5-2011

Birnbaum Importance Patterns and Their Applications in the Component Assignment Problem

Qingzhu Yao
qyao@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Qingzhu Yao entitled "Birnbaum Importance Patterns and Their Applications in the Component Assignment Problem." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

Way Kuo, Xiaoyan Zhu, Major Professor

We have read this dissertation and recommend its acceptance:

Xueping Li, Frank M. Guess

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Birnbaum Importance Patterns and Their Applications in the Component Assignment Problem

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Qingzhu Yao

May 2011

To my parents and wife

Acknowledgements

I would like to express my gratitude to my co-advisor Dr. Way Kuo for his continued help and support. I have been very fortunate to have him guide me through my PhD program and point me in the right direction. I am deeply grateful to my co-advisor Dr. Xiaoyan Zhu for holding me to a high research standard. Dr. Zhu has always been there to help me out, encourage me, and give me advices. Special thanks go to Dr. Xueping Li and Dr. Frank Guess for their valuable advices on the research. I have learned much from them. It has been a tremendous pleasure and a privilege to be with such excellent professors.

I am also thankful to Dr. Tao Yuan, Dr. Rui Wan, Mr. Chia-Han Yang, Dr. Yuerong Chen, Dr. Dengfeng Yang, Dr. Laigang Song, and Mr. Qi Yuan. Thanks to all of you for your time and encouragement, as well as your willingness to share. You have made my adventure in Industrial Engineering go as smoothly as possible. I would also like to thank Ms. Xinyue Tang for reading my dissertation and for her comments.

This research was partially supported by a National Science Foundation Project # CMMI-0825908.

Abstract

The Birnbaum importance (BI) is a well-known measure that evaluates the relative contribution of components to system reliability. It has been successfully applied to tackling some reliability problems. This dissertation investigates two topics related to the BI in the reliability field: the patterns of component BIs and the BI-based heuristics and meta-heuristics for solving the component assignment problem (CAP).

There exist certain patterns of component BIs (i.e., the relative order of the BI values to the individual components) for linear consecutive-k-out-of-n ($Lin/Con/k/n$) systems when all components have the same reliability p. This study summarizes and annotates the existing BI patterns for $Lin/Con/k/n$ systems, proves new BI patterns conditioned on the value of p, disproves some patterns that were conjectured or claimed in the literature, and makes new conjectures based on comprehensive computational tests and analysis. More importantly, this study defines a concept of segment in $Lin/Con/k/n$ systems for analyzing the BI patterns, and investigates the relationship between the BI and the common component reliability p and the relationship between the BI and the system size n. One can then use these relationships to further understand the proved, disproved, and conjectured BI patterns.

The CAP is to find the optimal assignment of n available components to n positions in a system such that the system reliability is maximized. The ordering of component BIs has been successfully used to design heuristics for the CAP. This study proposes five new BI-based heuristics and discusses their corresponding

properties. Based on comprehensive numerical experiments, a BI-based two-stage approach (BITA) is proposed for solving the CAP with each stage using different BI-based heuristics. The two-stage approach is much more efficient and capable to generate solutions of higher quality than the GAMS/CoinBonmin solver and a randomization method.

This dissertation then presents a meta-heuristic, i.e., a BI-based genetic local search (BIGLS) algorithm, for the CAP in which a BI-based local search is embedded into the genetic algorithm. Comprehensive numerical experiments show the robustness and effectiveness of the BIGLS algorithm and especially its advantages over the BITA in terms of solution quality.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The importance measures of components evaluate the contribution of a component to system performance (e.g., system reliability). Various importance measures have been proposed for coherent systems (Freixas and Pons, 2008; Kuo et al., 2001, 1990; Kuo and Zuo, 2002; Zhu and Kuo, 2008) and have been successfully applied in tackling some reliability optimization problems. The Birnbaum importance (BI) (Birnbaum, 1969) has been used to solve the component assignment problem (CAP) (Kontoleon, 1979; Lin and Kuo, 2002; Zuo and Kuo, 1990; Zuo and Shen, 1992). The improvement potential/risk achievement, criticality, Fussell-Vesely, risk achievement worth, risk reduction worth, and differential importance measures are widely used for probabilistic safety assessment and other reliability and risk analysis (Aven and Nokland, 2010; Borgonovo and Apostolakis, 2001; van der Borst and Schoonakker, 2001). The redundancy (Boland et al., 1991) and yield (Xie and Lai, 1996; Xie and Shen, 1989) importance measures have been applied to the parallel and series redundancy allocation problems. The joint (Armstrong, 1995) and total order (Borgonovo, 2010) importance measures provide insights to the interaction of components in determining system reliability. In addition, Beeson and Andrews (2003), Lu and Jiang (2007), and Borgonovo (2010) extended some importance

measures (e.g., the BI, Fussell-Vesely, joint, and differential importance measures) to noncoherent systems.

The BI is one of the most widely investigated importance measures since it was first proposed by Birnbaum (1969). In a coherent system of n components with independent failures, the BI of component i is defined as the probability that component i becomes critical to system failure, and can be calculated as (Birnbaum, 1969)

$$I_n(i) = \frac{\partial R(p_1, p_2, \ldots, p_n)}{\partial p_i},$$

where p_i denotes the reliability of the component in position i for $i = 1, 2, \ldots, n$ and $R(p_1, p_2, \ldots, p_n)$ the system reliability. The BI measures the relative importance and contribution of a component to the system reliability by the rate at which the system reliability improves as the component reliability improves. Note that the BI is defined based on the change of the system reliability caused by the uniform changes of component reliabilities (i.e., all component reliabilities are changed by the same small amount) (Borgonovo and Apostolakis, 2001). This study focuses on the BI and its applications in two aspects: the patterns of component BIs in linear consecutive-k-out-of-n ($Lin/Con/k/n$) systems and the BI-based heuristics and meta-heuristics for solving the component assignment problem (CAP).

1.1 BI Patterns in $Lin/Con/k/n$ systems

A $Lin/Con/k/n$:F (G) system is a linear sequence of n components such that the system fails (works) if and only if at least k consecutive components fail (work). A $Lin/Con/k/n$ system can represent oil pipeline networks Kuo et al. (2001), computer ring networks Hwang (1989), street lights and microwave towers Chao and Lin (1984), parking spaces Kuo et al. (1990), quality control lot acceptance sampling Shen

and Zuo (1994), etc. Eryilmaz (2010) presented a review of recent developments in reliability of consecutive k-out-of-n systems[†].

To determine the Birnbaum importance of components, we need to know not only the structure of the coherent system but also the component reliabilities involved. In the early stage of system design, the component reliabilities may be unknown. Thus, in order to investigate the BI patterns (i.e., the relative order of the BI values to the individual components) in $Lin/Con/k/n$ systems, we assume that

$$0 < p_1 = p_2 = \ldots = p_n = p < 1. \tag{1.1}$$

Under assumption (1.1), the BI is also referred to as the B-importance (Lin and Kuo, 2002), and the common component reliability p is an arbitrary parameter.

The BI patterns may be different for distinct values of p. Chang et al. (2002), Chang and Hwang (2002), and Lin et al. (1999) investigated the BI patterns in $Lin/Con/k/n$ systems for different values or ranges of p. Recall from Chang et al. (2002) that component i is said to be more *uniformly* B-important than component j (denoted by $i \succ_u j$) if $I_n(i) > I_n(j)$ for all $0 < p < 1$, more *half-line* B-important (denoted by $i \succ_h j$) if $I_n(i) > I_n(j)$ for all $\frac{1}{2} \leq p < 1$ (this condition can be expected in most cases), and more *combinatorially* B-important (denoted by $i \succ_c j$) if $I_n(i) > I_n(j)$ for $p = \frac{1}{2}$. A BI pattern in the uniform case is stronger than the half-line case, which, in turn, is stronger than the combinatorial case, i.e., $i \succ_u j \Rightarrow i \succ_h j \Rightarrow i \succ_c j$. Components i and j are said to be equally B-important in the uniform case (denoted by $i =_u j$) if $I_n(i) = I_n(j)$ for all $0 < p < 1$.

[†]Consecutive k-out-of-n systems include $Lin/Con/k/n$ and circular consecutive k-out-of-n ($Cir/Con/k/n$) systems which are same as $Lin/Con/k/n$ systems except all components are arranged in a circle rather than a line (Kuo and Zuo, 2002).

1.2 The CAP

Given a coherent system of n positions and n available components with reliabilities $0 < \hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n < 1$, different assignments of n components to n positions result in different values of the system reliability. Assuming that each position must be assigned one and only one component, let π_i be the index of the component assigned to position i, $i = 1, 2, \ldots, n$. Then vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ is a permutation of component indices $1, 2, \ldots, n$, representing an arrangement of the n components to the n positions. The CAP is to find an optimal permutation $\boldsymbol{\pi}^*$ under which the system reliability is maximized. The CAP is a NP-hard problem due to its combinatorial nature. When the optimal arrangement depends only on the ordering of component reliabilities, it is called the *invariant optimal arrangement* (Lin and Kuo, 2002) since it exists as such regardless of the magnitude of the component reliabilities.

The CAP has the applications in the case where a given set of components are functionally exchangeable and need to be assigned to different positions in a system. By optimally assigning these components, the resulting system reliability is maximized. For example, in the oil pipeline pumping system that is a $Lin/Con/k/n$:F system (Zuo and Shen, 1992), there are n pump stations along an oil pipeline, and each pump station is used to pump oil to the next k pump stations. The pumps in this pumping system are functionally exchangeable, and thus, given n pumps with different reliabilities, the reliability of this pumping system is determined by the assignment of the n pumps to the n pump stations. Intuitively, the high reliable pump should be assigned to the important position in the system. Other examples in two-terminal networks include wireless and hard-wired telecommunication networks, computer networks, and electric power distribution networks (Gebre and Ramirez-Marques, 2007; Kontoleon, 1979). In a two-terminal network, the links are regarded as the components and have different reliabilities due to the different supporting resources behind the links. Optimally arranging the components (i.e., allocating the

different supporting resources) can maximize the reliability of the network given a fixed set of supporting resources.

In addition, the components degrade differently due to their positions and uses in the system and thus have different reliabilities as time goes on. Generally, the relatively expensive and new ones have higher quality and are more reliable. Rearranging them has the potential to improve the system reliability. The corresponding CAP can provide the new assignment of these components based on their current reliabilities (i.e., the extents of degradation) so that the system reliability is improved to a possibly maximal level.

Generally, most of systems use multiple types of components, and different types of components are not exchangeable. At the early stage of system reliability design, we assume that only the reliabilities of one type of components that perform the same function are known and that all the other components are perfect (with reliability of 1). Then, the remaining system design of allocating this type of components into their positions can be completed by solving the corresponding CAP. As an extension, suppose that certain types of components have been fixed in the system and their reliabilities are known. Built on that information, the system reliability design is to allocate the remaining types of components type-wise one-by-one in the order of their importance to the system until all types of components are assigned. For each type of components, the CAP is to determine the allocation of these functionally exchangeable components to the system so as to maximize the system reliability whose calculation is based on the assignments of the allocated types of components and the assumption that all the undealt types of components are perfect.

1.3 Solution methods for the CAP

To the best of our knowledge, there is no study on exact optimization method except the trivial enumeration methods for solving the CAP. The exact enumeration method evaluates the system reliability for every possible permutation and chooses the one

that achieves the highest system reliability as the optimal assignment; thus, it requires to calculate the system reliability for $n!$ times. As n increases, the computation time of the exact enumeration would be intolerably long. For example, in a system with $n = 20$ positions, the exact enumeration method takes several days without obtaining the optimal assignment. Hence, the fast and high-performance heuristics are highly demanded. Some heuristic type methods have been proposed for the CAP and can mainly be classified into two groups. One group is the BI-based heuristics as reviewed in Subsection 1.3.1, and another group is the meta-heuristics including simulated annealing and genetic algorithm in Subsection 1.3.2.

1.3.1 The BI-based heuristics

The ordering of component BIs is a good indicator for the solution of the CAP. Lin and Kuo (2002) proved that when the component reliabilities are close enough, the optimal assignment of the CAP is always consistent with the BI ordering of components that is calculated using the lowest component reliability. In other words, the optimal assignment is to assign the most reliable component to the position with the largest BI value, the second most reliable component to the position with the second largest BI value, and so on. Some $Lin/Con/k/n$ systems admit invariant optimal assignments (Malon, 1984, 1985; Zuo and Kuo, 1990); then these systems must have a consistent BI ordering, and the invariant optimal arrangement assigns components according to the consistent BI ordering (Lin and Kuo, 2002). Motivated by these properties of the BI, research has been conducted to design heuristics for the CAP as illustrated below.

Kontoleon (1979) presented an iterative algorithm, which starts with all positions assigned the same component of the lowest reliability and then iteratively assigns the available components to the system. At each iteration, two possible positions are selected as candidates to receive the component of higher reliability. The first candidate is the position whose BI is the largest and for which there exists an available

component of higher reliability. The second candidate is the position which currently has the highest reliability among the positions whose reliabilities are lower than the reliability of the component to be assigned. Setting the reliability of the second position equal to that of the first position and recalculating the BIs of all positions, if any updated BI is larger than the original BI of the first position, then the second position is chosen; otherwise, the first position is chosen. The iterations continue until all components are assigned to the system.

Zuo and Kuo (1990) designed two related heuristics, referred to as the ZKA and ZKB heuristics, for $Lin/Con/k/n$ systems. They start with a feasible initial solution (i.e., arrangement) and try to improve the arrangement by pairwise exchanging the allocations of components to match the BI ordering. Lin and Kuo (2002) proposed a greedy algorithm, referred to as the LKA heuristic, with analytically calculated error terms. It initializes all positions with the least reliable component and iteratively assigns the available most reliable component to the unassigned position with the largest BI. Zuo and Shen (1992) also proposed a heuristic, which is actually the same as the LKA heuristic except that it is specialized for redundant $Lin/Con/k/n$:F systems whereas the LKA heuristic is for any general system.

1.3.2 The meta-heuristics for $Cir/Con/k/n$:F systems

The meta-heuristics (e.g., genetic algorithm and simulated annealing) have shown great potential to break local optimum and are expected to obtain better solutions than the local search algorithms. They have been successfully applied to solve some difficult combinatorial optimization problems. For example, genetic algorithm has been used in the traveling salesman problem (Grefenstette et al., 1985), the scheduling problem (Yamada and Nakano, 1992), and the assignment problem (Pentico, 2007), and simulated annealing in scheduling, routing, assignment problems (Koulamas et al., 1994).

A few genetic algorithms and simulated annealing algorithms have been proposed for the CAP in $Cir/Con/k/n$:F systems in the literature. Note that for a $Cir/Con/k/n$:F system, two assignments are equivalent if one can be obtained by rotating or reversing another because the components are arranged in a circle. Shingyoch et al. (2009) developed a genetic algorithm that modifies Grefenstette's ordinal representation schema (Grefenstette et al., 1985) by keeping only one of the equivalent assignments and eliminating others. Their genetic algorithm always assigns the most reliable components at every kth positions because the $Cir/Con/k/n$:F system works if the components at every kth positions work.

In addition to this genetic algorithm, Shingyoch et al. (2010) proposed two simulated annealing algorithms – a standard simulated annealing and an improved one that further eliminates equivalent assignments as in Shingyoch et al. (2009). The numerical experiments showed that these two simulated annealing algorithms usually generate better solutions than the genetic algorithm in Shingyoch et al. (2009) but take longer computation time; the improved simulated annealing can generate as good solutions as the standard one but takes only half of the computation time due to its reduced search space.

1.4 Dissertation overview

The remainder of this dissertation is structured as follows. Chapter 2 conducts literature review on the BI patterns and, more importantly, studies the nature of BI patterns in $Lin/Con/k/n$ systems by utilizing a new division method for $Lin/Con/k/n$ systems. Chapter 3 proposes five new BI-based heuristics based on the schemes of the existing BI-based heuristics and a two-stage approach for solving the CAP. Chapter 4 proposes a BI-based genetic local search algorithm for the CAP. Chapter 5 concludes this dissertation.

Chapter 2

BI Patterns in $Lin/Con/k/n$ Systems[†]

Some valuable research has been conducted on the BI patterns in $Lin/Con/k/n$ systems (Chadjiconstantinidis and Koutras, 1999; Chang et al., 1999, 2000, 2002; Chang and Hwang, 2002; Kuo et al., 1990; Lin et al., 1999; Zuo, 1993). However, the existing results are dispersive and some of them are misleading or inaccurate. To the best of our knowledge, no research work has been done to unscramble the results and explore the nature of BI patterns for $Lin/Con/k/n$ systems. This chapter first summarizes and clarifies all existing BI patterns, and, most importantly, presents our new findings, which discover the nature of BI patterns for $Lin/Con/k/n$ systems. Then, some new BI patterns are proved conditioned on the value of p. Further, based on systematic computational tests, we disprove some conjectures and claims that were made in the literature, and make new conjectures. With the discovery of the nature of BI patterns, we can understand the proved and disproved patterns and analyze the conjectures in-depth.

Now, we list the notation used in this chapter.

[†]Reused with permission from Xiaoyan Zhu, Qingzhu Yao, and Way Kuo (2011) Patterns of the Birnbaum importance in linear consecutive-k-out-of-n systems, IIE Transactions, in press. Copyright IIE Transactions 2011.

Notation

- p: common component reliability, i.e., $p_1 = p_2 = \ldots = p_n = p$
- $q = 1 - p$: common component unreliability
- $R(k, n)$: reliability of a $Lin/Con/k/n$:F system with components of common reliability p
- $R'(k, n)$: reliability of a $Lin/Con/k/n$:G system with components of common reliability p
- $I_n(i)$: BI of component i
- $i \succ_u j, i \succ_h j, i \succ_c j$: component i is more uniformly, half-line, and combinatorially B-important than component j, respectively
- $i =_u j$: components i and j are equally B-important in the uniform case

Note that a $Lin/Con/k/n$ system is structurally symmetric with respect to the middle component(s); thus in a $Lin/Con/k/n$:F or G system, for any $0 < p < 1$,

$$I_n(i) = I_n(n - i + 1). \tag{2.1}$$

Throughout this chapter, the BI is discussed for the first half components $i \leq \lceil \frac{n}{2} \rceil$, unless otherwise specified. The BIs of components $\lceil \frac{n}{2} \rceil + 1, \ldots, n$ can be compared using the symmetric property of components i and $n-i+1$ in (2.1). For convenience, we do not explicitly state the upper bound $\lceil \frac{n}{2} \rceil$ each time. Consequently, the default range of n is assumed so that each component i involved is in the first half part of the system, i.e., $n \geq 2i$. Moreover, the results in this chapter are mainly presented for $Lin/Con/k/n$:F systems because the results for the F system are also true for the G system via changing the common component reliability p to $1-p$ (Kuo et al., 1990).

2.1 Existing BI patterns

For a $Lin/Con/k/n$ system, the BI patterns are complex. This section summarizes all existing BI patterns for $Lin/Con/k/n$ systems in the uniform case (Subsection 2.1.1) and the half-line case (Subsection 2.1.2).

2.1.1 The uniform BI patterns

Based on previous research, we summarize all existing uniform BI patterns for $Lin/Con/k/n$:F and G systems in Lemma 1.

Lemma 1. *The uniform BI in a $Lin/Con/k/n$ (either F or G) system has the following patterns:*

(i)	*when $k = 1$ or n:*	$i =_u j$		*for* $1 \le i < j \le n$
(ii)	*when $k = 2$:*	*a.*	$2t \succ_u 2t-1$	
		b.	$2t \succ_u 2t+2$	
		c.	$2t+1 \succ_u 2t-1$	
(iii)	*when $\frac{n}{2} \le k < n$:*	*a.*	$i+1 \succ_u i$	*for* $1 \le i \le n-k$
		b.	$i =_u i+1$	*for* $i > n-k$
(iv)	*when $2 < k < \frac{n}{2}$:*	*a.*	$i+1 \succ_u i$	*for* $1 \le i < k$
		b.	$i \succ_u 1$	*for* $i > 1$
		c.	$k \succ_u i$	*for* $i \ge 1, i \ne k$
(v)	*when $n \ge 2k+3$:*	$k+2 \succ_u k+1$		
(vi)	*when $n \ge 4k+1$:*	$2k \succ_u 2k+1$		
(vii)	*when $n = 4k-1$:*	$2k \succ_u 2k-1$		
(viii)	*when $n = 6k+1$:*	$3k \succ_u 3k+1$.		

In part (ii), $t \ge 1$ is an integer.

Lemma 1 is accompanied for illustration by Figure 2.1 in which the horizontal axis stands for the component index i and the vertical axis for the corresponding BI. Figure 2.1 shows the combinatorial BI patterns for $Lin/Con/k/14$:F (the same for G) systems with $k = 2, 3, 7$, and 12, representing the different relative magnitudes of k and n in Lemma 1. Figure 2.1 also demonstrates the symmetric property of the BI around the middle (two) component(s) in (2.1). For all other figures in this chapter, only components 1 to $\lceil \frac{n}{2} \rceil$ are shown to save space.

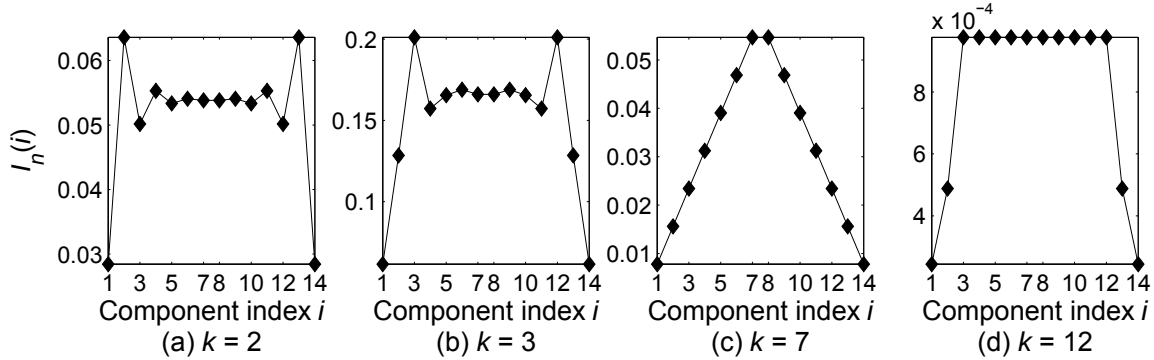

Figure 2.1: The combinatorial BI for $Lin/Con/k/14$:F systems with $k = 2, 3, 7$, and 12

Part (i) in Lemma 1 indicates that when $k = 1$ in an F system or $k = n$ in a G system (i.e., a series system) or when $k = n$ in an F system or $k = 1$ in a G system (i.e., a parallel system), all components are equally important according to the BI. For the series system, the failure of any component is equally likely to cause system failure, and for the parallel system, the survival of any one of n components guarantees the working of the system.

