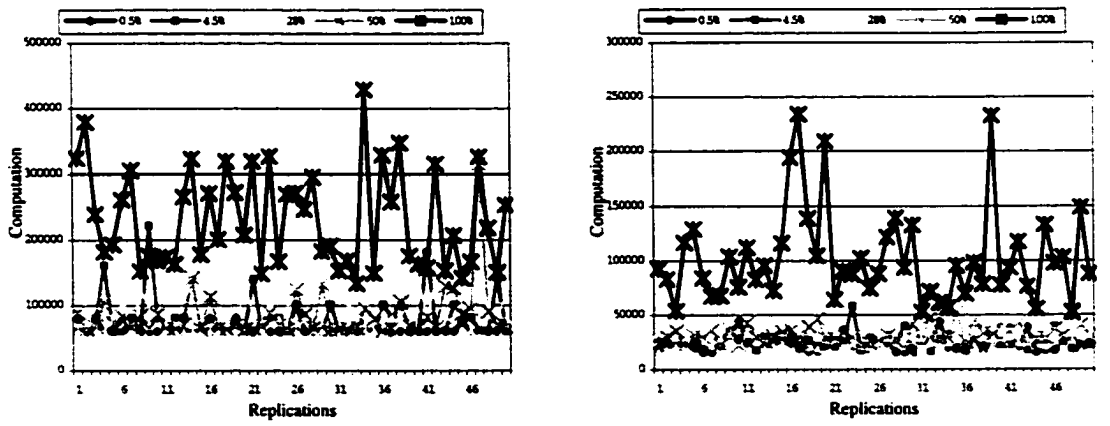
| Algorithm | Inheritance | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | No | $\tau(699)-\tau(350)$ | 293112.0 | 44656110.50 | [279685.8, 306536.3] |
| | | $\tau(350)-\tau(200)$ | 57871.5 | 1821877.94 | [55159.8, 60583.2] |
| | | $\tau(350)-\tau(30)$ | 4828.8 | 4287784.60 | [3514.8, 6142.8] |
| | | **$\tau(3)-\tau(30)$** | **5099.8** | **934601.70** | **[3157.6, 7042.0]** |
| | Yes | $\tau(699)-\tau(350)$ | 275852.3 | 7916703.57 | [270199.7, 281505.0] |
| | | $\tau(350)-\tau(200)$ | 55397.8 | 469235.84 | [54021.6, 56773.9] |
| | | $\tau(200)-\tau(30)$ | 4595.6 | 30965.96 | [4242.1, 4949.1] |
| | | **$\tau(3)-\tau(30)$** | **3906.1** | **4461.07** | **[3771.9, 4040.2]** |
| NP/NM/Genetic | No | $\tau(699)-\tau(350)$ | 252352.8 | 557756722.50 | [204906.5, 299799.1] |
| | | $\tau(350)-\tau(200)$ | 32007.6 | 95879826.60 | [12335.8, 51679.3] |
| | | **$\tau(200)-\tau(30)$** | **-9028.8** | **104906984.50** | **[-29605.8, 11548.2]** |
| | | $\tau(200)-\tau(3)$ | 4411.4 | 73052166.33 | [-12759.6, 21582.4] |
| | Yes | $\tau(699)-\tau(350)$ | 115716.1 | 42115329.06 | [102678.4, 128753.7] |
| | | **$\tau(350)-\tau(200)$** | **12602.5** | **4109306.46** | **[8529.9, 16675.0]** |
| | | $\tau(200)-\tau(30)$ | 1314.5 | 4796274.00 | [-3085.3, 5714.2] |
| | | $\tau(3)-\tau(30)$ | 3620.3 | 3193699.00 | [30.0, 7210.5] |
| NP/NM/Genetic/K-means | No | **$\tau(699)-\tau(350)$** | **146280.1** | **118284295.70** | **[124430.5, 168129.7]** |
| | | $\tau(350)-\tau(200)$ | 4846.1 | 53839269.07 | [-9894.9, 19587.2] |
| | | $\tau(350)-\tau(30)$ | 3949.1 | 57173370.00 | [-11241.5, 19139.7] |
| | | $\tau(350)-\tau(3)$ | 21793.1 | 31479374.57 | [10521.3, 33064.8] |
| | Yes | $\tau(699)-\tau(350)$ | 69605.9 | 36691463.38 | [57436.7, 81775.2] |
| | | **$\tau(350)-\tau(200)$** | **6706.5** | **2153790.90** | **[3758.0, 9654.8]** |
| | | $\tau(200)-\tau(30)$ | 572.6 | 1695801.00 | [-2043.5, 3188.7] |
| | | $\tau(200)-\tau(3)$ | 1390.6 | 1666363.74 | [-1202.7, 3983.9] |




NP/NM/K-means algorithm




NP/NM/Genetic algorithm




NP/NM/Genetic/K-means algorithm

Figure 5.3: Computation Time of Each Algorithm with No Inheritance (Left) and Inheritance (Right) for the Large Data Set when the Nelson-Matejcik Sampling Method is Used

Figure 5.4, Figure 5.5, Table 5.5, Table 5.6, Table 5.7, and Table 5.8 show the results of a small data set when Nelson-Matejcik sampling method is used. $\mu(i)$ is defined as the expected similarity value of each system and $\tau(i)$ is defined as the expected computation time when the number of instances are $i \in \{1, 12, 82, 143, 286\}$.

The NP/NM/K-means algorithm results show that 143 instances give the best similarity value when inheritance is not used. Computation time is minimized between 12~143 instances. When inheritance is used, fewer sample instances, 82, are needed to get a minimum similarity value than without inheritance. Computation time is minimized between 82 ~ 143 instances are used.

The NP/NM/Genetic algorithm results show that 143 instances give the best similarity value and more than 143 instances make no improvement in similarity value (see the confidence interval of $\mu(143) - \mu(286)$ in Table 5.7) when inheritance is not used. Furthermore, there is no difference in computation time. When inheritance is used, 82 instances gave the best similarity value and computation time (see the confidence interval of $\mu(12) - \mu(82)$ in Table 5.3 and $\tau(12) - \tau(82)$ in Table 5.8).

The NP/NM/Genetic/K-means algorithm results show that whether the inheritance is used or not, 143 instances gave the best similarity value and no difference above 143 instances (see the confidence interval of $\mu(143) - \mu(286)$ in Table 5.7). Also, there is no difference in computation time.

Table 5.5 and Table 5.6 show the mean and the variance of accuracy, computation speed, and the amount of backtrackings of the small data set when Nelson-Matejcik method is used. The results are similar with the large data set.

Table 5.5: Effect of Using Fraction of Instance Space for the Small Data Set without Inheritance when the Nelson-Matejcik Sampling Method is Used

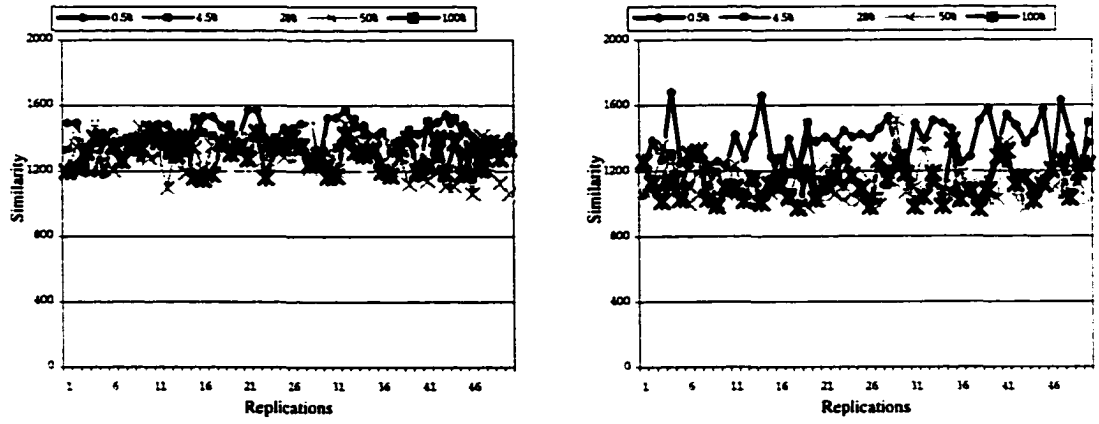
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NM/K-means | 100% | 1302.3 | 13 | 84115 | 3166 | 0.72 | 0.103 |
| | 50% | 1276.6 | 15 | 27295 | 901 | 0.30 | 0.071 |
| | 28% | 1322.9 | 12 | 25699 | 689 | 0.56 | 0.186 |
| | 4.5% | 1363.1 | 13 | 27698 | 960 | 0.44 | 0.122 |
| | 0.5% | 1430.9 | 12 | 33773 | 1203 | 0.34 | 0.079 |
| NP/NM/Genetic | 100% | 1317.1 | 12 | 128158 | 8684 | 2.32 | 0.292 |
| | 50% | 1277.7 | 11 | 134849 | 10827 | 2.32 | 0.324 |
| | 28% | 1367.6 | 11 | 116872 | 9181 | 2.06 | 0.331 |
| | 4.5% | 1416.1 | 11 | 130608 | 10622 | 2.34 | 0.312 |
| | 0.5% | 1452.2 | 12 | 114601 | 10317 | 1.82 | 0.314 |
| NP/NM/Genetic/K-means | 100% | 1267.8 | 11 | 72739 | 4528 | 1.48 | 0.234 |
| | 50% | 1285.9 | 9 | 70752 | 4701 | 2.22 | 0.264 |
| | 28% | 1363.0 | 10 | 70175 | 4589 | 2.16 | 0.316 |
| | 4.5% | 1421.0 | 9 | 59104 | 3929 | 1.30 | 0.241 |
| | 0.5% | 1516.4 | 13 | 54164 | 3241 | 1.34 | 0.199 |

Table 5.6: Effect of Using Fraction of Instance Space for the Small Data Set with Inheritance when the Nelson-Matejcik Sampling Method is Used

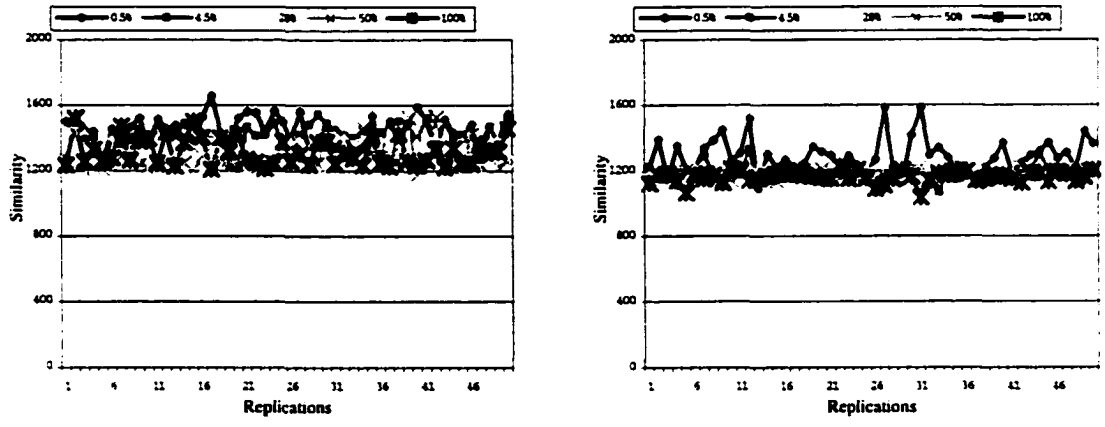
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NM/K-means | 100% | 1128.0 | 16 | 69111 | 960 | 0.00 | 0.000 |
| | 50% | 1114.3 | 16 | 27382 | 635 | 0.06 | 0.044 |
| | 28% | 1102.1 | 12 | 26227 | 359 | 0.00 | 0.000 |
| | 4.5% | 1147.7 | 16 | 29928 | 502 | 0.04 | 0.040 |
| | 0.5% | 1385.0 | 18 | 37405 | 943 | 0.10 | 0.059 |
| NP/NM/Genetic | 100% | 1160.3 | 5 | 31606 | 753 | 0.04 | 0.028 |
| | 50% | 1172.1 | 4 | 30929 | 594 | 0.00 | 0.000 |
| | 28% | 1168.9 | 5 | 31391 | 748 | 0.04 | 0.028 |
| | 4.5% | 1187.8 | 7 | 34923 | 1148 | 0.08 | 0.039 |
| | 0.5% | 1292.4 | 15 | 40317 | 1472 | 0.10 | 0.052 |
| NP/NM/Genetic/K-means | 100% | 1125.6 | 8 | 20025 | 467 | 0.00 | 0.000 |
| | 50% | 1147.1 | 7 | 21995 | 879 | 0.02 | 0.020 |
| | 28% | 1173.0 | 7 | 20588 | 825 | 0.10 | 0.043 |
| | 4.5% | 1243.3 | 10 | 18753 | 598 | 0.10 | 0.043 |
| | 0.5% | 1393.6 | 15 | 18571 | 1294 | 0.34 | 0.113 |

Table 5.7: Similarity Results of *t*-test with 95% Confidence Interval for the Small Data Set when the Nelson-Matejcik Sampling Method is Used

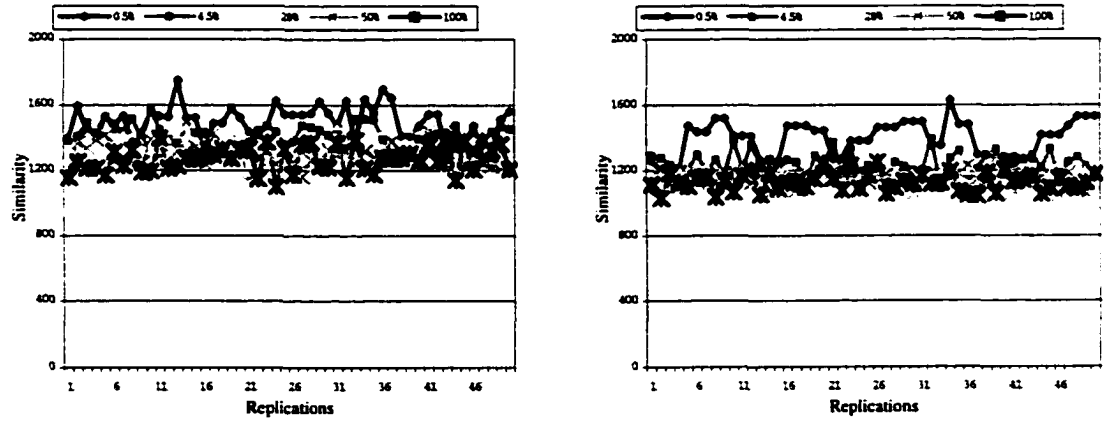
| Algorithm | Inheritance | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | No | $\mu(1)-\mu(12)$ | 67.8 | 284.48 | [33.9, 101.6] |
| | | $\mu(12)-\mu(82)$ | 40.2 | 376.44 | [1.2, 79.2] |
| | | **$\mu(82)-\mu(143)$** | **46.2** | **426.05** | **[4.7, 87.7]** |
| | | $\mu(143)-\mu(286)$ | -25.7 | 367.20 | [-64.2, 12.8] |
| | Yes | $\mu(1)-\mu(12)$ | 237.3 | 663.48 | [185.5, 289.0] |
| | | **$\mu(12)-\mu(82)$** | **42.5** | **375.10** | **[3.6, 81.5]** |
| | | $\mu(82)-\mu(143)$ | 33.4 | 523.02 | [-12.5, 79.3] |
| | | $\mu(82)-\mu(286)$ | 19.7 | 384.69 | [-19.6, 59.1] |
| NP/NM/Genetic | No | $\mu(1)-\mu(12)$ | 36.1 | 3753.09 | [6.6, 65.6] |
| | | $\mu(12)-\mu(82)$ | 48.4 | 282.71 | [14.6, 82.2] |
| | | **$\mu(82)-\mu(143)$** | **89.9** | **225.68** | **[59.7, 120.1]** |
| | | $\mu(143)-\mu(286)$ | -39.3 | 252.14 | [-71.2, -7.4] |
| | Yes | $\mu(1)-\mu(12)$ | 104.5 | 237.12 | [73.6, 135.4] |
| | | **$\mu(12)-\mu(82)$** | **18.9** | **61.80** | **[3.1, 34.7]** |
| | | $\mu(82)-\mu(143)$ | -3.2 | 44.06 | [-16.5, 10.0] |
| | | $\mu(82)-\mu(286)$ | 8.5 | 51.62 | [-5.8, 23.0] |
| NP/NM/Genetic/K-means | No | $\mu(1)-\mu(12)$ | 74.8 | 315.34 | [39.1, 110.5] |
| | | $\mu(12)-\mu(82)$ | 58.0 | 137.10 | [34.5, 81.5] |
| | | **$\mu(82)-\mu(143)$** | **77.0** | **200.04** | **[48.6, 105.4]** |
| | | $\mu(143)-\mu(286)$ | 18.1 | 260.58 | [-14.3, 50.5] |
| | Yes | $\mu(1)-\mu(12)$ | 150.3 | 371.94 | [111.5, 189.0] |
| | | $\mu(12)-\mu(82)$ | 70.2 | 163.79 | [44.5, 95.9] |
| | | **$\mu(82)-\mu(143)$** | **25.9** | **97.04** | **[6.1, 45.7]** |
| | | $\mu(143)-\mu(286)$ | 21.5 | 125.03 | [-0.9, 44.0] |




NP/NM/K-means algorithm




NP/NM/Genetic algorithm

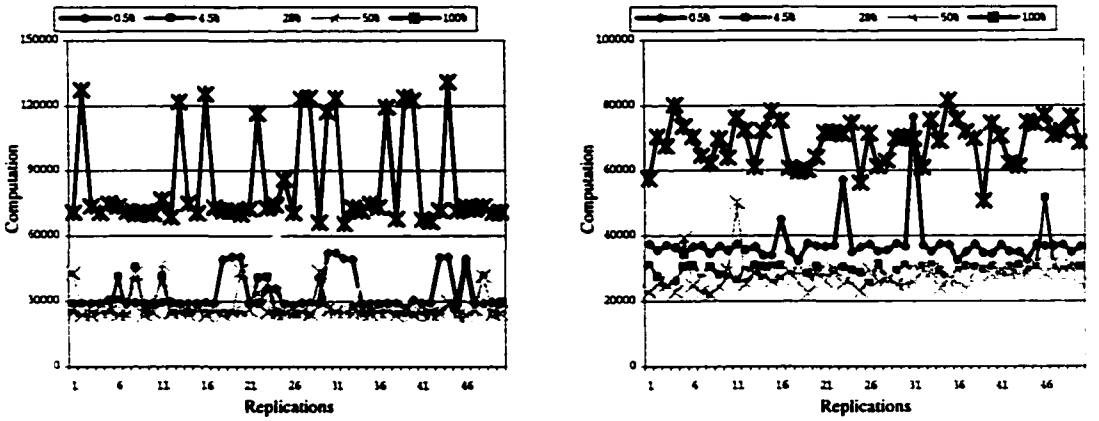



NP/NM/Genetic/K-means algorithm

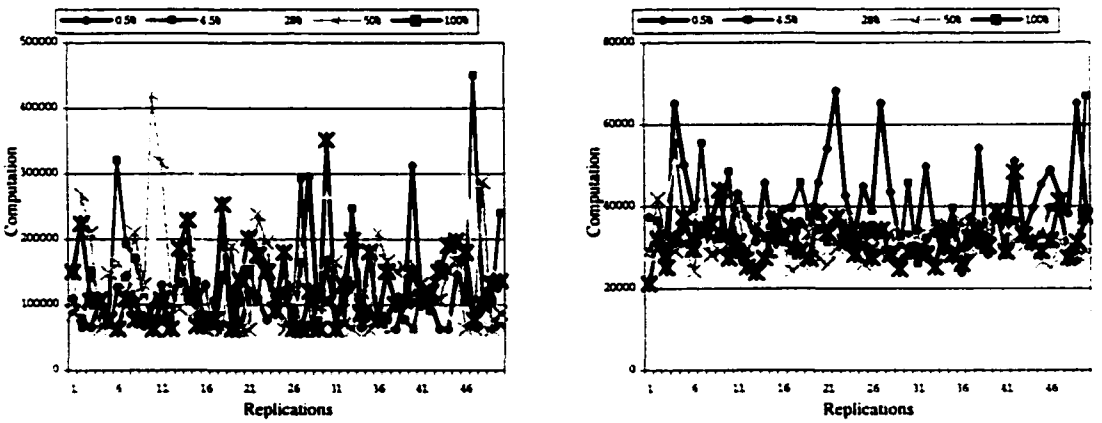
Figure 5.4: Similarity Value of Each Algorithm with No Inheritance (Left) and Inheritance (Right) for the Small Data Set when the Nelson-Matejcik Sampling Method is Used

Table 5.8: Computation Time Results of $t$-test with 95% Confidence Interval for the Small Size Data set when the Nelson-Matejcik Sampling Method is Used

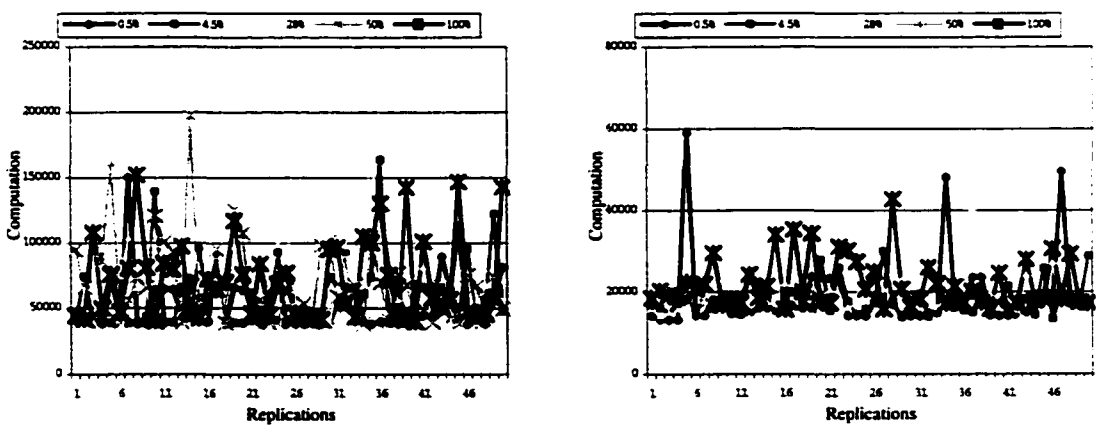
| Algorithm | Inheritance | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | No | $r(286)-r(143)$ | 56851.7 | 1154940.00 | [50025.6, 63677.9] |
| | | $r(143)-r(82)$ | 724.2 | 18883956.66 | [-2033.2, 3481.7] |
| | | $r(143)-r(12)$ | -435.1 | 843224.58 | [-2279.9, 1409.6] |
| | | $r(1)-r(12)$ | 6510.2 | 2305501.98 | [3459.8, 9560.7] |
| | Yes | $r(286)-r(143)$ | 41728.8 | 1006545.95 | [39713.2, 43744.4] |
| | | $r(143)-r(82)$ | 1120.8 | 651458.48 | [-500.6, 2742.4] |
| | | $r(12)-r(82)$ | 2546.1 | 650712.63 | [925.5, 4166.7] |
| | | $r(1)-r(12)$ | 7477.0 | 1138584.00 | [5333.3, 9260.7] |
| NP/NM/Genetic | No | $r(286)-r(143)$ | -6691.6 | 214701206.90 | [-36128.8, 22745.6] |
| | | $r(286)-r(82)$ | 11285.2 | 149730595.50 | [-13297.8, 35868.2] |
| | | $r(286)-r(12)$ | -2450.0 | 189676967.00 | [-30118.6, 25218.6] |
| | | $r(286)-r(1)$ | 13556.4 | 160702337.20 | [-11911.4, 39024.1] |
| | Yes | $r(286)-r(143)$ | 676.5 | 883792.14 | [-1212.1, 2565.1] |
| | | $r(286)-r(82)$ | 215.0 | 1197948.19 | [-1983.7, 2413.9] |
| | | $r(12)-r(82)$ | 4010.7 | 1669548.11 | [1414.9, 6606.6] |
| | | $r(1)-r(12)$ | 5394.5 | 3386494.00 | [1697.4, 9091.5] |
| NP/NM/Genetic/K-means | No | $r(286)-r(143)$ | 1986.9 | 42365775.27 | [-11089.4, 15063.3] |
| | | $r(286)-r(82)$ | 2564.6 | 38017073.37 | [-9822.4, 14951.7] |
| | | $r(286)-r(12)$ | 11783.6 | 41472002.69 | [-11917.3, 13020.4] |
| | | $r(286)-r(1)$ | 5123.0 | 27790075.30 | [-5467.6, 15713.7] |
| | Yes | $r(286)-r(143)$ | -1969.3 | 1209276.43 | [-4178.6, 239.8] |
| | | $r(286)-r(82)$ | -562.7 | 884566.63 | [-2452.2, 1326.7] |
| | | $r(286)-r(12)$ | 1272.1 | 475199.55 | [-112.7, 2657.0] |
| | | $r(286)-r(1)$ | 1454.4 | 1972903.40 | [-1367.4, 4276.2] |




NP/NM/K-means algorithm




NP/NM/Genetic algorithm




NP/NM/Genetic/K-means algorithm

Figure 5.5: Computation Time of Each Algorithm with No Inheritance (Left) and Inheritance (Right) for the Small Data Set when the Nelson-Matejcik Sampling Method is Used

### 5.4.2 Rinott's Sampling

Figure 5.6, Figure 5.7, Table 5.9, Table 5.10, Table 5.11, and Table 5.12 show the results of the large data set when Rinott's sampling method is used. As in the Nelson-Matejcik's sampling method, $\mu(i)$ is defined as the expected similarity value for each system and $\tau(i)$ is defined as the expected computation time when the number of instances are $i \in \{3, 30, 200, 350, 699\}$.