Part (ii) completely identifies the uniform BI patterns for a $Lin/Con/2/n$ system, which was discovered by Zuo and Kuo (1990). For the $Lin/Con/2/n$ system (e.g., the $Lin/Con/2/14$:F system in Figure 2.1(a)), the BI value alternatively jumps up and down and the relative difference of the BI values of two consecutive components decreases as component index i increases from 1 to $\lceil \frac{n}{2} \rceil$.

Part (iii) was presented by Zuo (1993), completely identifying the patterns of the uniform BI when $k \geq \frac{n}{2}$. When n is even and $k = \frac{n}{2}$ (or n is odd and $k = \frac{n+1}{2}$) (e.g., the $Lin/Con/7/14$:F system in Figure 2.1(c)), the middle two components (or the middle component) have (has) the largest BI value and the BI value increases from components 1 to k and decreases from components $k+1$ (or k) to n. When $k > \frac{n}{2}$ as shown in Figure 2.1(d), the BI value increases from components 1 to $n-k+1$ and decreases from components k to n because beginning with component 1 until component $n-k+1$, each component gradually has more components adjacent to it, and this has a heavier impact on system reliability than its preceding components. Moreover, when $k > \frac{n}{2}$, the components between $n-k+1$ and k maintain the same BI value:

$$I_n(i) = \frac{1}{q}[1 - R(k, n)] \quad \text{for F systems}$$

and

$$I_n(i) = \frac{1}{p}R'(k, n) \quad \text{for G systems}$$

because the survival (failure) of any one of these components guarantees the working (failure) of the F (G) system. Further, these components are more uniformly B-important than others.

When $2 < k < \frac{n}{2}$, parts (iv)-(viii) identify the partial ordering of component BIs; however, the BI patterns, particularly for components k to $n-k+1$, have not been completely identified. Specifically, part (iv) indicates that the BI value monotonically increases in the first k-interval (see Definition 1 below) and that the BIs of components 1 and k, respectively, are the lower and upper bounds on the BIs of all other components (e.g., the $Lin/Con/3/14$:F system in Figure 2.1(b)). Chang et al. (2002) presented a review on parts (iv)-(vii) and also proved part (viii). Note that part (v) was also independently proved by Chadjiconstantinidis and Koutras (1999) using the Markov chain approach. Chang et al. (2002) proved that in a $Lin/Con/k/(tk-1)$:F system $(t-2)k \succ_u (t-2)k-1$ for $t \geq 3$. When $t = 3$, it is equivalent to $k \succ_u k-1$ which is covered by part (iv)c; when $t \geq 5$, it is equivalent to $2k \succ_u 2k+1$ which is covered by part (vi); when $t = 4$, it is part (vii).

Defination 1. For a $Lin/Con/k/n$ system with $n > 2k$, a k-interval (or simply, interval) is a set of k consecutive components starting from component $(t-1)k+1$ for each $1 \leq t \leq \lceil \frac{n}{2k} \rceil$, except that the last interval ($t = \lceil \frac{n}{2k} \rceil$) may include less than k components: $(t-1)k+1, \ldots, \lceil \frac{n}{2} \rceil$.

2.1.2 The half-line BI patterns

Except for the uniform BI patterns presented in Lemma 1, Lemma 2 presents all additional patterns for the half-line BI for the $Lin/Con/k/n$:F system with $2 < k < \frac{n}{2}$. The half-line BI patterns in Lemma 2 were proved by Chang et al. (2002), and some of them are extensions of the combinatorial BI patterns presented by Lin et al. (1999).

Lemma 2. *The half-line BI in a $Lin/Con/k/n$:F system has the following patterns:*

(i) $k+1 \succ_h k-1$

(ii) $i+1 \succ_h i$ *for* $k < i < 2k$

(iii) $i \succ_h k+1$ *for* $i > k+1$

(iv) $2k \succ_h i$ *for* $i > 2k$

(v) $2k+2 \succ_h 2k+1$

(vi) $3k \succ_h 3k+1$.

These BI patterns also hold for a $Lin/Con/k/n$:G system with $0 < p \leq \frac{1}{2}$.

With the restriction of $p \geq \frac{1}{2}$, patterns in part (iv) in Lemma 1 are extended to parts (ii)-(iv) in Lemma 2. Note that part (iv) in Lemma 1 specifies the uniform BI patterns of components in the first k-interval, and parts (ii)-(iv) in Lemma 2 specify the same half-line BI patterns of components in the second k-interval (i.e., components $k+1$ to $2k$), e.g., Figure 2.1(b). However, even with the addition of patterns in Lemma 2, the half-line BI patterns for the $Lin/Con/k/n$:F system with $2 < k < \frac{n}{2}$ has not been completely determined. If we specify $p = \frac{1}{2}$ and consider the combinatorial BI, no additional pattern is obtained. As will be shown in Section 2.3, the half-line BI patterns in parts (i), (ii), (iv) and (vi) in Lemma 2 cannot be generalized to the uniform case and the ones in parts (iii) and (v) are conjectured to the uniform case.

Remark 1. *Summarizing Section 2.1, we present three points based on Lemmas 1 and 2.*

(i) *Regardless of the relative magnitudes of k and n and the value of p, component k has the largest BI value, component 1 has the smallest BI value, and the BI values of components 1 to k are nondecreasing (see parts (i)-(iv) in Lemma 1).*

(ii) *The patterns of the uniform BI for a $Lin/Con/k/n$ system with $k = 1, 2$, or $k \geq \frac{n}{2}$ are completely identified (see parts (i)-(iii) in Lemma 1).*

(iii) *For a $Lin/Con/k/n$ system with $2 < k < \frac{n}{2}$, the BI patterns have not been completely identified yet. In addition to the patterns in point (i), the half-line BI in the second k-interval (see parts (ii)-(iv) in Lemma 2) has the same patterns as the uniform BI in the first k-interval (see part (iv) in Lemma 1).*

2.2 The nature of BI patterns

The rest of this chapter investigates the BI patterns for the $Lin/Con/k/n$ systems with $2 < k < \frac{n}{2}$ because the BI patterns for the cases of $k \leq 2$ or $n \leq 2k$ have been

completely identified in parts (i)-(iii) in Lemma 1. Research has been done towards completely identifying the BI patterns for such systems but instead has turned in some disproofs of the "expected" patterns on change and left some patterns for conjectures (see Section 2.3). This section investigates the fundamental behaviors of BI patterns for the $Lin/Con/k/n$:F systems. Subsection 2.2.1 demonstrates the nonexistence of "expected" patterns, and Subsection 2.2.2 proposes a new direction for analyzing the BI patterns for the $Lin/Con/k/n$ systems. Then, Subsections 2.2.3 and 2.2.4 investigate the relationship between the BI patterns and p and the relationship between the BI patterns and n, respectively.

2.2.1 The nonexistence of "expected" patterns

Previous research on the BI patterns for a $Lin/Con/k/n$ system (Chang et al., 1999, 2000, 2002; Chang and Hwang, 2002; Zuo, 1993) explicitly or implicitly used a concept of the k-interval (see Definition 1) and tried to find the similar patterns within an interval and/or among intervals. Shingyoch and Yamamoto (2009) and Shingyochi et al. (2009) also proposed that using k-interval can usually improve the performance of their meta-heuristics. Note that the length of intervals is fixed to k, possibly except the last interval. Motivated by the results in Section 2.1 and particularly in Remark 1, one may expect a general "fixed-length BI ordering" for some range of p (e.g., $0 < p < 1$, $p \geq \frac{1}{2}$, or $p = \frac{1}{2}$), which embodies the BI patterns in Lemmas 1 and 2 and can be specified by the following patterns (here $1 \leq t \leq \lceil \frac{n}{2k} \rceil$):

(i) $I_n(i+1) > I_n(i)$ for $(t-1)k < i < tk$

(ii) $I_n(i) > I_n((t-1)k+1)$ for $i > (t-1)k+1$

(iii) $I_n(tk) > I_n(i)$ for $i > tk$

(iv) $I_n(tk+1) > I_n(tk-1)$.

As its name implies, the "fixed-length BI ordering" is based on the k-interval division, and the patterns look like a water wave with the wave form repeating itself at intervals of *fixed* length, k. Mimicking part (iv) in Lemma 1 and parts (ii)-(iv)

in Lemma 2, patterns (i)-(iii) describe that the BI values of components within a k-interval strictly increases, and that the BI of the first (last) component in a k-interval is the lower (upper) bound of the BIs of its succeeding components. Mimicking part (i) in Lemma 2, pattern (iv) shows that the BI of component $tk+1$ is between the BIs of components $tk-1$ and tk, which describes the relation of the component BIs among two consecutive intervals.

However, our computational tests show that the "fixed-length BI ordering" does not exist in general when $2 < k < \frac{n}{2}$, although it may hold for some special cases determined by the values of k, n, and p (e.g., the $Lin/Con/3/14$:F system in Figure 2.1(b) with $p = \frac{1}{2}$). Thus, when $2 < k < \frac{n}{2}$, the effort for finding similar BI patterns within each of all intervals and among intervals is futile and cannot bring about general BI patterns. This is illustrated by the example in a $Lin/Con/8/128$:F system ($n > 2k$) with $p = 0.2$ in Figures 2.2 and 2.3. Figure 2.2 illustrates the whole picture of how the BI changes in the first half (i.e., components 1 to 64) of the $Lin/Con/8/128$:F system. Noting that the BIs after the first two intervals are not easy to be compared based on Figure 2.2, Figure 2.3 draws the BIs in the eight intervals separately in order to further amplify the differences of the BI values. In Figure 2.3, the vertical axis represents the BI values scaled by multiplying 10^4.

As in Figure 2.3(a), a nice uniform BI pattern exists for the first interval in which the BI value increases as component index i increases from 1 to k. However, such a pattern cannot be extended even to the second interval. In some cases (e.g., Figure 2.3(b)), the component BI value within the second interval first increases and then decreases as component index i increases. If we treat the monotone-increasing pattern in the first interval as a special case of the increasing-then-decreasing pattern in the second interval, the patterns are consistent in this sense. However, we note that the increasing-then-decreasing pattern does not hold for all intervals (e.g., the fifth, sixth, and seventh intervals). Moreover, the last interval has a completely reversed pattern: first decreasing and then increasing. We observe the same phenomenon for various values of $0 < p < 1$.

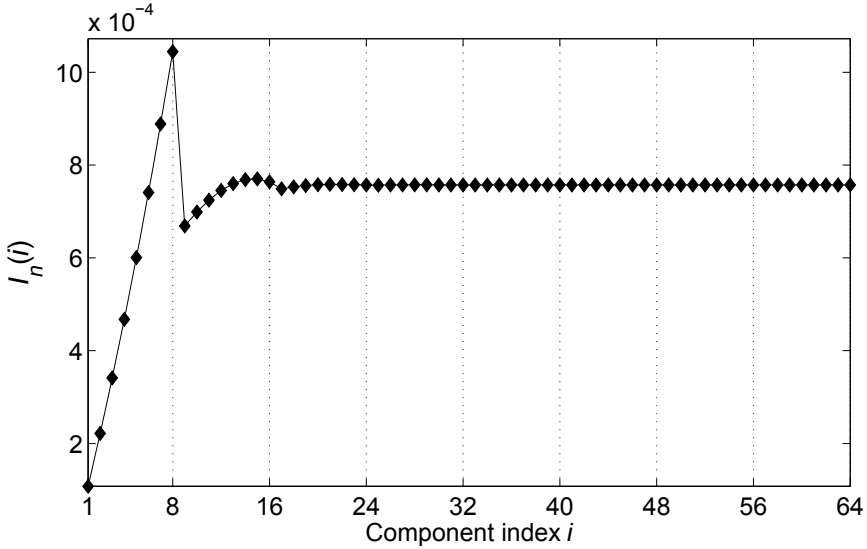

Figure 2.2: BI patterns for a $Lin/Con/8/128$:F system with $p = 0.2$

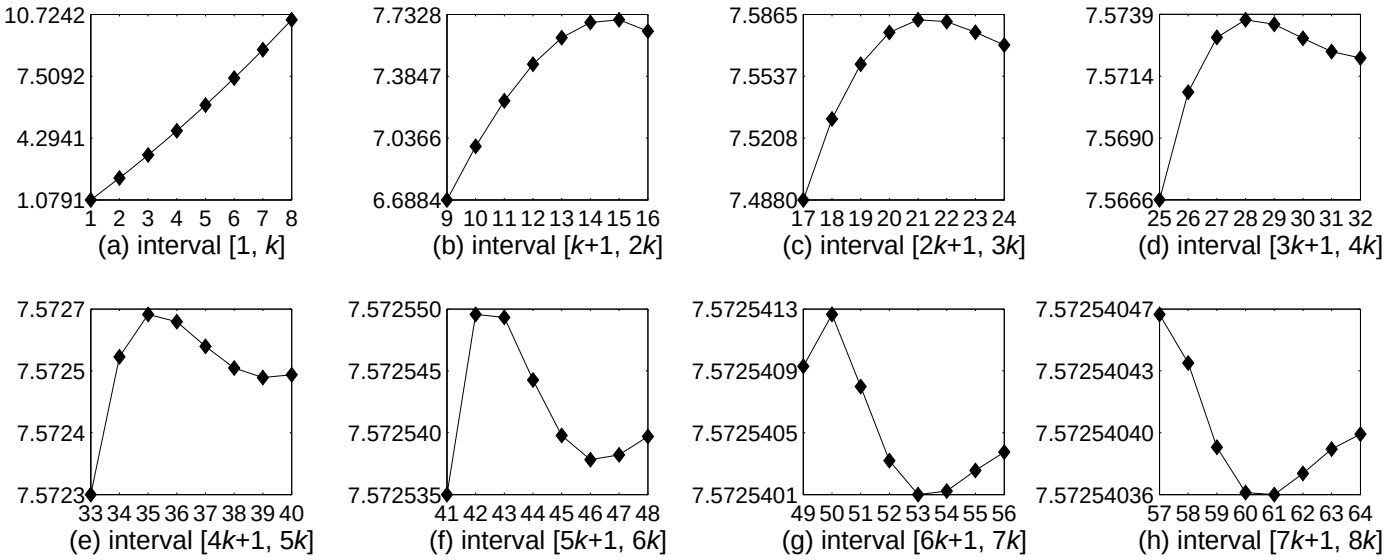

Figure 2.3: Intervals for a $Lin/Con/8/128$:F system with $p = 0.2$

2.2.2 Segments and their lengths and peaks

The observations in Subsection 2.2.1 reveal that the "fixed-length BI ordering" based on the k-interval division does not always hold. The fundamental problem is that the k-interval division (i.e., all intervals of equal length k except for the last one) is not appropriate. Now, we propose a new segment-based division for grouping the first half components (i.e., components 1 to $\lceil \frac{n}{2} \rceil$) in a $Lin/Con/k/n$ system with $n > 2k$ and define a segment and its length and peak as below.

Defination 2. For a $Lin/Con/k/n$ system with $n > 2k$, a segment is a maximal set of consecutive first-half components whose BI values first increase and then decrease, satisfying that the union of all segments covers the first half components and two consecutive segments contain exactly one common component.

Defination 3. The length of a segment is the number of components in the segment minus 1 since one common component is shared by two consecutive segments.

Defination 4. The peak of a segment is the index of the component corresponding to the maximal BI value in the segment.

Then, according to parts (iv) and (v) in Lemma 1, in any $Lin/Con/k/n$ system with $n > 2k$, the first segment always contains components 1 to k and $k+1$, and both its length and its peak are k. Figure 2.4 shows the newly-defined segments for the $Lin/Con/8/128$:F system in Figure 2.2. In Figure 2.4, the vertical axis represents the BI values scaled by multiplying 10^4. There are 9 segments, and each segment shows an increasing-then-decreasing pattern except for the last segment, which may be monotonically increasing. Actually, the last segment is truncated by the middle component of the system.

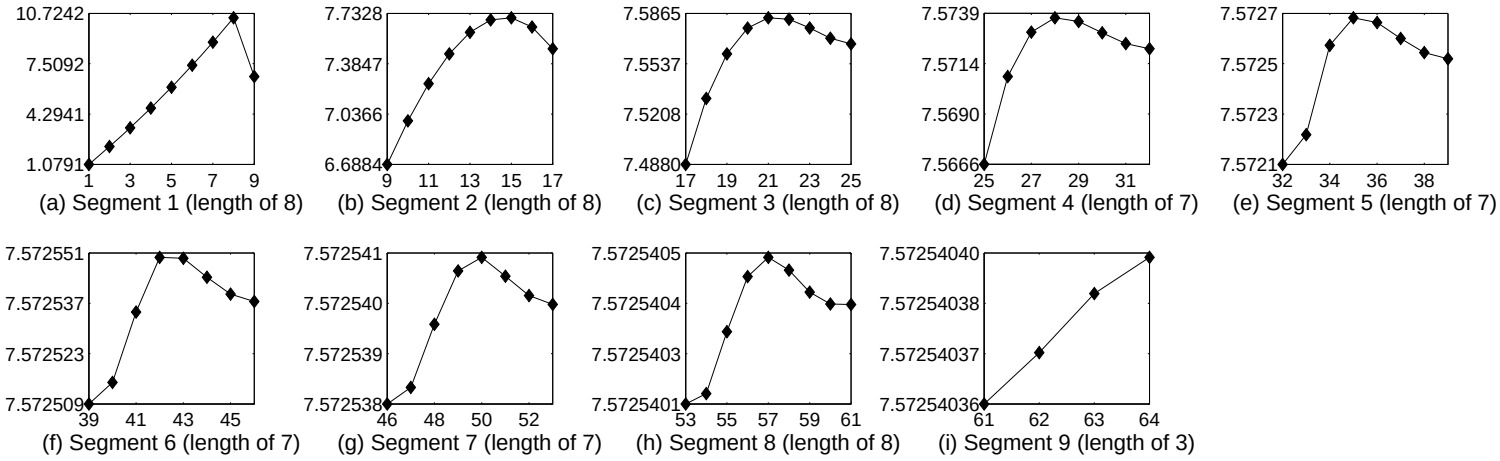

Figure 2.4: Segments for a $Lin/Con/8/128$:F system with $p = 0.2$

From Figure 2.4 and all of our computational tests, we observe that the lengths of segments can be different. Starting from k for the first segment, the length of segments may decrease in the subsequent segments and then may increase again, and so on. It looks like a water wave but the wave form repeats at intervals of *different* lengths. Although we do not know the exact formula to determine the length of each segment (it may be hard or impossible), at least the increasing-then-decreasing BI pattern is consistent in all segments. We also observe from our computational tests that the lengths of segments are never greater than k. We conjecture that this observation is true for any general cases. Additionally, the peaks can appear at different relative positions in the segments. In Figure 2.4, for segments 1 and 2, the peaks appear at the back end of each segment; for the other segments except the last one, the peaks appear at the middle of each segment. Because the length and the relative position of the peak vary with the segments, the "fixed-length BI ordering", of course, do not hold for general. In the next subsection, we investigate the peak and the length of segments with respect to p.

2.2.3 Patterns with respect to p

Theorem 2.1, presented below, gives the smallest possible index of the peak of the second segment for different p values, which also implies the lower bound on the length of the second segment.

Theorem 2.1. *Considering a Lin/Con/k/n:F system with components of the common reliability p and $k > 2$, if $p \geq \frac{1}{k-s}$, where $0 \leq s \leq k-3$, then*

(i) $I_n(k+1) > I_n(s+1)$ *for* $n \geq 2k+1$, *and*

(ii) $I_n(i+1) > I_n(i)$ *for* $k+1 \leq i \leq k+s+1$ *and* $n \geq 4k+s+2$.

Proof. We only prove the statements for the case of $s = k-3 \geq 1$ (i.e., $p \geq \frac{1}{3}$). The other cases of $0 \leq s \leq k-4$ can be proved similarly.

(i) According to Kuo and Zuo (2002),

$$R(k,n) = R(k,n-1) - pq^k R(k,n-k-1), \tag{2.2}$$

$$I_n(i) = \frac{R(k,i-1)R(k,n-i) - R(k,n)}{q}, \tag{2.3}$$

where $R(k,n) = 1$ for $0 \le n < k$, and $R(k,k) = 1 - q^k$. Using (2.2) and (2.3), for any $n \ge 2k+1$,

$$\begin{aligned}
&I_n(k+1) - I_n(k-2) \\
&= \frac{R(k,k)R(k,n-k-1) - R(k,n)}{q} - \frac{R(k,k-3)R(k,n-k+2) - R(k,n)}{q} \\
&= \frac{1}{q}\left[(1-q^k)R(k,n-k-1) - R(k,n-k+2)\right] \\
&= \frac{1}{q}\left[(1-q^k)R(k,n-k-1)\right. \\
&\qquad \left. -R(k,n-k-1) + pq^k\left(R(k,n-2k-1) + R(k,n-2k) + R(k,n-2k+1)\right)\right] \\
&= q^{k-1}\left[p\left(R(k,n-2k-1) + R(k,n-2k) + R(k,n-2k+1)\right) - R(k,n-k-1)\right],
\end{aligned}$$

where the third equation is obtained by iteratively using (2.2) for three times. Further, by (2.2), we know that $R(k,n)$ decreases as n increases; thus, each of $R(k,n-2k-1)$, $R(k,n-2k)$, and $R(k,n-2k+1)$ is greater than $R(k,n-k-1)$. Therefore, $p \ge \frac{1}{3}$ implies that

$$I_n(k+1) > I_n(k-2). \tag{2.4}$$

(ii) Chang et al. (2002) proved that for $i \ge k+1$ and $0 < j - i < k$,

$$I_n(j) - I_n(i) = pq^k \sum_{r=1}^{j-i} \left[I_{n-k-r}(i) - I_{n-k-r}(j-k-r)\right]. \tag{2.5}$$

Letting $j = i+1$ in (2.5), for $i \ge k+1$,

$$I_n(i+1) - I_n(i) = pq^k \left[I_{n-k-1}(i) - I_{n-k-1}(i-k)\right]. \tag{2.6}$$

For any $n \geq 4k+s+2 = 5k-1$ and $k+2 \leq i \leq k+s+1 = 2k-2$, by (2.5), we have

$$I_{n-k-1}(i) - I_{n-k-1}(k+1) = pq^k \sum_{r=1}^{i-k-1} \left[I_{n-2k-r-1}(k+1) - I_{n-2k-r-1}(i-k-r) \right]. \quad (2.7)$$

For any $1 \leq r \leq i-k-1$, we know $n-2k-r-1 \geq n-k-i \geq 2k+1$ and $1 \leq i-k-r \leq k-3$. Thus, by part (iv)a in Lemma 1, $I_{n-2k-r-1}(k-2) > I_{n-2k-r-1}(i-k-r)$, and by (2.4), $I_{n-2k-r-1}(k+1) > I_{n-2k-r-1}(i-k-r)$. Then, from (2.7), for any $n \geq 5k-1$ and $k+2 \leq i \leq 2k-2$,

$$I_{n-k-1}(i) > I_{n-k-1}(k+1). \quad (2.8)$$

On the other hand, by part (iv)a in Lemma 1 and (2.4), we have

$$I_{n-k-1}(k+1) > I_{n-k-1}(i-k) \quad (2.9)$$

for $n-k-1 > 2k+1$ and $2 \leq i-k \leq k-2$, which are satisfied when $n \geq 5k-1$ and $k+2 \leq i \leq 2k-2$.