In the NP/NSR/K-means algorithm with no inheritance, there is no difference in the similarity values. Runs using between 30 ~ 200 instances required the least computation time. When using inheritance, the best similarity value is found to be only 30 instances with no difference above 30 instances (see the confidence interval of $\mu(30)-\mu(200)$ in Table 5.12). Runs using 200 instances required the least computation time (see the confidence interval of $\tau(30)-\tau(200)$ in Table 5.12).

The NP/NSR/Genetic algorithm results show that to get the minimum similarity value, at least 350 instances are required but the difference in similarity value cannot be found even if more than 350 instances are used (see the confidence interval of $\mu(350)-\mu(699)$ in Table 5.11) when inheritance is not used. When using inheritance, fewer sample instances, 200, are required to get a minimum similarity value than without inheritance and more than 200 instances give no difference in similarity value (see the confidence interval of $\mu(200)-\mu(350)$ in Table 5.11). Up to 200 instances there are no difference in computation time (see the confidence interval of $\tau(200)-\tau(30)$ in Table 5.12), but above 200 instances, computation time increased.

The NP/NSR/Genetic/K-means algorithm results show that to get the minimum similarity value, at least 200 instances are required and there is no difference in similarity value even if more than 200 instances are used (see the confidence interval of $\mu(200) - \mu(350)$ in Table 5.11). Finally, there is no difference in computation time up to 350 instances (see the confidence interval of $\tau(699) - \tau(350)$ in Table 5.12), however more than 350 instances make computation time increase.

Table 5.9 and Table 5.10 show the mean and variance of accuracy, computation speed and the amount of backtrackings of the large data set when Rinott's sampling method is used. When only half of the instances is used, the computation time and variance are decreased by a quarter without solution quality. Also, when using inheritance, relatively better accuracy, better speed, and fewer amount of backtrackings are needed than without inheritance. This is noticeable for the NP/NM/K-means algorithm.

Figure 5.8, Figure 5.9, Table 5.15, and Table 5.16 show the results of a small data set when the Rinott's sampling method is used. As in the Nelson-Matejcik's sampling method, $\mu(i)$ is defined as the expected similarity value of each system and $\tau(i)$ is defined as the expected computation time when the number of instances are $i \in \{1, 12, 82, 143, 286\}$.

The NP/NSR/K-means algorithm results show almost same results as in the Nelson-Matejcik's sampling method. The best similarity value is obtained when 143 instances are used and there is no difference in similarity value even if and more than 143 instances are used (see the confidence interval of $\mu(143) - \mu(286)$ in Table 5.15) when inheritance is not used. Computation time is minimized between 12~143 instances. When using inheritance, fewer sample instances, 82 instances, are required to get a minimum similarity value than

Table 5.9: Effect of Using Fraction of Instance Space for the Large Data Set without Inheritance when the Rinott's Sampling Method is Used

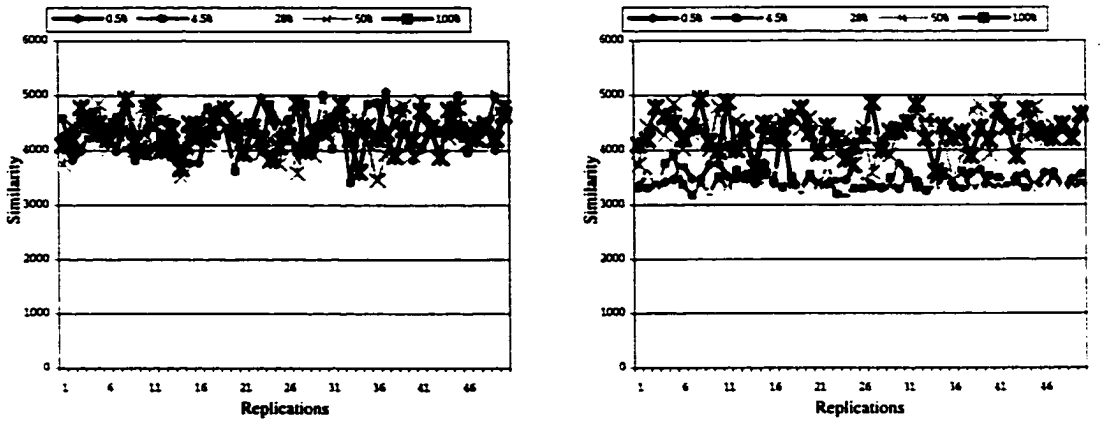
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NSR/K-means | 100% | 4327.7 | 44 | 114530 | 2590 | 0.14 | 0.057 |
| | 50% | 4301.7 | 50 | 42178 | 547 | 0.10 | 0.043 |
| | 28% | 4295.7 | 48 | 39490 | 659 | 0.26 | 0.069 |
| | 4.5% | 4376.8 | 55 | 39791 | 647 | 0.18 | 0.062 |
| | 0.5% | 4354.9 | 42 | 42316 | 213 | 0.02 | 0.020 |
| NP/NSR/Genetic | 100% | 4196.6 | 49 | 522436 | 33318 | 1.06 | 0.193 |
| | 50% | 4218.1 | 44 | 179204 | 10221 | 0.94 | 0.190 |
| | 28% | 4434.9 | 44 | 114039 | 4661 | 0.68 | 0.147 |
| | 4.5% | 4432.4 | 42 | 122786 | 6256 | 10.2 | 0.193 |
| | 0.5% | 4667.5 | 31 | 125134 | 5729 | 1.12 | 0.017 |
| NP/NSR/Genetic/K-means | 100% | 4403.3 | 40 | 108895 | 70004 | 1.06 | 0.245 |
| | 50% | 4481.8 | 34 | 79793 | 4579 | 0.98 | 0.231 |
| | 28% | 4391.3 | 40 | 72567 | 3298 | 0.64 | 0.171 |
| | 4.5% | 4559.1 | 42 | 73423 | 3187 | 0.66 | 0.180 |
| | 0.5% | 4661.6 | 53 | 71228 | 2663 | 0.56 | 0.131 |

Table 5.10: Effect of Using Fraction of Instance Space for the Large Data Set with Inheritance when the Rinott's Sampling Method is Used

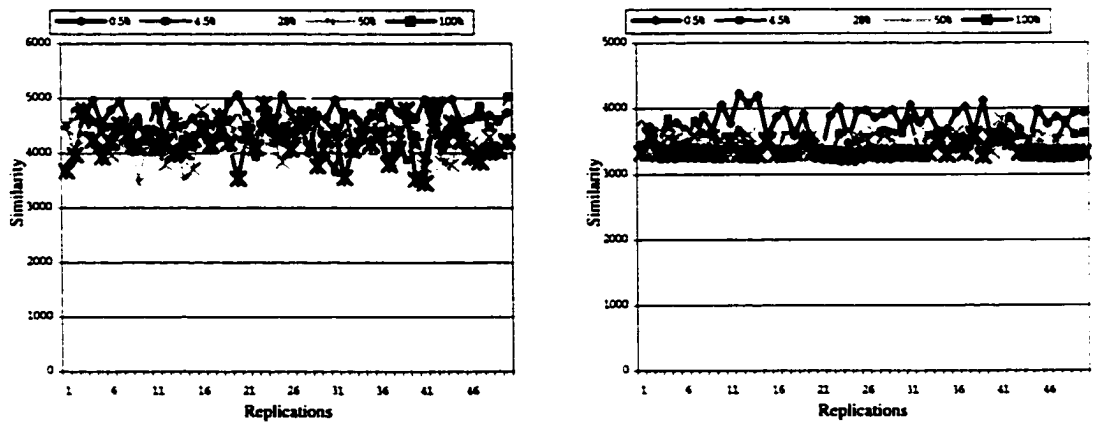
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NSR/K-means | 100% | 3339.3 | 24 | 190673 | 1439 | 0.00 | 0.000 |
| | 50% | 3363.5 | 27 | 56990 | 273 | 0.00 | 0.000 |
| | 28% | 3409.9 | 25 | 37091 | 36 | 0.00 | 0.000 |
| | 4.5% | 3405.7 | 21 | 37890 | 41 | 0.00 | 0.000 |
| | 0.5% | 3493.2 | 28 | 41712 | 52 | 0.00 | 0.000 |
| NP/NSR/Genetic | 100% | 3405.6 | 17 | 212895 | 10154 | 0.02 | 0.020 |
| | 50% | 3418.2 | 20 | 114857 | 2950 | 0.02 | 0.020 |
| | 28% | 3459.9 | 21 | 60296 | 1776 | 0.04 | 0.028 |
| | 4.5% | 3542.9 | 20 | 58701 | 1183 | 0.04 | 0.028 |
| | 0.5% | 3808.0 | 29 | 61792 | 1053 | 0.00 | 0.000 |
| NP/NSR/Genetic/K-means | 100% | 3834.8 | 37 | 51932 | 2159 | 0.14 | 0.057 |
| | 50% | 3879.7 | 34 | 27948 | 938 | 0.16 | 0.059 |
| | 28% | 3836.0 | 35 | 27330 | 999 | 0.10 | 0.051 |
| | 4.5% | 3982.4 | 33 | 27199 | 1297 | 0.36 | 0.106 |
| | 0.5% | 4425.3 | 44 | 25419 | 955 | 0.26 | 0.075 |

Table 5.11: Similarity Results of *t*-test with 95% Confidence Interval for the Large Data Set when the Rinott's Sampling Method is Used

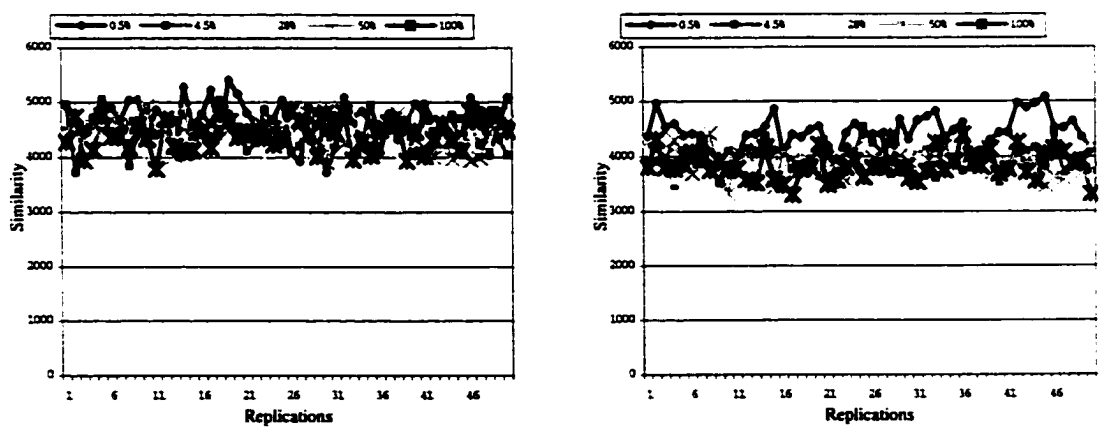
| Algorithm | Inheritance | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NSR/K-means | No | $\mu(3)-\mu(30)$ | -21.9 | 4202.57 | [-152.1, 108.3] |
| | | $\mu(30)-\mu(200)$ | 81.1 | 4658.49 | [-55.9, 218.2] |
| | | $\mu(30)-\mu(350)$ | 75.1 | 4437.55 | [-58.6, 209.0] |
| | | $\mu(30)-\mu(699)$ | 49.1 | 5487.75 | [-99.7, 197.9] |
| | Yes | **$\mu(3)-\mu(30)$** | **87.5** | **1377.90** | **[12.8, 162.0]** |
| | | $\mu(30)-\mu(200)$ | -4.1 | 1173.94 | [-72.9, 64.6] |
| | | $\mu(30)-\mu(350)$ | 42.2 | 1163.15 | [-26.2, 110.7] |
| | | $\mu(30)-\mu(699)$ | 66.4 | 1208.00 | [-3.4, 136.2] |
| NP/NSR/Genetic | No | $\mu(3)-\mu(30)$ | 235.1 | 3053.40 | [124.1, 346.1] |
| | | $\mu(30)-\mu(200)$ | 97.5 | 3444.05 | [-20.4, 215.4] |
| | | **$\mu(30)-\mu(350)$** | **214.3** | **3076.31** | **[102.8, 325.7]** |
| | | $\mu(350)-\mu(699)$ | 21.4 | 3210.72 | [-92.3, 135.3] |
| | Yes | $\mu(3)-\mu(30)$ | 265.0 | 1292.98 | [192.8, 337.2] |
| | | **$\mu(30)-\mu(200)$** | **76.9** | **970.72** | **[14.3, 139.5]** |
| | | $\mu(200)-\mu(350)$ | 41.7 | 857.40 | [-17.1, 100.5] |
| | | $\mu(200)-\mu(699)$ | 54.2 | 779.60 | [-1.8, 110.3] |
| NP/NSR/Genetic/K-means | No | $\mu(3)-\mu(30)$ | 102.4 | 4893.82 | [-38.0, 243.0] |
| | | **$\mu(30)-\mu(200)$** | **167.8** | **4759.23** | **[29.2, 306.4]** |
| | | $\mu(200)-\mu(350)$ | -50.4 | 3084.42 | [-162.0, 61.1] |
| | | $\mu(200)-\mu(699)$ | -12.0 | 2792.92 | [-118.1, 94.1] |
| | Yes | $\mu(3)-\mu(30)$ | 442.9 | 2963.47 | [333.5, 552.2] |
| | | **$\mu(30)-\mu(200)$** | **146.4** | **2631.59** | **[43.3, 249.4]** |
| | | $\mu(200)-\mu(350)$ | -43.6 | 2516.07 | [-144.4, 57.1] |
| | | $\mu(200)-\mu(699)$ | 1.2 | 2955.02 | [-107.9, 110.4] |




NP/NSR/K-means algorithm




NP/NSR/Genetic algorithm

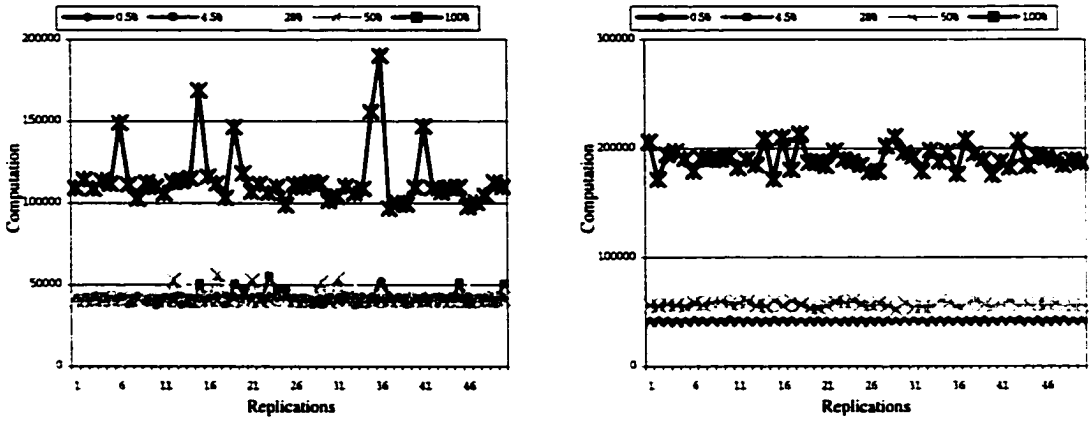



NP/NSR/Genetic/K-means algorithm

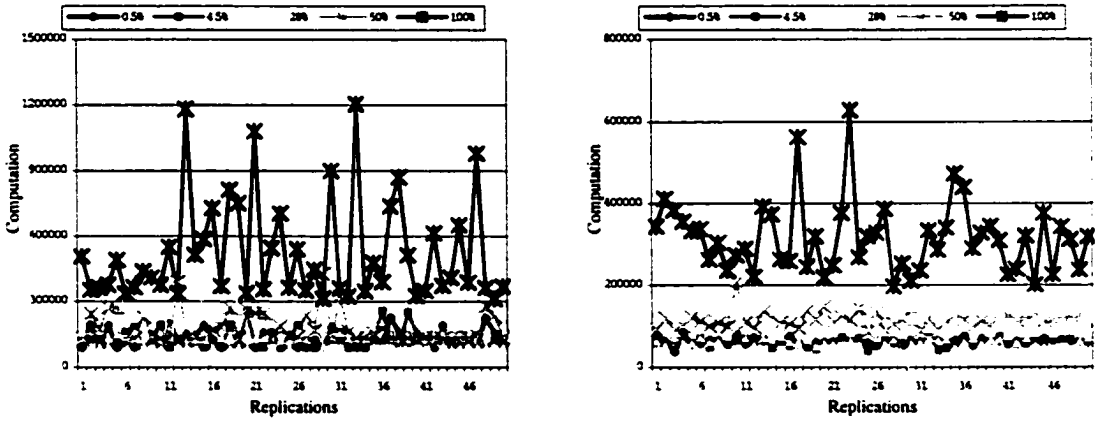
Figure 5.6: Similarity Value of Each Algorithm with No inheritance (Left) and Inheritance Right) for the Large Data set when the Rinott's Sampling Method is Used

Table 5.12: Computation Time Results of $t$-test with 95% Confidence Interval for the Large Data Set when the Rinott's Sampling Method is Used

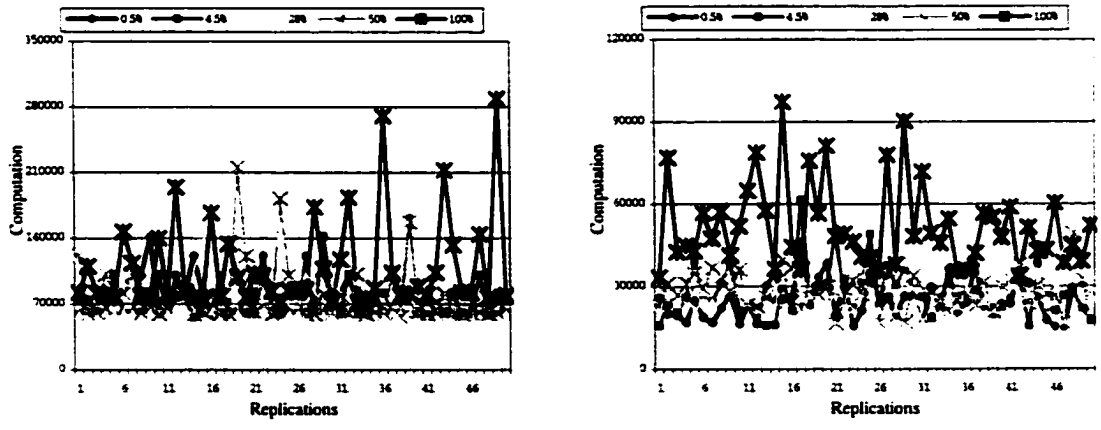
| Algorithm | Inheritance | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NSR/K-means | No | $\tau(699)-\tau(350)$ | 72351.0 | 7185759.00 | [66965.6, 77736.4] |
| | | $\tau(350)-\tau(200)$ | 2688.9 | 796050.37 | [896.4, 4481.3] |
| | | $\tau(200)-\tau(30)$ | -30.1 | 1049028.03 | [-2359.3, 1755.9] |
| | | **$\tau(3)-\tau(30)$** | **2826.3** | **5081177.60** | **[1394.1, 4258.4]** |
| | Yes | $\tau(699)-\tau(350)$ | 133682.8 | 2289860.30 | [130642.7, 136722.8] |
| | | $\tau(350)-\tau(200)$ | 19899.0 | 74702.71 | [19349.9, 20448.1] |
| | | **$\tau(30)-\tau(200)$** | **799.0** | **3505.92** | **[680.0, 917.9]** |
| | | $\tau(3)-\tau(30)$ | 3821.56.0 | 5468.40 | [3672.9, 3970.1] |
| NP/NSR/Genetic | No | $\tau(699)-\tau(350)$ | 343232.0 | 1383888532.00 | [268495, 417968.0] |
| | | $\tau(350)-\tau(200)$ | 65165.0 | 138973863.00 | [41482.1, 88849.1] |
| | | **$\tau(200)-\tau(30)$** | **-8747.2** | **67415142.78** | **[-25242.4, 7748.0]** |
| | | $\tau(200)-\tau(3)$ | -12077.1 | 50620185.00 | [-26370.7, 2216.4] |
| | Yes | $\tau(699)-\tau(350)$ | 200948.2 | 16835724.00 | [174878.8, 227017.8] |
| | | **$\tau(350)-\tau(200)$** | **52955.7** | **12366315.40** | **[45890.9 60020.5]** |
| | | $\tau(200)-\tau(30)$ | 2635.1 | 3194934.40 | [-955.7, 6226.1] |
| | | $\tau(200)-\tau(3)$ | -144.8 | 3635724.48 | [-3975.5, 3685.8] |
| NP/NSR/Genetic/K-means | No | **$\tau(699)-\tau(350)$** | **29101.5** | **81822810.80** | **[10928.9, 47274.1]** |
| | | $\tau(350)-\tau(200)$ | 7226.5 | 27939739.80 | [-3392.6, 17845.7] |
| | | $\tau(350)-\tau(30)$ | 6370.5 | 28311895.38 | [-4319.1, 17060.2] |
| | | $\tau(350)-\tau(3)$ | 8565.5 | 31626919.60 | [-2732.6, 19863.7] |
| | Yes | **$\tau(699)-\tau(350)$** | **23532.6** | **5184708.85** | **[18958.2, 28107.1]** |
| | | $\tau(350)-\tau(200)$ | 1068.8 | 1583978.80 | [-1459.6, 3597.2] |
| | | $\tau(350)-\tau(30)$ | 1199.7 | 2481275.22 | [-1964.8, 4364.3] |
| | | $\tau(3)-\tau(350)$ | 2529.1 | 1605678.30 | [-16.5, 5074.8] |




NP/NSR/K-means algorithm




NP/NSR/Genetic algorithm




NP/NSR/Genetic/K-means algorithm

Figure 5.7: Computation Time of Each Algorithm with No inheritance (Left) and Inheritance (Right) for the Large Data Set when the Rinott's Sampling Method is Used

without inheritance but there is no need to use more than 82 instances (see the confidence interval of $\mu(82) - \mu(143)$ in Table 5.15). Computation time is minimized between 82~143 instances.

The NP/NSR/Genetic algorithm results show that 143 instances give the best similarity value but more than 143 instances make no difference in similarity value (see the confidence interval of $\mu(143) - \mu(286)$ in Table 5.15) when inheritance is not used. When inheritance is used, 82 instances give the best similarity value and there is no difference in similarity value more than 82 instances are used (see the confidence interval of $\mu(82) - \mu(143)$ in Table 5.15). Computation time is minimized with 82 instances when inheritance is not used (see the confidence interval of $\tau(286) - \tau(82)$ in Table 5.16). However, when using inheritance, fewer sample instances, 12 instances, are required to get a minimum similarity value.

The NP/NSR/Genetic/K-means algorithm results show that whether the inheritance is used or not, 143 instances require the smallest similarity value but the difference in similarity cannot be found more than 143 instances are used (see the confidence interval of $\mu(143) - \mu(286)$ in Table 5.15). There is no difference in computation time.

Table 5.13 and Table 5.14 show that the mean and the variance of accuracy, computation speed and the amount of backtracking of the small data set when Rinott's sampling method is used. The results are similar with large data set.

Table 5.13: Effect of Using Fraction of Instance Space for the Small Data Set without Inheritance when the Rinott's Sampling Method is Used

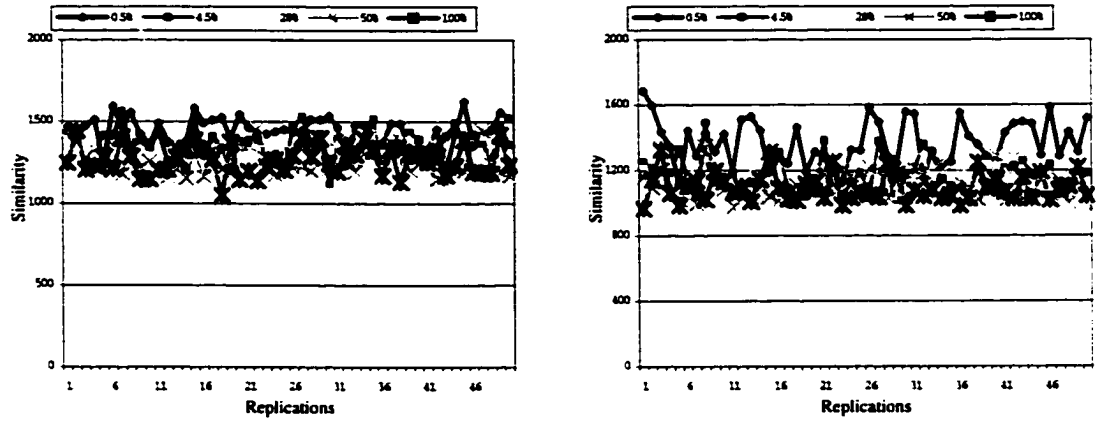
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NSR/K-means | 100% | 1265.2 | 13 | 27134 | 884 | 0.66 | 0.123 |
| | 50% | 1268.0 | 13 | 24862 | 478 | 0.70 | 0.222 |
| | 28% | 1319.9 | 11 | 24472 | 468 | 0.22 | 0.059 |
| | 4.5% | 1361.4 | 13 | 25678 | 647 | 0.12 | 0.054 |
| | 0.5% | 1450.1 | 12 | 33202 | 1537 | 0.32 | 0.109 |
| NP/NSR/Genetic | 100% | 1262.7 | 8 | 121930 | 9268 | 2.24 | 0.295 |
| | 50% | 1161.0 | 8 | 23470 | 6429 | 0.00 | 0.219 |
| | 28% | 1348.8 | 14 | 105136 | 8352 | 1.74 | 0.277 |
| | 4.5% | 1387.5 | 11 | 179993 | 13807 | 3.86 | 0.403 |
| | 0.5% | 1486.4 | 11 | 147007 | 12118 | 3.32 | 0.439 |
| NP/NSR/Genetic/K-means | 100% | 1265.0 | 7 | 67901 | 4425 | 2.00 | 0.285 |
| | 50% | 1259.0 | 10 | 64803 | 4269 | 2.20 | 0.297 |
| | 28% | 1333.2 | 10 | 77988 | 7366 | 2.30 | 0.355 |
| | 4.5% | 1402.8 | 9 | 68307 | 4502 | 2.06 | 0.249 |
| | 0.5% | 1486.2 | 13 | 737002 | 6140 | 2.44 | 0.364 |

Table 5.14: Effect of Using Fraction of Instance Space for the Small Data Set with Inheritance when the Rinott's Sampling Method is Used

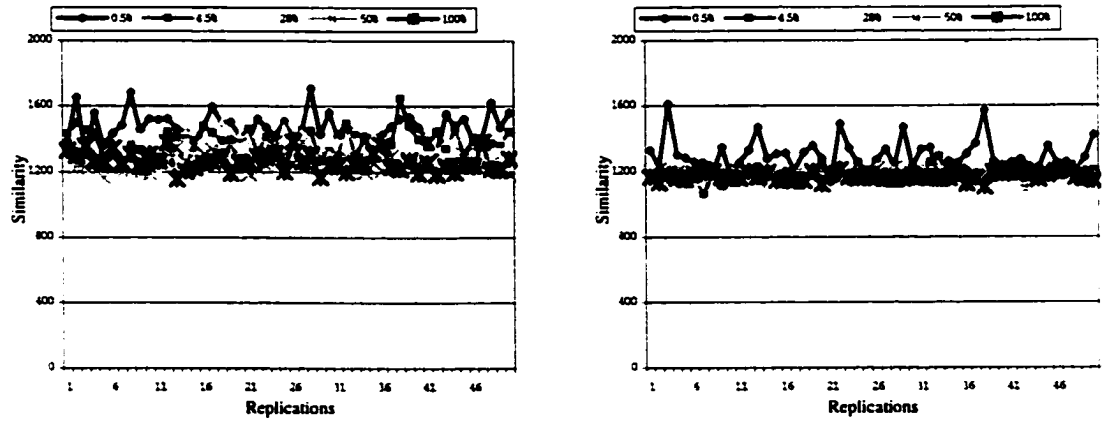
| Algorithm | Fraction | Similarity Value | | Computation Time | | Backtracking | |
|---|---|---|---|---|---|---|---|
| NP/NSR/K-means | 100% | 1093.7 | 12 | 27475 | 433 | 0.02 | 0.020 |
| | 50% | 1106.6 | 13 | 26060 | 522 | 0.04 | 0.040 |
| | 28% | 1124.0 | 13 | 27878 | 738 | 0.10 | 0.052 |
| | 4.5% | 1171.1 | 15 | 29657 | 397 | 0.02 | 0.020 |
| | 0.5% | 1371.9 | 19 | 36823 | 651 | 0.08 | 0.048 |
| NP/NSR/Genetic | 100% | 1168.8 | 4 | 31453 | 984 | 0.06 | 0.034 |
| | 50% | 1174.3 | 4 | 31091 | 707 | 0.04 | 0.028 |
| | 28% | 1166.6 | 5 | 32854 | 1154 | 0.10 | 0.043 |
| | 4.5% | 1194.6 | 5 | 35478 | 1435 | 0.10 | 0.065 |
| | 0.5% | 1298.7 | 14 | 40793 | 1331 | 0.12 | 0.055 |
| NP/NSR/Genetic/K-means | 100% | 1144.2 | 7 | 19722 | 597 | 0.00 | 0 .000 |
| | 50% | 1161.4 | 7 | 20931 | 861 | 0.06 | 0.044 |
| | 28% | 1180.8 | 7 | 19167 | 632 | 0.00 | 0.000 |
| | 4.5% | 1229.9 | 11 | 18986 | 729 | 0.10 | 0.043 |
| | 0.5% | 1391.0 | 21 | 18130 | 615 | 0.12 | 0.046 |

Table 5.15: Similarity Results of *t*-test with 95% Confidence Interval for the Small Data Se when the Rinott's Sampling Method is Used

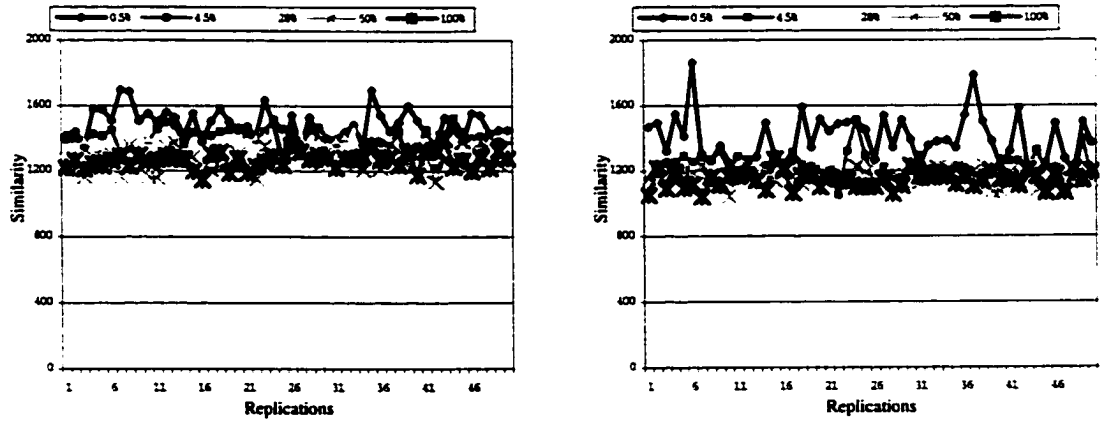
| Algorithm | Inheritance | $\delta$ | $\bar{Z}(n)$ | $Var(\bar{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NSR/K-means | No | $\mu(1)-\mu(12)$ | 82.5 | 289.49 | [48.3, 116.6] |
| | | $\mu(12)-\mu(82)$ | 41.5 | 252.53 | [9.6, 73.4] |
| | | **$\mu(82)-\mu(143)$** | **52.8** | **309.14** | **[16.4, 87.1]** |
| | | $\mu(143)-\mu(286)$ | 2.8 | 351.40 | [-34.8, 40.4] |
| | Yes | $\mu(1)-\mu(12)$ | 200.8 | 524.35 | [154.8, 246.8] |
| | | **$\mu(12)-\mu(82)$** | **47.1** | **226.32** | **[16.9, 77.3]** |
| | | $\mu(82)-\mu(143)$ | 17.3 | 258.18 | [-14.9, 49.6] |
| | | $\mu(82)-\mu(286)$ | 30.2 | 334.55 | [-6.4, 67.0] |
| NP/NSR/Genetic | No | $\mu(1)-\mu(12)$ | 98.8 | 218.71 | [69.1, 128.6] |
| | | $\mu(12)-\mu(82)$ | 38.6 | 357.70 | [0.6, 76.7] |
| | | **$\mu(82)-\mu(143)$** | **89.4** | **299.00** | **[54.6, 124.1]** |
| | | $\mu(143)-\mu(286)$ | -3.3 | 147.79 | [-27.7, 21.0] |
| | Yes | $\mu(1)-\mu(12)$ | 104.1 | 237.94 | [73.1, 135.1] |
| | | **$\mu(12)-\mu(82)$** | **27.9** | **58.63** | **[12.5, 43.3]** |
| | | $\mu(82)-\mu(143)$ | -7.6 | 29.43 | [-18.5, 3.2] |
| | | $\mu(82)-\mu(286)$ | -2.1 | 33.40 | [-13.7, 9.4] |
| NP/NSR/Genetic/K-means | No | $\mu(1)-\mu(12)$ | 83.3 | 256.83 | [51.1, 115.5] |
| | | $\mu(12)-\mu(82)$ | 69.6 | 164.38 | [43.8, 95.3] |
| | | **$\mu(82)-\mu(143)$** | **74.1** | **188.79** | **[46.5, 101.7]** |
| | | $\mu(143)-\mu(286)$ | -6.0 | 105.71 | [-26.6, 14.6] |
| | Yes | $\mu(1)-\mu(12)$ | 161.2 | 560.15 | [113.6, 208.7] |
| | | $\mu(12)-\mu(82)$ | 49.0 | 163.22 | [23.3, 74.7] |
| | | **$\mu(82)-\mu(143)$** | **19.4** | **85.54** | **[0.8, 38.0]** |
| | | $\mu(143)-\mu(286)$ | 17.2 | 96.12 | [-2.5, 36.8] |




NP/NSR/K-means algorithm




NP/NSR/Genetic algorithm

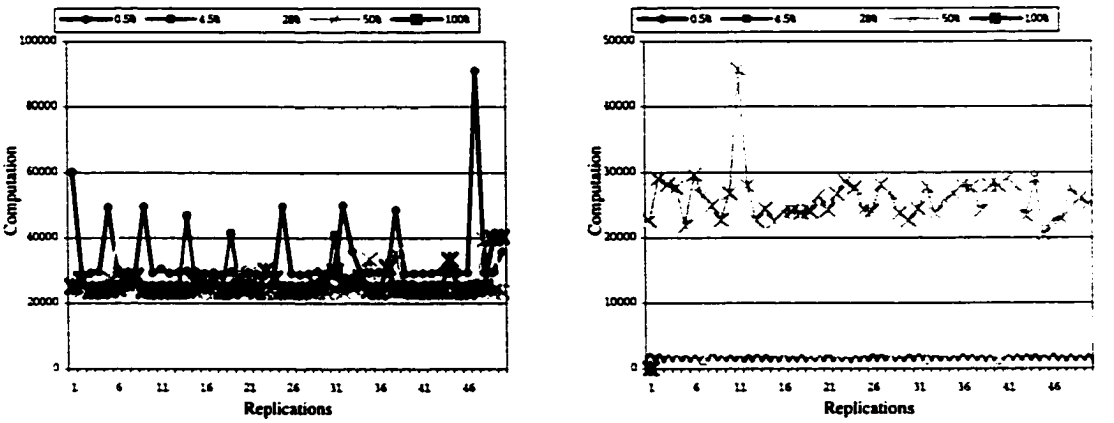



NP/NSR/Genetic/K-means algorithm

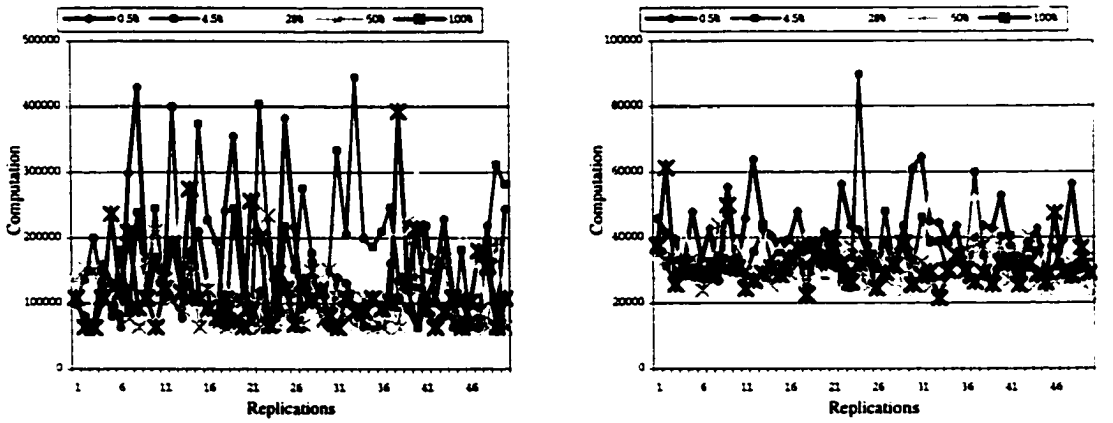
Figure 5.8: Similarity Value of Each Algorithm with No Inheritance (Left) and Inheritance (Right) for the Small Data Set when the Rinott Sampling Method is Used

Table 5.16: Computation Time Results of $t$-test with 95% Confidence Interval for the Small Size Data Set when the Rinott's Sampling Method is Used

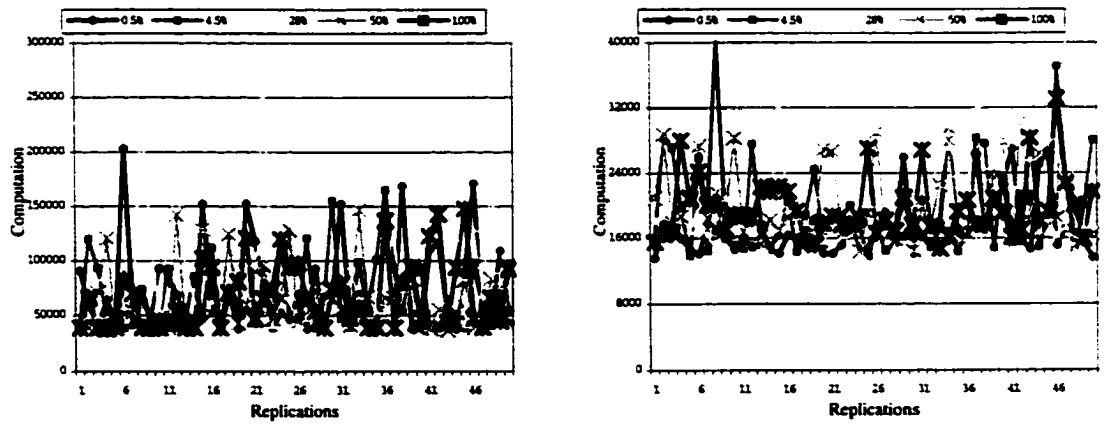
| Algorithm | Inheritance | $\delta$ | $\bar{Z}(n)$ | $Var(\bar{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NSR/K-means | No | **$r(286)-r(143)$** | **2271.7** | **1123707.80** | **[142.078, 4401.36]** |
| | | $r(143)-r(82)$ | 389.8 | 464828.60 | [-979.88, 1759.5] |
| | | $r(143)-r(12)$ | -1206.2 | 616884.40 | [-2784.1, 371.7] |
| | | $r(1)-r(12)$ | 7794.0 | 3005681.98 | [4311.06, 11277] |
| | Yes | **$r(286)-r(143)$** | **1415.4** | **384102.80** | **[170.3, 2660.52]** |
| | | $r(143)-r(82)$ | -1817.7 | 845360.67 | [-3664.86, 29.42] |
| | | $r(12)-r(82)$ | 1779.1 | 719928.10 | [74.5, 3483.7] |
| | | $r(1)-r(12)$ | 7166.0 | 286124.70 | [6091.41, 8240.67] |
| NP/NSR/Genetic | No | $r(286)-r(143)$ | 9417.6 | 155167148.60 | [-15607.7, 34442.4] |
| | | $r(286)-r(82)$ | 16793.6 | 161053385.00 | [-8701.98, 42289.2] |
| | | **$r(12)-r(82)$** | **58063.2** | **332618210.10** | **[21423.41, 94703]** |
| | | $r(1)-r(12)$ | -32986.4 | 280561626.90 | [-66637.1, 664.29] |
| | Yes | $r(286)-r(143)$ | 361.5 | 1387880.96 | [-2005.27, 2728.27] |
| | | $r(286)-r(82)$ | -1401.1 | 2947921.28 | [-4850.5, 2048.2] |
| | | **$r(12)-r(82)$** | **4387.3** | **2654011.30** | **[1114.4, 7660.2]** |
| | | $r(1)-r(12)$ | 5314.2 | 3743559.20 | [1427.1, 9201.2] |
| NP/NSR/Genetic/K-means | No | $r(286)-r(143)$ | 3097.5 | 30997419.71 | [-8087.7, 14282.67] |
| | | $r(286)-r(82)$ | -10087.6 | 88230398.80 | [-28958.4, 8783.146] |
| | | $r(286)-r(12)$ | -406.3 | 38997985.10 | [-12952.3, 12139.5] |
| | | $r(286)-r(1)$ | -510.5 | 3592367.01 | [-17142.8, 6939.65] |
| | Yes | $r(286)-r(12)$ | -1208.9 | 1074139.04 | [-3291, 873.24] |
| | | $r(286)-r(1)$ | 555.4 | 740201.10 | [-1173.1, 2283.82] |
| | | $r(286)-r(12)$ | 736.4 | 1046753.62 | [-1318.9, 2791.9] |
| | | $r(286)-r(1)$ | 1592.1 | 794362.10 | [-198.4, 3382.7] |




NP/NSR/K-means algorithm




NP/NSR/Genetic algorithm



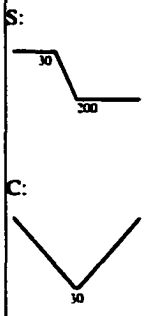
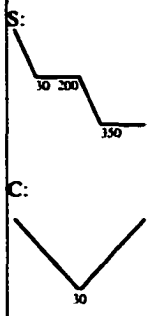
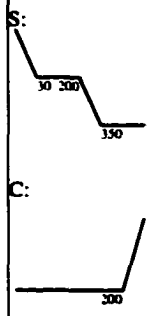
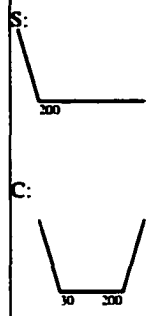
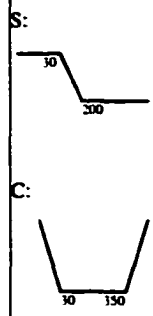
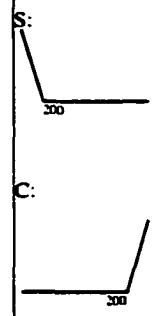
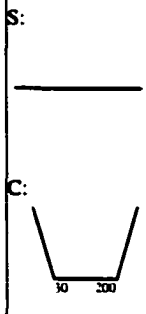
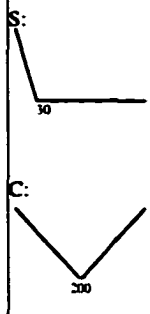
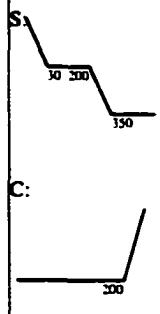
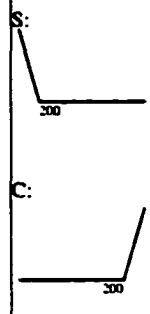
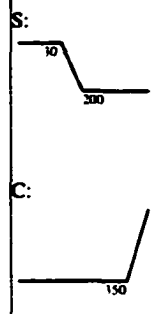
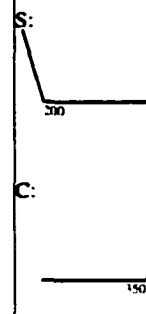
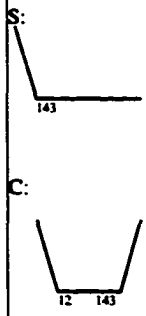
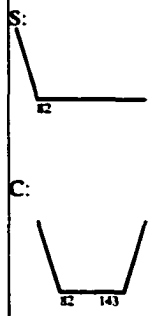
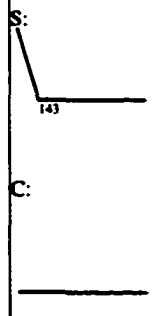
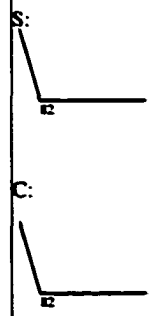
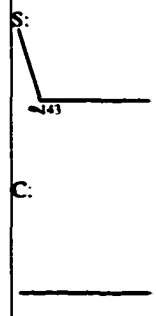
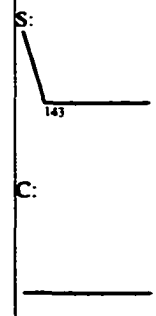
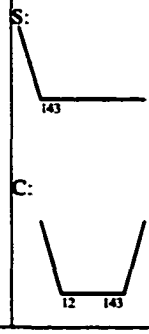
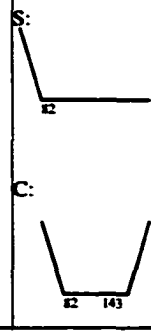
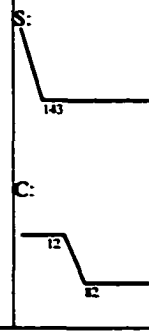
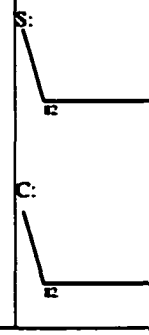
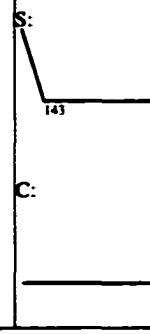
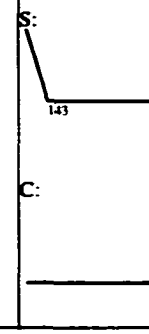

NP/NSR/Genetic/K-means algorithm

Figure 5.9: Computation Time of Each Algorithm with No Inheritance (Left) and Inheritance (Right) for the Small Data Set when the Rinott's Sampling Method is Used

## 5.5 Conclusions

Table 5.17 shows the results of all the cases that are described as an *x*-*y* plot. The *x*-axis and *y*-axis represent the number of replication and the similarity value. The computation time is reduced by sampling of the instances rather than using all the instances. This is more noticeable in cases of large data problem. When using only half of the instances, the computation time is decreased without affecting solution quality. In addition, the variance is decreased, which means computation time is getting stable. With too few instances, the solution quality becomes significantly worse while the computation time goes up. When sampling 4.5~50% of instances, there is a trade-off between the similarity value and the computation time. Most of the K-means algorithm runs had concave shaped plots and needed relatively less computation time than the other algorithms. This is most noticeable when the large size data set is considered.