From (2.8) and (2.9), we obtain $I_{n-k-1}(i) > I_{n-k-1}(i-k)$ for $n \geq 5k-1$ and $k+2 \leq i \leq 2k-2$. By part (iv)b in Lemma 1, $I_{n-k-1}(i) > I_{n-k-1}(i-k)$ also holds for $n \geq 5k-1$ and $i = k+1$. Therefore, by (2.6), we obtain $I_n(i+1) > I_n(i)$ for $n \geq 5k-1$ and $k+1 \leq i \leq 2k-2$. □

Note that in Theorem 2.1, when $s = 0$, parts (i) and (ii) come directly from parts (iv)b and (v) in Lemma 1, respectively. As shown in part (ii) in Theorem 2.1, $I_n(k+s+1) > I_n(k+s) > \cdots > I_n(k+1)$ when $p \geq \frac{1}{k-s}$ with $1 \leq s \leq k-3$. Thus, the peak of the second segment is no less than $k+s+1$, and, consequently, the length of the second segment is no less than $s+1$. As $p \geq \frac{1}{k-s}$ increases, s and $k+s+1$ increase, which means that the lower bounds of the peak and the length increase as p increases.

We now conduct the numerical studies on the relations of the segment length and the segment peak to p. As an example, Figure 2.5 presents the BIs for the $Lin/Con/5/29$:F system for $p = 0.1, 0.3, 0.5, 0.7$ and 0.9. We focus on the contrast of component BI values; thus, Figure 2.5 does not present the BIs of the first k components since the patterns of these components are known and fixed. We observe the following results for the segment length and peak from our numerical studies.

(i) Given a $Lin/Con/k/n$:F system (e.g., Figure 2.5), as p becomes smaller, it is more likely to have short segment length and the segment length decreases more quickly. With a lower p (e.g., 0.1 in Figure 2.5(a)), the segment length may become less than k starting from the third segment, but with a higher p, it may maintain k for the first few segments.

(ii) As p increases (e.g., from 0.1 to 0.9 as in Figure 2.5), the BI patterns change gradually. With a lower p (e.g., 0.1 in Figure 2.5(a)), each segment except the first segment has clearly both increasing and decreasing parts, and its peak usually appears before the second component from the end of each segment. With a higher p, the increasing part is much longer than the decreasing part in each segment, and the peak usually is the second component from the end of each segment. When p is very high (e.g., 0.9 in Figure 2.5(e)), the BI values are almost the same starting from the second segment, which will be proved in Theorem 2.2.

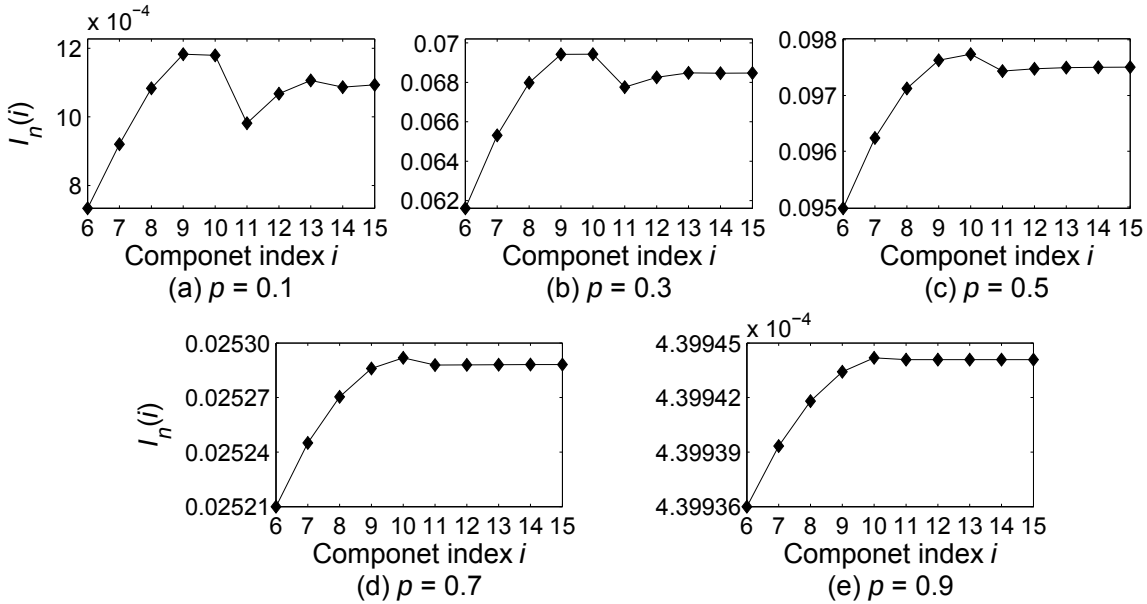

Figure 2.5: The BI for $Lin/Con/5/29$:F systems with $p = 0.1, 0.3, 0.5, 0.7$, and 0.9

Remark 2. *In summarizing the theoretical and numerical findings, we give three points about the peaks and lengths of the first three segments.*

(i) *For any* $0 < p < 1$*, the first segment always has the length equal to* k*, and the peak of the first segment is* k *(see part (iv) in Lemma 1).*

(ii) *When* $p \geq \frac{1}{2}$*, the length of the second segment is always* k*, and the peak of the second segment is* $2k$ *(see parts (ii), (iv), and (v) in Lemma 2). When* $p < \frac{1}{2}$*, we observe from the computational tests that the BI value in the second segment first increases and then decreases until component* $2k+1$ *and that the peak of the second segment can appear earlier than* $2k$*, but the length of the second segment is always* k*. This observation leads to conjecture (2.37), i.e.,* $2k+2 \succ_u 2k+1$ *(see Subsection 2.3.2).*

(iii) *The third segment may have a length less than* k*. For example, in Figure 2.5(a), the third segment from component* 11 *to component* 14 *has a length of* 3.

Remark 3. *With the discoveries in this subsection, we can explain why some half-line BI patterns in Lemma 2 hold for* $p \geq \frac{1}{2}$ *but cannot be extended to* $p < \frac{1}{2}$*. This is because the BI patterns for a system, in particular the lengths and peaks of segments, may vary for different* p *values. For example, considering part (ii) in Lemma 2 and the Lin/Con/5/29:F system in Figure 2.5, if* $p \geq \frac{1}{2}$ *(e.g.,* $p = 0.5, 0.7, 0.9$ *in Figures 2.5(c)-(e)), the peak of the second segment is* $2k = 10$ *according to part (ii) in Lemma 2. However, when* p *becomes smaller (e.g.,* $p = 0.1$ *in Figure 2.5(a)), the peak of the second segment is* 9*, less than* $2k$*, and the decreasing part in the second segment starts from component* 9 *until component* 11*, which implies that part (ii) in Lemma 2 cannot be extended to the uniform BI.*

Kuo et al. (1990) claimed that the illustrations of the BI patterns for $p = \frac{1}{2}$ are also true for $p \neq \frac{1}{2}$ except that contrast is less sensitive to variations. From this subsection, we see that the BI patterns for a $Lin/Con/k/n$ system may differ for distinct values of p.

2.2.4 Patterns with respect to n

For the $Lin/Con/k/n$:F system with $2 < k < \frac{n}{2}$, Lemma 1 presents the uniform BI patterns for the first $k+2$ components, and Lemma 2 presents the half-line BI patterns for the first $2k+2$ components. Because the lengths of segments can be different, the BI patterns that exist for the first or first few k-intervals cannot generally be extended further. Fortunately, for the half-line case, only the BIs of components in the first few segments make a difference as n increases, and the components in the middle of the n-component line have almost the same BI values. This subsection supports this statement via investigating the changes of BI patterns with respect to the system size n. In the following, we first study the difference of the BI values of some special pairs of components in one $Lin/Con/k/n$ system and then compare the BI values in two different systems: $Lin/Con/k/n$ and $Lin/Con/k/n+1$.

Difference of BIs of two components in one system

Theorem 2.2 shows that the difference of $I_n(k) - I_n(k+1)$ decreases as n increases or p increases.

Theorem 2.2. *For a $Lin/Con/k/n$:F system with components of the common reliability $0 < p < 1$ and $n > 2k$, the difference of $I_n(k) - I_n(k+1)$ decreases as n increases and approaches 0 as p approaches 1.*

Proof. Considering a $Lin/Con/k/(n-k-1)$:F system with $n-k-1 \geq k$ (i.e., $n > 2k$) and making a pivotal decomposition on component k, the $Lin/Con/k/(n-k-1)$:F system is decomposed into two subsystems: a $Lin/Con/k/(k-1)$:F system and a $Lin/Con/k/(n-2k-1)$:F system. Then we have

$$
\begin{aligned}
& R(k, n-k-1) \\
&\quad = \; p \Pr\{\text{the } Lin/Con/k/(n-k-1)\text{:F system works} \mid \text{component } k \text{ works}\} \\
&\qquad + q \Pr\{\text{the } Lin/Con/k/(n-k-1)\text{:F system works} \mid \text{component } k \text{ fails}\} \\
&\quad = \; p\, R(k, n-2k-1) \\
&\qquad + q \Pr\{\text{the } Lin/Con/k/(n-k-1)\text{:F system works} \mid \text{component } k \text{ fails}\}\,.
\end{aligned}
$$

Using (2.2) and (2.3), we have

$$
\begin{aligned}
& I_n(k) - I_n(k+1) \\
&\quad = \frac{R(k, n-k) - R(k,k)R(k, n-k-1)}{q} \\
&\quad = \frac{R(k, n-k-1) - pq^k R(k, n-2k-1) - (1-q^k)R(k, n-k-1)}{q} \\
&\quad = q^{k-1}\left[R(k, n-k-1) - pR(k, n-2k-1)\right] \\
&\quad = q^k \Pr\{\text{the } Lin/Con/k/(n-k-1)\text{:F system works} \mid \text{component } k \text{ fails}\}\,. \qquad (2.10)
\end{aligned}
$$

Note that $\Pr\{\text{the } Lin/Con/k/(n-k-1)\text{:F system works} \mid \text{component } k \text{ fails}\}$ is just the system reliability of a $Lin/Con/k/n$:F system with component reliabilities $p_k = 0$ and $p_i = p$ for $i \neq k$, and by (2.2), we know that the system reliability decreases as n increases. Therefore, we have from (2.10) that $I_n(k) - I_n(k+1)$ decreases as n increases.

Furthermore, from (2.10), it is straightforward that $I_n(k) - I_n(k+1)$ approaches 0 as p approaches 1 since q^k approaches 0 and the second term in (2.10) approaches 1 as p approaches 1. □

In the half-line case, $I_n(k+1)$ and $I_n(k)$ are the lower and upper bounds of the BIs of components $k+1 < i < n-k$, respectively, according to part (iv)c in Lemma 1 and part (iii) in Lemma 2 (note that $I_n(2k)$ is a tighter upper bound). Thus, the significance of Theorem 2.2 is that, for the half-line case, when n is large enough or p

approaches 1, there is no significant difference between the BIs of any two components i and j for $k+1 < i, j < n-k$, and thus we may not need to investigate the BI patterns of these components. This is consistent with the numerical observation in Subsection 2.2.3 that the BI values are almost the same starting from the second segment when p is very high. In addition, if conjecture (2.36) (see Subsection 2.3.2) can be proved, then $I_n(k+1)$ and $I_n(k)$ will be the lower and upper bounds of the BIs of components $k+1 < i < n-k$ in the uniform case, and the above statements will be extended to the uniform case.

For the half-line or uniform BI, results for some other special pairs of components are presented in Theorem 2.3, whose proof needs Lemma 3. Theorem 2.3 shows that when n is large, if $p \geq \frac{1}{2}$, $I_n(i) \approx I_n(i-k)$ for $k < i < 2k$, $i = 2k+1$, and $i = 3k$; if $0 < p < 1$, $I_n(i) \approx I_n(i-k)$ for $i = 2k$.

Lemma 3. *Considering a Lin/Con/k/n:F system and a Lin/Con/k/(n+1):F system with components of the common reliability* $0 < p < 1$ *and* $k > 2$, *for* $k+2 \leq i \leq \lceil \frac{n+1}{2} \rceil$,

$$\begin{aligned} & I_{n+1}(i-1) - I_{n+1}(i) \\ & \quad = [I_{n+1}(i-1) - I_{n+1}(i-k-1)] - [I_n(i-1) - I_n(i-k-1)]\,. \end{aligned} \tag{2.11}$$

Proof. Considering a $Lin/Con/k/(n+k+2)$:F system with $k > 2$, for any $k+2 \leq i \leq \lceil \frac{n+1}{2} \rceil$, we obtain from (2.5),

$$\begin{aligned} & I_{n+k+2}(i+1) - I_{n+k+2}(i-1) \\ & \quad = pq^k[I_{n+1}(i-1) - I_{n+1}(i-k) + I_n(i-1) - I_n(i-k-1)], \end{aligned} \tag{2.12}$$

and from (2.6),

$$\begin{aligned} & I_{n+k+2}(i+1) - I_{n+k+2}(i-1) \\ & \quad = I_{n+k+2}(i+1) - I_{n+k+2}(i) + I_{n+k+2}(i) - I_{n+k+2}(i-1) \\ & \quad = pq^k[I_{n+1}(i) - I_{n+1}(i-k) + I_{n+1}(i-1) - I_{n+1}(i-k-1)], \end{aligned} \tag{2.13}$$

where (2.13) uses (2.6) twice. Making (2.12) equal (2.13) gives (2.11). □

Theorem 2.3. *For a Lin/Con/k/n:F system with components of the common reliability p and $k > 2$, the following statements hold.*

(i) *The difference of $I_n(i)-I_n(i-k)$ decreases as n increases for $p \geq \frac{1}{2}$, $k < i < 2k$, and $n \geq 4k-1$.*

(ii) *The difference of $I_n(k) - I_n(2k)$ decreases as n increases for $0 < p < 1$ and $n \geq 4k$.*

(iii) *The difference of $I_n(2k) - I_n(3k)$ decreases as n increases for $p \geq \frac{1}{2}$ and $n \geq 6k$.*

(iv) *The difference of $I_n(2k+1) - I_n(k+1)$ decreases as n increases for $p \geq \frac{1}{2}$ and $n \geq 4k+2$.*

Proof. (i) According to part (ii) in Theorem 2, we have $I^h_{n+1}(i)-I^h_{n+1}(i+1) < 0$ for $k < i < 2k$ and $n \geq 4k-1$. Then, by (2.11), $\left[I^h_{n+1}(i) - I^h_{n+1}(i-k)\right] - \left[I^h_n(i) - I^h_n(i-k)\right] < 0$, which implies that $I^h_n(i) - I^h_n(i-k)$ decreases as n increases.

(ii) Letting $i = 2k+1$ in (2.11), then

$$[I_{n+1}(2k) - I_{n+1}(k)] - [I_n(2k) - I_n(k)] = I_{n+1}(2k) - I_{n+1}(2k+1) > 0,$$

where the inequality holds for $n \geq 4k$ because of part (vi) in Theorem 1. Thus, $I_n(k) - I_n(2k)$ decreases as n increases.

(iii) Similar to (ii), part (iii) could be proved by letting $i = 3k+1$ in (2.11) and using part (vi) in Theorem 2 (i.e., $I^h_{n+1}(3k+1) < I^h_{n+1}(3k)$).

(iv) Similar to (ii), part (iv) could be proved by letting $i = 2k+2$ in (2.11) and using part (v) in Theorem 2 (i.e., $I^h_{n+1}(2k+1) < I^h_{n+1}(2k+2)$). □

Comparison of BIs in systems of different sizes

Considering a $Lin/Con/k/n$:F system and a $Lin/Con/k/(n+1)$:F system with components of the common reliability $0 < p < 1$, we define

$$\Delta I_n(i) = I_n(i) - I_{n+1}(i) \text{ for } 1 \le i \le n, \tag{2.14}$$

$$\Delta \tilde{I}_n(i) = \begin{cases} I_n(i) - I_{n+1}(i) & \text{for } 1 \le i \le \lceil \frac{n}{2} \rceil, \\ I_n(i) - I_{n+1}(i+1) & \text{for } \lceil \frac{n}{2} \rceil + 1 \le i \le n. \end{cases} \tag{2.15}$$

Note that a $Lin/Con/k/(n+1)$:F system has one more component than a $Lin/Con/k/n$:F system. $\Delta I_n(i)$ and $\Delta \tilde{I}_n(i)$ represent two different ways of comparing the BIs of components in these two systems. By definition, $\Delta \tilde{I}_n(i)$ has the symmetric property

$$\Delta \tilde{I}_n(i) = \Delta \tilde{I}_n(n-i+1) \quad \text{for } 1 \le i \le n, \tag{2.16}$$

inheriting the symmetric property of BI in (2.1). Further, because $\Delta \tilde{I}_n(i) = \Delta I_n(i)$ for $1 \le i \le \lceil \frac{n}{2} \rceil$, it is enough to only investigate the properties of $\Delta I_n(i)$. Theorem 2.4 presents the main results related to $\Delta I_n(i)$, which demonstrates how the BI value of component i changes as n increases.

Theorem 2.4. *For a $Lin/Con/k/n$:F system and a $Lin/Con/k/(n+1)$:F system with components of the common reliability $0 < p < 1$ and $n > 2k$, the following statements hold.*

(i) $\Delta I_n(i) > 0$ *(i.e., $I_n(i)$ decreases as n increases) and $\Delta I_n(i)$ approaches 0 (i.e., $I_{n+1}(i) \approx I_n(i)$) as p approaches 1 for $1 \le i \le n-k$.*

(ii) $\Delta I_n(i)$ *decreases (i.e., the decreasing rate of $I_n(i)$ slows down) as n increases for $1 \le i \le n-2k$.*

Proof. (i) Chang et al. (2002) proved that for $1 \le i \le n-k+1$,

$$I_{n+1}(i) = pI_n(i) + pqI_{n-1}(i) + \cdots + pq^{k-1}I_{n-k+1}(i).$$

Then, for $1 \le i \le n-k$, we have

$$\begin{aligned} I_{n+1}(i) &= pI_n(i) + q\left[pI_{n-1}(i) + pqI_{n-2}(i) + \cdots + pq^{k-2}I_{n-k+1}(i)\right] \\ &= pI_n(i) + q\left[pI_{n-1}(i) + \cdots + pq^{k-2}I_{n-k+1}(i) + pq^{k-1}I_{n-k}(i)\right] - pq^kI_{n-k}(i) \\ &= pI_n(i) + qI_n(i) - pq^kI_{n-k}(i) \\ &= I_n(i) - pq^kI_{n-k}(i). \end{aligned}$$

Thus, we have

$$\Delta I_n(i) = pq^kI_{n-k}(i), \text{ for } 1 \le i \le n-k. \tag{2.17}$$

Note that Chang and Hwang (2003) gave (2.17) without proof. From (2.17), it is straightforward that $\Delta I_n(i) > 0$ since $I_{n-k}(i) > 0$ for $i \le n-k$. Then, by (2.14), we have $I_n(i) > I_{n+1}(i)$, which means that $I_n(i)$ decreases as n increases for any component $1 \le i \le n-k$. Moreover, pq^k approaches 0 as p approaches 1 and $I_{n-k}(i)$ in (2.17) is bounded; thus $\Delta I_n(i)$ approaches 0 as p approaches 1, which means that $I_{n+1}(i) \approx I_n(i)$ when p is close to 1, for $1 \le i \le n-k$.

(ii) According to part (i), the value $I_{n-k}(i)$ decreases as n increases for $1 \le i \le n-2k$. Then, for fixed p and k, the value $\Delta I_n(i)$ in (2.17) decreases as n increases for all $1 \le i \le n-2k$. □

Figure 2.6 demonstrates the changes of the BI values for the $Lin/Con/4/n$:F systems for $n = 14, 15, 16, 17$, and 18, in which (a) and (b) are for the case of $p = 0.2$, and (c) and (d) for $p = 0.7$. As in Theorem 2.4, Figures 2.6(a) and (c) draw the BI values for components $1 \le i \le n-k$, and (b) and (d) show the changes of $\Delta I_n(i)$ with n for components $1 \le i \le n-2k$. Figure 2.6(a) clearly verifies that $I_n(i)$ decreases as n increases for $1 \le i \le n-k$ as shown in part (i) in Theorem 2.4. Further, Figure 2.6(b) indicates that $\Delta I_n(i)$ is positive and decreases as n increases for $1 \le i \le n-2k$, that is, the decreasing rate of $I_n(i)$ slows down as n increases, as shown in part (ii) in Theorem 2.4. When $p = 0.7$, a relatively high component

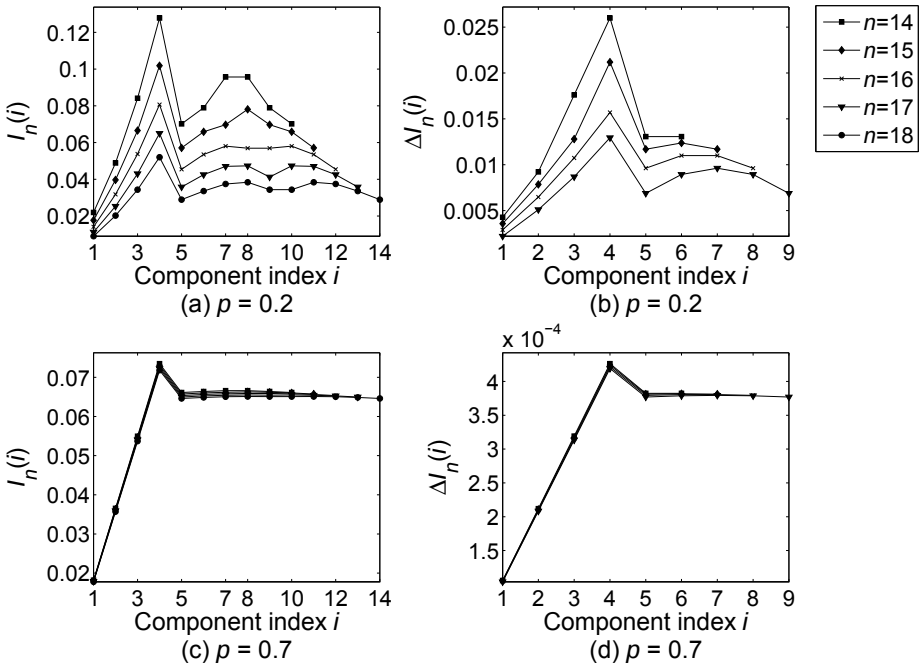

Figure 2.6: The changes of BIs for $Lin/Con/4/n$:F systems for $n = 14, 15, 16, 17$, and 18

reliability, Figure 2.6(c) shows that $I_{n+1}(i) \approx I_n(i)$, and Figure 2.6(d) shows that $\Delta I_n(i)$ is almost zero ($< 5 \times 10^{-4}$). That is, for a given component, there is no significant decrease of its BI value in the $Lin/Con/4/n$:F systems as n changes from 14 to 18 as proved in part (i) in Theorem 2.4.