Table 5.17: Summary of all the Results (S: Similarity Value, C: Computation Time)

| | | K-means | | Genetic | | Genetic/ K-means | |
|---|---|---|---|---|---|---|---|
| | | No Inheritance | Inheritance | No Inheritance | Inheritance | No Inheritance | Inheritance |
| Large | NM |  |  |  |  |  |  |
| | NSR |  |  |  |  |  |  |
| Small | NM |  |  |  |  |  |  |
| | NSR |  |  |  |  |  |  |

# Chapter 6
# Evaluation of New Methodology

In Chapter 2, the new concept of inheritance in the NP method is introduced and evaluated using two different scale problems. In this chapter, the importance of inheritance is evaluated further by numerically verifying that inheritance is beneficial and suggests guidelines to establish the best level of inheritance. Also the different statistical selection methods are evaluated such as Nelson-Matejcik and Rinott's two-stage sampling, which are proposed in Chapter 3 to overcome certain shortcomings of the pure NP method. Lastly, comparison results with several algorithms such as PAM, CLARA, CLARANS are reported. For numerical testing, the problem addressed in Chapter 5 is used.

## 6.1 Amount of Inheritance

One parameter that can affect the quality and computation time of the algorithm is the amount of sample points to be inherited by the next iteration. The amount of inheritance is varied: 0, 3, 6, 10, and 20 sample points for the large data set; 0, 1, 2, 4, and 8 sample points for the small data set (0, 0.5, 0.85, 1.5, and 3% of the data size, respectively). The $t$-test is used to compare different systems based on the difference of performance. Figure 6.1, Figure 6.2, Table 6.1, and Table 6.2 show the results of the large data set when the Nelson-Matejcik sampling method is used. $\mu(i)$ is defined as the expected similarity value of each system and $\tau(i)$ is defined as the expected computation time when the amounts of inheritance are $i \in \{0, 3, 6, 10, 20\}$.

For the NP/NM/K-means algorithm, numerical results show that by inheriting sample points to the next iterations, similarity values are decreased (see the confidence interval of $\mu(0)-\mu(3)$ and $\mu(3)-\mu(6)$ in Table 6.1). However, when considering more than 10 sample points, there is not much difference in the similarity value (see the confidence interval of $\mu(6)-\mu(10)$ and $\mu(10)-\mu(20)$ in Table 6.1). The computation time shows that 3 inherited sample points require the smallest computation time in both cases (see the confidence interval of $\tau(0)-\tau(3)$ in Table 6.2). This reason can be explained by NP partitioning with inheritance and backtracking. As more sample points are inherited to the next iteration, it increases the subset size which thereby causes an increase in computation time. Between different inheritances, there could be a trade-off between the amount of backtracking caused by sampling bias and an increase in computation time caused by a larger subset. Also, the NP method can have sampling bias which can lead to an increase in backtracking without considering inheritance.

As in the NP/NM/K-means algorithm, the NP/NM/Genetic algorithm, by inheriting sample points to the next iteration, makes the similarity value decrease; but with more than 3 sample points, there is no difference in similarity value (see the confidence interval of $\mu(0)-\mu(3)$ in Table 6.1). However, the pattern looks a little different compared to the NP/NM/K-means algorithm. In the NP/NM/K-means algorithm, the gap (difference) between no inheritance and inheritances (except with 3 inheritances) is not clear even if confidence intervals show there is a difference. In the NP/NM/Genetic algorithm, however, this gap looks very clear, and there is not much gap between different inheritances. This can be explained by a GA search and backtracking of the NP method. If the algorithm starts with

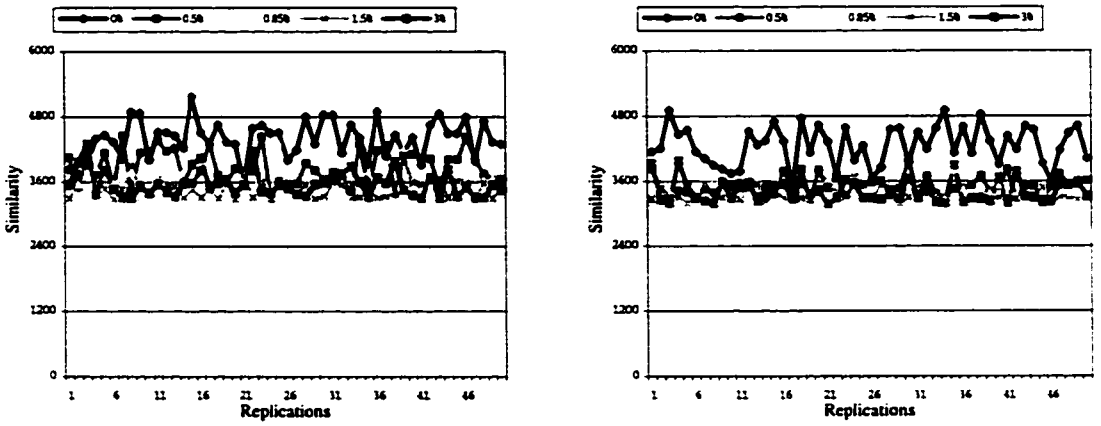
randomly selected parents (without using inheritance), at any iteration, the number of backtracking is going to be increased, thereby increasing the computation time. The reason is the same as with the NP/NM/K-means algorithm. The GA search makes the population form the best two parents that are acquired from the previous iteration within the GA search. Through cross-over and mutation, the quality of these populations improves. So inheriting the initial two best samples of parents by the next iteration avoids an increase in the amount of backtracking. The NP/NM/Genetic algorithm is implemented to choose parents randomly if more than 2 samples are inherited. If not, the top 2 inherited points are considered as parents for making the population. This could degrade the quality of the solution and increase the computation time by invoking more backtracking.

Figure 6.3, Figure 6.4, Table 6.3, and Table 6.4 show the results of the small data set when the Nelson-Matejcik sampling method is used. $\mu(i)$ is defined as the expected similarity value of each system and $\tau(i)$ is defined as the expected computation time when the amount of inheritance is $i \in \{0, 1, 2, 4, 8\}$. Similar results is obtained from the small data set from the NP/NM/K-means algorithm. Similarity values is decreased by using inheritance, but there is no difference in similarity value when inheriting more than 2 sample points (see the confidence interval of $\mu(1)-\mu(2)$ in Table 6.3) and 1 sample point inheritance requires the smallest computation time (see the confidence interval of $\tau(0)-\tau(1)$ in Table 6.4).

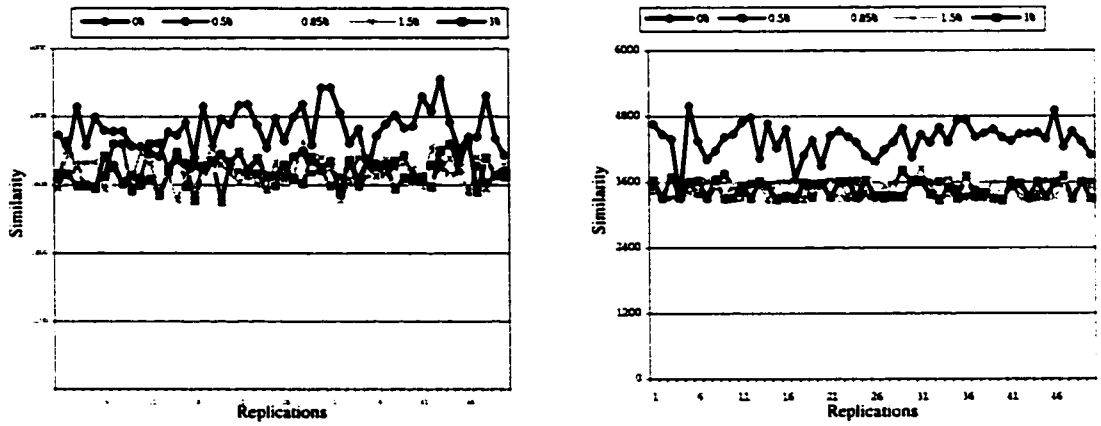
In the NP/NM/Genetic algorithm, the similarity value is decreased but there is no need for more than 1 inheritance and inheriting 1 sample point requires the smallest computation time (see the confidence interval of $\tau(0)-\tau(1)$ in Table 6.4). The similarity

Table 6.1: Similarity Results of $t$-test with 95% Confidence Interval for the Large Data Set when the Nelson-Matejcik Sampling Method is Used

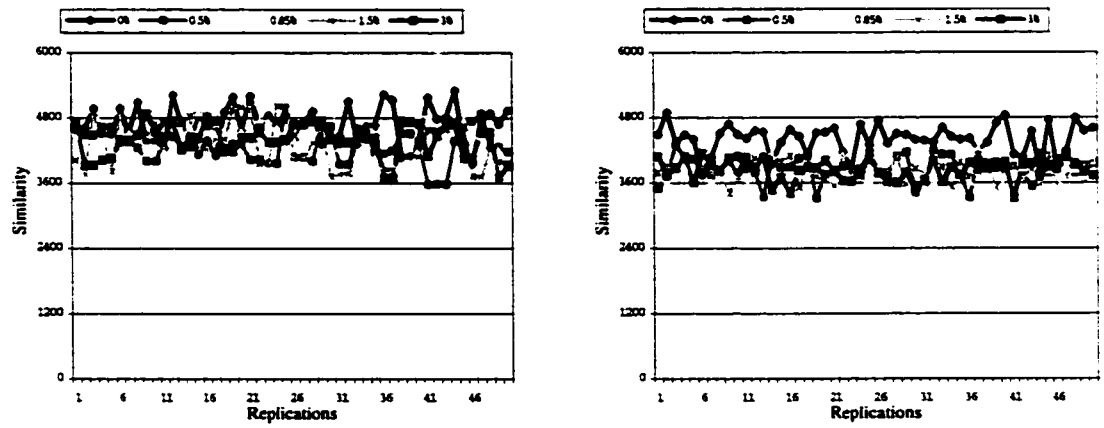
| Algorithm | Instances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | 3(0.5%) | $\mu(0)-\mu(3)$ | 528.6 | 4682.23 | [391.1, 666.1] |
| | | $\mu(3)-\mu(6)$ | 204.2 | 3774.14 | [80.8, 327.6] |
| | | **$\mu(6)-\mu(10)$** | **168.4** | **1467.90** | **[91.4, 245.3]** |
| | | $\mu(10)-\mu(20)$ | -69.7 | 1875.98 | [-156.7,17.2] |
| | 200(28%) | $\mu(0)-\mu(3)$ | 754.9 | 3345.20 | [638.7, 871.1] |
| | | **$\mu(3)-\mu(6)$** | **106.9** | **1023.01** | **[42.6, 171.1]** |
| | | $\mu(6)-\mu(10)$ | 54.5 | 737.30 | [-0.1, 109.0] |
| | | $\mu(6)-\mu(20)$ | -9.2 | 1283.28 | [-81.2, 62.7] |
| NP/NM/Genetic | 3(0.5%) | **$\mu(0)-\mu(3)$** | **744.0** | **4249.89** | **[613.1, 875.0]** |
| | | $\mu(3)-\mu(6)$ | 38.5 | 1986.14 | [-50.9, 128.0] |
| | | $\mu(3)-\mu(10)$ | 16.6 | 2015.94 | [-73.5, 106.8] |
| | | $\mu(3)-\mu(20)$ | 42.9 | 2987.78 | [-66.8, 152.7] |
| | 200(28%) | **$\mu(0)-\mu(3)$** | **898.4** | **1983.81** | **[808.9, 987.8]** |
| | | $\mu(3)-\mu(6)$ | -0.1 | 1046.80 | [-65.1, 64.8] |
| | | $\mu(3)-\mu(10)$ | 21.7 | 718.92 | [-32.1, 75.6] |
| | | $\mu(3)-\mu(20)$ | 17.6 | 821.72 | [-39.9, 75.2] |
| NP/NM/Genetic/K-means | 3(0.5%) | **$\mu(0)-\mu(3)$** | **342.8** | **6483.34** | **[181.0, 504.6]** |
| | | $\mu(3)-\mu(6)$ | -172.8 | 3954.42 | [-299.1, -46.4] |
| | | $\mu(3)-\mu(10)$ | -22.0 | 5845.19 | [-175.5, 131.5] |
| | | $\mu(3)-\mu(20)$ | -75.5 | 4088.27 | [-204.2, 52.6] |
| | 200(28%) | **$\mu(0)-\mu(3)$** | **552.5** | **2695.56** | **[448.2, 656.8]** |
| | | $\mu(3)-\mu(6)$ | 64.3 | 2153.06 | [-28.9, 157.5] |
| | | $\mu(3)-\mu(10)$ | 25.3 | 1737.24 | [-58.4, 109.0] |
| | | $\mu(3)-\mu(20)$ | 6.7 | 1970.01 | [-82.4, 95.8] |




NP/NM/K-means algorithm




NP/NM/Genetic algorithm




NP/NM/Genetic/K-means algorithm

Figure 6.1: Similarity Value of Each Algorithm when the Numbers of Instances are 3 (Left), and 200 (Right) of the Large Data Set when the Nelson-Matejcik Sampling Method is Used

Table 6.2: Computation Time Results of $t$-test with 95% Confidence Interval for the Large Data Set when the Nelson-Matejcik Sampling Method is Used

| Algorithm | Instances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | 3(0.5%) | $\tau(20)-\tau(10)$ | 116.5 | 5330.06 | [-30.1, 263.2] |
| | | $\tau(20)-\tau(6)$ | 159.9 | 15941.31 | [-93.6, 413.6] |
| | | $\tau(20)-\tau(3)$ | 7688.8 | 161148.20 | [6882.4, 8495.4] |
| | | $\tau(0)-\tau(3)$ | 9950.4 | 819072.60 | [8132.2, 11768.7] |
| | 200(28%) | $\tau(20)-\tau(10)$ | 48.4 | 69038.00 | [-479.5, 576.2] |
| | | $\tau(20)-\tau(6)$ | 639.0 | 73666.00 | [93.6, 1184.2] |
| | | $\tau(6)-\tau(3)$ | 10724.0 | 330525.00 | [9568.6, 11878.6] |
| | | $\tau(0)-\tau(3)$ | 12663.5 | 502978.20 | [11238.7, 14088.4] |
| NP/NM/Genetic | 3(0.5%) | $\tau(20)-\tau(10)$ | 14166.0 | 2999346.00 | [10687.2, 17645.8] |
| | | $\tau(20)-\tau(6)$ | 6465.9 | 3403612.10 | [2759.6, 10172.3] |
| | | $\tau(6)-\tau(3)$ | 4219.2 | 4491934.20 | [-38.7, 8477.1] |
| | | $\tau(0)-\tau(3)$ | 78673.6 | 26044356.00 | [68421.0, 88926.3] |
| | 200(28%) | $\tau(20)-\tau(10)$ | 19133.1 | 4203785.00 | [15014.0, 23252.2] |
| | | $\tau(10)-\tau(6)$ | 2183.3 | 2184388.30 | [-785.9, 5152.6] |
| | | $\tau(10)-\tau(3)$ | 7982.9 | 1606192.00 | [5436.8, 10529.0] |
| | | $\tau(0)-\tau(3)$ | 88642.6 | 29323923.00 | [77763.2, 99521.3] |
| NP/NM/Genetic/K-means | 3(0.5%) | $\tau(20)-\tau(10)$ | 1869.3 | 2046660.30 | [-1004.8, 4743.4] |
| | | $\tau(20)-\tau(6)$ | 1631.0 | 1109214.00 | [-484.9, 3746.8] |
| | | $\tau(20)-\tau(3)$ | 7440.3 | 1039346.00 | [-1185.0, 3677.3] |
| | | $\tau(0)-\tau(3)$ | 40400.3 | 1959527.10 | [37588.1, 43212.6] |
| | 200(28%) | $\tau(20)-\tau(10)$ | -1882.8 | 2373735.00 | [-4978.1, 1212.3] |
| | | $\tau(20)-\tau(6)$ | -989.5 | 1440955.00 | [-3401.2, 1422.0] |
| | | $\tau(20)-\tau(3)$ | 381.2 | 1500706.00 | [-2079.6, 2842.3] |
| | | $\tau(0)-\tau(3)$ | 55091.7 | 12733533.60 | [47922.8, 62260.7] |

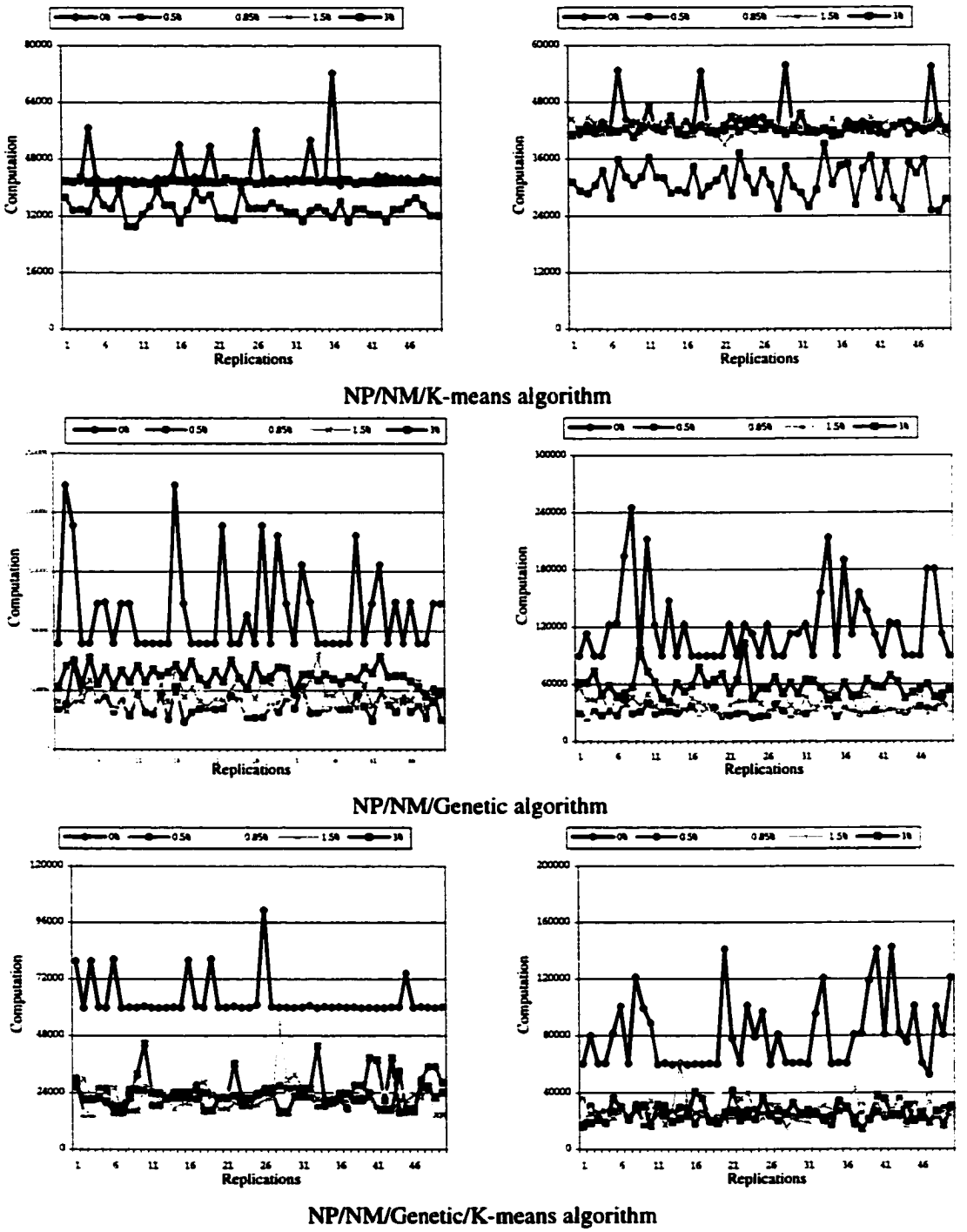



Figure 6.2: Computation Time of Each Algorithm when the Numbers of Instances are 3 (Left), and 200 (Right) of the Large Data Set when the Nelson-Matejcik Sampling Method is Used

value decreases by inheriting more sample points (see the confidence interval of $\mu(0)-\mu(1)$ in Table 6.1) but there is no need for more than 1 sample point. Moreover, 1 sample point inheritance requires the smallest computation time (see the confidence interval of $\tau(0)-\tau(1)$ in Table 6.4).

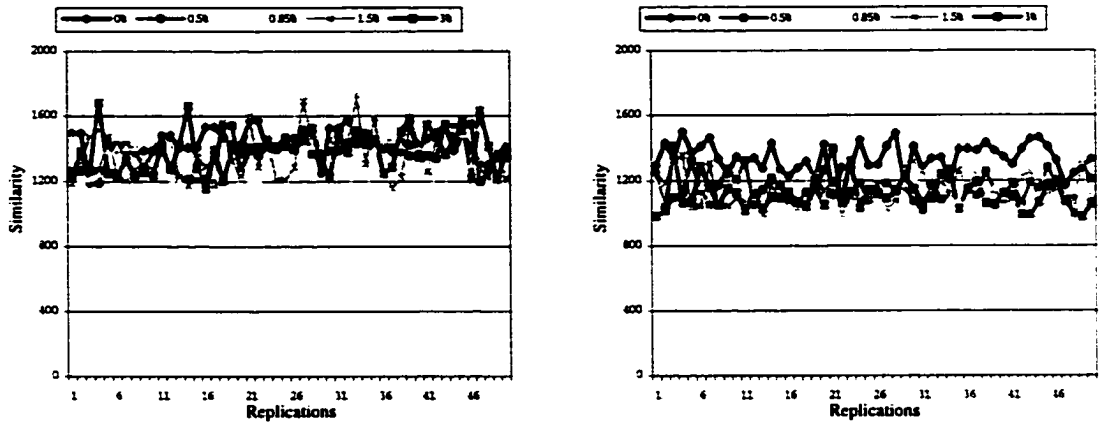
In conclusion, sample points inherited from the previous iteration decreases the similarity value in the next iteration but no more than 0.5% inheritance for the small data set and, 0.5~0.85% inheritance for the large data set are needed. Most of the computation time is minimized when the inheritance level is 0.5% and is stabilized by using inheritance. Computation time shows different patterns that depended on what algorithms are used, not on the size of the data set as already explained above. Rinott's sampling method shows similar results.

## 6.2 Statistical Sampling

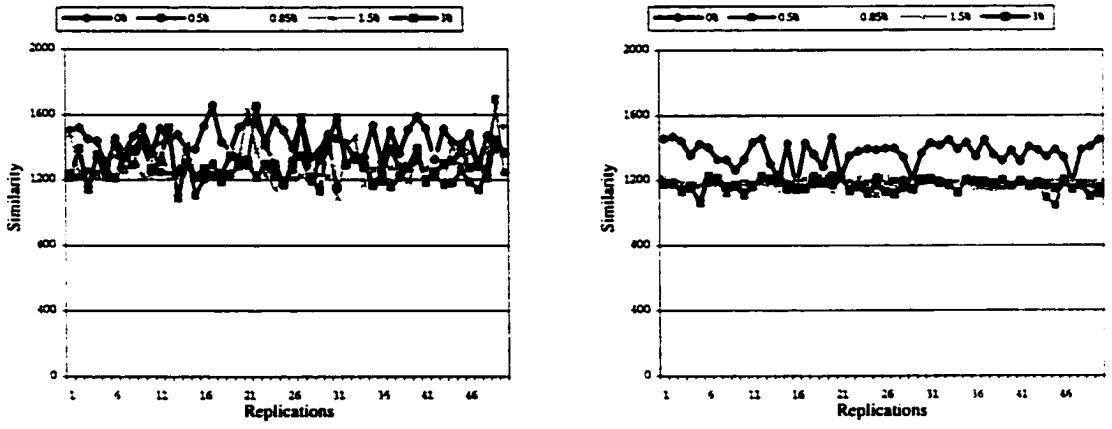
As already discussed in the previous section, there are two shortcomings of the pure NP method: the probability of success in each iteration is not guaranteed, and there may be considerable waste involved in the allocation of sample points. These can be handled by combining pure NP with a statistical sampling method (Nelson-Matejcik and Rinott's sampling). Figure 6.5, Figure 6.6, Figure 6.7, and Table 6.5 show the results of the large data set when the numbers of instances are 3, 30, 200. $\mu$(P), $\mu$(NM), and $\mu$(NSR) are defined as the expected similarity value when pure NP, NP/NM, and NP/NSR algorithms are used. Combining algorithms with the NP method are definitely better than the pure algorithm in terms of similarity value (all the confidence interval does not contain 0 in Table 6.5).

Table 6.3: Similarity Results of *t*-test with 95% Confidence Interval for the Small Data Set when the Nelson-Matejcik Sampling Method is Used

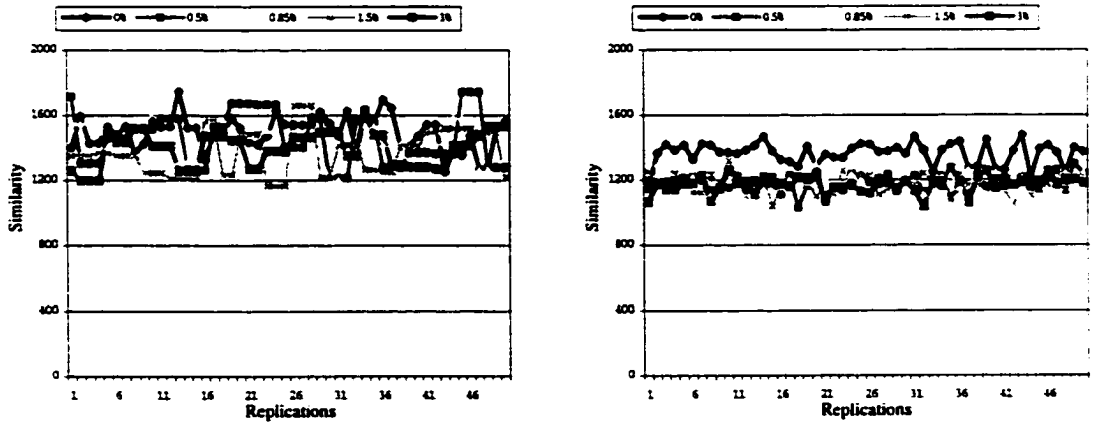
| Algorithm | Instances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/K-means | 1(0.5%) | $\mu(0)-\mu(1)$ | 52.3 | 246.35 | [20.8, 83.8] |
| | | **$\mu(1)-\mu(2)$** | **51.9** | **620.00** | **[1.9, 102.0]** |
| | | $\mu(2)-\mu(4)$ | -35.7 | 875.10 | [-23.7, 95.1] |
| | | $\mu(2)-\mu(8)$ | -58.4 | 700.70 | [-111.6, -5.2] |
| | 82(28%) | $\mu(0)-\mu(1)$ | 166.4 | 325.70 | [130.1, 202.7] |
| | | **$\mu(1)-\mu(2)$** | **36.2** | **247.30** | **[4.5, 67.8]** |
| | | $\mu(2)-\mu(4)$ | -20.3 | 360.80 | [-58.4, 17.9] |
| | | $\mu(2)-\mu(8)$ | 18.2 | 205.80 | [-10.6,47.0] |
| NP/NM/Genetic | 1(0.5%) | **$\mu(0)-\mu(1)$** | **166.8** | **367.00** | **[128.3, 205.3]** |
| | | $\mu(1)-\mu(2)$ | -9.8 | 412.50 | [-50.6, 30.9] |
| | | $\mu(1)-\mu(4)$ | -5.8 | 297.20 | [-40.4, 28.8] |
| | | $\mu(1)-\mu(8)$ | -6.9 | 405.10 | [-47.4, 33.5] |
| | 82(28%) | **$\mu(0)-\mu(1)$** | **195.6** | **139.50** | **[171.9, 219.3]** |
| | | $\mu(1)-\mu(2)$ | 1.6 | 45.80 | [-11.9, 15.3] |
| | | $\mu(1)-\mu(4)$ | -0.5 | 32.38 | [-16.8, 6.0] |
| | | $\mu(1)-\mu(8)$ | 3.1 | 36.99 | [-9.0, 15.3] |
| NP/NM/Genetic/K-means | 1(0.5%) | **$\mu(0)-\mu(1)$** | **57.2** | **791.30** | **[0.7, 113.8]** |
| | | $\mu(1)-\mu(2)$ | 33.9 | 594.60 | [-15.0, 82.9] |
| | | $\mu(1)-\mu(4)$ | 85.6 | 955.26 | [23.5, 147.7] |
| | | $\mu(4)-\mu(8)$ | -20.0 | 617.13 | [-69.9, 29.8] |
| | 82(28%) | **$\mu(0)-\mu(1)$** | **184.9** | **139.70** | **[161.2, 208.7]** |
| | | $\mu(1)-\mu(2)$ | -11.2 | 110.60 | [-32.3, 9.9] |
| | | $\mu(1)-\mu(4)$ | 3.1 | 118.60 | [-18.7, 25.0] |
| | | $\mu(1)-\mu(8)$ | 5.0 | 94.88 | [-14.5, 24.6] |




NP/NM/K-means algorithm




NP/NM/Genetic algorithm




NP/NM/Genetic/K-means algorithm

Figure 6.3: Similarity Value of Each Algorithm when the Numbers of Instances are 1 (Left), and 82 (Right) of the Small Data Set when the Nelson-Matejcik Sampling Method is Used

Table 6.4: Computation Time Results of $t$-test with 95% Confidence Interval of Small Data when Nelson-Matejcik Sampling Method is Used

| Algorithm | Instances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| NP/NM/k-Means | 1(0.5%) | $\tau(8)-\tau(4)$ | 7328.2 | 1263366.50 | [5070.1, 9586.3] |
| | | $\tau(4)-\tau(2)$ | 11634.0 | 907747.23 | [9720.6, 13548.8] |
| | | $\tau(2)-\tau(1)$ | 4399.6 | 1034573.50 | [2356.2, 6443.1] |
| | | $\tau(0)-\tau(1)$ | 19730.7 | 2156646.10 | [16780.4, 22681.1] |
| | 82(28%) | $\tau(8)-\tau(4)$ | 6693.5 | 495405.40 | [5279.5, 8107.5] |
| | | $\tau(4)-\tau(2)$ | 7397.9 | 325284.27 | [6252.1, 8543.7] |
| | | $\tau(2)-\tau(1)$ | 2717.9 | 90731.90 | [2112.7, 3323.1] |
| | | $\tau(0)-\tau(1)$ | 16281.2 | 507203.15 | [14850.5, 17712.0] |
| NP/NM/Genetic | 1(0.5%) | $\tau(8)-\tau(4)$ | 4040.1 | 4122266.20 | [-38.7, 8119.1] |
| | | $\tau(8)-\tau(2)$ | 10929.6 | 2628330.20 | [7672.6, 14186.6] |
| | | $\tau(2)-\tau(1)$ | 3427.4 | 2215261.90 | [437.3, 6417.6] |
| | | $\tau(0)-\tau(1)$ | 88640.8 | 110502160.50 | [67522.3, 109759.5] |
| | 82(28%) | $\tau(8)-\tau(4)$ | 2069.0 | 1015423.60 | [44.6, 4093.5] |
| | | $\tau(4)-\tau(2)$ | 2419.4 | 908391.05 | [504.7, 4334.2] |
| | | $\tau(2)-\tau(1)$ | 2648.6 | 659898.69 | [1016.7, 4280.6] |
| | | $\tau(0)-\tau(1)$ | 92618.8 | 83748658.00 | [74233.5, 111003.9] |
| NP/NM/Genetic/k-Means | 1(0.5%) | $\tau(8)-\tau(4)$ | 1861.6 | 1857355.40 | [-876.2, 4599.6] |
| | | $\tau(8)-\tau(2)$ | 3288.4 | 1582839.50 | [760.9, 5816.0] |
| | | $\tau(2)-\tau(1)$ | 672.2 | 54202.65 | [204.5, 1139.9] |
| | | $\tau(0)-\tau(1)$ | 39553.7 | 10582267.00 | [33018.3, 46089.0] |
| | 82(28%) | $\tau(8)-\tau(4)$ | 636.4 | 1031686.10 | [-1404.2, 2677.0] |
| | | $\tau(8)-\tau(2)$ | 4044.6 | 819583.70 | [2225.8, 5863.3] |
| | | $\tau(2)-\tau(1)$ | 1388.3 | 215079.08 | [456.6, 2320.0] |
| | | $\tau(0)-\tau(1)$ | 55019.6 | 21487199.00 | [45584.9, 64454.4] |

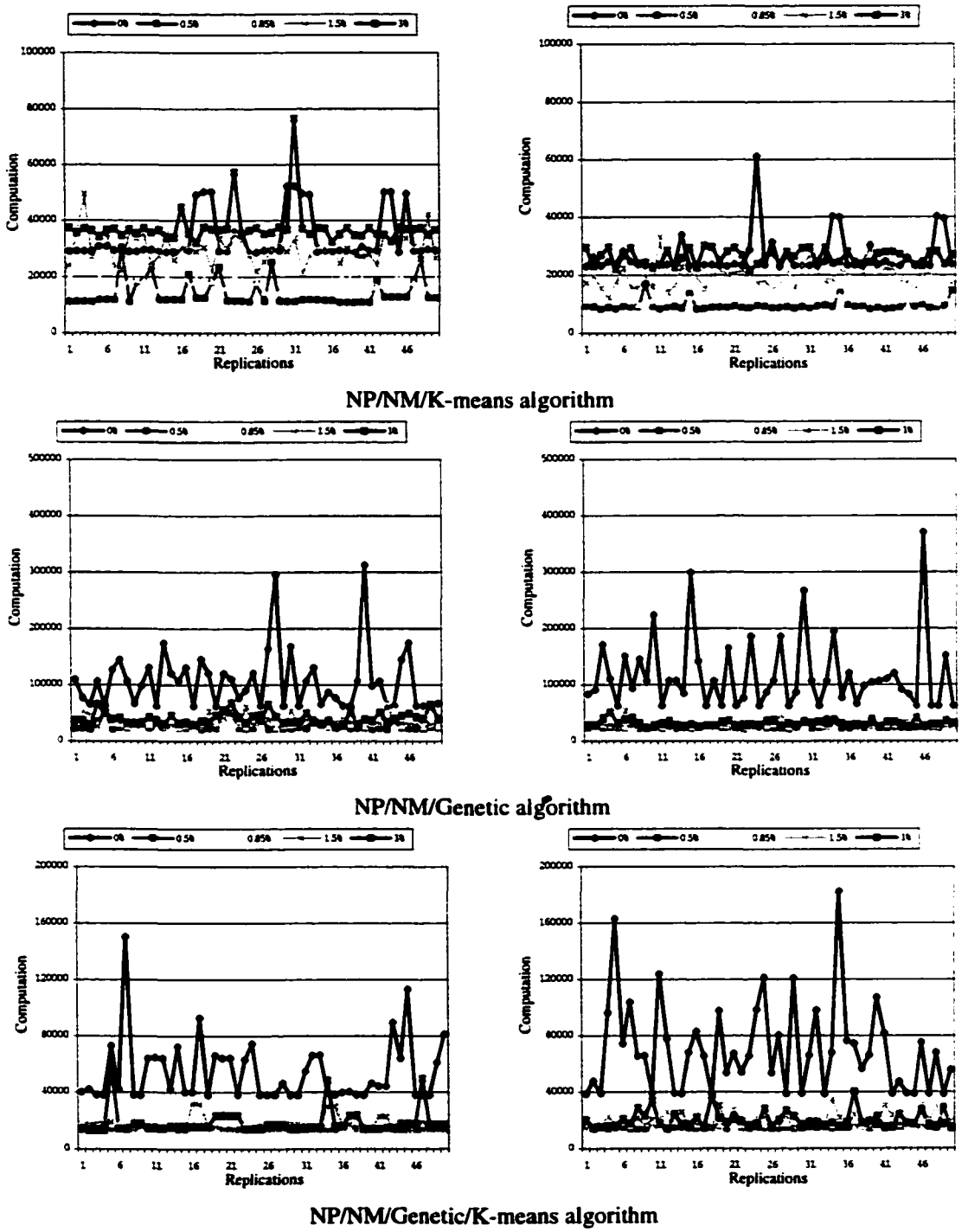


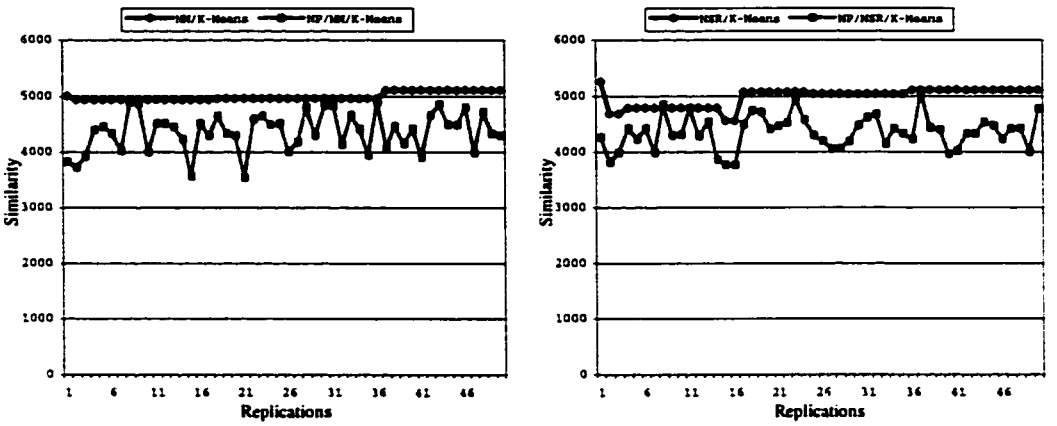

Figure 6.4: Computation Time of Each Algorithm when the Numbers of Instances are 1 (Left), and 82 (Right) of the Small Data Set when the Nelson-Matejcik Sampling Method is Used

However, the NP method required more computation time. Similar results are obtained from the small data set.

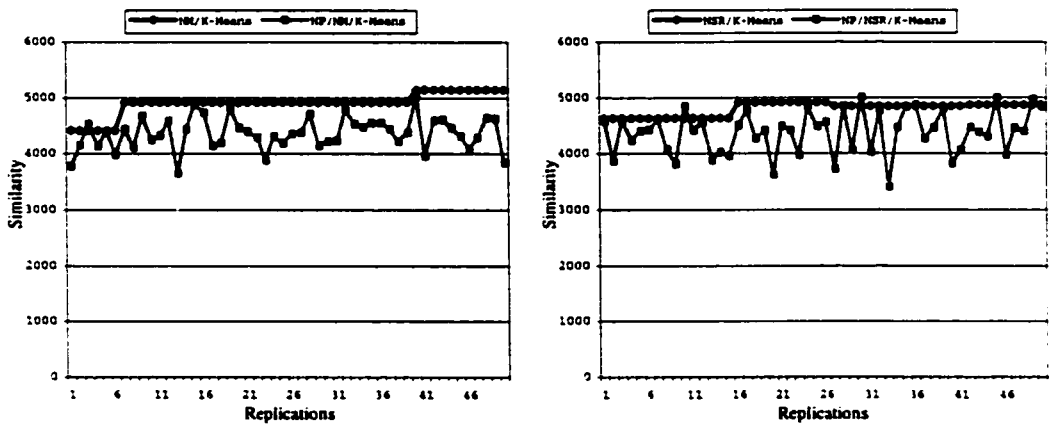
It has been observed that combining NP algorithms are better than pure NP algorithm in terms of the similarity value. Next, comparisons between different sampling methods are observed. Figure 6.8, Figure 6.9, Figure 6.10, and Table 6.6 show the results of the similarity value for the large size data set when the numbers of instances are 30 and 200. $\mu(\text{NM}\_30)$ and $\mu(\text{NSR}\_30)$ are defined as the expected similarity value and $\tau(\text{NM}\_30)$ and $\tau(\text{NSR}\_30)$ are defined as the expected computation time for the NP/NM and NP/NSR algorithms when 30 instances are used. $\mu(\text{NM}\_200)$ and $\mu(\text{NSR}\_200)$ are defined as the expected similarity value and $\tau(\text{NM}\_200)$ and $\tau(\text{NSR}\_200)$ are defined as the expected computation time for the NP/NM and NP/NSR algorithms when 200 instances are used. Table 6.6 shows all the confidence intervals containing 0, which means there are no significant difference in similarity values between the Nelson-Matejcik sampling and the Rinott's sampling method. Computation results show the same results (see the Figure 6.8, Figure 6.9, Figure 6.10, and Table 6.6).

Table 6.5: Similarity Results of $t$-test with 95% Confidence Interval for the Large Data Set

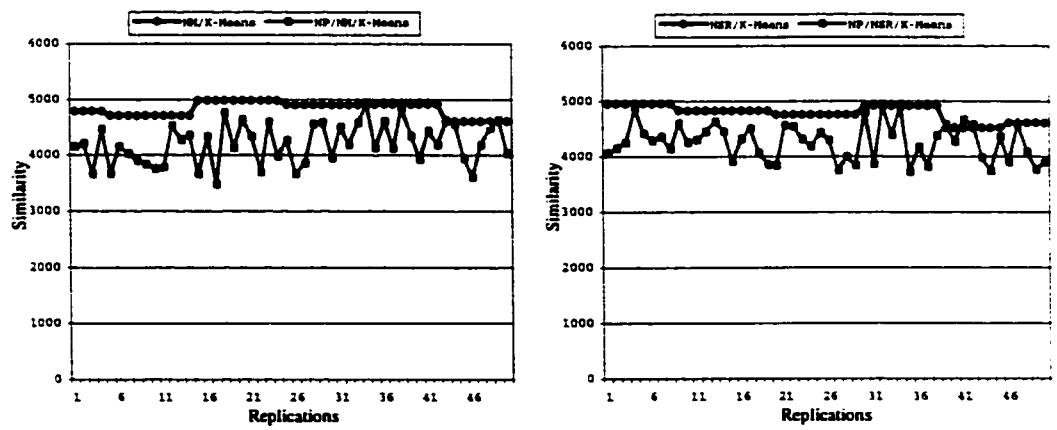
| Algorithm | Instances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| K-means | 3(0.5%) | $\mu$(P) – $\mu$(NM) | 597.42 | 2516.087 | [496.6, 698.1] |
| | | $\mu$(P) – $\mu$(NSR) | 627.84 | 1590.046 | [547.7, 707.9] |
| | 30(4.5%) | $\mu$(P) – $\mu$(NM) | 541.34 | 1942.268 | [452.8, 629.8] |
| | | $\mu$(P) – $\mu$(NSR) | 422.74 | 3008.832 | [312.5, 532.9] |
| | 200(28%) | $\mu$(P) – $\mu$(NM) | 560.66 | 2756.138 | [455.1, 666.1] |
| | | $\mu$(P) – $\mu$(NSR) | 488.68 | 2714.066 | [384.0, 593.3] |
| Genetic | 3(0.5%) | $\mu$(P) – $\mu$(NM) | 234.98 | 2693.210 | [130.7, 339.2] |
| | | $\mu$(P) – $\mu$(NSR) | 530.56 | 1459.407 | [453.8, 607.3] |
| | 30(4.5%) | $\mu$(P) – $\mu$(NM) | 582.96 | 1886.153 | [495.7, 670.2] |
| | | $\mu$(P) – $\mu$(NSR) | 578.88 | 2767.782 | [473.1, 684.5] |
| | 200(28%) | $\mu$(P) – $\mu$(NM) | 523.44 | 1952.084 | [434.6, 612.2] |
| | | $\mu$(P) – $\mu$(NSR) | 518.04 | 2641.413 | [414.7, 621.2] |
| Genetic/K-means | 3(0.5%) | $\mu$(P) – $\mu$(NM) | 788.06 | 8311.132 | [604.9, 971.2] |
| | | $\mu$(P) – $\mu$(NSR) | 398.00 | 5370.542 | [250.7, 545.2] |
| | 30(4.5%) | $\mu$(P) – $\mu$(NM) | 306.32 | 4335.522 | [174.0, 438.6] |
| | | $\mu$(P) – $\mu$(NSR) | 620.40 | 4028.181 | [492.8, 747.9] |
| | 200(28%) | $\mu$(P) – $\mu$(NM) | 558.46 | 3448.608 | [440.4, 676.4] |
| | | $\mu$(P) – $\mu$(NSR) | 1112.50 | 9195.504 | [919.8, 1305.1] |