2.3 Disproved patterns and conjectures

Based on computations of the BIs for the $Lin/Con/k/n$:F systems with $3 \le k \le 8$, $1 \le n \le 50$, and also $n = 60, 70, \ldots, 200$, Chang et al. (2002) disproved some patterns by counterexamples and made some conjectures. Along this line, we conduct more comprehensive computational tests with $3 \le k \le 20$, $2k+1 \le n \le 10k+1$, and $p = 0.01, 0.02, \ldots, 0.99$, and, consequently, more patterns are disproved and conjectured. These tests are computed using symbolic operations in Matlab and thus are accurate enough even when p is small.

Our new observations concerning the BI patterns in Section 2.2 give a better understanding of the disproved patterns and conjectures. With the knowledge of the segments and their lengths and peaks of the BI patterns, we can effectively find the counterexamples and intuitively disprove some previous conjectures. All the patterns that have been disproved and conjectured (including our new findings) are summarized in Subsections 2.3.1 and 2.3.2, respectively.

2.3.1 Disproved patterns

For a $Lin/Con/k/n$ (either F or G) system, the following previously claimed or conjectured patterns are now disproved by counterexamples. These patterns are disproved in the sense that they do not always hold for all the specified values of parameters (i.e., p, t, k, and/or i). Detailed illustrations on each of disproved patterns (2.18) – (2.31) are provided below. Note that if a BI pattern is disproved for the combinatorial case, it does not hold for the half-line and uniform cases because the

combinatorial BI is a special case of the half-line BI and uniform BI; therefore, in this situation, we will only claim the disproof of the pattern for the combinatorial case.

$$k+1 \succ_u n-2k \qquad \text{for } 2k+2 \le n < 3k \tag{2.18}$$

$$tk \succ_c tk+1 \qquad \text{for } t \ge 4 \tag{2.19}$$

$$tk \succ_c tk-1 \qquad \text{for } n = 2tk-1 \text{ and } t \ge 3 \tag{2.20}$$

$$tk+2 \succ_u tk+1 \qquad \text{for } t \ge 4 \tag{2.21}$$

$$i \succ_u tk+1 \qquad \text{for } i > tk+1 \text{ and } t \ge 3 \tag{2.22}$$

$$i+1 \succ_u i \qquad \text{for } 2k < i < 3k-1 \tag{2.23}$$

$$i+1 \succ_c i \qquad \text{for } 3k < i < 4k-1 \tag{2.24}$$

$$tk \succ_c (t+1)k \qquad \text{for } t \ge 3 \tag{2.25}$$

$$k+1 \succ_u k-1 \tag{2.26}$$

$$i+1 \succ_u i \qquad \text{for } k < i < 2k \tag{2.27}$$

$$3k \succ_u 3k+1 \tag{2.28}$$

$$2k \succ_u 3k \tag{2.29}$$

$$2k \succ_u i \qquad \text{for } i > 2k \tag{2.30}$$

$$tk \succ_u tk+1 \qquad \text{for } n = 2tk+1 \text{ and } t \ge 4 \tag{2.31}$$

Chadjiconstantinidis and Koutras (1999) claimed that they proved (2.18), which is, however, a false statement, as disproved by our next counterexample.

Example 1. For $k = 4, n = 11$, and $p = 0.01, 0.02, \ldots, 0.34$, $I_n(k+1) < I_n(n-2k)$. For example, when $p = 0.01$, $I_n(k+1) = 0.0006810248 < I_n(n-2k) = 0.0008673900$.

Chang et al. (2002) disproved (2.19), (2.20), and (2.21) using numerical counterexamples. For (2.21), we find another counterexample as shown in Example 2 in which t is reduced to 4 compared with $t = 7$ in the counterexample in Chang et al.

(2002). Note that (2.21) is disproved only for $t \geq 4$ and is conjectured for $t = 2$ and 3 as shown in conjectures (2.37) and (2.39) in Subsection 2.3.2.

Example 2. For $k = 18, n = 145$, and $p = 0.06, 0.07, 0.08$, $I_n(4k+2) < I_n(4k+1)$. For example, when $p = 0.06$, $I_n(4k+2) = 0.0017098693 < I_n(4k+1) = 0.0017098709$.

Uniform BI pattern (2.22) is conjectured for $t = 1$ and 2 as shown in conjectures (2.36) and (2.35) in Subsection 2.3.2, and is disproved for $t \geq 3$ by Chang et al. (2002). They also disproved patterns (2.23), (2.24), and (2.25). Although the half-line BI patterns in parts (i), (ii), and (vi) in Lemma 2 have been proved, the corresponding uniform BI patterns (2.26) – (2.28) do not hold. Chang et al. (2002) disproved (2.26) and (2.27), and our next counterexample disproves (2.28).

Example 3. For $k = 14, n = 92$, and $p = 0.03, 0.04, \ldots, 0.1$, $I_n(3k) < I_n(3k+1)$. For example, when $p = 0.03$, $I_n(3k) = 0.0000492933 < I_n(3k+1) = 0.0000493310$.

Because the half-line BI pattern in part (iv) in Lemma 2 has been proved, the corresponding uniform BI patterns (2.29) and (2.30) were conjectured by Chang et al. (2002). In fact, pattern (2.29) is a special case of (2.30) and can be disproved by Example 4. Chang and Hwang (2002) disproved (2.30) by a counterexample of the rare-event importance, which is equivalent to the BI as p approaches 0.

Example 4. For $k = 15, n = 96$, and $p = 0.04, 0.05, 0.06, 0.07$, $I_n(2k) < I_n(3k)$. For example, when $p = 0.04$, $I_n(2k) = 0.0004558502 < I_n(3k) = 0.0004559212$.

Motivated by parts (iv)c, (vi), and (viii) in Lemma 1 where $tk \succ_u tk+1$ for $t = 1, 2$, and 3, respectively, Chang et al. (2002) conjectured (2.31) for all $t \geq 4$. However, Example 5 disproves it.

Example 5. For $k = 4, n = 57$, $t = 7$, and $p = 0.08, 0.09, \ldots, 0.46$, $I_n(7k) < I_n(7k+1)$. For example, when $p = 0.30$, $I_n(7k) = 0.0006396002 < I_n(7k+1) = 0.0006396003$.

Our observations to the nature of BI patterns in Section 2.2 facilitate the process of seeking counterexamples. Take the disproved patterns (2.28) and (2.29) as examples.

For a period of time, one can neither prove nor disprove these patterns. Now, utilizing the concepts of segments, we can conclude that these patterns have been conjectured inappropriately. As stated in Subsections 2.2.2 and 2.2.3, the length of segments may decrease, especially when the component reliability p is small. If the length of the third segment decreases to $k-1$ or smaller, then both components $3k$ and $3k+1$ are in the increasing part of the fourth segment, resulting in $I_n(3k+1) > I_n(3k)$. Following this analysis, we use small p values and easily find the counterexample to (2.28), as shown in Example 3. As for (2.29), if p is small enough, the peak of the second segment may appear earlier than component $2k$ as discussed in Remark 2(ii). Consequently, component $2k$ is not the peak of the second segment any more. Then, the BI value of component $2k$ could be less than the BI values of the peak and its surrounding components in the third segment. Thus, the counterexample for (2.29) is found by using small p values as in Example 4.

2.3.2 Conjectures

We present the following conjectures on the BI patterns for a $Lin/Con/k/n$ (either F or G) system. We neither prove nor find counterexamples to disprove them. Chang et al. (2002) made the conjectures of (2.32) – (2.36) and (2.38), and we make the new conjectures of (2.37) and (2.39) for the reasons illustrated below.

$$2k-1 \succ_c 2k+1 \quad \text{for } k \geq 4 \tag{2.32}$$

$$i+1 \succ_c i \quad \text{for } 2k < i < 3k-1 \tag{2.33}$$

$$i \succ_c tk+1 \quad \text{for } i > tk+1 \text{ and } t \geq 2 \tag{2.34}$$

$$i \succ_u 2k+1 \quad \text{for } i > 2k+1 \tag{2.35}$$

$$i \succ_u k+1 \quad \text{for } i > k+1 \tag{2.36}$$

$$2k+2 \succ_u 2k+1 \tag{2.37}$$

$$2k+1 \succ_u k+1 \tag{2.38}$$

$$3k+2 \succ_u 3k+1 \tag{2.39}$$

Note that (2.33) and (2.34) cannot be extended to the uniform case as in disproved patterns (2.23) and (2.22), respectively, and (2.33) can not be extended to the next k consecutive components as in disproved pattern (2.24). Conjecture (2.35) is an extension of (2.34) with $t = 2$. Uniform BI patterns (2.36) and (2.37) are conjectured because the corresponding half-line BI patterns in parts (iii) and (v) in Lemma 2 have been proved. Patterns (2.37) and (2.38) are important special cases of (2.35) and (2.36), respectively. It is worthy to present them separately because it may be relatively easier for the future research to prove the special cases than the general cases. Pattern (2.39) is an extension of (2.37) and is conjectured because we do not find its counterexample in our computational tests, but related pattern (2.21) has been disproved.

The above conjectures do not have obvious violations against our findings about the BI patterns in Section 2.2. For example, conjecture (2.37) is consistent with the observation that the length of the second segment is always k as in Remark 2(ii). We notice that all conjectures except for (2.34) are made for the first few segments rather than for all segments. Because the length of segments varies and complex increasing-then-decreasing patterns exist for the general case, it becomes very hard to make the conjectures which may hold in general. Further, conjecture (2.34) is made for the case of $p = \frac{1}{2}$ only, which may simplify the change of the length of segments and the complexity of the increasing-then-decreasing patterns, and thus make (2.34) possibly hold for all segments under the special condition of $p = \frac{1}{2}$.

2.4 Summary

This chapter summarizes the current existing BI patterns for $Lin/Con/k/n$ systems, disproves some plausible patterns using counterexamples, and gives illustrations

based on our findings on the BI patterns. Through theoretical analysis, we prove new BI patterns conditioned on the value of p (see Theorem 2.1). Based on both computational tests and systematic analysis on the rules of BI patterns, we make conjectures as shown in (2.37) and (2.39).

An investigation has been conducted to explore the nature of BI patterns for $Lin/Con/k/n$:F systems. We observe that the "fixed-length BI ordering" that is based on the k-interval division and expected for a period of time does not exist. This chapter proposes a nonequal-length segment division method and discovers the increasing-then-decreasing pattern in each segment. Both theoretical and numerical studies on the relationship between the BI patterns and p and the relationship between the BI patterns and n have been conducted. It is observed that BI patterns are sensitive to the value of p. The relationship between the BI patterns and p also shows that in the half-line case only the BI values of components in the first few segments make a difference, and that the components in the middle of the n-component line have almost the same BI values when n is large or p is high.

These new findings make a better understanding of the BI patterns (e.g., see Remarks 2 and 3) as well as help to intuitively disprove some conjectures and effectively find the counterexamples (see Section 2.3). Because the BI measures the contribution of a component to the system reliability, it can be used to direct the assignment of components or the determination of component reliabilities when designing $Lin/Con/k/n$ systems. For example, according to Remark 1, an engineer, at least in the early stage of design, should place the most reliable component in position k and the least reliable component in position 1, and assign the components from positions 1 to k in the increasing order of component reliabilities. The relationship between the BI patterns and p inspires the design of $Lin/Con/k/n$ systems with $p \geq \frac{1}{2}$ to focus on the first few segments and pay less resources in designing the middle components of the n-component line.

In addition, the relationship between the BI patterns and n is useful for the redesign of existing $Lin/Con/k/n$ systems. For example, consider the problem of

shrinking (expanding) a $Lin/Con/k/n$:F system to a $Lin/Con/k/n'$:F system with $n' < n$ ($n' > n$) by removing $n-n'$ (adding $n'-n$) components in the middle. Suppose that the component assignment in the $Lin/Con/k/n$:F system is optimal. According to Theorem 2.4, when the system size n is large or the component reliabilities are high, the component assignment in the $Lin/Con/k/n'$:F system remains or is close to be optimal because the BI patterns do not change significantly. Otherwise, the BI patterns change significantly, and thus the component assignment should be reexamined to account for the new BI patterns in the $Lin/Con/k/n'$:F system.

As future research, the conjectures presented in Subsection 2.3.2 need to be proved. There is a need for further theoretical work on the length of segments and the positions of peaks. Moreover, it is also worthy to study the theoretical aspects rather than the computational tests on how the value of p affects the BI patterns.

Chapter 3

BI-based heuristics for the CAP[†]

Given a set of n functionally exchangeable components with different reliabilities, the CAP is to determine the allocation or rearrangement of these n components into the n positions in the system with the objective of maximizing the system reliability. There is no effective exact optimization method except the enumeration method which evaluates all $n!$ possible assignments. In this situation, heuristics become more important for solving CAPs to find optimal or sub-optimal solution in a reasonable time. The BI has been applied to design heuristics for CAPs by trying to assign more reliable components to the positions with larger BI values (Kontoleon, 1979; Lin and Kuo, 2002; Zuo and Kuo, 1990; Zuo and Shen, 1992).

This chapter focuses on the BI-based heuristics for the CAP. We propose three new heuristics based on the scheme of the LKA heuristic (Lin and Kuo, 2002) and two new heuristics based on the schemes of the ZKA and ZKB heuristics (Zuo and Kuo, 1990). We investigate the properties of these heuristics, particularly the relations between invariant optimal arrangements and the arrangements generated by these heuristics. Significantly, we conduct comprehensive numerical tests to evaluate the performances of all these heuristics on various coherent systems and different types

[†]Reused with permission from Qingzhu Yao, Xiaoyan Zhu, and Way Kuo (2011) Heuristics for component assignment problems based on the Birnbaum importance, IIE Transactions, in press. Copyright IIE Transactions 2011.

of components, filling the gap in this field. Based on statistical analysis of this part of computational results, we make suggestions on choosing appropriate heuristics and further propose a BI-based two-stage approach (BITA) with each stage using different BI-based heuristics. The BITA is evaluated by various computational tests (including both small and large systems), benchmarking on the GAMS/CoinBonmin solver and a randomization method. This study does not further investigate the heuristic by Kontoleon (1979) because our preliminary numerical tests show that the LKA heuristic always outperforms it.

The following notation and assumptions are used in the rest of this dissertation.

Notation

N	$= \{1, 2, \ldots, n\}$, the index set of n positions
π_i	The index of the component assigned to position $i, i \in N$
$\boldsymbol{\pi}$	$= (\pi_1, \pi_2, \ldots, \pi_n)$, a permutation of components $1, 2, \ldots, n$
$\boldsymbol{\pi}(i, j)$	a permutation formed from permutation $\boldsymbol{\pi}$ by interchanging the integers in positions i and j, $i, j \in N$
$\hat{p}_i$	the reliability of available *component* $i, i = 1, 2, \ldots, n$
p_i	$= \hat{p}_{\pi_i}$, the reliability of *position* i, i.e., the reliability of component π_i, the one assigned to position $i, i \in N$
$\mathbf{p}_{\boldsymbol{\pi}}$	$= (\hat{p}_{\pi_1}, \hat{p}_{\pi_2}, \ldots, \hat{p}_{\pi_n})$, vector of the position reliabilities specified by a permutation $\boldsymbol{\pi}$
$I(i; \cdot)$	the BI of position i under the position reliability vector $\cdot$, $i \in N$
$I(i; p)$	the BI of position i with all positions having a common reliability p, $i \in N$
$R(\cdot)$	the system reliability under the position reliability vector $\cdot$

For simplification, the reliability (BI) of the component in position i is referred to as the reliability (BI) of position i. Using the above notation, the BI of position i, $i \in N$, can also be calculated as (Birnbaum, 1969)

$$I(i; \mathbf{p}_{\boldsymbol{\pi}}) = R(\hat{p}_{\pi_1}, \ldots, \hat{p}_{\pi_{i-1}}, 1, \hat{p}_{\pi_{i+1}}, \ldots, \hat{p}_{\pi_n}) - R(\hat{p}_{\pi_1}, \ldots, \hat{p}_{\pi_{i-1}}, 0, \hat{p}_{\pi_{i+1}}, \ldots, \hat{p}_{\pi_n}). \tag{3.1}$$

Assumptions

1. The system is coherent and consists of n positions.
2. Components are functionally interchangeable and can be assigned to any position.
3. The number of available components equals the number of positions, and each position can be assigned exactly one component.
4. Component reliabilities are independent of positions.
5. Component failures are independent.
6. The component reliabilities are in nondecreasing order, i.e., $\hat{p}_1 \leq \hat{p}_2 \leq \cdots \leq \hat{p}_n$.

According to Assumption 6, $\pi_i < \pi_j$ implies that the component reliability in position i is not higher than that in position j, i.e., $\hat{p}_{\pi_i} \leq \hat{p}_{\pi_j}$.

3.1 LK type heuristics

This section reviews the LKA heuristic (Lin and Kuo, 2002) and proposes three new heuristics based on its scheme. We call these four heuristics LK type heuristics.

3.1.1 The LKA heuristic

The LKA heuristic uses the BI as an indicator to assign components to system positions one by one. The detailed procedure is shown below, which generates a solution of arrangement $\boldsymbol{\pi}$.

The LKA heuristic

1. Initially, all positions are assigned component 1, i.e., $\pi_i = 1$ for $i \in N$.
2. Let $S = N$.
3. Do loop, for $k = n$ to 2
   (a) compute $I(i; \mathbf{p}_{\boldsymbol{\pi}})$ for all $i \in S$ according to (3.1),
   (b) find position m_k such that $I(m_k; \mathbf{p}_{\boldsymbol{\pi}}) = \max_{i \in S} I(i; \mathbf{p}_{\boldsymbol{\pi}})$ (break tie arbitrarily),
   (c) let $S = S \backslash \{m_k\}$,
   (d) assign component k to position m_k, i.e., $\pi_{m_k} = k$.

Initially, all positions are temporarily assigned component 1, the least reliable component. Set S stands for the collection of the positions that are not assigned any components rather than component 1. At each iteration in Step 3, the BIs of the positions in S are computed, and the available component of the highest reliability is assigned to the position with the largest BI in S. This process is repeated until all components are assigned to the system.

3.1.2 Three new LK type heuristics

The rationale of the LKA heuristic is that a position with a larger BI should be assigned a component with higher reliability. Continuing with the scheme of the LKA heuristic, we propose three alternative ways to implement the rationale of the LKA heuristic by modifying the initialization in Step 1 and/or the assignment rule in Step 3. Then, as shown in Table 3.1, three new LK type heuristics are generated, namely the LKB, LKC, and LKD heuristics. Each of them is delineated by replacing the partial contents in Steps 1, 3, 3b, and 3d of the LKA heuristic with the corresponding contents of the new heuristics in Table 3.1.

Table 3.1: The LK type heuristics

Heu.	Step 1	Step 3	Step 3b	Step 3d
LKA	component 1, i.e., $\pi_i = 1$	n to 2	$I(m_k; \mathbf{p}_{\boldsymbol{\pi}}) = \max_{i \in S} I(i; \mathbf{p}_{\boldsymbol{\pi}})$	component k to position m_k, i.e., $\pi_{m_k} = k$
LKB	component n, i.e., $\pi_i = n$	1 to $n-1$	$I(m_k; \mathbf{p}_{\boldsymbol{\pi}}) = \min_{i \in S} I(i; \mathbf{p}_{\boldsymbol{\pi}})$	component k to position m_k, i.e., $\pi_{m_k} = k$[a]
LKC	component 1, i.e., $\pi_i = 1$[a]	2 to n	$I(m_k; \mathbf{p}_{\boldsymbol{\pi}}) = \min_{i \in S} I(i; \mathbf{p}_{\boldsymbol{\pi}})$	component k to positions in S, i.e., $\pi_i = k$ for $i \in S$
LKD	component n, i.e., $\pi_i = n$	$n-1$ to 1	$I(m_k; \mathbf{p}_{\boldsymbol{\pi}}) = \max_{i \in S} I(i; \mathbf{p}_{\boldsymbol{\pi}})$[a]	component k to positions in S, i.e., $\pi_i = k$ for $i \in S$

[a]No change is made in this step compared to the LKA heuristic.

The LKB heuristic initializes all positions by the most reliable component (i.e., component n) and then starts the assignment from the least reliable component, i.e., assigning the current available component of the lowest reliability to position m_k, which has the smallest BI among the positions currently with component n. The LKC heuristic starts from the same initialization as the LKA heuristic but uses a different assignment rule, which remains the assignment of position m_k that has the smallest BI in S and allocates the least reliable unassigned component to all other positions in S. The LKD heuristic uses the same assignment rule as the LKC heuristic, but it iterates differently from the most reliable component to the least reliable component.

Lin and Kuo (2002) proved that the existence of an invariant optimal arrangement for a system guarantees the LKA heuristic to produce the invariant optimal arrangement. This statement holds in general but can fail for a special case where more than one position achieve the same largest BI in Step 3b in some iteration. Under this situation (i.e., a tie is present in Step 3b in some iteration), the LKA heuristic may choose a position which is not consistent with the invariant optimal arrangement and, consequently, the generated arrangement may be not an invariant optimal and even not optimal arrangement, as shown in Example 6 below.