3(0.5%) Instances

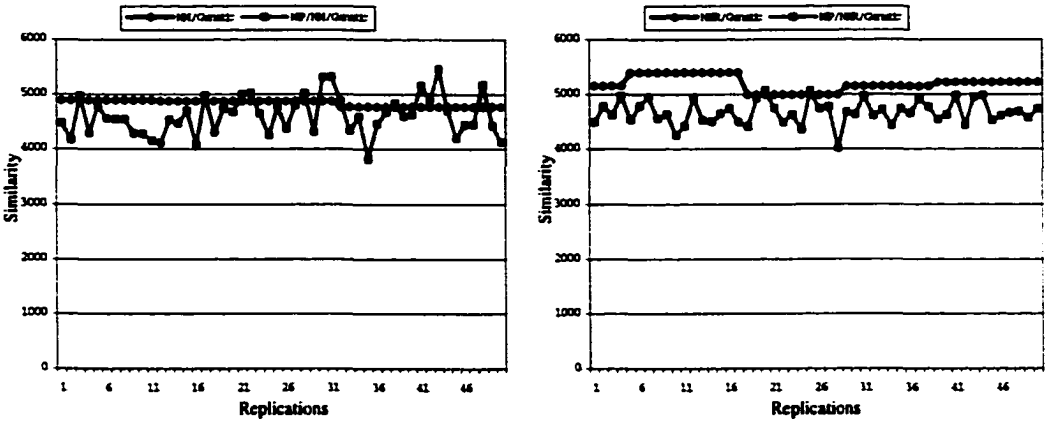



30(4.5%) Instances

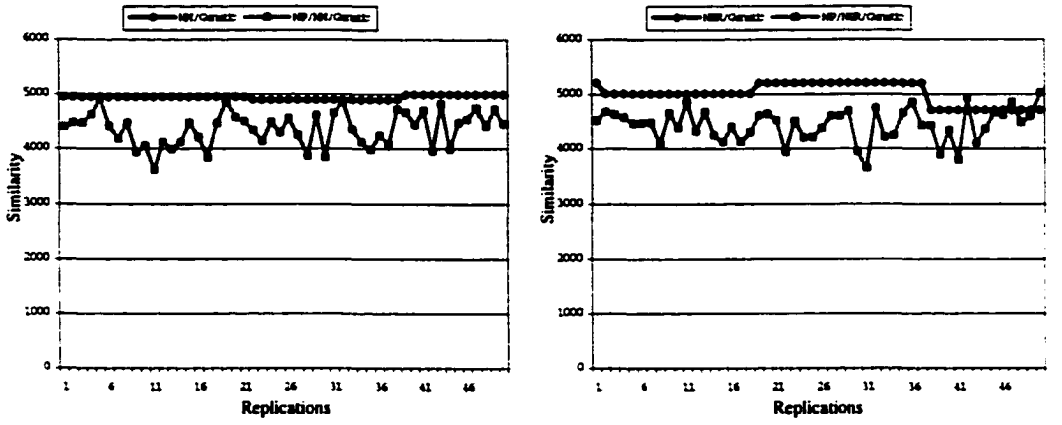



200(28%) Instances

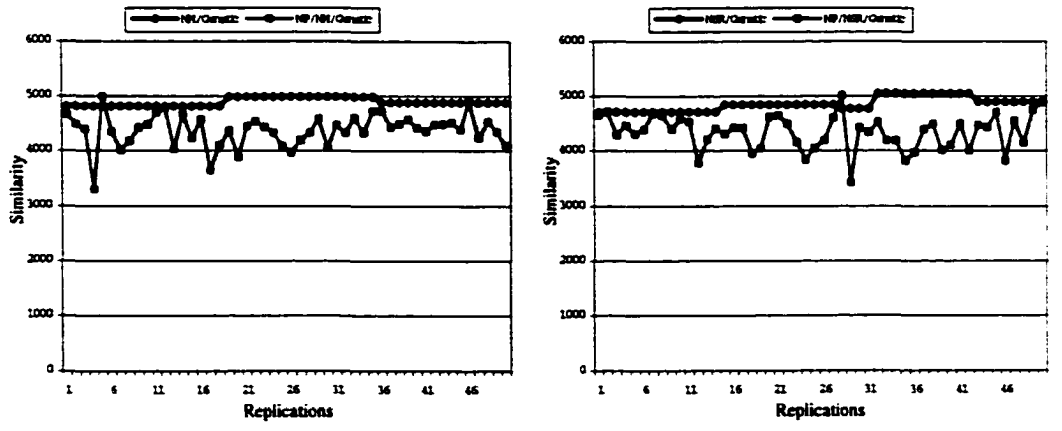
Figure 6.5: Similarity Value of the Pure K-means and Combined K-means with the NP for the Large Data Set




3(0.5%) Instances

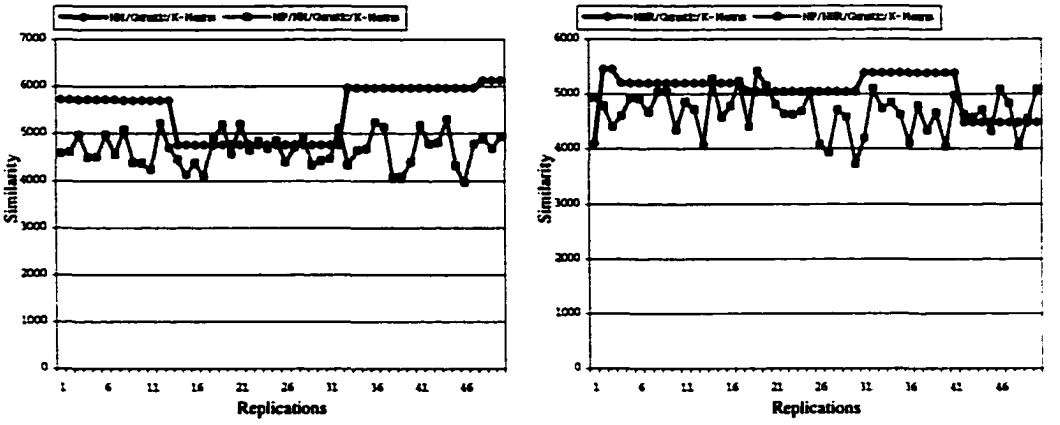



30(4.5%) Instances

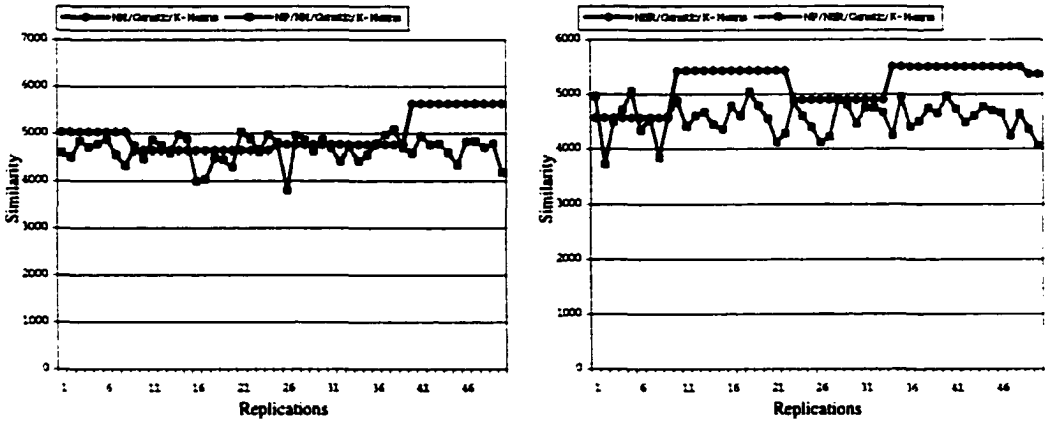



200(28%) Instances

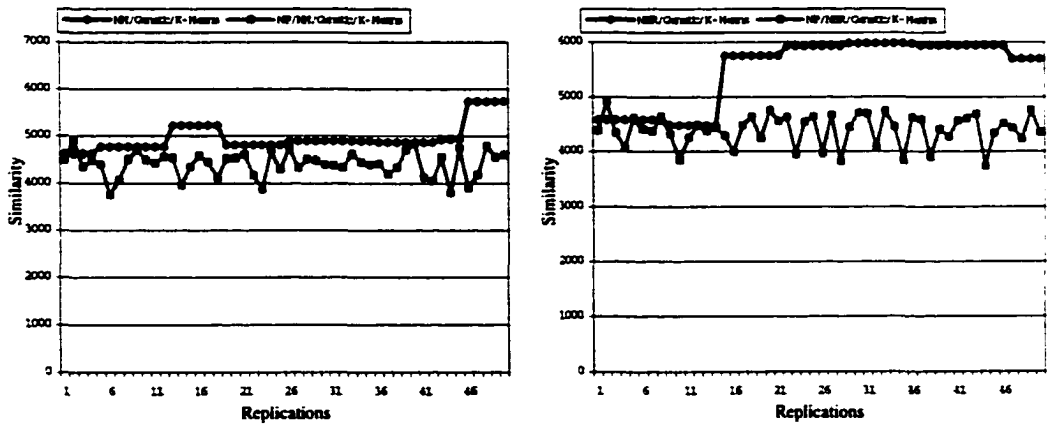
Figure 6.6: Similarity Value of Pure Genetic and Combined Genetic with the NP for the Large Data Set




3(0.5%) Instances




30(4.5%) Instances




200(28%) Instances

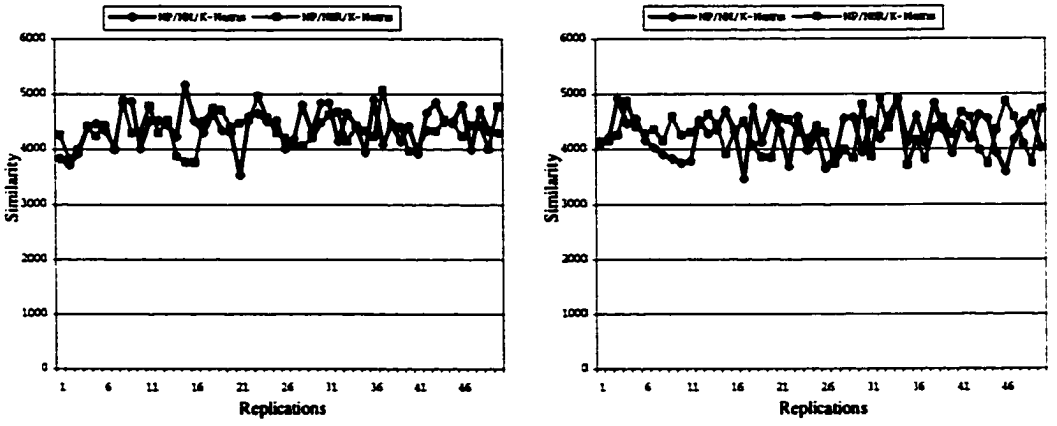
Figure 6.7: Similarity Value of Genetic/K-means and Combined Genetic/K-means with the NP for the Large Data Set

Table 6.6: Similarity Results of $t$-test with 95% Confidence Interval for the Large Data Set

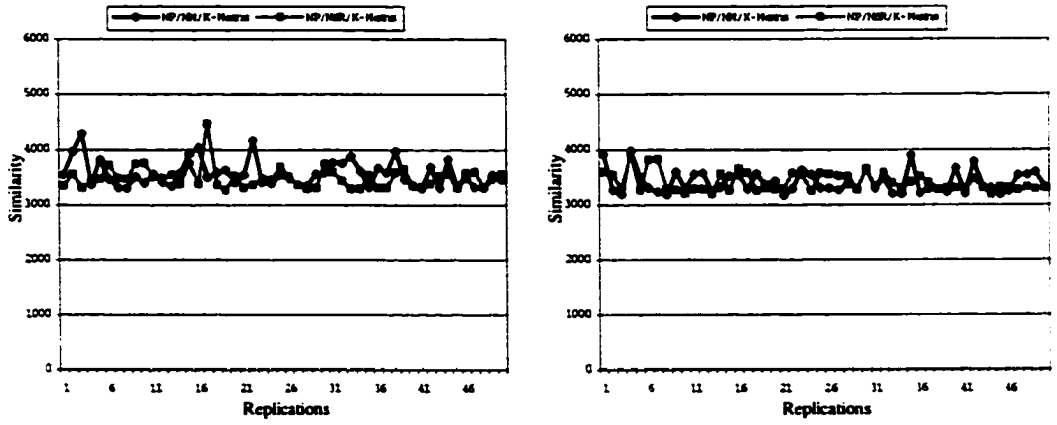
| Algorithm | Inheritances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| K-means | 0(0%) | $\mu(NM\_30) - \mu(NSR\_30)$ | -13.44 | 4488 | [-148.0, 121.1] |
| | | $\mu(NM\_200) - \mu(NSR\_200)$ | -30.9 | 5850 | [-184.5, 122.7] |
| | 20(3%) | $\mu(NM\_30) - \mu(NSR\_30)$ | 17.3 | 785 | [-38.9, 73.6] |
| | | $\mu(NM\_200) - \mu(NSR\_200)$ | 2.2 | 1142 | [-65.6, 70.1] |
| Genetic | 0(0%) | $\mu(NM\_30) - \mu(NSR\_30)$ | -70.1 | 4091 | [-198.6, 58.3] |
| | | $\mu(NM\_200) - \mu(NSR\_200)$ | 34.5 | 4874 | [-105.7, 174.8] |
| | 20(3%) | $\mu(NM\_30) - \mu(NSR\_30)$ | 24.3 | 1012 | [-39.5, 88.2] |
| | | $\mu(NM\_200) - \mu(NSR\_200)$ | -12.6 | 889 | [-72.5, 47.2] |
| Genetic/K-means | 0(0%) | $\mu(NM\_30) - \mu(NSR\_30)$ | 93.2 | 3145 | [-19.4, 205.9] |
| | | $\mu(NM\_200) - \mu(NSR\_200)$ | 9.9 | 2809 | [-95.5, 116.4] |
| | 20(3%) | $\mu(NM\_30) - \mu(NSR\_30)$ | 50.9 | 2327 | [-46.0, 147.8] |
| | | $\mu(NM\_200) - \mu(NSR\_200)$ | 5.9 | 1910 | [-81.8, 93.7] |

Table 6.7: Computation Time Results of $t$-test with 95% Confidence Interval for the Large Data Set

| Algorithm | Inheritances | $\delta$ | $\overline{Z}(n)$ | $Var(\overline{Z}(n))$ | Confidence Interval |
|---|---|---|---|---|---|
| K-means | 0(0%) | $\tau(NM\_30) - \tau(NSR\_30)$ | 825.6 | 689577 | [-2493.9, 842.6] |
| | | $\tau(NM\_200) - \tau(NSR\_200)$ | 4304.9 | 453621 | [2951.8, 5657.9] |
| | 20(3%) | $\tau(NM\_30) - \tau(NSR\_30)$ | 7.3 | 3115 | [-104.7, 119.5] |
| | | $\tau(NM\_200) - \tau(NSR\_200)$ | 5402.0 | 31995 | [5042.7, 5761.4] |
| Genetic | 0(0%) | $\tau(NM\_30) - \tau(NSR\_30)$ | 6416.4 | 93000000 | [-12943.0, 25775.7] |
| | | $\tau(NM\_200) - \tau(NSR\_200)$ | 6134.8 | 49000000 | [-7998.2, 20267.8] |
| | 20(3%) | $\tau(NM\_30) - \tau(NSR\_30)$ | -1368.7 | 3619518 | [-5190.8, 2453.4] |
| | | $\tau(NM\_200) - \tau(NSR\_200)$ | -2689.4 | 4771738 | [-7077.9, 1699.1] |
| Genetic/K-means | 0(0%) | $\tau(NM\_30) - \tau(NSR\_30)$ | -7548.4 | 39000000 | [-200044.0, 4947.4] |
| | | $\tau(NM\_200) - \tau(NSR\_200)$ | 7507.3 | 22000000 | [-1911.2, 16925.9] |
| | 20(3%) | $\tau(NM\_30) - \tau(NSR\_30)$ | -2408.3 | 2582494 | [-5636.8, 820.1] |
| | | $\tau(NM\_200) - \tau(NSR\_200)$ | -1966.6 | 1600965 | [-575.3, 4508.5] |




0(0%) Inheritance




20(3%) Inheritances

Figure 6.8: Similarity Value of K-means when then Numbers of Instances are 30(Left) and 200(right) for the Large Data Set

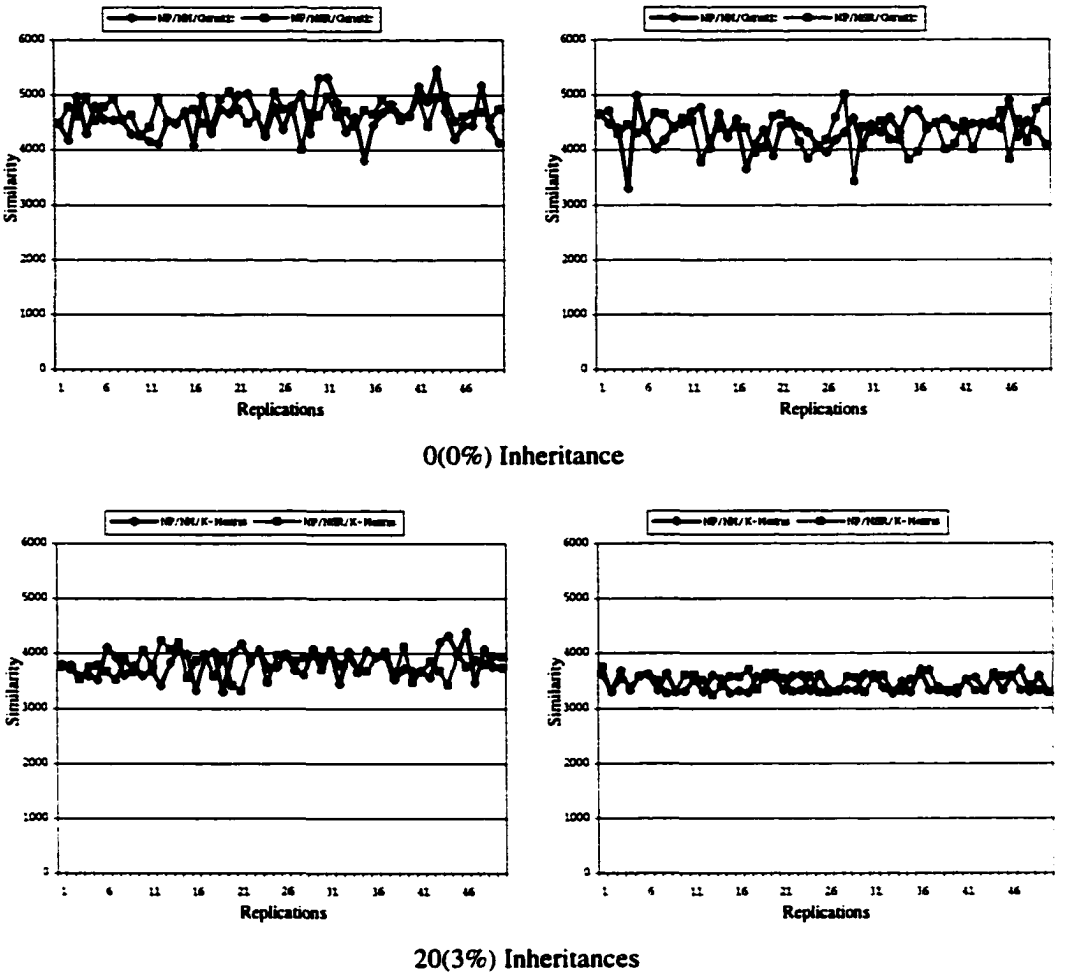
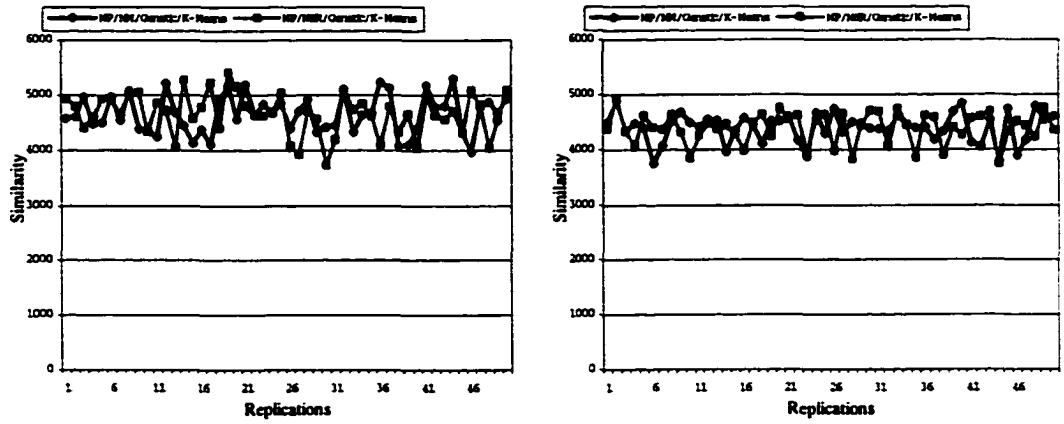
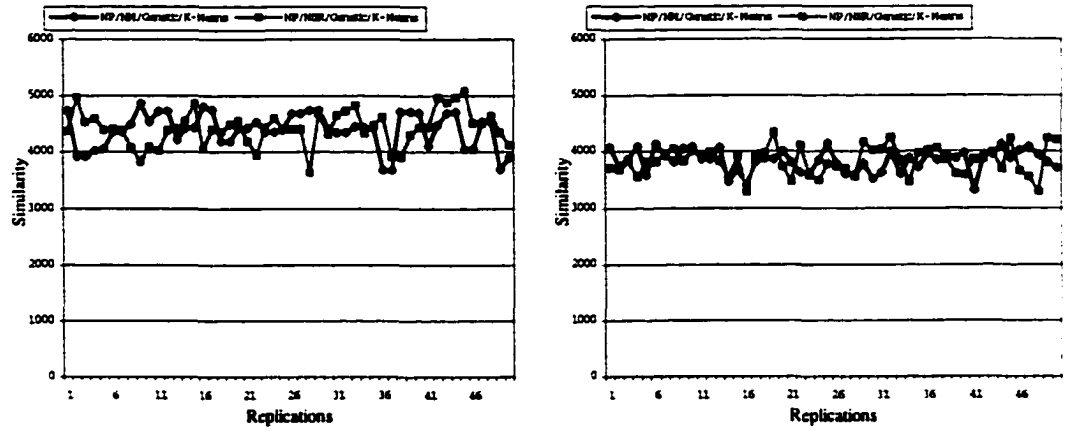



Figure 6.9: Similarity Value of Genetic when the Numbers of Instances are 30(Left) and 200(Right) for the Large Data Set

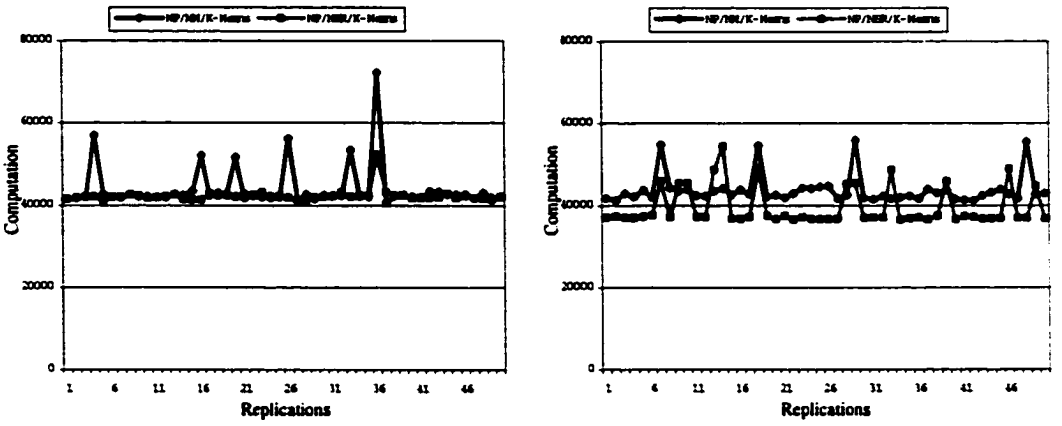



0(0%) Inheritances

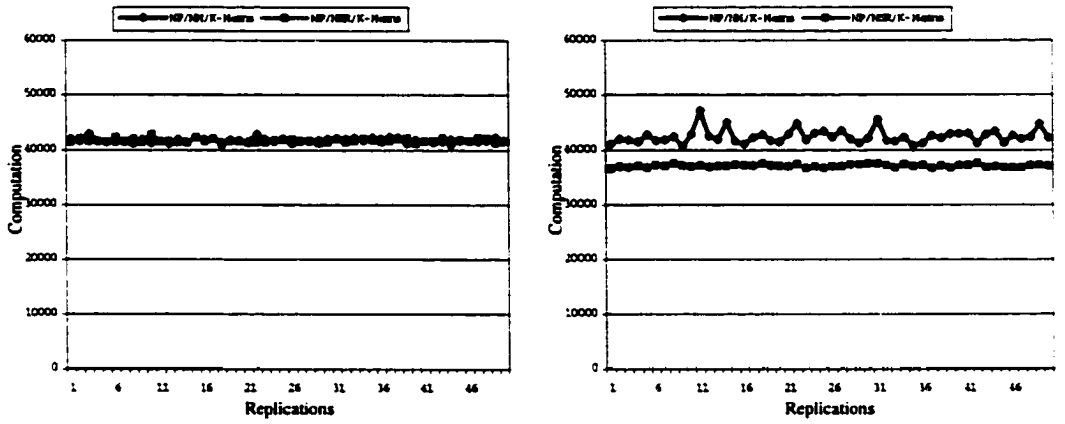



20(3%) Inheritances

Figure 6.10: Similarity Value of Genetic/K-means when the Numbers of Instances are 30(Left) and 200(Right) for the Large Data Set