Example 6. Consider the CAP instance of assigning seven components with reliabilities (0.806, 0.809, 0.818, 0.833, 0.853, 0.925, 0.934) to the $Lin/Con/2/7$:F system which has an invariant optimal arrangement $\boldsymbol{\pi}^* = (1, 7, 3, 5, 4, 6, 2)$ (treating the inverse $(2, 6, 4, 5, 3, 7, 1)$ as the same invariant optimal arrangement due to the symmetry of the $Lin/Con/2/7$:F system). Under the invariant optimal arrangement, the maximal system reliability is 0.910892. To solve this instance using the LKA heuristic, any tie in Step 3b is broken by choosing the position with the lowest index among tied positions. In Step 3 of the LKA heuristic, at the first iteration ($k = 7$), positions 2 and 6 have the same largest BI value (i.e., a tie), and the rule of breaking ties chooses position 2 to receive the candidate component (i.e., component $k = 7$). At the following four iterations ($k = 6, 5, 4, 3$), there is no tie and the intermediate

assignment after all these five iterations is $(1, 7, 3, 5, 4, 6, 1)$, that is so far consistent with invariant optimal arrangement $\boldsymbol{\pi}^*$. At the last iteration $(k=2)$, a tie is present that positions 1 and 7 have the same BI value, and the rule of breaking ties chooses position 1 to receive component $k = 2$. Thus, the final generated arrangement is $(2, 7, 3, 5, 4, 6, 1)$ with the system reliability of 0.910868, which is not even an optimal arrangement.

Therefore, the statement and its proof in Lin and Kuo (2002) must be amended by adding the condition that no tie is present in Step 3b in all iterations, as shown in Theorem 3.1 below. Meanwhile, Theorem 3.1 also states that the same property can be proved for the LKB, LKC, and LKD heuristics. Note that as in the proof of Theorem 3.1, if no such tie is present in Step 3b, the LK type heuristics assign the components according to the invariant optimal arrangement.

Theorem 3.1. *If a system admits an invariant optimal arrangement, the arrangement generated by each of the LKA, LKB, LKC, and LKD heuristics is the invariant optimal arrangement given that no tie is present in Step 3b in all iterations.*

Proof. We only prove that the statement is true for the LKB heuristic by contradiction, and the proofs for the LKA, LKC, and LKD heuristics can be similarly completed. Recall that in the kth iteration, Step 3d of the LKB heuristic in Table 3.1 assigns component k to position m_k for $k = 1, 2, \ldots, n-1$.

Assume to the contrary that the arrangement specified by m_k for $k = 1, 2, \ldots, n-1$ (i.e., generated by the LKB heuristic) does not match any invariant optimal arrangement. Consider an arbitrary invariant optimal arrangement, which assigns component k to position i_k for $k = 1, 2, \ldots, n$. Then, there exists an integer r, $1 \leq r < n$, such that $m_k = i_k$ for $k = 1, 2, \ldots, r-1$ and $m_r \neq i_r$. Furthermore, $\hat{p}_r < \hat{p}_n$; otherwise (i.e., $\hat{p}_r = \hat{p}_n$), $\hat{p}_r = \hat{p}_{r+1} = \cdots = \hat{p}_n$ by Assumption 6, which means that any assignment of components r to n after the $(r-1)$th iteration would make the arrangement generated by the LKB heuristic invariant optimal.

If this is the case, then the invariant optimal arrangement implies

$$R((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}},(\hat{p}_r)_{i_r};\hat{p}_n) \;\geq\; R((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}},(\hat{p}_r)_{m_r};\hat{p}_n), \qquad (3.2)$$

where the notation $((\hat{p}_1)_{i_1},\ldots,(\hat{p}_r)_{i_r};\hat{p}_n)$ denotes the vector of the position reliabilities where all positions have a common reliability $\hat{p}_n$ except that positions $i_1, i_2, \ldots, i_r$ have different reliabilities $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_r$, respectively. Using the relationship between system reliability and the BI in Lin and Kuo (2002) on both sides of (3.2), Equation (3.2) becomes

$$\begin{aligned}
&R((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n) + (\hat{p}_r-\hat{p}_n)I(i_r;((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n))\\
\geq &R((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n)+(\hat{p}_r-\hat{p}_n)I(m_r;((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n)).
\end{aligned}$$

Because $\hat{p}_r < \hat{p}_n$, then

$$I(i_r;((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n)) \leq I(m_r;((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n)),$$

which contradicts the fact that in Step 3b of the rth iteration,

$$I(m_r;\mathbf{p}_{\boldsymbol{\pi}}) < I(i;\mathbf{p}_{\boldsymbol{\pi}}),$$

for any $i \in S = N\backslash\{i_1,\ldots,i_{r-1}\}$ where $\mathbf{p}_{\boldsymbol{\pi}} = ((\hat{p}_1)_{i_1},\ldots,(\hat{p}_{r-1})_{i_{r-1}};\hat{p}_n)$, noting that $I(m_r;\mathbf{p}_{\boldsymbol{\pi}})$ is strictly less than $I(i;\mathbf{p}_{\boldsymbol{\pi}})$ because no tie is present. Therefore, the arrangement generated by the LKB heuristic is the invariant optimal arrangement. □

In fact, it is equivalent for the LK type heuristics to use the improvement potential or criticality importance measure instead of the BI because the improvement potential and criticality importance measures of position i equal $(1-p_i)I(i;\mathbf{p}_{\boldsymbol{\pi}})$ and $\frac{p_i}{R(\mathbf{p}_{\boldsymbol{\pi}})}I(i;\mathbf{p}_{\boldsymbol{\pi}})$ (Kuo and Zuo, 2002), respectively, and in Step 3a, all positions $i \in S$ temporarily have the same reliability. Therefore, the importance ordering of positions in S is the same according to the BI, improvement potential, or criticality importance

measure, and the heuristics choose the same position m_k in Step 3b. In turn, even if the LK type heuristics use the improvement potential or criticality importance measure, Theorem 3.1 still holds true essentially due to the above relations of these two importance measures to the BI. In solving the CAP, the BI is better than the other two because they are built upon the BI.

3.2 ZK type heuristics

This section reviews the ZKA and ZKB heuristics (Zuo and Kuo, 1990) and proposes two new heuristics based on their scheme. We call these heuristics ZK type heuristics.

3.2.1 The ZKA and ZKB heuristics

The ZKA and ZKB heuristics try to match the BI ordering by pairwise exchanges of components. They are originally designed for the CAPs in $Lin/Con/k/n$ (either F or G) systems and can be extended to solve the CAPs in any coherent system. The detailed procedure is shown below.

The ZKA and ZKB heuristics

1. Generate an initial arrangement $\boldsymbol{\pi}$.
2. Compute $I(i;\mathbf{p}_{\boldsymbol{\pi}})$ for all positions $i \in N$ according to (3.1).
3. Do loop, for $k = 1$ to $n-1$
   (a) (*Alternative 1*:ZKA) Find positions i and j such that $\pi_i = k$ and $\pi_j = k+1$,
   (*Alternative 2*:ZKB) Find positions i and j such that $\pi_i = k$ and $I(j;\mathbf{p}_{\boldsymbol{\pi}}) = \min_{r:p_r>p_i} I(r;\mathbf{p}_{\boldsymbol{\pi}})$ (break tie arbitrarily),
   (b) If $I(i;\mathbf{p}_{\boldsymbol{\pi}}) > I(j;\mathbf{p}_{\boldsymbol{\pi}})$ and $R(\mathbf{p}_{\boldsymbol{\pi}(i,j)}) > R(\mathbf{p}_{\boldsymbol{\pi}})$, then exchange the positions of components π_i and π_j, i.e., $\boldsymbol{\pi} = \boldsymbol{\pi}(i,j)$, and update the position BIs.
4. If there is no exchange in Step 3, stop; otherwise go to Step 3.

Note that the ZKA and ZKB heuristics use Alternatives 1 and 2 in Step 3a, respectively. Recall that the LK type heuristics and the heuristic in Kontoleon (1979)

start from an infeasible arrangement that duplicates a single component and assigns it to all positions, and obtain a feasible arrangement only at the end of the heuristic procedures that assigns each component to a distinct position. Differently, the ZKA and ZKB heuristics start from a feasible initial arrangement in Step 1 (i.e., each component is initially assigned to a distinct position) and maintain the feasibility throughout the procedure. Zuo and Kuo (1990) presented four initial arrangements below for solving the CAPs in $Lin/Con/k/n$ systems.

Initial Arrangement 1: $(1, n-1, 3, n-3, \ldots, n-2, 4, n, 2)$.
Initial Arrangement 2: $(1, n, 3, n-2, \ldots, n-3, 4, n-1, 2)$.
Initial Arrangement 3: $(1, 3, 5, \ldots, n, \ldots, 6, 4, 2)$.
Initial Arrangement 4: $(1, 2, 3, 4, \ldots, n-1, n)$.

Zuo and Kuo (1990) claimed that Initial Arrangement 1 is the B-importance ordering of a consecutive-2-out-of-n ($Con/2/n$) system; however, Initial Arrangement 1 is neither the B-importance ordering of linear nor circular $Con/2/n$ systems. Initial Arrangement 2 is the invariant optimal arrangement of a $Lin/Con/2/n$:F system. Initial Arrangement 3 is an arrangement found to be optimal for many tests on the $Lin/Con/k/n$ systems in Zuo and Kuo (1990). Initial Arrangement 4 is a naturally ordered arrangement. We introduce more initial arrangements in Subsection 3.3.4.

Step 3 conducts pairwise exchanges iteratively from the least reliable component to the most reliable component and is repeated until no more exchange can be made (see Step 4). In loop k of Step 3, two alternative methods are provided in Step 3a to find positions i and j for exchange. Alternative 1 associated with the ZKA heuristic compares component k which is in position i with the next more reliable component $k+1$ which is in position j, while Alternative 2 associated with the ZKB heuristic compares component k in position i with component π_j in position j that has the smallest BI among the positions whose reliabilities are higher than the reliability of component k (i.e., p_i). In either alternative/heuristic, if the BI of the less reliable component π_i is larger than the BI of the more reliable component π_j and the exchange

of these two components can improve the system reliability, then exchange them (see Step 3b).

3.2.2 Two new ZK type heuristics

The ZKA and ZKB heuristics start pairwise exchanges from the least reliable component. We propose two derivative methods, the ZKC and ZKD heuristics, which start pairwise exchange from the most reliable component and use the revised Alternatives 1 and 2, respectively. The ZKC (ZKD) heuristic is delineated by replacing the partial contents in Steps 3, 3a, and 3b of the ZKA (ZKB) heuristic with the corresponding contents as shown in Table 3.2.

The new Alternative 1 associated with the ZKC heuristic compares component $\pi_i = k$ with the next less reliable component $\pi_j = k - 1$, while the new Alternative 2 associated with the ZKD heuristic compares component $\pi_i = k$ with component π_j in position j that has the largest BI among the positions whose reliabilities are lower than p_i. Noting that $\hat{p}_{\pi_i} \geq \hat{p}_{\pi_j}$ in both new alternatives, if the BI of component π_i is smaller than the BI of component π_j and the exchange of these two components can improve the system reliability, then exchange them (see Step 3b).

Different from the LK type heuristics, when a system admits an invariant optimal arrangement, the ZK type heuristics may not find the invariant optimal arrangement, as shown in Example 7 for the ZKA and ZKB heuristics.

Example 7. Considering the CAP instance of assigning seven components with reliabilities $\hat{p}_i = 0.1 \times i$ for $i = 1, 2, \ldots, 7$ to the $Lin/Con/2/7$:F system, it has an invariant optimal arrangement $\boldsymbol{\pi}^* = (1, 7, 3, 5, 4, 6, 2)$ (see also Example 6), under which the maximal system reliability is 0.2538. Using Initial Arrangement 1, 2, or 3 in Step 1, both the ZKA and ZKB heuristics generate the optimal arrangement $\boldsymbol{\pi}^*$. However, using Initial Arrangement 4 in Step 1, the ZKA heuristic generates

Table 3.2: The ZK type heuristics

Heu.	Step 3	Step 3a		Step 3b
		Alternative 1	Alternative 2	
ZKA	1 to $n-1$	$\pi_j = k+1$	—	$I(i;\mathbf{p}_{\boldsymbol{\pi}}) > I(j;\mathbf{p}_{\boldsymbol{\pi}})$
ZKB	1 to $n-1$	—	$I(j;\mathbf{p}_{\boldsymbol{\pi}}) = \min_{r:p_r>p_i} I(r;\mathbf{p}_{\boldsymbol{\pi}})$	$I(i;\mathbf{p}_{\boldsymbol{\pi}}) > I(j;\mathbf{p}_{\boldsymbol{\pi}})$
ZKC	n to 2	$\pi_j = k-1$	—	$I(i;\mathbf{p}_{\boldsymbol{\pi}}) < I(j;\mathbf{p}_{\boldsymbol{\pi}})$
ZKD	n to 2	—	$I(j;\mathbf{p}_{\boldsymbol{\pi}}) = \max_{r:p_r<p_i} I(r;\mathbf{p}_{\boldsymbol{\pi}})$	$I(i;\mathbf{p}_{\boldsymbol{\pi}}) < I(j;\mathbf{p}_{\boldsymbol{\pi}})$

arrangement $\boldsymbol{\pi}^1 = (1, 7, 2, 6, 4, 5, 3)$ with the system reliability of 0.2524, and the ZKB heuristic generates arrangement $\boldsymbol{\pi}^2 = (3, 4, 6, 1, 7, 2, 5)$ with the system reliability of 0.1559. Both $\boldsymbol{\pi}^1$ and $\boldsymbol{\pi}^2$ differ from $\boldsymbol{\pi}^*$.

Note that the ZK type heuristics using an initial arrangement with high system reliability may produce a worse final arrangement than using one with low system reliability, as shown in the next example. Therefore, it is not sufficient to judge the performances of initial arrangements by simply comparing the system reliabilities associated with them.

Example 8. Table 3.3 presents the results that the ZKA heuristic solves the instance of assigning ten components with reliabilities $\hat{p}_i = 0.61 + 0.03 \times (i-1), i = 1, 2, \ldots, 10$ to the C3 system (see Figure 3.1(c) in Subsection 3.3.1). As shown in Table 3.3, the system reliability associated with the final arrangement generated using better Initial Arrangement 4 is lower than that generated using worse Initial Arrangement 3.

Table 3.3: Comparison of initial and final arrangements in the ZKA heuristic

Initial Arrangement $\boldsymbol{\pi}$	$R(\mathbf{p}_{\boldsymbol{\pi}})$	Final Arrangement $\boldsymbol{\pi}'$	$R(\mathbf{p}_{\boldsymbol{\pi}'})$
3: $(1, 3, 5, 7, 9, 10, 8, 6, 4, 2)$	0.9781	$(5, 6, 1, 4, 7, 3, 2, 9, 10, 8)$	0.9928
4: $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$	0.9918	$(4, 5, 1, 3, 6, 2, 7, 8, 9, 10)$	0.9927

3.3 Comparisons of LK and ZK type heuristics

This section conducts numerical experiments to evaluate the LK and ZK type heuristics in Tables 3.1 and 3.2, respectively. Section 3.3.1 illustrates the design of the experiments. Section 3.3.2 gives a preliminary example that details the descriptions of the computational results. Sections 3.3.3 and 3.3.4 present the computational results related to the LK and ZK type heuristics, respectively.

3.3.1 Design of experiments

For the computational tests, we choose ten complex systems, as shown in Table 3.4, including four general coherent systems (C1 - C4 systems in Figure 3.1) and six $Lin/Con/k/n$ systems. These systems were previously studied in the field of the CAP. The C1 system in Figure 3.1(a) is given by Boland et al. (1989) and has six components. The C2 and C3 systems in Figures 3.1(b) and 3.1(c) are studied by Lin and Kuo (2002). The C2 system has five components, and the C3 system has ten components and is a parallel combination of the C2 system and a bridge system. The C4 system in Figure 3.1(d) is studied by Prasad and Kuo (2000) and has ten components. The $Lin/Con/k/n$:F and G systems with $k = 2, 3$ and $n = 7, 8$ are studied except for the $Lin/Con/2/7$:F and $Lin/Con/2/8$:F systems for which the invariant optimal arrangements exist (Du and Hwang, 1986; Malon, 1984). The $Lin/Con/k/n$ systems are studied by Zuo and Kuo (1990) and Zuo and Shen (1992). None of these ten tested systems admits invariant optimal arrangements.

In order to see if the performances of the heuristics depend on the component reliabilities, we consider three types of available components – High, Low, and Arbitrary reliable components, which have the component reliabilities randomly generated from uniform distributions on $[0.8, 0.99]$, $[0.01, 0.2]$, and $[0.01, 0.99]$, respectively.

A CAP instance is defined on a given system and involves a set of available components that are of specific reliabilities and are to be assigned to the system. Given a system (one of the ten systems) and a type of component (one of the three types), we generate 100 instances by populating 100 sets of component reliabilities

Table 3.4: Tested systems

C1: Figure 3.1(a)	C2: Figure 3.1(b)	C3: Figure 3.1(c)
C4: Figure 3.1(d)	G1: $Lin/Con/2/7$:G	G2: $Lin/Con/2/8$:G
G3: $Lin/Con/3/7$:G	G4: $Lin/Con/3/8$:G	F1: $Lin/Con/3/7$:F
F2: $Lin/Con/3/8$:F		

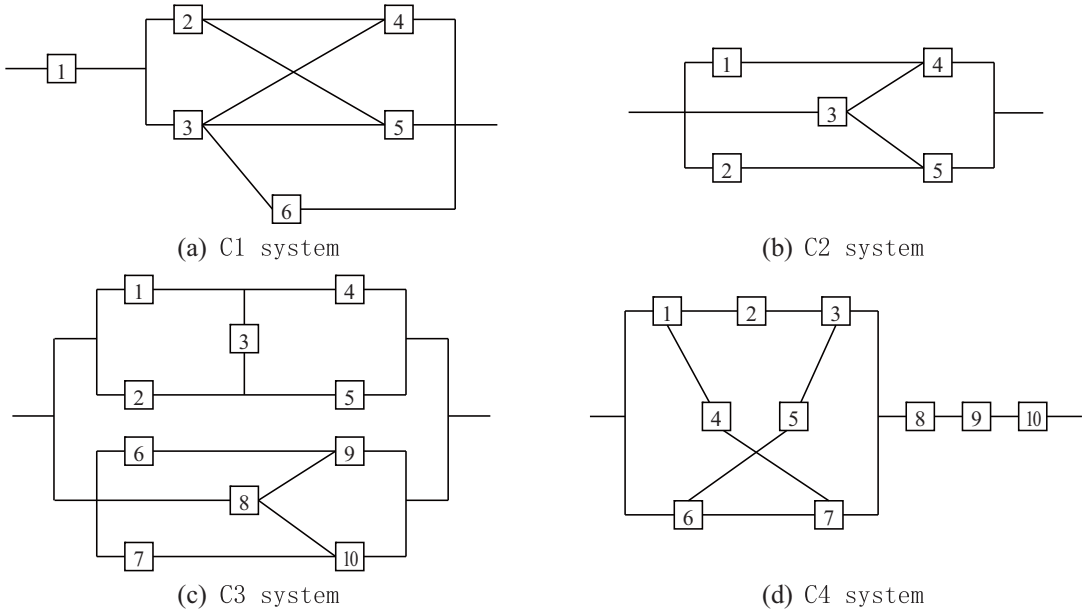

Figure 3.1: Diagrams of four coherent systems

independently according to the distribution specified by the component type. *We call each set of such* 100 *instances a trial, which is characterized by the system and the type of components.* In our computational tests, we consider ten systems in Table 3.4 and three types of components, thus producing $10 \times 3 = 30$ trials with $3,000$ instances.

We solve each CAP instance using each of the eight heuristics in Tables 3.1 and 3.2. To evaluate the quality of the solution generated by a heuristic on a single instance, we calculate a standardized system reliability (SSR):

$$SSR = \frac{R_{heu} - R_{wor}}{R_{opt} - R_{wor}}, \tag{3.3}$$

where R_{heu} stands for the system reliability corresponding to the arrangement generated by the heuristic, R_{opt} the system reliability corresponding to the optimal arrangement, and R_{wor} the system reliability corresponding to the worst arrangement. The enumeration method, which evaluates all the arrangements of components, is used to find the optimal and the worst arrangements (i.e., R_{opt} and R_{wor}). Thus, for each trial, we obtain 100 SSRs when we use a heuristic to solve the 100 instances in the trial, each SSR corresponding to an instance. Since a single instance may be biased, we evaluate the performances of the heuristics at the level of trials. To evaluate the performance of a heuristic on a trial, we use the mean (MSSR), standard deviation, minimum, and maximum over the 100 SSRs associated with the trial, representing the on-average, variability of, worst, and best performances of the heuristic on the trial, respectively.

For example, considering a trial where the system is the $Lin/Con/3/8$:F system and the available components are High reliable, we generate 100 instances by specifying 100 sets of component reliabilities according to uniform distribution on $[0.01, 0.2]$. We use the LKA heuristic to solve the 100 instances in this trial, obtain 100 SSRs, and then evaluate the performance of the LKA heuristic on this trial by calculating the MSSR, standard deviation, minimum, and maximum of these 100 SSRs. The corresponding values are presented in the bold cell in Table 3.5.

Zuo and Kuo (1990) and Zuo and Shen (1992) used a measure $\frac{R_{heu}}{R_{opt}}$ to evaluate the performance of a heuristic on an instance. A drawback of this measure is that it ignores the difference between the worst arrangement and the arrangement generated by a heuristic. For example, we use a heuristic to solve an instance and obtain $R_{heu} = 0.96$, and use the enumeration method to obtain $R_{opt} = 0.98$ and $R_{wor} = 0.94$. Then, the measure $\frac{R_{heu}}{R_{opt}} = \frac{0.96}{0.98} \approx 0.9796$ can lead to the wrong conclusion that this heuristic performs well and is almost 98% accurate. In fact, the heuristic does not perform well and only achieves the middle point between the optimal and the worst arrangements, which is precisely demonstrated by our $SSR = \frac{0.96-0.94}{0.98-0.94} = 0.5$.

We program all heuristics in MATLAB 7.8 and conduct all experiments on a server with two Intel Dual-Core Xeon 5160 3.0 GHz processors and 8GB RAM. We do not focus on the computation time in this section because all heuristics are very efficient compared with the enumeration method. For example, to solve a single instance in the C4 system, the heuristics take 0.006 to 0.02 seconds while the enumeration method takes about 280 seconds. All LK type heuristics execute $(n-1)$ iterations in Step 3 and thus have almost the same computation time for an instance. Observed from our computational tests, all ZK type heuristics with the same initial arrangement have a similar computation time for an instance. In Section 3.4, we report computation time of various solution methods on both small and large CAP instances.