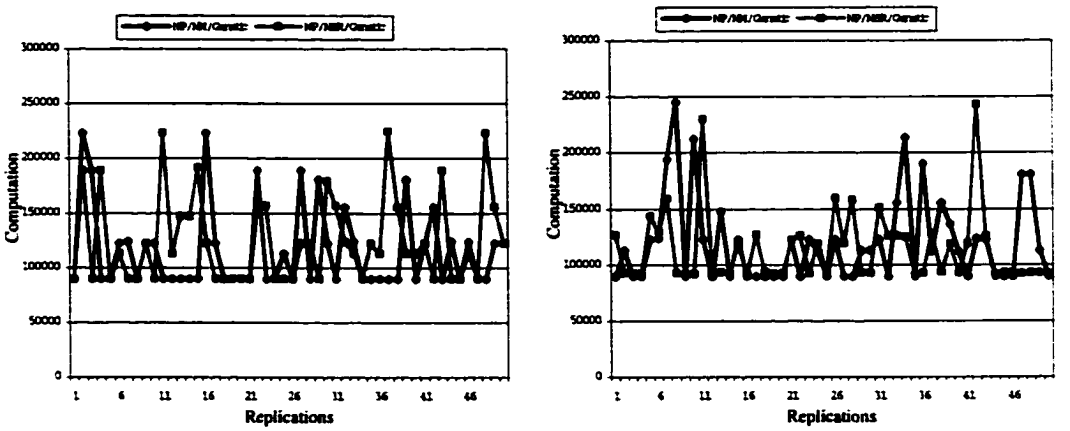



0(0%) Inheritance

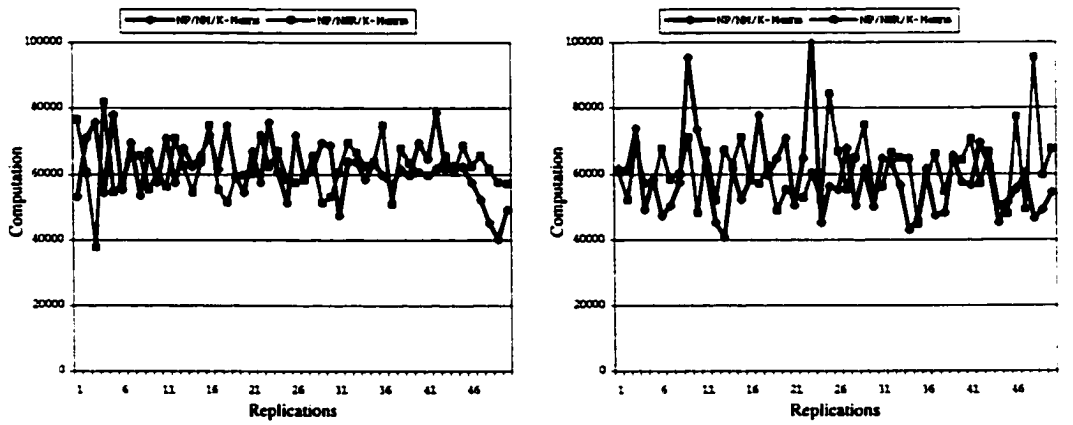



20(3%) Inheritances

Figure 6.11: Computation Time of K-means when the Numbers of Instances are 30(Left) and 200(Right) for the Large Data Set

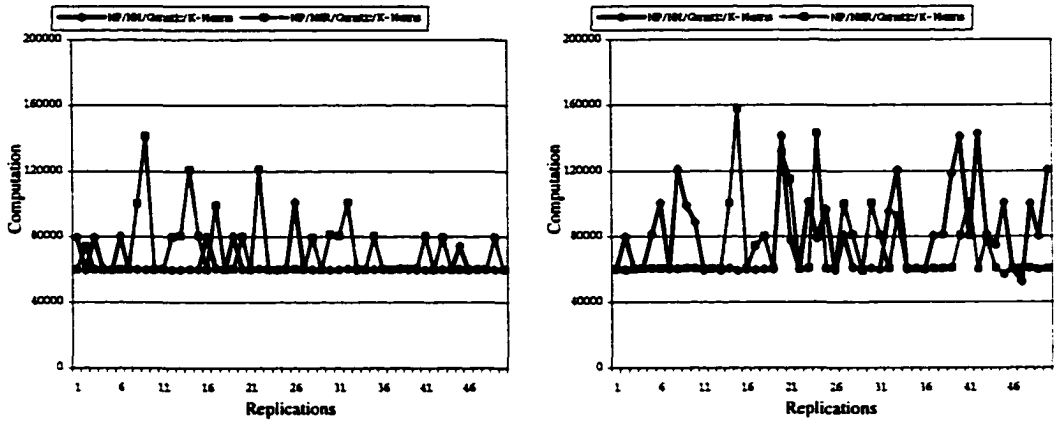



0(0%) Inheritance

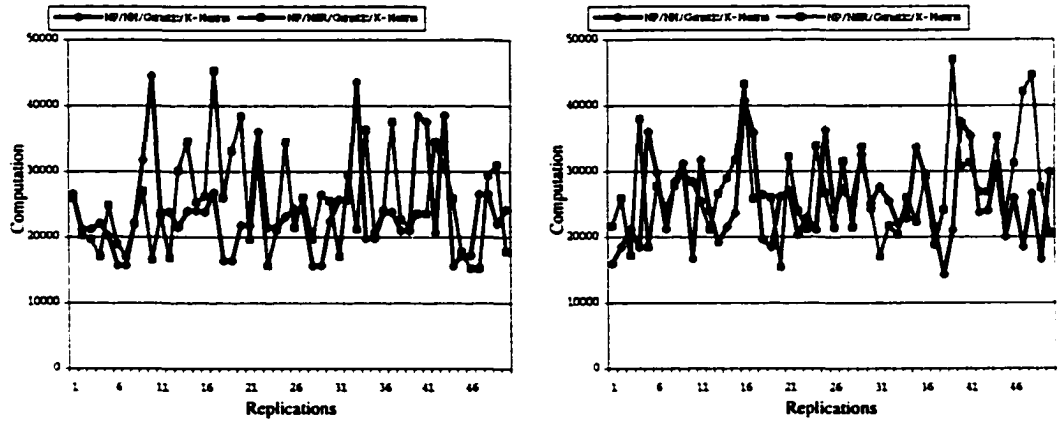



20(3%) Inheritances

Figure 6.12: Computation Time of Genetic when the Numbers of Instances are 30(Left) and 200(Right) for the Large Data Set




0(0%) Inheritance




20(3%) Inheritances

Figure 6.13: Computation Time of Genetic/K-means when the Numbers of Instances are 30(Left) and 200(Right) for the Large Data Set

## 6.3 Comparison with K-medoid methods

In this section, comparison results between suggested algorithms and several algorithms such as PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications), CLARANS (Clustering Large Applications with RANdomized Search) that employing K-medoid method are showed. PAM was developed by Kaufman and Rosseeuw (Kaufman and Rosseeuw, 1990). To find $k$ clusters, PAM's approach is to determine a representative object for each cluster. Instead of finding representative objects for the entire set like in PAM, CLARA draws a sample of the data set, applies PAM on the sample, and finds the medoids of the sample. Like CLARA, CLARANS does not check every neighbor of a node. But unlike CLARA, it does not restrict its search to particular subset. In other words, while CLARA draws a sample of nodes at the beginning of a search, CLARANS draws a sample of neighbors in each step of search. Detail algorithms can be found in Ng, R. *et al.* (1994).

Table 6.8 and Table 6.9 show the results of mean and variance of similarity value and computation time of large data set. As expected PAM gives the best similarity value but requires too much computation time. CLARA requires the smallest computation time but the similarity value is too high. Unlike these, the suggested algorithms and CLARANS give relatively small similarity value and computation time. There is a trade-off between similarity value and computation time in choosing an algorithm. However, when inheritance is used, the results are different. The suggested algorithms are better than PAM, CLARA, and CLARANS in terms of both similarity value and computation time.

Table 6.8: Comparison Results of Large Data Set when Inheritance is Not Used

| Algorithm | Similarity Value | | Computation Time | |
|---|---|---|---|---|
| NP/NM/K-means | 4259.0 | 46 | 394777 | 6668 |
| NP/NM/Genetic | 4203.7 | 40 | 404534 | 23118 |
| NP/NM/Genetic/K-means | 4444.2 | 39 | 231200 | 10487 |
| PAM | 3165.6 | 0.48 | 10255267 | 151560 |
| CLARA | 5026.3 | 54 | 174861 | 4682 |
| CLARANS | 3619.7 | 29 | 524103 | 15 |

Table 6.9: Comparison Results of Large Data Set when Inheritance is Used

| Algorithm | Similarity Value | | Computation Time | |
|---|---|---|---|---|
| NP/NM/K-means | 3292.1 | 16 | 373743 | 2692 |
| NP/NM/Genetic | 3388.3 | 17 | 186965 | 6782 |
| NP/NM/Genetic/K-means | 3811.9 | 33 | 101676 | 6001 |
| PAM | 3165.6 | 0.48 | 10255267 | 151560 |
| CLARA | 5026.3 | 54 | 174861 | 4682 |
| CLARANS | 3619.7 | 29 | 524103 | 15 |

Table 6.10 and Table 6.11 show the results of mean and variance of similarity value and computation time of small data set. As expected, PAM and CLARA give the worst computation time and similarity value each. The results are almost same as large data set. Unlike the large data set, in small data set, the suggested algorithms give always better computation time than PAM, CLARA, and CLARANS whether the inheritance is used or not. Furthermore, there is not much difference in similarity value. In conclusion, the suggested algorithms are always much more efficient than PAM, CLARA, and CLARANS in

terms of similarity value and computation time. Even if the data size is increased, the suggested algorithms with inheritance are better than k-medoids algorithms in similarity value and computation time.

Table 6.10: Comparison Results of Small Data Set when Inheritance is Not Used

| Algorithm | Similarity Value | | Computation Time | |
|---|---|---|---|---|
| NP/NM/K-means | 1302.3 | 13 | 84115 | 3166 |
| NP/NM/Genetic | 1317.1 | 12 | 128158 | 8684 |
| NP/NM/Genetic/K-means | 1267.8 | 11 | 72739 | 4528 |
| PAM | 977.8 | 2 | 1924105 | 52992 |
| CLARA | 1971.3 | 21 | 178816 | 4880 |
| CLARANS | 1151.3 | 9 | 213114 | 15 |

Table 6.11: Comparison Results of Small Data Set when Inheritance is Used

| Algorithm | Similarity Value | | Computation Time | |
|---|---|---|---|---|
| NP/NM/K-means | 1128.0 | 16 | 69111 | 960 |
| NP/NM/Genetic | 1160.3 | 5 | 31606 | 753 |
| NP/NM/Genetic/K-means | 1125.6 | 8 | 20025 | 467 |
| PAM | 977.8 | 2 | 1924105 | 52992 |
| CLARA | 1971.3 | 21 | 178816 | 4880 |
| CLARANS | 1151.3 | 9 | 213114 | 15 |

## 6.4 Conclusions

In this chapter, two things have been demonstrated: first, by varying the amount of inheritance the most effective amount of inheritance is determined for the next iteration; second, by combining pure NP with statistical sampling, shortcomings of the pure NP method are overcome.

From the numerical results, 0.5% is the best level for the small data set and 0.5~0.85% is ideal for the large data set when inheriting samples in the next iteration. Most of computation time is minimized when the inheritance level is 0.5% and is stabilized by using inheritance. Computation time showed different patterns that depend on what algorithms are used rather than the size of the data set. In addition, by combining pure NP with statistical sampling method, the similarity value could be improved over the pure NP, but this also required more computation time. Also, there is no difference in similarity value and computation time when using either Nelson-Matejcik or Rinott's sampling method. Finally, the comparison results between suggested algorithms and PAM, CLARA, and CLARANS shows suggested algorithms always better in terms of solution quality and computation time when the data size is small whether the inheritance is used or not. Even if the data size is increased, the suggested algorithms are better than k-medoids algorithms in similarity value and computation time when inheritance is used. However, without inheritance, there is a trade-off between solution quality and computation time for choosing algorithm.

# Chapter 7
# Conclusions and Future Research

In this thesis, a new optimization technique under uncertainty is presented that extends the pure Nested Partition (NP) algorithm. This method is called Nested Partition with inheritance. Furthermore, statistical selection methods and random search methods are introduced to overcome certain shortcomings of the pure NP algorithm. For the numerical evaluation, both the Monte Carlo problem and queuing problem are used to test the proposed methods. Finally, all suggested algorithms are applied to a data mining problem with noisy performance estimates.

## 7.1 NP with Inheritance

In the original NP, independent sampling is performed in each iteration such that knowledge that gained by each iteration is partially lost. The basic idea of this new algorithm, called NP with inheritance, is to retain certain good solutions by inheriting these solutions in the next iteration. Therefore, better solutions are formed in the next iteration than the previous iteration.

In addition to developing NP with inheritance (both with and without statistical selection), numerical results are presented to determine its effectiveness and suggest guidelines for the amount of inheritance. These results show that these algorithms, when employing inheritance, give a higher quality solution than without whether a statistical selection method used or not. Furthermore, the improvement increases with the number of the GA iterations used, such that, when more effort is put into local search, saving those

solutions is more critical. By applying this approach to a specific problem, namely data clustering, some empirical experience regarding the best amount of inheritance or number of points to be carried over to the next iteration has been determined. For the data clustering problem, two different problem sizes are tested. From the numerical results, 0.5% of sample points appears to be the best amount for the smaller sized problem and 0.5~0.85% for the larger sized problem for inheriting samples to the next iteration. Also, most of computation effort is minimized when the inheritance amount is 0.5% and is stabilized by using inheritance. These results indicate that while inheritance is beneficial, the level of inheritance, that is the number of points carried to the next iteration, should be low.

## 7.2 NP with Statistical Selection Method and Random Search Method

Even though the pure NP method guarantees the convergence to the optimal solution, its efficiency and convergence properties can still be improved. To address these, two extensions to the pure NP method are suggested: the statistical selection method and random search method. First, to have more intelligent sampling, four statistical selection methods are implemented, which include Nelson Matejcik's procedure, Rinott's procedure, and Dudewicz and Dalal's procedure, as well as subset selection. Second, Genetic Algorithms (GAs) are used to speed convergence and to overcome the difficulty in the backtracking stage of the NP algorithm. For the numerical evaluation, both the Monte Carlo problem and queuing problem are used.

Chapter 3 deals mainly with combining statistical methods with NP. The Nelson Matejcik's procedure is suggested for independent sampling. On the other hand, Rinott's procedure and Dudewicz and Dalal's procedure are suggested for dependent sampling.

Theoretically, employing dependent sampling using Common Random Numbers (CRNs) needs less computation than independent sampling. Therefore, it is expected that dependent sampling will save computation time. As expected, the numerical results indicate that the Nelson Matejcik's procedure needs less computational effort. With a limited computation budget, the solution quality degenerates very quickly when the desired probability of the correct solution ( $P^*$ ) is set too high.

In addition, when using the subset selection procedure, inferior regions can be deleted in advance. This procedure is combined with an independent sampling method because of the independence assumption. Numerical results show that when using a subset selection better results are obtained.

The statistical selection method shows several things. First, which algorithm is better depends on the problem that has to be solved. For example, Nelson Matejcik's procedure performs well for the Monte Carlo test problem. On the other hand, the Rinott's procedure and Dudewicz and Dalal's procedure perform better for the queuing test problem. Second, two-stage sampling can save computation effort over pure NP. Finally, low $P^*$ values are advisable because the amount of computational effort increases exponentially in $P^*$.

The pure NP is also combined with the statistical selection method, Genetic Algorithms (GA) and inheritance. There are two noteworthy results: First, the statistical selection method has better results than no statistical selection method whether inheritance is used or not. Second, the difference of the probability of correct selection between with and without inheritance is smaller with than without statistical selection.

In addition to the development of new optimization methodology, an application to data clustering is presented.

## 7.3 Application to Data Clustering

In data mining, it is necessary to effectively deal with the scalability problem because of the extensive amount of data involved. To effectively deal with large amounts of data, random sampling may be unavoidable instead of learning from every instance. Therefore, when using clustering for data mining, two things are critically important: scalability and high dimensionality. Fortunately, the NP methodology can handle both. In addition, combining the pure NP with the methods which are previously introduced, better results are obtained in data clustering. In particular, the well-known clustering algorithm, K-means, is incorporated with statistical selection, random search, and inheritance. For the numerical evaluation, two different sizes of cancer data are used.

The numerical results show that with the combined NP method, the computation effort can be greatly reduced by using a sample of instances instead of all instances in the case of the large problem. When using half of the instances instead of all instances, the computation effort is decreased without affecting solution quality. On the other hand, with too few instances solution quality becomes significantly worse at the same time as computation effort goes up. When sampling between 4.5~50% of instances, there is a trade-off between solution quality and computation effort. Finally, the combined algorithm with K-means needs relatively less computation effort than the other algorithms, especially with the larger sized problem.

Chapter 6 focuses on the evaluation of the new methodology, Nested Partitions method with inheritance, for the data clustering problem. In this chapter, several points are made. First, the best amount of inheritance for the next iteration is suggested. Second, the shortcoming of the pure NP is overcome by combining pure NP with statistical sampling.

Combining pure NP with statistical sampling gives better solutions than pure NP but needs more computation effort. Moreover, the pattern of computation effort depends on the algorithms used, not the problem size. And there is no difference in solution quality and computation time between the two different sampling methods: Nelson-Matejcik and Rinott's sampling. Finally, the comparison results between suggested algorithms and PAM, CLARA, and CLARANS shows suggested algorithms always better, especially when the data size is small.

## 7.4 Future Research

For future research several things can be considered. In this thesis, only the new algorithm has been introduced and demonstrated using numerical results. By inheriting sample points from the previous iteration to the next iteration, the Markovian property, which is used to prove the convergence for the pure NP methodology, is not applicable anymore. Therefore, rigorous convergence analysis of the new methodology still remains. Also, further testing of different kinds of data should be performed. Finally, instead of GA, other random search methods can be considered with combining with NP.

# APPENDIX

**Algorithm NP/NM/K-means**

Step 1. *Initialization*

Set $k=0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0 \quad j=1,2,\ldots,M_{\sigma(k)}$

Specify the constants $\varepsilon, \alpha, n_k$ and $n_0$. Let $g=T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer et al. (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

If $d(\sigma(k)) \neq d^*$, that is, $\sigma(k) \neq \sum_0$, partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ sub-regions $\sigma_1(k),\ldots,\sigma_{M_{\sigma(k)}}(k)$. If $d(\sigma(k)) \neq 0$, that is, $\sigma(k) \neq \Theta$, aggregate the surrounding region $\Theta \setminus \sigma(k)$ into one region $\sigma_{M_{\sigma(k)+1}}(k)$.

Stage I Sampling

Step 3. *First-Stage Sampling*

Step 3-1. Set $t=0$.

Step 3-2. *K-Means Algorithm*

Step 3-2-1. $h=1$.

Step 3-2-2. *Randomly Assign Instances to the Clusters*

Use random sampling to obtain $N$ instances and assign to the centers for each of the regions $\sigma_j(k), \quad j=1,2,\ldots,M_{\sigma(k)}+1$.

Step 3-2-3. *Calculate the Squared Error Criterion Function*

Calculate squared error criterion function

$$\hat{L}_h(\sigma_j(k)) = \sum_{i=1}^{N_C} \sum_{x \in I_j^h} | x - z_j^h |, \quad i = 1,\ldots,N_c, \quad j = 1,2,\ldots,M_{\sigma(k)} + 1.$$

$$\text{if } \hat{L}_h(\sigma_j(k)) < \hat{L}_{h-1}(\sigma_j(k)) \text{ then } X_{ij}(k) = \hat{L}_h(\sigma_j(k))$$

Step 3-2-4. If $h = n_k$ continue to Step 3-3. Otherwise let $h = h + 1$ and go back to Step 3-2-2.

Step 3-2-4. *Change the Center of each Subregion*

Change the centers of the value of the features $> d(\sigma(k))$ for each cluster of each subregion and back to Step 3-2-2.

Step 3-3. If $t = n_0$ continue to Step 4. Otherwise let $u = u + 1$ and go back to Step 3-2-2.

Stage II Sampling

Step 4. *Estimating Mean and Variance of First-Stage Sampling*

Compute the approximate sample variance of the difference of the sample means

$$S^2 = \frac{2\sum_{j=1}^{k} \sum_{t=1}^{n_0} (X_{ij} - \bar{X}_{t\cdot} - \bar{X}_{\cdot j} + \bar{X}_{\cdot\cdot})^2}{(k-1)(n_0-1)}.$$

Where $\bar{X}_{i\cdot} = \sum_{j=1}^{k} X_{ij}/k$, $\bar{X}_{\cdot j} = \sum_{u=1}^{n_0} X_{ij}/n_0$, and $\bar{X}_{\cdot\cdot} = \sum_{t=1}^{n_0} \sum_{j=1}^{k} X_{ij}/kn_0$

Step 5. *Computing Total Sample Size for Second-Stage Sampling*

Compute the total sample size for all $j$

Compute the total sample size $N(k) = \max\left\{n_0, \left\lceil \left(\frac{gS}{\varepsilon}\right)^2 \right\rceil\right\}$.

for $j = 1, 2, \ldots, M_{\sigma(k)} + 1$

Step 6. *Second-Stage Sampling*

Obtain $N(k) - n_0$ more sample estimates of the system performance for all $j$ as in Step 3 above.

Step 7. *Estimating Mean of Second-Stage Sampling*

Let the overall sample mean be the promising index for all $j \in I$,

$$\hat{I}(\sigma_j(k)) = \bar{X}_j(k) = \frac{\sum_{i=1}^{N(k)} X_{ij}(k)}{N_j(k)}$$

Step 8. *Calculating the Promising Index*

Calculate the index of the region with the smallest squared error criterion function (most promising region);

$$\hat{j}_k \in \arg\ \min \hat{I}(\sigma_j) \ \text{ for all } j \in I.$$

If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a sub-region of $\sigma(k)$, then let this sub-region be the most promising region of next iteration, that is, $\sigma(k+1) = \sigma_j(k)$, $j < M_{\sigma(k)}$ Otherwise, if the index corresponds to the surrounding region, backtrack to the region, $s(\sigma(k))$, of the current most promising region. That is, let $\sigma(k+1) = s(\sigma(k))$.

Step 9. *Checking the Stopping Rule*

If $\sigma(k+1) \in \Sigma_0$ stop and $\sigma_{opt} = \sigma(k+1)$ else $k = k+1$ and go back to Step 2.

**Algorithm NP/NM/Genetic**

Step 1. *Initialization*

Set $k=0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0$ $j=1,2,\ldots,M_{\sigma(k)}$.

Specify the constants $\varepsilon$, $\alpha$, and $n_0$. Let $g=T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer et al. (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

If $d(\sigma(k))\neq d^*$, that is, $\sigma(k)\neq\sum_0$, partition the fittest region, $\sigma(k)$, into $M_{\sigma(k)}$ sub-regions $\sigma_1(k),\ldots,\sigma_{M_{\sigma(k)}}(k)$. If $d(\sigma(k))\neq 0$, that is, $\sigma(k)\neq\Theta$, aggregate the surrounding region $\Theta\backslash\sigma(k)$ into one region $\sigma_{M_{\sigma(k)+1}}(k)$.