3.3.2 A preliminary example

As an example, we use the four LK type heuristics to solve three trials (i.e., 100 instances in each trial) that correspond to the $Lin/Con/3/8$:F system and three component types. Table 3.5 presents the computational results in terms of the MSSR, standard deviation, minimum, and maximum of SSRs. From Table 3.5, the LKB performs best for Low and Arbitrary reliable components in terms of all four statistics. For High reliable components, the LKB performs best in terms of the MSSR and the minimum of SSRs, but the LKA heuristic performs a little more robustly with the

smallest standard deviation, which, however, just slightly differs from that of the LKB heuristic.

Through this example and all of our computational tests, we find that almost all of the maximum SSRs are 1, implying that almost all heuristics are able to find the optimal arrangement for some instances in every trial (with some exceptions, e.g., in Table 3.5, the LKA and LKD heuristics for Arbitrary reliable components). Furthermore, ignoring slight difference, the MSSR, standard deviation, minimum, and maximum of SSRs give consistent conclusions. Therefore, we only present the computational results in terms of MSSR hereafter.

3.3.3 Comparison of the LK type heuristics

Figure 3.2 presents the computational results of the LK type heuristics on solving all 30 trials, in which each graph involves ten trials corresponding to one component type and ten systems, and associated with each trial, four MSSRs (data points) correspond to the four LK type heuristics. As shown in Figure 3.2, the LK type heuristics have no clear system preference. For a trial, the best performance is always achieved by either the LKA (in 12 out of 30 trials) or LKB (in the other 18 trials) heuristic. The LKC and LKD heuristics never outperform both of the LKA and LKB heuristics, and the LKD heuristic is more likely to generate extremely low MSSRs than the others. Therefore, we recommend using both the LKA and LKB heuristics so that the best arrangement from the LK type heuristics can be obtained. In addition, these observations imply that in Step 3d, updating reliability of only one position (as in the LKA and LKB heuristics) is more effective than updating reliabilities of multiple positions (as in the LKC and LKD heuristics).

Table 3.5: Test results for the LK type heuristics on the $Lin/Con/3/8$:F system

Heuristic	Component Type		
	High	Low	Arbitrary
LKA	**0.992768/0.008709**[a]	0.990222/0.008306	0.984292/0.013921
	0.945815/1.000000	0.966053/1.000000	0.934308/0.999438
LKB	0.992851/0.010327	0.999188/0.001509	0.990570/0.012027
	0.954624/1.000000	0.993580/1.000000	0.941072/1.000000
LKC	0.988928/0.011870	0.997797/0.004272	0.987222/0.015851
	0.958660/1.000000	0.976579/1.000000	0.909321/1.000000
LKD	0.988882/0.010626	0.984776/0.012151	0.975388/0.021366
	0.945815/1.000000	0.936150/1.000000	0.894853/0.998664

[a]In each cell, the two numbers in the first row are the MSSR and standard deviation of SSRs; in the second row are the minimum and maximum SSRs.

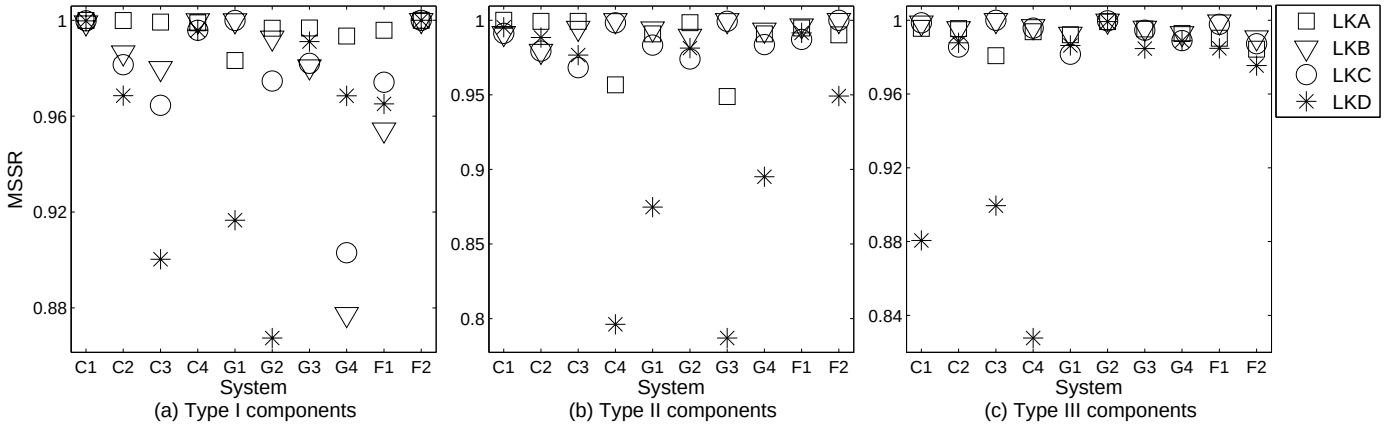

Figure 3.2: MSSRs associated with 30 trials and obtained by the LK type heuristics

Table 3.6 presents the overall performances of the LK type heuristics with respect to the component types, in which each number stands for the mean of ten MSSRs associated with ten trials of the same component type but ten different systems. Over all ten systems, the LKA heuristic performs best for High reliable components, and the LKB heuristic performs best for Low and Arbitrary reliable components. Table 3.6 provides a good reference for choosing a single LK type heuristic according to the component types.

Table 3.6: Overall performances of the LK type heuristics on ten systems

Heu.	Component Type		
	High	Low	Arbitrary
LKA	0.9964[a]	0.9869	0.9920
LKB	0.9770	0.9939[a]	0.9961[a]
LKC	0.9775	0.9865	0.9929
LKD	0.9573	0.9236	0.9515

[a]The best heuristic for each type of components

3.3.4 Comparison of the ZK type heuristics

The ZK type heuristics need an initial arrangement to start the procedure. In this subsection, we first discuss the choice of initial arrangements for the ZK type heuristics and then evaluate the ZK type heuristics.

The choice of initial arrangements

Initial arrangements are critical for the ZK type heuristics. As listed in Subsection 3.2.1, Initial Arrangements 1 and 2 are designed for the $Lin/Con/2/n$ systems; therefore, only Initial Arrangements 3 and 4 are tested here for general systems. In addition, we propose and test

Initial Arrangements 5 – 8: arrangements generated by the LKA, LKB, LKC, and LKD heuristics, respectively.

Noting that Initial Arrangements 3 and 4 are fixed arrangements regardless of system structures, Initial Arrangements 5 – 8 are instance-specialized, particularly, the solutions from the LK type heuristics. Using them ensures that the ZK type heuristics do not generate bad solutions. Although it takes time to generate them by the LK type heuristics, the high quality of Initial Arrangements 5 – 8 can accelerate the termination of the ZK type heuristics. Thus, generating Initial Arrangements 5 – 8 does not significantly increase the overall computation time of the ZK type heuristics. For example, solving the trial of the C4 system with High reliable components, the ZKD heuristic takes 0.99 seconds with Initial Arrangement 3, and 1.01 seconds with Initial Arrangement 6.

As illustrated in Subsection 3.2.2, to compare Initial Arrangements 3 – 8, we can not simply comparing the system reliabilities associated with them. Instead, we use all of them in each ZK type heuristic to solve all 30 trials. Figure 3.3 presents the number of trials on which the heuristic (x-axis) using the initial arrangement (the legend) achieves the highest MSSRs compared to using other initial arrangements. Because some of Initial Arrangements 5 – 8 may be the same for an instance and a heuristic using different initial arrangements may achieve the same highest MSSR on a trial, the ties exist and thus for each heuristic the total number of trials summed over all initial arrangements is more than 30. As shown in Figure 3.3, for all ZK type heuristics, the first three choices should be Initial Arrangements 6, 5 and 7, respectively.

An alternative is to use multiple initial arrangements in the ZK type heuristics and then choose the best arrangement generated as the final solution. Table 3.7 tallies the number of trials on which the highest MSSRs are achieved using the initial arrangement(s) listed in the first column. When the ZK type heuristics use the first two best choices – Initial Arrangements 6 and 5, they achieve the highest MSSRs on almost all trials (27 – 29 trials), increasing at least 9 trials compared to using Initial

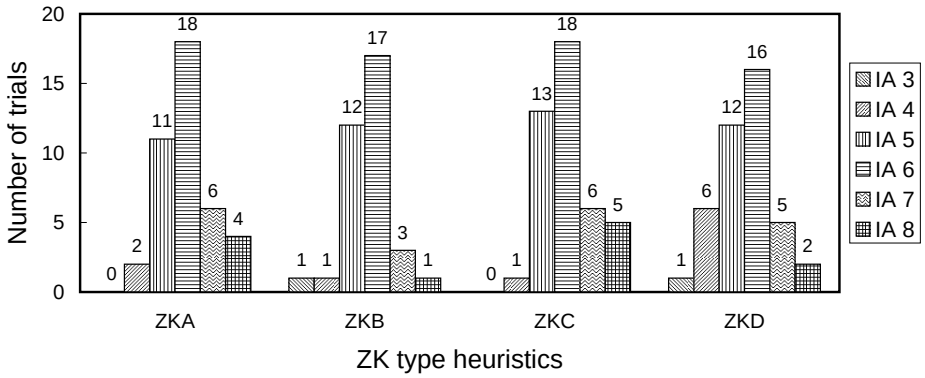

Figure 3.3: Comparison of the six initial arrangements

Arrangement 6 alone. But there are no more improvement when using the first three best choices – Initial Arrangements 6, 5, and 7. Thus, we suggest to use Initial Arrangements 6 and 5.

Table 3.7: Comparison of using multiple initial arrangements

Initial Arrangement(s) used	ZKA	ZKB	ZKC	ZKD
Initial Arrangement 6	18	17	18	16
Initial Arrangements 6 and 5	27	28	29	27
Initial Arrangements 6, 5 and 7	27	28	29	27

Test results of the ZK type heuristics

To evaluate each ZK type heuristic, we use both Initial Arrangements 5 and 6 and select the better arrangement generated as the final solution from that heuristic, as suggested above. Similar to Figure 3.2, Figure 3.4 presents the computational results of the ZK type heuristics on all 30 trials, and shows that none of them outperforms the others on all trials. Further analysis based on Figure 3.4 shows that one of the ZKB and ZKD heuristics that use Alternative 2 in Step 3a achieves the highest MSSRs on 28 out of 30 trials, while one of the ZKA and ZKC heuristics that use Alternative 1 achieves the highest MSSRs on 20 trials. Thus, Alternative 2 is superior to Alternative 1 overall.

Similar to Table 3.6, Table 3.8 demonstrates the overall performances of the ZK type heuristics with respect to the component types. Over all ten systems, for High and Arbitrary reliable components the ZKD heuristic performs best and thus is recommended; for Low reliable components, the ZKB heuristic performs best and is recommended. These conclusions are consistent with the above result that Alternative 2 (in the ZKB and ZKD heuristics) outperforms Alternative 1 (in the ZKA and ZKC heuristics). Since all the ZK type heuristics perform similarly, there is no need to use more than one ZK type heuristic to solve the CAP.

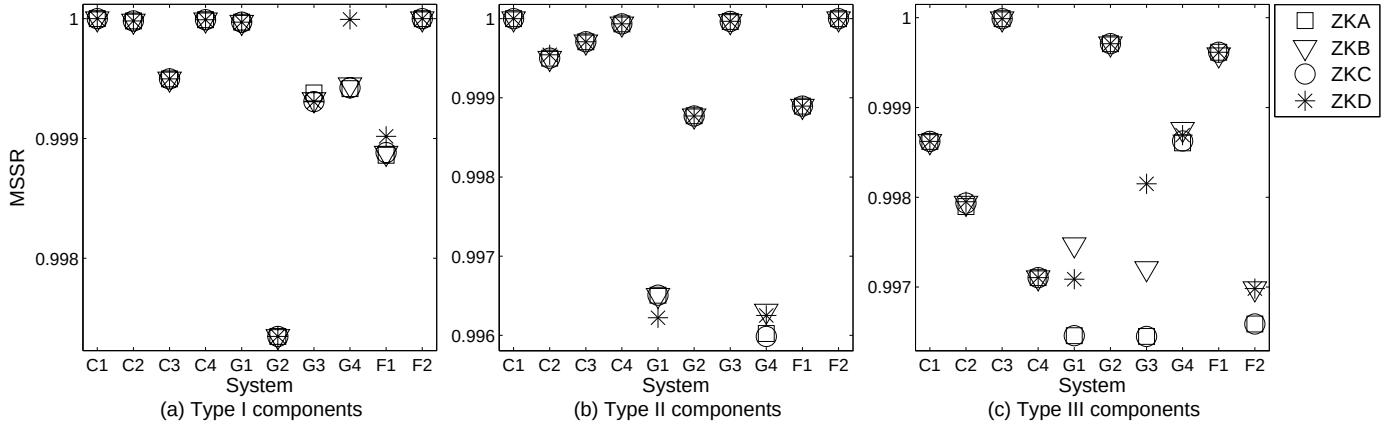

Figure 3.4: MSSRs associated with 30 trials and obtained by the ZK type heuristics

Table 3.8: Overall performance of the ZK type heuristics on ten systems

Heu.	Component Type		
	High	Low	Arbitrary
ZKA	0.999444	0.998931	0.998104
ZKB	0.999444	0.998959[a]	0.998337
ZKC	0.999440	0.998927	0.998110
ZKD	0.999511[a]	0.998929	0.998391[a]

[a]The best heuristic for each type of components

From our tests as shown in Figure 3.4, one of the ZKA and ZKC heuristics that use Alternative 1 in Step 3a achieves the highest MSSRs on 20 out of 30 trials, while one of the ZKB and ZKD heuristics that use Alternative 2 achieves the highest MSSRs on 28 trials. Thus, Alternative 2 has better overall performance than Alternative 1. From this point of view, the ZKB and ZKD heuristics, which use Alternative 2, are outstanding, and our computational results in Table 3.8 justify it.

3.4 A BI-based two-stage approach (BITA)

3.4.1 The BITA

As stated in Subsections 3.3.3 and 3.3.4, we suggest using the solutions of the LKA and LKB heuristics as the initial arrangements in the ZK type heuristics, the ZKB heuristic for solving the CAPs of low reliable components, and the ZKD heuristic for solving the CAPs of other types of components. Now, integrating these results, we propose the BITA for solving the CAP:

Stage 1: Use both the LKA and LKB heuristics to generate two initial arrangements, i.e., Initial Arrangements 5 and 6.

Stage 2: If the CAP involves only low reliable components (≤ 0.2), choose the ZKB heuristic; otherwise, choose the ZKD heuristic. Using Initial Arrangements 5 and 6 separately, solve the instance by the chosen ZK type heuristic (either

the ZKB or ZKD heuristic), and the final solution is the better one of the two arrangements generated.

Stage 2 can be regarded as an improvement stage that further improves the arrangements generated in Stage 1. If computation time is restricted, use only Stage 1 (i.e., the LKA and LKB heuristics) and then choose the better arrangement as the final solution, or even choose one single LK type heuristic according to Table 3.6.

3.4.2 Computational tests on small instances

Note that the CAP can be formulated as a mixed integer nonlinear programming problem and solved by GAMS/CoinBonmin solver, which basically uses a branch and bound algorithm. This subsection compares the BITA with GAMS/CoinBonmin solver by solving the 30 trials corresponding to the ten small systems (with no more than ten components) in Subsection 3.3.1. Table 3.9 presents the results in terms of MSSR and computation time. Note that GAMS/CoinBonmin solver does not guarantee to generate optimal solutions for the nonconvex problems including the CAP. Thus, the computation time of the enumeration method for finding optimal solutions is also listed in the last column. The "Time" column represents the seconds used to solve the whole trial, i.e., 100 instances.

As shown in Table 3.9, the BITA achieves higher MSSRs than GAMS/CoinBonmin solver in all trials except the one of the F1 system and Low reliable components, and is also much faster than GAMS/CoinBonmin solver by around 54 – 1268 times. Furthermore, the BITA takes much shorter computation time than the enumeration method, especially for those systems with more than seven components.

From Table 3.9, we can conclude that the BITA has clear advantage over GAMS/CoinBonmin solver in terms of both solution quality and computation time, especially when the system size (i.e., the number of components) gets larger and larger. Although the enumeration method can find optimal solutions for the small instances, it takes extremely long time for the medium-to-large instances and even

Table 3.9: Comparison of the BITA, GAMS, and enumeration method on 30 trials of small systems

System	Component	MSSR		Time (second)		
	Type	BITA	GAMS	BITA	GAMS	Enumeration
C1	High	1.000000	0.999886	0.56	46.58	2.59
	Low	0.999981	0.997689	0.50	70.97	2.60
	Arbitrary	0.999497	0.993666	0.47	62.26	2.58
C2	High	0.999991	0.995693	0.45	28.64	0.49
	Low	0.999971	0.998134	0.45	25.59	0.49
	Arbitrary	0.997348	0.988374	0.45	27.33	0.49
C3	High	0.999307	0.998270	1.55	1971.98	20058.76
	Low	0.999450	0.999375	1.83	714.92	19492.95
	Arbitrary	0.999018	0.997354	1.63	1291.07	19337.29
C4	High	1.000000	0.999995	2.05	749.54	27880.76
	Low	1.000000	0.997216	2.59	2511.12	28918.37
	Arbitrary	0.999544	0.994482	2.45	1503.31	27920.94
G1	High	0.999708	0.994352	0.82	162.75	45.45
	Low	0.999934	0.998317	0.78	70.74	45.31
	Arbitrary	0.996222	0.994522	0.76	49.80	46.66
G2	High	0.998771	0.989447	0.77	536.09	381.39
	Low	0.999965	0.997088	1.05	119.76	377.61
	Arbitrary	0.996252	0.994360	0.86	94.58	510.24
G3	High	0.998896	0.998267	0.65	66.56	44.15
	Low	1.000000	0.993257	0.67	109.25	45.06
	Arbitrary	0.998622	0.995506	0.60	74.47	49.63
G4	High	0.997950	0.996138	0.86	143.19	364.25
	Low	0.999989	0.995767	0.84	284.05	362.69
	Arbitrary	0.997106	0.994336	0.74	176.65	370.51
F1	High	0.997088	0.994183	0.78	124.20	51.92
	Low	0.999711	0.999894	0.65	71.73	50.63
	Arbitrary	0.998151	0.980612	0.78	52.68	58.90
F2	High	0.998692	0.996627	0.84	141.11	442.57
	Low	0.999589	0.997331	0.97	253.16	438.88
	Arbitrary	0.996984	0.979267	0.93	90.61	534.25

results in out-of-memory, being impractical for achieving the optimal solutions. For example, to solve the instance in the $Lin/Con/3/20$:F system with High reliable components, the BITA takes only 0.02 seconds with the system reliability of 0.993657, but GAMS/CoinBonmin solver takes 49.09 seconds with the system reliability of

0.993606, and the enumeration method cannot achieve the optimal solution after several days of computation. Therefore, in the next subsection of computational tests on large instances (with 20 or more components), we only test the BITA because it can generate solutions of better quality in much shorter time than GAMS/CoinBonmin solver and the enumeration method cannot solve these large systems in a reasonable time.

3.4.3 Computational tests on large instances

We now evaluate the performance of the BITA on 28 large $Lin/Con/k/n$ systems, including 12 $Lin/Con/k/n$:F systems with $k = 3, 4, 5$ and $n = 20, 30, 50, 100$ and 16 $Lin/Con/k/n$:G systems with $k = 2, 3, 4, 5$ and $n = 20, 30, 50, 100$. As mentioned above, the enumeration method is extremely slow and even out-of-memory to calculate the R_{opt} and R_{wor} in (3.3) for a CAP instance. Instead, a randomization method is used to approximate the R_{opt} and R_{wor}. To solve a CAP instance, 10,000 random arrangements are generated, and the R_{opt} and R_{wor} are the system reliabilities corresponding to the best and worst arrangements out of the 10,000 random arrangements. Then, the approximate SSR of the instance and the MSSR of the trial can be calculated. Table 3.10 presents the approximate MSSRs and the computation time for the BITA to solve the trials on the 28 large systems. The "Time" column still represents the seconds used to solve the whole trial, i.e., 100 instances. Note that there is no MSSR for the trial of the $Lin/Con/3/100$:F system and Low reliable components because in this trial the system reliabilities of any instance found by both the randomization method and the BITA are less than 10^{-19} and the calculation of the SSRs and MSSR is misleading under this situation.

As shown in Table 3.10, for the trials associated with High reliable components, most of the MSSRs are larger than 1 which means that the BITA performs better than the randomization method, though there are four exceptions on the $Lin/Con/2/50$:G, $Lin/Con/2/100$:G, $Lin/Con/3/100$:G, and $Lin/Con/4/100$:G systems where the

Table 3.10: MSSRs and computation time of the BITA on the trials of large systems

System	Component Type					
	High		Low		Arbitrary	
	MSSR	Time	MSSR	Time	MSSR	Time
Lin/Con/3/20:F	1.0954	3.87	1.4454	3.76	1.2204	3.76
Lin/Con/3/30:F	1.1727	8.33	2.4771	9.60	1.6664	9.57
Lin/Con/3/50:F	1.3168	26.80	7.9320	39.37	3.3948	34.36
Lin/Con/3/100:F	1.5966	133.67	—	116.98	34.2925	233.79
Lin/Con/4/20:F	1.0411	3.95	1.3203	3.93	1.0793	4.33
Lin/Con/4/30:F	1.0900	8.39	1.8716	7.75	1.2350	8.70
Lin/Con/4/50:F	1.1669	25.97	3.5567	39.10	1.6296	28.23
Lin/Con/4/100:F	1.3239	158.96	26.6616	336.66	4.1843	267.61
Lin/Con/5/20:F	1.0237	4.31	1.2420	3.63	1.0313	4.40
Lin/Con/5/30:F	1.0473	11.66	1.6418	9.51	1.0879	12.70
Lin/Con/5/50:F	1.0934	39.40	2.7746	38.23	1.2370	50.82
Lin/Con/5/100:F	1.2035	203.43	9.8065	379.43	1.8477	347.29
Lin/Con/2/20:G	1.0039	2.57	1.1340	3.21	1.0092	2.69
Lin/Con/2/30:G	1.0036	4.02	1.2410	6.28	1.0048	5.46
Lin/Con/2/50:G	0.6790	7.14	1.3860	16.08	1.0012	15.57
Lin/Con/2/100:G	0.6309	23.14	1.6408	62.68	1.0001	34.42
Lin/Con/3/20:G	1.0154	2.89	1.2134	3.16	1.0337	2.35
Lin/Con/3/30:G	1.0192	5.73	1.3830	6.47	1.0258	4.51
Lin/Con/3/50:G	1.0160	9.74	1.6377	15.84	1.0117	17.19
Lin/Con/3/100:G	0.6183	24.34	2.0558	63.35	1.0014	59.29
Lin/Con/4/20:G	1.0337	2.84	1.2702	2.73	1.0820	2.27
Lin/Con/4/30:G	1.0461	6.43	1.5150	6.10	1.0955	4.55
Lin/Con/4/50:G	1.0469	16.02	1.8418	15.97	1.0576	14.56
Lin/Con/4/100:G	0.9000	28.27	2.4501	61.91	1.0159	60.38
Lin/Con/5/20:G	1.0568	2.69	1.3312	2.45	1.1274	2.20
Lin/Con/5/30:G	1.0783	6.11	1.5872	5.76	1.1952	4.53
Lin/Con/5/50:G	1.0885	15.34	2.1065	15.21	1.1505	12.15
Lin/Con/5/100:G	1.0622	31.83	2.9050	58.86	1.0818	72.44

MSSRs are less than 1. For the trials associated with Low and Arbitrary reliable components, the BITA always finds the MSSRs larger than 1, and in some trials the MSSRs are even larger than 2, which implies that the BITA generates much better arrangements than the randomization method in these trials. The larger the MSSR is, the better the BITA performs. Moreover, the BITA takes a reasonable time

for solving the CAPs associated with the large systems. For example, the longest computation time is about 380 seconds for solving the 100 instances in the trial of the $Lin/Con/5/100$:F system and Low reliable components. On average, the BITA takes only 1/6 time of the randomization method for the same instance in Table 3.10. We also test even larger systems, for example, a trial of the $Lin/Con/3/300$:F system and High reliable components. Solving this trial, the MSSR is 2.3973, the BITA takes 1,425 seconds, and the randomization method takes 2,537 seconds. That is, the BITA uses only about 3/5 time of and obtains far better solutions than the randomization method. In summary, the BITA is capable of solving the large instances with high accuracy in short computation time. The BITA is very practically implementable and absolutely outperforms the randomization method.

3.5 Summary

This chapter proposes three new LK type heuristics (i.e., the LKB, LKC, and LKD heuristics) and two new ZK type heuristics (i.e., the ZKC and ZKD heuristics) for solving the CAP. We prove that if a system admits an invariant optimal arrangement, the arrangements generated by the LK type heuristics must be the invariant optimal arrangement given that there is no tie in Step 3b in all iterations, but the ZK type heuristics do not have such a property. Based on the numerical analysis on the LK and ZK type heuristics, we propose the BITA whose first stage uses the LKA and LKB heuristics, and the second stage uses the ZKB or ZKD heuristic depending on the component types. Our numerical experiments involving both small and large systems show that the BITA absolutely performs much better than GAMS/CoinBonmin solver, the enumeration method, and the randomization method, especially for those CAPs with large system sizes. The BITA is efficient and capable to generate solutions of high quality.

Chapter 4

A BI-based Genetic Local Search Algorithm for the CAP

In Chapter 3, we unsurprisingly observe that the BITA may converge to a local optimal arrangement as illustrated in Table 3.9. The meta-heuristics have the potential to break local optimal arrangements and thus are expected to obtain solutions of higher quality. As shown in Subsection 1.3.2, some simulated annealing and genetic algorithms have been proposed for solving the CAP. However, these methods were designed specially for the CAP in the $Cir/Con/k/n$:F systems and did not utilize the BI which can be used to improve the performance of heuristics. In the literature, there is no meta-heuristic for solving the CAP in general systems, and none of them utilized the BI. These observations encourage us to develop BI-based meta-heuristics for the CAP in general systems.

In this chapter, we propose a BI-based genetic local search (BIGLS) for tackling the CAP in general coherent systems. The genetic local search is a hybrid genetic algorithm that integrates both the population-based search and the local optimization. The genetic operations explore the broad search space in order to identify promising subregions, meanwhile the local search exploits good solutions in these small subregions by iteratively moving from one solution to a better one

in its neighborhood until a local optimum is reached. Genetic local search has been successfully applied to the traveling salesman (Freisleben and Merz, 1996), quadratic assignment (Lim et al., 2000), and flowshop scheduling (Tseng and Lin, 2009) problems but never been used for solving the CAP.

The BIGLS uses the BI in directing the local improvement and uses the general scheme of the genetic local search to overcome the potential shortcoming of premature that the BI-based heuristics in Chapter 3 might arise. The BIGLS has a great chance of achieving solutions of high quality effectively because it takes advantages of both the efficient local search directed by the BI and the population-based global search (broad exploration) of genetic algorithm. Comprehensive numerical tests compare the BIGLS with two benchmarking methods (i.e., the BITA and a simple genetic algorithm) on various coherent systems and different types of components, and verify the importance of embedding the BI in the genetic evolutionary mechanism and the effectiveness of the BIGLS.

4.1 The BIGLS algorithm

Based on the BIGLS framework shown below, we develop the BIGLS algorithm for the CAP using specially designed operators in each step, which are detailed in Subsections 4.1.1 – 4.1.7. Particularly, the BIGLS uses an initial assignment generated by the BITA and applies the efficient BI-based local search.

The BIGLS framework

1. Generate an initial population with m chromosomes. Perform the BI-based local search on each initial chromosome.

2. If the current population satisfies the termination conditions, perform the BI-based local search on the best chromosome and stop.

3. Evaluate the scaled fitness of each chromosome according to (4.1) (introduced later).

4. Perform elitism strategy, i.e., the $m\beta$ best chromosomes in the current population are directly reproduced into the next population.

5. Perform crossover on the current population to generate $m(1-\beta)$ offspring chromosomes into the next population.

6. Perform mutation on the selected offspring chromosomes, and perform the BI-based local search on each mutated offspring chromosome and on the unmutated offspring chromosomes that are better than the best one in the current population.

7. Replace the current population with the next population and go to Step 2.

Starting with the initial population generated in Step 1, an iterative procedure from Steps 2 to 7 is performed on each generation of population, until the termination conditions in Step 2 is satisfied. In Step 3, the scaled fitness values are computed for all chromosomes. Then, in Step 4, the $m\beta$ best chromosomes in the current population as the elitists are directly reproduced into the next population, where β is called the elitist rate, $0 < \beta < 1$. In Step 5, a crossover operator generates $m(1-\beta)$ offspring chromosomes into the next population. In Step 6, mutation is conducted on a small portion of the offspring chromosomes, and the BI-based local search is performed on the mutated and some promising unmutated offspring chromosomes. Considering the solution quality and the computational efficiency, Lim et al. (2000) found that for the quadratic assignment problems the local search should not be performed for each new chromosome, and our preliminary numerical tests for the CAP obtain the same finding. Thus, our BIGLS algorithm performs the BI-based local search in Step 6 only on some promising rather than all offspring chromosomes. After Step 6, a new population of size m is generated and replaces the current population in Step 7.

4.1.1 Gene, chromosome, and population

In the BIGLS, a chromosome is coded as an assignment $\boldsymbol{\pi}$, and each integer π_i, $i = 1, 2, \ldots, n$, in the chromosome is a gene. Each population has m chromosomes, and the population size m is constant through the generations. The initial population includes $m - 1$ randomly generated chromosomes and one special chromosome generated by the BITA. According to Chapter 3, the BITA is the most accurate BI-based heuristic. Thus, including the assignment generated by the BITA in the initial population can improve the quality of the initial population while the other $m - 1$ random chromosomes maintain the diversity of the population.

4.1.2 The BI-based local search

Steps 1, 2, and 6 of the BIGLS use a BI-based local search to boost a given assignment $\boldsymbol{\pi}$ in its neighborhood. That is essentially a pairwise exchange method based on the BI as detailed below.

The BI-based local search

Given an assignment $\boldsymbol{\pi}$, do loop for $k = n$ to 2,

1. Compute $I(i; \mathbf{p}_{\boldsymbol{\pi}})$ for all $i = 1, 2, \ldots, n$ according to (3.1).
2. Find positions i and j such that $\pi_i = k$ and $I(j; \mathbf{p}_{\boldsymbol{\pi}}) = \max\limits_{r: p_{\pi_r} < p_{\pi_i}} I(r; \mathbf{p}_{\boldsymbol{\pi}})$ (break tie arbitrarily).
3. If $I(i; \mathbf{p}_{\boldsymbol{\pi}}) < I(j; \mathbf{p}_{\boldsymbol{\pi}})$ and exchanging the allocations of π_i and π_j can improve the system reliability, then exchange them and update $\boldsymbol{\pi}$.

This BI-based local search conducts pairwise exchange iteratively (for k) from the most reliable component n to the second least reliable component 2. Position i associated with component k is compared with position j that has the largest BI value among the positions with reliabilities lower than p_{π_i}. If position i is less important than position j in terms of the BI and exchanging p_{π_i} and p_{π_j} can improve the system reliability, then exchange the allocations of components π_i and π_j.

4.1.3 Fitness scaling

In the CAP, the objective is to maximize the system reliability which is bounded by 0 and 1. It is natural to use the system reliability $R(\mathbf{p}_{\boldsymbol{\pi}_t})$ as the fitness of chromosome $\boldsymbol{\pi}_t$, denoted by $f_t = R(\mathbf{p}_{\boldsymbol{\pi}_t})$, for $t = 1, 2, \ldots, m$. Meanwhile, Step 3 calculates the scaled fitness of chromosome $\boldsymbol{\pi}_t$, denoted by F_t, which is defined using a linear scaling function with coefficients α_1 and α_2 as

$$F_t = \alpha_1 \times f_t + \alpha_2. \tag{4.1}$$

Let f_{max}, f_{min}, and f_{avg} denote the maximum, minimum, and average fitness values of the m chromosomes before the scaling, respectively, and F_{max}, F_{min}, and F_{avg} denote the maximum, minimum, and average fitness values after the scaling, respectively. The coefficients α_1 and α_2 are chosen such that $F_{avg} = f_{avg}$ and $F_{max} = S_f \times F_{min}$, which yield

$$\alpha_1 = \frac{(S_f - 1) \times f_{avg}}{(f_{avg} - f_{min}) \times S_f + (f_{max} - f_{avg})} \tag{4.2}$$

$$\alpha_2 = (1 - \alpha_1) \times f_{avg} \tag{4.3}$$

where $S_f > 1$ is the scaling factor. Note that α_1 is always positive but α_2 may not be.

The scaled fitness in (4.1) is used with the aim of balancing the premature in convergence and the divergence of the population (Goldberg, 1989). When $S_f \gg 1$, a super good chromosome quickly dominates the population and results in premature and a suboptimal solution. When $S_f \approx 1$, the population always keeps large dispersion, which makes the convergence very slow and even diverge. A proper scaling factor S_f can prevent the premature in the early stage and the divergence in the late stage.

4.1.4 Termination conditions

The denominator of (4.2) reflects the extent of convergence of the population (Lim et al., 2000), and thus is used as a termination condition in Step 2 of the BIGLS:

$$(f_{avg} - f_{min}) \times S_f + (f_{max} - f_{avg}) \quad < \quad \varepsilon, \tag{4.4}$$

where ε is a very small positive number. Another termination condition in Step 2 is that the number of generations can not exceed a pre-specified threshold (e.g., 200 generations). The BIGLS stops whenever one of the two termination conditions is met.

4.1.5 Elitism

Step 4 of the BIGLS uses an elitism strategy. A proportion (i.e., elitist rate β) of the best chromosomes (i.e., those with high fitness values) in the current generation are directly copied to the next generation. These chromosomes are called elitists, which carry the best genes from the current generation to the next generation. This strategy can improve the performance of the genetic local search (Davis, 1991). Recall that the initial population includes the assignment generated by the BITA as a chromosome. Therefore, by the elitism strategy, the BIGLS could never generate a worse assignment than the BITA.

4.1.6 Crossover

Step 5 of the BIGLS executes the crossover operator in order to produce offspring chromosomes for the next population. The biased roulette wheel criterion (Goldberg, 1989) is used to select a pair of parent chromosomes from the current population; each chromosome $\boldsymbol{\pi}_t$, $t = 1, 2, \ldots, m$, is selected as a parent with the probability $F_t \Big/ \sum_{j=1}^{m} F_j$, proportional to its scaled fitness value. Note that $F_{max} = S_f \times F_{min}$, which means that the best chromosome has S_f times probability to be selected

as the worst chromosome. After a chromosome is selected, it is replaced back to the current population and is eligible to be re-selected. Totally, $m(1-\beta)$ pairs of parent chromosomes are selected for reproducing $m(1-\beta)$ offsprings into the next population. With probability P_{cro}, a crossover operation is executed on a pair of parent chromosomes to generate an offspring chromosome into the next generation; with probability $1-P_{cro}$, no crossover operation is taken and the first selected parent is reproduced to the next population.

In the CAP, the absolute positions of the components is more important than their relative positions. Therefore, the partial matched crossover (PMX) (Goldberg and Lingle, 1985) operator is implemented. Under this crossover operator, two crossing points are picked uniformly at random. These two points define a matching section between the parent chromosomes according to which a crossing of position-by-position exchanges is implemented. As an example, consider a pair of parent chromosomes $\boldsymbol{\pi}_a$ and $\boldsymbol{\pi}_b$:

$$\begin{aligned} \boldsymbol{\pi}_a &= 8\ 7 \mid 6\ 4\ 2 \mid 1\ 5\ 3 \\ \boldsymbol{\pi}_b &= 2\ 5 \mid 1\ 7\ 3 \mid 8\ 4\ 6 \end{aligned}$$

and two randomly selected crossing points 3 and 6 as indicated above. Then, relative to chromosome $\boldsymbol{\pi}_a$, exchanging the positions of genes 6 with 1, 4 with 7, and 2 with 3 produces an offspring chromosome:

$$\boldsymbol{\pi}' = 8\ 4 \mid 1\ 7\ 3 \mid 6\ 5\ 2.$$

4.1.7 Mutation

Following Step 5 of the crossover operations, Step 6 of the BIGLS mutates each offspring chromosome with probability P_{mut} to maintain the variation of the population and spread out the search regions. The mutation operator randomly picks two positions and swaps the genes in these two positions. For example, if the

above chromosome $\boldsymbol{\pi}'$ is chosen for mutation and the second and fourth positions are randomly picked, then genes 4 and 7 in these positions are exchanged, producing a mutated chromosome:

$$\boldsymbol{\pi}'' = 8\ 7\ 1\ 4\ 3\ 6\ 5\ 2.$$

In Step 6, the mutated offspring chromosomes and the unmutated ones with the scaled fitness value higher than the best chromosome in the current population are further improved by the BI-based local search, and the chromosomes generated by the BI-based local search replace the original ones to enter the next population.

4.2 Numerical experiments

This section conducts numerical experiments to evaluate the performance of the BIGLS on solving the CAP in various systems and with different types of component reliabilities. To demonstrate the effectiveness of the utilization of the BI and BI-based local search in the BIGLS, we propose a simple genetic algorithm (SGA) which uses the same framework and parameter setting as the BIGLS but does not use any local search in Steps 1, 2, and 6. We benchmark on the SGA and BITA to evaluate the performance of the BIGLS. Note that the assignment generated by the BITA is included in the initial populations of both the BIGLS and SGA, and their elitism strategy makes the best assignment(s) evolve from one generation to another. Thus, the BIGLS and SGA never generate solutions worse than the BITA.

The numerical experiments use the same setting as in Chapter 3. The systems are divided into two groups by the system size: the ten small systems and the 28 large $Lin/Con/k/n$ systems in Chapter 3. We keep considering three types of components – High, Low, and Arbitrary reliable components, which have the component reliabilities randomly generated from uniform distributions on $[0.8, 0.99]$, $[0.01, 0.2]$, and $[0.01, 0.99]$, respectively, and the performances of the BIGLS, BITA, and SGA are evaluated with respect to these three component types.

We program the BIGLS and SGA in MATLAB 7.8 and conduct all experiments on a server with two Intel Dual-Core Xeon 5160 3.0 GHz processors and 8GB RAM. All BIGLS and SGA runs are performed with population size $m = 50$, fitness scaling factor $S_f = 3$, crossover probability $P_{cro} = 0.8$, and mutation probability $P_{mut} = 0.05$. The BIGLS is terminated after 200 generations or when the convergence criterion in (4.4) is satisfied with $\varepsilon = 0.0001$.

For the CAP instances associated with the small systems, the enumeration method can find the optimal assignments in a reasonable time (less than two minutes in our computational setting), and thus we compare the assignments generated by the BIGLS to the optimal assignments in addition to the assignments generated by the benchmarking BITA and SGA. However, the enumeration method fails to find the optimal solutions for the CAP instances associated with the large systems as illustrated in Section 1.3. Therefore, in the large systems, we evaluate the performance of the BIGLS only benchmarking on the BITA and SGA.

4.2.1 Preliminary test results

Because the execution of the BIGLS depends on the realization of the random operations, preliminary tests on the large systems are conducted to examine the robustness of the BIGLS to its random operations. For each large system with n positions and each type of High, Low, and Arbitrary reliable components, an instance is created by randomly generating n component reliabilities according to the distribution specified by the component type. We execute ten random BIGLS runs on each instance, and obtain ten assignments and correspondingly ten system reliabilities which may be different due to the random operations. Fig. 4.1 shows the coefficient of variation (i.e., σ/μ) of the ten system reliabilities for each instance where σ is the standard deviation and μ the mean. The horizontal axis stands for the systems of the instances with the omission of the phrase "$Lin/Con/$". An exponential transformation is conducted on the coefficients of variation in order to

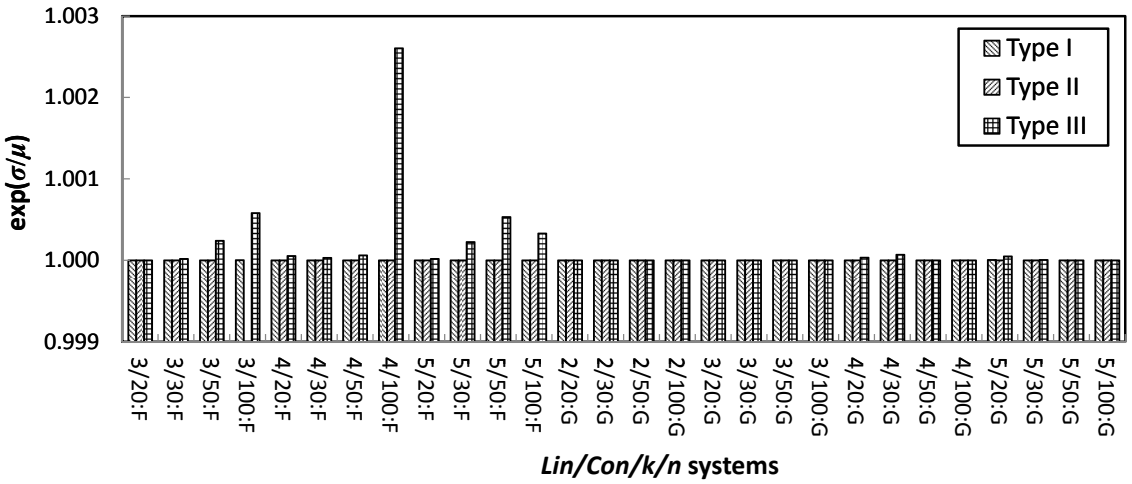

Figure 4.1: Robustness of the BIGLS in ten random runs for the $Lin/Con/k/n$ systems

clearly show the small values. Note that there is no value for the instance in the $Lin/Con/3/100$:F system with Low reliable components because in this instance the ten system reliabilities found by the ten BIGLS runs are less than 10^{-19} and then the calculation of σ/μ is inaccurate and misleading.

In Fig. 4.1, the coefficients of variation are exact zero for 44 out of 83 instances (i.e., those with value 1 above the corresponding bars) and are near zero for the others with the largest value of 0.0026 associated with the instance in the $Lin/Con/4/100$:F system with Arbitrary reliable components. The instances with Arbitrary reliable components tend to have larger coefficients of variation, but all the coefficients of variation are extremely small and close to zero. This implies that the standard deviation of the ten random BIGLS runs is negligible compared to their mean. Therefore, the BIGLS is robust to its random operations, and there is no need to execute multiple BIGLS runs to solve a CAP instance. In the following tests, only one BIGLS run is executed for each instance. Note that the same observation holds for the SGA.

4.2.2 Computational results on small instances

As in Section 3.3, for each system and each component type, 100 instances are generated, each instance corresponding to a set of different randomly-generated component reliabilities. There are ten small systems and three component types and thus 30 trials, i.e., 30 sets of small CAP instances. Table 4.1 presents computational results of the BITA, BIGLS, and SGA on solving the instances in the 30 trials. The "Opt. #" columns represent the numbers out of 100 instances in each trial for which the BITA, BIGLS, and SGA find the optimal assignments, respectively. Note that the optimal assignments are found by the enumeration method. The "Imp. #" columns show the numbers out of 100 instances in each trail for which the BIGLS and SGA can improve the assignments generated by the BITA, respectively. Recall that the BIGLS and SGA never generate solutions worse than the BITA as illustrated at the beginning of Section 4.2. The "Time" columns record the average computation time of the BITA, BIGLS, and SGA for solving one of the 100 instances in each trial, respectively.