Step 3. *Initial Population*

If $k=0$ and $d(\sigma(k))\neq d^*$, use random sampling to obtain an initial center population $N$ strings from each of the regions $\sigma_j(k)$, $j=1,2,\ldots,M_{\sigma(k)}+1$,

$$POP_I^j=[z_I^{j1},\ z_I^{j2},\ \ldots,\ z_I^{jN}],\quad j=1,2,\ldots,M_{\sigma(k)}+1$$

else use the population $INH_{k-1}=[z^{j1},\ z^{j2},\ \ldots,\ z^{jN}]$, $j=\hat{j}_k$ as the initial population.

Part of the lacks should be fulfilled using uniform sampling.

Stage I Sampling

Step 4. *First-Stage Sampling*

Step 4-1. Set $h=1$.

Step 4-2. *GA Search*

Apply the GA to each initial population $POP_I^j$ individually, obtaining a final population for each region $\sigma_j(k)$, $j = 1,2,...,M_{\sigma(k)}+1$

$$POP_F^j = [z_F^{j1}, z_F^{j2}, \ldots, z_F^{jN}], \quad j = 1,2,\ldots,M_{\sigma(k)}+1.$$

Step 4-3. *Calculate the Squared Error Criterion Function (Overall Fitness)*

Randomly assign instances to the best of final population to calculate squared error criterion function

$$\hat{L}_h(\sigma_j(k)) = \sum_{i=1}^{N_c} \sum_{x \in I_j^*} | x - z_j^{jBest} |, \quad i = 1,\ldots,N_c, \quad j = 1,2,\ldots,M_{\sigma(k)}+1.$$

If $\hat{L}_h(\sigma_j(k)) < \hat{L}_{h-1}(\sigma_j(k))$ then $X_{ij}(k) = \hat{L}_h(\sigma_j(k))$

Step 4-4. If $h = n_0$ continue to Step 5.Otherwise let $h = h + 1$ and go back to Step 4-2.

Stage II Sampling

Step 5. *Estimating Mean and Variance of First-Stage Sampling*

See Step 4 in Algorithm NP/NM/K-means

Step 6. *Computing Total Sample Size for Second-Stage Sampling*

See Step 5 in Algorithm NP/NM/K-means

Step 7. *Second-Stage Sampling*

Obtain $N_j(k) - n_0$ more sample estimates of the system performance for all $j$ as in Step 4 above.

Step 8. *Estimating Mean of Second-Stage Sampling*

See Step 7 in Algorithm NP/NM/K-means

Step 9. *Calculating the Promising Index*

See Step 8 in Algorithm NP/NM/K-means

Step10.*Checking the Stopping Rule*

See Step 9 in Algorithm NP/NM/K-means

**Algorithm NP/NM/K-means/Genetic**

Step 1. *Initialization*

Set $k=0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0 \quad j=1,2,\ldots,M_{\sigma(k)}$

Specify the constants $\varepsilon, \alpha, n_k$ and $n_0$. Let $g=T^{(\alpha)}_{k-1,(k-1)(n_0-1),0.5}$, an equicoordinate critical point of the equicorrelated multivariate central $t$-distribution; the constant can be found in Hochberg and Tamhane (1987), Appendix 3, Table 4; Bechhofer et al, (1995); or by using the FORTRAN program AS251 of Dunnet (1989).

Step 2. *Partitioning*

See Step 4 in Algorithm NP/NM/Genetic

Step 3. *Initial Population*

See Step 4 in Algorithm NP/NM/Genetic

Step 4. *GA Search*

Apply the GA to each initial population $POP_I^j$ individually, obtaining a final population for each region $\sigma_j(k), \quad j=1,2,\ldots,M_{\sigma(k)}+1$

$$POP_F^j=[z_F^{j1}, z_F^{j2}, \ldots, z_F^{jN}], \quad j=1,2,\ldots,M_{\sigma(k)}+1$$

Step 5. *First-Stage Sampling*

Step 5-1. Set $t=0$.

Step 5-2. *K-means Algorithm*

See Step 3-2 in Algorithm NP/NM/K-means

Step 5-3. See Step 3-3 in Algorithm NP/NM/K-means

Stage II Sampling

Step 6. *Estimating Mean and Variance of First-Stage Sampling*

See Step 4 in Algorithm NP/NM/K-means

Step 7. *Computing Total Sample Size for Second-Stage Sampling*

See Step 5 in Algorithm NP/NM/K-means

Step 8. *Second-Stage Sampling*

See Step 6 in Algorithm NP/NM/K-means

Step 9. *Estimating Mean of Second-Stage Sampling*

See Step 7 in Algorithm NP/NM/K-means

Step 10. *Calculating the Promising Index*

See Step 8 in Algorithm NP/NM/K-means

Step 11. *Checking the Stopping Rule*

See Step 9 in Algorithm NP/NM/K-means

**Algorithm NP/NSR/K-means**

Step 1. *Initialization*

Set $k=0$ and $\sigma(k)=\Theta$. Specify the value of $z_j^0 \quad j=1,2,\ldots,M_{\sigma(k)}$

Specify the overall desired probability $P^*$ of correct selection and indifference zone $\varepsilon$, termination parameter of K-means algorithm $n_k$, the common initial sample size $n_0 \geq 2$, the number of sub-regions $M$. Determine $h_2$ for Rinott's integral. $h_2$ are constants which are determined by $n_0$, the minimum probability $P^*$ of correct selection, and $M$ (See the tables in Bechhofer et al., 1995).

Step 2. *Partitioning*

See Step 2 in Algorithm NP/NM/K-means

Stage I Sampling

Step 3. *First-Stage Sampling*

See Step 3 in Algorithm NP/NM/K-means

Stage II Sampling

Step 4. *Estimating Mean and Variance of First-Stage Sampling*

Calculate first-stage sample means and variances

$$\overline{X}_j^{(1)}(k)=\frac{1}{n_0}\sum_{i=1}^{n_0}X_{ij}(k),$$

and

$$S_j^2=\frac{\sum_{i=1}^{n_0}[X_{ij}(k)-\overline{X}_j^{(1)}(k)]^2}{n_0-1},$$

for $j=1,2,\ldots,M+1$.

Step 5. *Computing Total Sample Size for Second-Stage Sampling*

Compute the total sample size for all $j \in I$

$$N_j = \max\left\{n_0 + 1, \left\lceil \frac{h_2^2 S_j^2}{\varepsilon^2} \right\rceil\right\}$$

where $\varepsilon$ is the indifference zone and $h_2$ is a constant determined by $n_0$ and the minimum probability $P^*$ of correct selection.

Step 6. *Second-Stage Sampling*

See Step 6 in Algorithm NP/NM/K-means

Step 7. *Estimating Mean of Second-Stage Sampling*

See Step 7 in Algorithm NP/NM/K-means

Step 8. *Calculating the Promising Index*

See Step 8 in Algorithm NP/NM/K-means

Step 9. *Checking the Stopping Rule*

See Step 9 in Algorithm NP/NM/K-means

# BIBLOGRAPHY

Aarts, E. and Korst, J. 1989. "Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing," Wiley-Interscience series in discrete mathematics and optimization. John Wiley and Sons, Inc., New York, NY.

Agrawal, R. and Srikant, R. 1994. "Fast algorithms for mining association rules," in *Proceeding of the 20th Int'l Conf. on Very Large Databases*, Santiago, Chile.

Al-Sultan, K.S. 1995. "A tabu search approach to the clustering problem," *Pattern recognition*, **28**(9), 1443-1451.

Al-Sultan, K.S and Khan, M.M. 1996. "Computational experience on four algorithms for the hard clustering problem," *Pattern recognition Letters*, **17**, 295-308.

Andradóttir, S. 1989. "A review of Simulation Optimization Techniques," in *Proceedings of the Winter Simulation Conference*, 151-158.

Andradóttir, S. 1990. "A new algorithm for stochastic optimization," in *Proceedings of the Winter Simulation Conference*, 364-366.

Andradóttir, S. 1995. "A method for discrete stochastic optimization," *Management Science*, **41**, 1946-1961.

Andradóttir, S. 1996. "A global search method for discrete stochastic optimization," *SIAM J. Optimization*, **6**, 513-530.

Andradóttir, S. 1998. "Simulation Optimization Techniques," in J. Banks (ed.), *Handbook of Simulation*, 307-333.

Azadivar, F., Shu, J., and Ahmad, M. 1996. "Simulation optimization in strategic location of semi-finished products in a pull-type production system," in *Proceedings of the Winter Simulation Conference*, 1123-1128.

Bäck, T. 1994. "Selective pressure in evolutionary algorithms: a characterization of selection mechanisms," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, 57-62, IEEE Press, Piscataway NJ.

Babu, G.P. and Murty, M.N. 1993. "A near optimal initial seed value selection in K-means algorithm using a genetic algorithm," *Pattern Recognition Letters*, **14**(10) 763-769.

Ball, G.H. and Hall, D.J. 1964. "Some fundamental concepts and synthesis procedure for pattern recognition preprocessors," In *International Conference on Microwaves, Circuit Theory, and Information theory.*

Banks, J. 1998. *Handbook of Simulation*, Wiley-Interscience, New York.

Banks, J. 1999. "Introduction to simulation," in *Proceedings of the Winter Simulation Conference*, 7-13.

Bechhofer, R.E. 1954. "A single-sample multiple decision procedure for ranking means of normal populations with known variances," *Annals of Mathematical Statistics*, **25**, 16-39.

Bechhofer, R.E., Santner, T., and Goldsman, D. 1995. *Design and analysis of experiments for statistical selection, screening, and multiple comparisons*, Wiley-Interscience, New York.

Bechhofer, R.E., Kiefer, J., and Soble, M. 1968. *Sequential Identification and Ranking Procedures*, The University of Chicago Press, Chicago, Illinois.

Bnnett, K.P. 1994. "Global Tree Optimization: A Non-greedy Decision Tree Algorithm," *Computing Science and Statistics*, **26**, 156-160.

Berger, J.O. 1988. "Bayesian approach to ranking and selection of related means with alternatives to analysis-of-variance methodology," *Journal of the American Statistical Association*, **83**, 402, 364-373.

Blake, C.L. and Merz, C.J. 1998. *UCI Repository of machine learning database* [http://www.ics.uci.edu/mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine, CA.

Boley, D., Gini, M., Gross, R., Han, E.H., Hastings, K., Karypis, G., Kumar. V., Mobasher, B., and Moore, J. 1999. "Document categorization and query generation on the world wide web using WebACE," AI Review.

Bös, S. 1996. "A realizable learning task which exhibits overfitting," *Advances in Neural Information Processing Systems*, **8**, 218-224, MIT Press, Cambridge, MA.

Bradley, P., Fayyad, U., and Mangasarian, O. 1998. "*Data Mining: Overview and Optimization Opportunities*," *INFORMS: Journal of Computing*.

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. 1984 *Classification ana Regression Trees*, Monterey, CA: Wadsworth and Brooks/ Cole.

Byers, S. and Adrian, E. R. 1998. "Nearest neighbor clutter removal for estimating features in spatial point processes," *Journal of the American Statistical Association*, **93**, 577-584.

Carson, Y., and Maria, A. 1997. "Simulation Optimization: Methods and Applications," in *Proceedings of the Winter Simulation Conference*, 118-126.

Chen, C.-H., E.Yücesan, Y.Yuan, H.-C.Chen, and L.Dai. 1998. "Computing budget allocation for simulation experiments with different system structures," in *Proceedings of the Winter Simulation Conference*, 735-741.

Chen, C.-H., Chen, H.-C, and Dai, L., 1996. "A gradient approach for smartly allocating computing budget for discrete event simulation," in *Proceedings of the Winter Simulation Conference*, 398-405.

Chick, S.E. and Inoue, K. 1998. "A Sequential allocation procedures that reduce risk for multiple comparisons," in *Proceedings of the Winter Simulation Conference*, 669-676.

Chick, S.E. and Inoue, K. 1999. "New two-stage and sequential procedures for selecting the best simulated systems," Technical report, University of Michigan, Ann Arbor, Dept. of Industrial & Operations Engineering.

Clark, G.M. and Yang. W.N. 1986. "A Bonferroni selection procedure when using common random numbers with unknown variances," in *Proceedings of the Winter Simulation Conference*, 372-375.

David, L. 1991. *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York.

Deogun, J. S., Raghavan, V. V., Sarkar, A., and Server H. 1997. "Data mining: Research trends, challenges, and applications," in *Roughs Sets and Data Mining: Analysis of Imprecise Data* (Boston, MA), T. Y. Lin and N. Cercone, Eds., Kluwer Academic Publishers, 9–45.

Donohue, J.M., Houck, E.C., and Myers, R.H. 1990. "Some optimal simulation designs for estimating quadratic response surface functions," in *Proceedings of the Winter Simulation Conference*, 337-343.

Duda, R. and Hart, P. 1973. *Pattern Classification and Scene Analysis*, John Wiley and Sons.

Dudewicz, E.Z. and Taneja, V.S. 1978. "Multivariate ranking and selection without reduction to a univariate problem," in *Proceedings of the Winter Simulation Conference*, 207-210.

Dudewicz, E.J. and Dalal, S.R. 1975. "Allocation of Observations in Ranking and Selection with Unequal Variances," *Sankhya*, **37B**, 28-78.

Dudewicz, E.J., and Zaino. N.A. 1976 "Allowance for correlation in setting simulation run length via ranking and selection procedures," Technical Report. Department of Statistics, Stanford University

Dunnett, C. W. 1989. "Multivariate normal probability integrals with product correlation structure," *Applied Statistics*, **38**, 564-579. Correction: **42**, 709.

Faccenda J.F., and Tenga, R.F. 1992. "A combined simulation/optimization approach to process plant design," in *Proceedings of the Winter Simulation Conference*, 1256-1261.

Fleischer, M. 1995. "Simulated annealing: past, present, and future," in *Proceedings of the Winter Simulation Conference*, 155-161.

Fraley, C., and Raftery, A. 1998. "How many clusters? Which clustering method? Answers via Model-Based Cluster Analysis," Technical Report No. 329, Dept. of Statistics, University of Washington.

Frawely, W. J. Piatesky-Shapiro, G. and Matheus, C. J. 1991, "*Knowledge discovery in databases: an overview*," in Knowledge Discovery in Databases, AAAI/MIT Press, 1-27.

Fu, M.C., and Healy, K. 1992. "Simulation Optimization of (s,S) Inventory Systems," in *Proceedings of the Winter Simulation Conference*, 506-514.

Fu, M.C. 1994. "Stochastic Optimization using Simulation: A review," *Annals of Operations Research*, **53**, 506-514.

Gen, M., and R, Cheng. 2000. *Genetic algorithms and engineering optimization*, Wiley, New York .

Glasserman, P. and Yao, D.D. 1992. "Some Guidelines and Guarantees for Common Random Numbers," *Management Science*, **38**, 884-908,

Glover, F. 1977. "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, **8**, 156-166.

Glynn, P.W. 1989. "Optimization of Stochastic Systems Via Simulation," in *Proceedings of the Winter Simulation Conference*, 90-105.

Goldberg, D., Korb, B., and Deb, K. 1989. "Messy genetic algorithms: motivation, analysis. and first results," *Complex Systems*, **3**, 493-530.

Goldsman, D., and Nelson. B.L. 1994. "Ranking, selection, and multiple comparisons in computer simulation," in *Proceedings of the Winter Simulation Conference*, 192-199.

Goldsman, D., and Nelson, B.L. 1998. "Statistical screening, selection, and multiple comparison procedures in computer simulation," in *Proceedings of the Winter Simulation Conference*, 159-166.

Goldsman, D., Nelson, B.L., and Schmeiser, B. 1997. "Methods for selecting the best system," in *Proceedings of the Winter Simulation Conference*, 177-186.

Gong, D., Gen, M., Yamazaki, G., and Xu, W. 1997. "Hybrid evolutionary method for capacitated location-allocation problem," *Engineering design and Optimization*, **3**, 179-190.

Gupta, S.S. 1965. "On some multiple decision rules," *Technometrics*, **6**, 225-245.

Gupta, S. S. and Miescke. K. J. 1996. "Bayesian look ahead one-stage sampling allocations for selecting the best population," *Journal of Statistical Planning and Inference*, **54**, 229–244.

Gurkan, G., Ozge, A.Y., and Robinson, S.M. 1994. "Sample-path optimization in simulation," in *Proceedings of the Winter Simulation Conference*, 247-254.

Han, J. and Kamber, M. 2001. *Data mining: Concepts and Techniques*, Morgan Kaufmann.

Harris, N., Hunter, L, and States, D. 1992. "Mega-classification: Discovering motifs in massive datastreams," in *Proceedings of the Tenth International Conference on Artificial Intelligence* (AAAI).

Hertz, J. Krogh, A., and Palmer, R.D. 1991. *Introduction to the theory of neural computation Reading*, Addison-Wesley.

Hochberg, Y., and Tamhane. A. C. 1987. *Multiple Comparison Procedures*, NewYork: John Wiley.

Hofmann, T. and Buhmann, J. 1997. "Pairwise data clustering by deterministic annealing," *IEEE Trans. Pattern Anal. Mach. Intell.*, **1**, 1-14.

Holland, J.H. 1975. *Adaptation in natural and artificial systems*, MIT Press.

Hill, R.R. 1999. "A monte carlo study of genetic algorithm initial population generation methods," in *Proceedings of the Winter Simulation Conference*, 543-547.

Inoue, K. and Chick, S.E., and Chen, C.-H. 1999. "An Empirical Evaluation of Several Methods to Select the Best System," *ACM Transitions on Modeling and Computer Simulation*, **9**, 381 – 407.

Jacobson, S.H. and Schruben. L.W. 1989. "A review of techniques for simulation optimization," *Operations Research Letters*, **8**, 1-9.

Jain, A. K. and Dubes, R.C. 1988. *Algorithms for clustering data*, Prentice hall.

Jones, D.R. and Beltramo, M.A. 1991. "Solving partitioning problems with genetic algorithms," in *Proceedings of the fourth International Conference on Genetic Algorithms*, 442-449.

Karypis, G. and Kumar, V. 1998. "METIS 4.0: Unstructured graph partitioning and sparse matrix ordering system," Technical Report, Department of computer Science, University of Minnesota.

Karypis, G. Han, E. H., and Kumar, V. 1999. "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling," *COMPUTER*, **32**, 68-75.

Kaufman, L. and Rousseeuw, P.J. 1990. *Finding Groups in Data: an Introduction to Cluster analysis*, Academic Press.

Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi. M.P. 1983. "Optimization by Simulated Annealing," *Science*, 220, 4598, 671-680.

Koeing, L.W. and A.M. Law. 1985. "A procedure for selecting a subset of size m containing the l best of l independent normal populations, with applications to simulation," *Communications in Statistics*, **B14(3)**, 719-734.

Law, A.M. and Kelton, W.D. 1991. *Simulation Modeling & Analysis(2nd ed)*, McGrawHill, Inc, New York.

L'Ecuyer, P. 1991. "An Overview of Derivative Estimation," in *Proceedings of the Winter Simulation Conference*, 207-217.

L'Ecuyer, P. and Perron, G. 1994. "On the Convergence Rates of IPA and FDC Derivative Estimators for Finite-Horizon Stochastic Simulations," *Operations Research*, **42**(4), 643-656.

Lee, Y.H., Park, K.J., and Kim, T.K. 1999. "An approach for finding discrete variable design alternatives using a simulation optimization method," in *Proceedings of the Winter Simulation Conference*, 678 - 685.

Leung, Y.T., and Suri, R. 1990. "Finite - Time Behavior of two simulation optimization algorithms," in *Proceedings of the Winter Simulation Conference*, 372-376.

Lu, S.Y. and Fu, K.S. 1978. "A sentence-to-sentence clustering procedure for pattern analysis," *IEEE Trans. Syst. Man Cybern*, **8**, 381-389.

Man, K.F. 1999. *Genetic Algorithms: Concepts and Designs*, Springer, New York.

Mitchell, T.M. 1997. *Machine Learning*, McGrawHill, Inc.

Neal, R. M. Hinton, G. E. 1999. "A view of the EM algorithm that justifies incremental, sparse, and other variants," *Learning in Graphical Models*, 355-368, Cambridge, MA: MIT Press.

Nelson, B.L., and Matejcik, F.J. 1995. "Using common random numbers for indifference-zone selection and multiple comparisons in simulation," *Management Science*, **41**, 1935-1945.

Nelson, B.L. 1993. "Robust multiple comparisons under common random numbers," *ACM Transactions on Modeling and Computer Simulation*, **3**, 225-243.

Ng, R. and Han, J. 1994. "*Efficient and effective clustering methods for spatial data mining*," In *Proceedings of VLDB Conference*, 144-155.

Ólafsson, S. 1999. "Iterative ranking-and-selection for large-scale optimization," in *Proceedings of the Winter Simulation Conference*, 479 -485.

Ólafsson, S., and Gopinath, N. 2000. "Optimal selection probability in the two-stage nested partitions method for simulation-based optimization," in *Proceedings of the Winter Simulation Conference*, 736-742.

Paulson, E. 1964. "A sequential procedure for selecting the population with the largest mean from k normal populations," *Annals of Mathematical Statistics*, **35**, 174-180.

Piatetsky-Shapiro, G. and Frawley, W.J. 1991. *Knowledge Discovery in Databases*, AAAI/MIT Press.

Raghavan, V. V. and Birchand, K. 1979. "A clustering strategy based on a formalism of the reproductive process in a natural system," in *Proceedings of the Second International Conference on Information Storage and Retrieval*, 10-22.

Rinott, Y. 1978. "On two-stage selection procedures and related probability-in-equalities," *Communications in Statistics*, A1, 799-811.

Robinson, H., and Monro, S. 1951. "A stochastic approximation method," *Annals of Mathematical Statistics*, **22**, 400-407.

Rose, K., Gurewitz, E., and Fox, G.C. 1993. "Deterministic annealing approach to constrained clustering," *IEEE Trans. Pattern Anal. Mach. Intell*, **15**, 785-794.

Rubinstein, R..Y. and Shapiro, A. 1993, *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization*, John Wiley & Sons Inc.

Sanchez, S.M. 1997. "It is a far, far better mean I find...," in *Proceedings of the Winter Simulation Conference*, 31-38.

Shapiro, A. 1996. "Simulation based optimization," in *Proceedings of the Winter Simulation Conference*, 332-336.

Shavlik, J.W. and Dietterich, T.G. 1990. *Readings in Machine Learning*. Morgan Kaufmann.

Shi, L., Ólafsson, S., and Chen, Q. 1999. "A new hybrid optimization algorithm," *Computers & Industrial Engineering*, **36**, 409-426.

Shi, L., and Ólafsson, S. 2000a. "Nested Partitions Method for Global Optimization," *Operations Research*, **48**, 390-407.

Shi, L., and Ólafsson, S. 2000b. "Nested Partitions Method for Stochastic Optimization," *Methodology and computing in Applied Probability*, **2**, 271-291.

Sullivan, D.W., and Wilson, J.R. 1989. "Restricted subset selection procedures for simulation," *Operations Research*, 37, 52-71.

Sudipto, Guha, Rajeev, Rastogi, and Kyuseok Shim. 1998. "CURE: An efficient clustering algorithm for large databases," in *Proceeding of 1998 ACM-SIGMOD Int. Conference on Management of Data.*

Sudipto, Guha, Rajeev, Rastogi, and Kyuseok, Shim. 1999. "ROCK: a robust clustering algorithm for categorical attributes," in *Proceedings of the15$^{th}$ Int'l Conference on Data Eng.*, 512-521.

Swisher, J.R., and Jacobson, S.H. 1999. "A survey of ranking, selection, and multiple comparison procedures for discrete-event simulation," in *Proceedings of the Winter Simulation Conference*, 492-501.

Wen, M.J. and H.J. Chen. 1994. "Single-stage multiple comparison procedures under heteroscedasticity," *American Journal of Mathematical and Management Sciences*, **14(1,2)**, 1-48.

# ACKNOWLEDGEMENTS

I would like to especially thank to Dr. Olafsson, my major professor and Sameh, my friend for their loving guidance and financial assistance during the writing of this work.