As shown in Table 4.1, the BITA finds the optimal assignments for all 100 instances in ten trials (e.g., the trials corresponding to the C1 system with High, Low, and Arbitrary reliable components, respectively) and leaves no room for further improvement to the BIGLS and SGA. Whereas as long as there is any room for improvement left by the BITA (i.e., in the other 20 trials), the BIGLS and SGA can always improve some, if not all, of the non-optimal assignments generated by the BITA. Moreover, the BIGLS is superior to the SGA since the BIGLS achieves at least as many optimal assignments as the SGA. For example, in the trial associated with the $Lin/Con/2/7$:G system and the Arbitrary reliable components, the BITA can not find optimal assignments for 48 instances; the BIGLS improves the solutions for 47 out of these 48 instances to be the optimal assignments; the SGA improves the solutions for 38 instances and achieves the optimal assignments for 35 out of the 38 instances. Overall, the BIGLS finds optimal assignments for all 100 instances in

Table 4.1: Comparison of the BITA, BIGLS, and SGA on small systems

System	Component	Opt. #			Imp. #		Time (second)		
	Type	BITA	BIGLS	SGA	BIGLS	SGA	BITA	BIGLS	SGA
C1	High	100	100	100	–	–	0.004	0.19	1.97
	Low	100	100	100	–	–	0.004	0.25	1.32
	Arbitrary	100	100	100	–	–	0.004	0.50	1.77
C2	High	100	100	100	–	–	0.003	0.34	1.50
	Low	100	100	100	–	–	0.003	1.25	1.82
	Arbitrary	100	100	100	–	–	0.003	1.56	1.65
C3	High	55	81	72	35	30	0.012	0.41	0.58
	Low	98	100	98	2	1	0.015	2.68	2.08
	Arbitrary	63	100	91	37	29	0.013	3.63	2.54
C4	High	100	100	100	–	–	0.016	1.16	2.48
	Low	100	100	100	–	–	0.019	0.34	0.20
	Arbitrary	99	100	100	1	1	0.017	1.44	2.35
G1	High	40	100	100	60	60	0.006	1.19	1.77
	Low	59	100	100	41	41	0.005	0.42	2.23
	Arbitrary	37	100	100	63	63	0.006	1.88	2.47
G2	High	65	100	93	35	28	0.007	1.49	1.89
	Low	82	100	99	18	17	0.008	0.62	2.48
	Arbitrary	33	100	92	67	64	0.008	2.09	2.90
G3	High	80	99	99	19	19	0.006	0.94	1.16
	Low	93	100	100	7	7	0.006	1.99	2.05
	Arbitrary	52	99	87	47	38	0.005	2.70	2.43
G4	High	50	96	85	50	40	0.006	0.97	0.85
	Low	92	100	96	8	4	0.007	2.38	2.24
	Arbitrary	31	99	73	69	57	0.007	3.01	2.61
F1	High	92	100	100	8	8	0.005	1.41	2.17
	Low	100	100	100	–	–	0.005	1.18	1.88
	Arbitrary	74	100	91	26	17	0.005	2.22	2.23
F2	High	72	100	95	28	25	0.006	1.54	2.25
	Low	100	100	100	–	–	0.006	1.56	1.76
	Arbitrary	46	100	75	54	34	0.006	2.40	2.39

25 trials, 96-99 instances in four trials, and 81 instances in one trial; the SGA finds optimal optimal assignments for all 100 instances in 16 trials and 72-99 instances in the other 14 trials.

The computation time of the BITA (0.005-0.019 seconds) is much smaller than those of the BIGLS (0.19-3.63 seconds) and SGA (0.20-2.90 seconds). Thus, including the assignment generated by the BITA in the initial populations of the BIGLS and SGA does not significantly increase their computational burden. Moreover, averaging on the 100 instances in each trial, the BIGLS is faster than the SGA in 22 trials and

slower in other eight trials. For the small instances, because the solution space of $n!$ assignments is relatively small, the BIGLS with the aid of the BI-based local search can find assignments of high quality more easily than the SGA. Therefore, the BIGLS can converge in less number of generations than the SGA and consequently the overall time of the BIGLS can be smaller than the SGA although the BI-based local search in the BIGLS consumes additional time in each generation. In summary, the BIGLS is the best one among the three heuristics in terms of both solution quality and the computation time for small size instances.

4.2.3 Computational results on large instances

Similarly, we generate a set of 100 instances for each large system and each component type. There are 28 large systems and three component types and thus 84 trials of large CAP instances. Because the enumeration method can not find optimal assignments for the large systems in a reasonable time, we evaluate the BIGLS and SGA by the number out of the 100 instances in each trial for which the BIGLS/SGA is able to improve the assignments generated by the BITA. Table 4.2 presents this part of the computation results and computation times. The first column represents the $Lin/Con/k/n$ systems with the omission of the phrase "$Lin/Con/$". The "Imp. #" and "Time" columns have the same meanings as in Table 4.1. Due to the same reason as in Subsection 4.2.1, the results for the $Lin/Con/3/100$:F system with Low reliable components are not available.

As summarized in the "Imp. #" columns, over 84 trials of instances, the BIGLS improves the assignments generated by the BITA for more instances than the SGA with only one exceptional trial (i.e., the one associated with the $Lin/Con/5/20$:F system and the High reliable components). In the most trials, the BIGLS significantly outperforms the SGA. For example, in the trial associated with the $Lin/Con/3/50$:F system and the Low reliable components, the BIGLS can improve the BITA assignments for 43 out of 100 instances, but the SGA fails to improve any of them.

Table 4.2: Comparison of the BIGLS, BITA, and SGA on large systems

System	High reliable comp.					Low reliable comp.					Arbitrary reliable comp.				
	Imp. #		Time (sec.)			Imp. #		Time (sec.)			Imp. #		Time (sec.)		
	BIGLS	SGA	BITA	BIGLS	SGA	BIGLS	SGA	BITA	BIGLS	SGA	BIGLS	SGA	BITA	BIGLS	SGA
3/20:F	90	71	0.04	4.70	3.32	8	0	0.04	0.59	0.18	89	67	0.04	6.27	3.98
3/30:F	91	59	0.09	9.31	4.02	26	0	0.10	1.25	0.11	97	76	0.10	10.40	4.75
3/50:F	93	43	0.28	24.36	5.53	43	0	0.41	3.41	0.43	99	35	0.36	22.43	6.62
3/100:F	94	29	1.40	89.24	10.26	–	–	–	–	–	99	31	2.46	56.06	11.20
4/20:F	89	60	0.04	3.46	1.42	32	9	0.04	1.64	1.71	95	79	0.05	7.58	4.23
4/30:F	83	42	0.09	8.18	1.95	12	1	0.08	1.23	0.24	85	50	0.09	12.92	5.07
4/50:F	82	14	0.27	27.64	2.74	57	0	0.41	3.45	0.43	96	47	0.29	27.92	6.80
4/100:F	74	7	1.67	117.39	5.63	100	0	3.50	15.37	3.55	100	29	2.78	112.25	12.67
5/20:F	30	44	0.04	1.06	0.71	64	34	0.04	1.83	2.93	98	86	0.05	8.34	4.38
5/30:F	47	35	0.12	3.73	1.02	47	9	0.10	2.85	1.59	99	83	0.13	14.46	5.26
5/50:F	70	17	0.41	18.23	1.70	42	0	0.40	3.42	0.42	100	54	0.53	33.94	6.85
5/100:F	64	10	2.11	99.14	4.30	99	0	3.95	15.79	4.00	97	30	3.60	123.90	13.87
2/20:G	1	0	0.03	0.37	0.03	21	0	0.03	4.64	2.99	80	36	0.02	4.56	2.31
2/30:G	2	0	0.04	0.54	0.05	31	1	0.07	6.05	3.69	39	22	0.06	3.11	1.22
2/50:G	77	36	0.07	0.47	0.09	22	1	0.17	9.78	4.54	1	1	0.16	1.91	0.37
2/100:G	85	39	0.24	1.45	0.27	4	0	0.66	29.68	6.60	1	0	0.36	4.57	0.39
3/20:G	13	0	0.03	0.40	0.05	6	0	0.03	2.28	2.75	85	39	0.02	5.47	3.52
3/30:G	0	0	0.06	0.76	0.07	5	0	0.07	3.50	3.43	90	24	0.05	7.39	3.22
3/50:G	1	0	0.10	1.42	0.11	6	0	0.17	6.24	4.08	75	10	0.18	9.01	1.95
3/100:G	95	30	0.26	1.67	0.28	1	0	0.67	18.47	6.07	0	0	0.63	6.99	1.52
4/20:G	87	49	0.03	1.77	0.92	3	0	0.03	1.70	2.13	74	23	0.02	5.10	3.36
4/30:G	4	1	0.07	0.77	0.10	0	0	0.07	2.88	2.20	76	14	0.05	8.06	3.80
4/50:G	0	0	0.17	1.91	0.18	0	0	0.17	5.77	3.30	90	21	0.15	12.63	3.63
4/100:G	32	5	0.30	2.17	0.33	0	0	0.67	18.46	4.81	28	2	0.64	17.90	2.64
5/20:G	93	46	0.03	2.37	2.16	8	0	0.03	1.46	0.85	62	23	0.02	4.51	3.38
5/30:G	59	12	0.07	2.01	0.79	2	0	0.06	2.48	1.17	79	19	0.05	7.29	3.90
5/50:G	1	0	0.16	1.93	0.17	0	0	0.16	5.24	1.59	91	10	0.13	14.44	4.61
5/100:G	3	0	0.34	5.34	0.36	0	0	0.63	17.08	2.90	85	14	0.77	32.26	3.74

Note that, in the $Lin/Con/k/n$:G systems with Low reliable components (especially when $k \geq 3$), although the BIGLS still works better than the SGA, neither of the BIGLS or SGA can significantly improve the assignments generated by the BITA. A possible reason is that the BITA works very well on these instances and already achieve optimal or near-optimal assignments (Yao et al., 2011), and thus leaves very limited room for further improvement to the BIGLS and SGA.

Similar to the case of the small instances, over the 84 trials of large instances, the average computation time of the BITA for a single instance (0.04-3.95 seconds) is much smaller than those of the BIGLS (0.37-123.90 seconds) and SGA (0.03-13.87 seconds), again validating the reasonability of including the assignment generated by the BITA in the initial populations of the BIGLS and SGA. However, different from the case of the small instances, the BIGLS is usually slower than the SGA with a few exceptions on the trials associated with the systems of size $n = 20$. As the system size n increases, the BIGLS runs slower than the SGA. For the large CAP instances, the solution space of $n!$ assignments is large, and it is not easy to find assignments of high quality by the BI-based local search over local subregions. Consequently, the BI-based local search helps little on the convergence of the BIGLS, and the BIGLS takes as many generations as the SGA before termination. On the other hand, the BI-based local search consumes additional time in each generation of the BIGLS, resulting in longer overall computation time than the SGA. Although the BIGLS is the slowest one among the three heuristics in most of the large instances, the longer (but still reasonable) computation time is paid off by its highest quality of solutions.

4.3 Summary

This chapter proposes the BIGLS algorithm for solving the CAP which is a hybrid genetic algorithm with an embedded BI-based local search method. The BI is used not only in the local search but also in generating the initial population of chromosomes. The numerical experiments shows that the BIGLS behaves robust (hardly affected by

its random operations) and effective in solving the CAP. The BI-based local search in the BIGLS increases the solution quality for both the small and large CAP instances. It also accelerates the convergence of the BIGLS for the small instances but adds additional (still reasonable) computational burden for the large instances. The BIGLS improves almost all the non-optimal assignments generated by the BITA and finds the optimal assignments for most of the small instances. The BIGLS outperforms both the BITA and SGA significantly in terms of solution quality, demonstrating the critical role of the BI and the BI-based local search in combination with genetic algorithm.

Chapter 5

Conclusions

This dissertation studies two BI-related topics: the BI patterns in $Lin/Con/k/n$ systems and the BI-based heuristics and meta-heuristics for solving the CAP. The results can be summarized as follows.

5.1 Nature of BI patterns in $Lin/Con/k/n$ systems

We first summarize and clarify the existing results on BI patterns in $Lin/Con/k/n$ systems in the literature and notice that the existing research on BI patterns in $Lin/Con/k/n$ systems in the literature followed a fixed-length division method of $Lin/Con/k/n$ systems and tried to find repeated patterns among the divisions. However, through some numerical experiments, we find that such a division method is inappropriate and the repeated patterns do not exist under this division method.

This study explores the nature of BI patterns. We propose a new nonequal-length division of $Lin/Con/k/n$ system and discover consistent increasing-then-decreasing patterns among the new divisions. We further investigate the relationship between the BI patterns and the component reliability n and the relationship between the BI patterns and the system size p, which also generate some new BI patterns conditioned on p. These new findings also facilitate the understanding of the existing BI patterns,

the disproof of some conjectures, the construction of new conjectures, and even the future research on BI patterns in $Lin/Con/k/n$ systems.

5.2 BI-based heuristics for the CAP

BI has been successfully applied to design heuristics for the CAP. This study proposes three new heuristics (i.e., the LKB, LKC, and LKD heuristics) based on the scheme of the LKA heuristic and two new heuristics (i.e., the ZKC and ZKD heuristics) based on the schemes of the ZKA and ZKB heuristics. We study all the existing and new BI-based heuristics for the CAP thoroughly, including their theoretical properties and numerical performances.

Based on comprehensive numerical experiments, we recommend a two-stage approach (i.e., the BITA), in which Stage 1 uses both the LKA and LKB heuristics to generate two initial assignments, and Stage 2 uses either the ZKB or ZKD heuristic in combination with the two initial assignments obtained in Stage 1 to generate two improved assignments and then chooses the one with the higher system reliability as the final solution. In Stage 2, if all component reliabilities are less than or equal to 0.2, the ZKB is used; otherwise, the ZKD is used. Numerical tests show that the BITA can achieve the best performances of all the BI-based heuristics in most cases and outperforms the randomization method and the commercial GAMS/CoinBonmin solver.

5.3 A hybrid genetic algorithm for the CAP

A drawback with the BI-based heuristics as well as the BITA is that they may stop at local optimal or suboptimal solutions. Meta-heuristics are supposed to have better performances in solving the CAP due to their potential to escape from local optimums. Some simulated annealing and genetic algorithms have been proposed but are only applicable to some special systems and do not utilize the useful BI information.

To fill this gap, this study proposes a hybrid genetic algorithm (i.e., the BIGLS). It embeds the BI into the generation of the initial population and the local search approach (i.e., a BI-based pairwise exchange approach). We conduct comprehensive numerical experiments to evaluate the robustness and effectiveness of the BIGLS and the impact of embedding the BI in the BIGLS by comparing the BIGLS with the benchmarking methods (i.e., the BITA and a simple genetic algorithm). The results show that the BIGLS is robust to the random operations and is capable of generating higher quality of solutions than the benchmarking methods.

The outstanding performances of both the BITA and the BIGLS on solving the CAP inspire exploring the importance-measure based methods for solving various highly challenging problems in the field of reliability optimization (e.g., the reliability redundancy allocation problems (Kuo and Zuo, 2002)) and operations research (e.g., the logistics system design (Zhu et al., 2011)) by embedding the importance measures into the design of solution methods. The findings and tests on the BI-based heuristics in this dissertation will also benefit the development and evaluation of other BI-based methods.

Bibliography

Bibliography

Armstrong, M. J. (1995). Joint reliability-importance of elements. *IEEE Transactions on Reliability*, 44:408–412. 1

Aven, T. and Nokland, T. E. (2010). On the use of uncertainty importance measures in reliability and risk analysis. *Reliability Engineering and System Safety*, 95:127–133. 1

Beeson, S. and Andrews, J. D. (2003). Importance measures for noncoherent system analysis. *IEEE Transactions on Reliability*, 52:301–310. 1

Birnbaum, Z. W. (1969). On the importance of different components in a multicomponent system. In Krishnaiah, P. R., editor, *Multivariate analysis*, volume 2, pages 581–592. Academic Press, New York. 1, 2, 44

Boland, P. J., El-Neweihi, E., and Proschan, F. (1991). Redundancy importance and allocation of spares in coherent systems. *Journal of Statistical Planning and Inference*, 29:55–65. 1

Boland, P. J., Proschan, F., and Tong, Y. L. (1989). Optimal arrangement of components via pairwise rearrangements. *Naval Research Logistics*, 36:807–815. 55

Borgonovo, E. (2010). The reliability importance of components and prime implicants in coherent and non-coherent systems including total-order interactions. *European Journal of Operational Research*, 204:485–495. 1

Borgonovo, E. and Apostolakis, G. E. (2001). A new importance measure for risk informed decision-making. *Reliability Engineering and System Safety*, 72:193–212. 1, 2

Chadjiconstantinidis, S. and Koutras, M. V. (1999). Measures of component importance for markov chain imbeddable reliability structures. *Naval Research Logistics*, 46:613–639. 9, 14, 36

Chang, G. J., Cui, L., and Hwang, F. K. (1999). New comparisons in birnbaum importance for the consecutive-k-out-of-n system. *Probability in the Engineering and Informational Sciences*, 13:187–192. 9, 16

Chang, G. J., Hwang, F. K., and Cui, L. (2000). Corrigenda on "new comparisons in birnbaum importance for the consecutive-k-out-of-n system". *Probability in the Engineering and Informational Sciences*, 14:405–405. 9, 16

Chang, H. W., Chen, R. J., and Hwang, F. K. (2002). The structural birnbaum importance of consecutive-k systems. *Journal of Combinatorial Optimization*, 6:183–197. 3, 9, 14, 16, 23, 32, 35, 36, 37, 38

Chang, H. W. and Hwang, F. K. (2002). Rare-event component importance for the consecutive-k system. *Naval Research Logistics*, 49:159–166. 3, 9, 16, 37

Chang, J. C. and Hwang, F. K. (2003). Reliabilities of consecutive-k systems. In Pham, H., editor, *Handbook of Reliability Engineering*, pages 37–59. Springer, London. 33

Chao, M. T. and Lin, G. D. (1984). Economical design of large consecutive-k-out-of-n:f systems. *IEEE Transactions on Reliability*, 33:411–413. 2

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York. 78

Du, D. Z. and Hwang, F. K. (1986). Optimal consecutive-2-out-of-n systems. *Mathematics of Operations Research*, 11:187–191. 55

Eryilmaz, S. (2010). Review of recent advances in reliability of consecutive k-out-of-n and related systems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 224:225–237. 3

Freisleben, B. and Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 616–621, New York. IEEE Press. 74

Freixas, J. and Pons, M. (2008). Identifying optimal components in a reliability system. *IEEE Transactions on Reliability*, 57:163–170. 1

Gebre, B. A. and Ramirez-Marques, J. E. (2007). Element substitution algorithm for general two-terminal network reliability analyses. *IIE Transactions*, 39:265–275. 4

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA. 77, 78

Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 154–159, Hillsdale, NJ. Erlbaum. 79

Grefenstette, J. J., Gopal, R., Rosmaita, B., and Cucht, D. V. (1985). Genetic algorithms for the traveling salesman problem. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 160–168, Hillsdale, NJ. Erlbaum. 7, 8

Hwang, F. K. (1989). Invariant permutations for consecutive-k-out-of-n cycles. *IEEE Transactions on Reliability*, 38:65–67. 2

Kontoleon, J. M. (1979). Optimal link allocation of fixed topology networks. *IEEE Transaction on Reliability*, 28:145–147. 1, 4, 6, 42, 43, 50

Koulamas, C., Antony, S. R., and Jaen, R. (1994). A survey of simulated annealing applications to operations research problems. *Omega International Journal of Management Science*, 22:41–56. 7

Kuo, W., Velaga, R., Tillman, F. A., and Hwang, C. L. (2001). *Optimal Reliability Design: Fundamentals and Applications*. Cambridge University Press, U.K. 1, 2

Kuo, W., Zhang, W., and Zuo, M. J. (1990). A consecutive-k-out-of-n:g system: The mirror image of a consecutive-k-out-of-n:f system. *IEEE Transactions on Reliability*, 39:244–253. 1, 2, 9, 10, 27

Kuo, W. and Zuo, M. J. (2002). *Optimal reliability modeling: principles and applications*. Wiley & Sons, Hoboken, NJ. 1, 3, 23, 49, 91

Lim, M. H., Yuan, Y., and Omatu, S. (2000). Efficient genetic algorithms using simple gnens exchange local search policy for the quadratic assignment prolem. *Computational Optimization and Applications*, 15:249–268. 74, 75, 78

Lin, F. H. and Kuo, W. (2002). Reliability importance and invariant optimal allocation. *Journal of Heuristics*, 8:155–171. 1, 3, 4, 6, 7, 42, 44, 47, 48, 49, 55

Lin, F. H., Kuo, W., and Hwang, F. K. (1999). Structure importance of consecutive-k-out-of-n systems. *Operations Research Letters*, 25:101–107. 3, 9, 14

Lu, L. and Jiang, J. (2007). Joint failure importance for noncoherent fault trees. *IEEE Transactions on Reliability*, 56:435–443. 1

Malon, D. M. (1984). Optimal consecutive-2-out-of-n:f component sequencing. *IEEE Transaction on Reliability*, 33:414–418. 6, 55

Malon, D. M. (1985). Optimal consecutive-k-out-of-n:f component sequencing. *IEEE Transaction on Reliability*, 34:46–49. 6

Pentico, D. W. (2007). Assignment problems: a golden anniversary survey. *European Journal of Operational Research*, 176:774–793. 7

Prasad, V. R. and Kuo, W. (2000). Reliability optimization of coherent systems. *IEEE Transaction on Reliability*, 49:323–330. 55

Shen, J. and Zuo, M. J. (1994). Optimal design on series consecutive-k-out-of-n:g system with age-dependent minimal repair. *Reliability Engineering and System Safety*, 45:277–283. 2

Shingyoch, K. and Yamamoto, H. (2009). Efficient genetic algorithm for optimal arrangement in a linear consecutive-k-out-of-n:f system. *The Institute of Electronics, Information and Communication Engineering*, E92-A:1578–1584. 16

Shingyoch, K., Yamamoto, H., Tsujimura, Y., and Akiba, T. (2010). Proposal of simulated annealing algorithms for optimal arrangement in a circular consecutive-k-out-of-n: F system. *Quality Technology & Quantitative Management*, 7:395–405. 8

Shingyoch, K., Yamamoto, H., Tsujimura, Y., and Kambayashi, Y. (2009). Improvement of ordinal representation scheme for solving optimal component arrangement problem of circular consecutive-k-out-of-n:f system. *Quality Technology & Quantitative Management*, 6:11–22. 8

Shingyochi, K., Yamamoto, H., Kim, J. H., and Jang, K. W. (2009). Efficient genetic algorithm for optimal arrangement in a linear consecutive-k-out-of-n:f system. In *The Proceeding of The 9th Asian Pacific Industrial Engineering and Management Systems Conference (APIEMS2009)*, pages 1168–1176. 16

Tseng, L.-Y. and Lin, Y.-T. (2009). A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 198:84–92. 74

van der Borst, M. and Schoonakker, H. (2001). An overview of psa importance measures. *Reliability Engineering and System Safety*, 72:241–245. 1

Xie, M. and Lai, C. D. (1996). On the increase of the expected lifetime by parallel redundancy. *Asia-Pacific Journal of Operational Research*, 13:171–179. 1

Xie, M. and Shen, K. (1989). On ranking of system components with respect to different improvement actions. *Microelectronics Reliability*, 29:159–164. 1

Yamada, T. and Nakano, R. (1992). A genetic algorithm applicable to large scale job-shop problems. In Manner, R. and Manderick, B., editors, *Parallel Problem Solving from Nature*, volume II, pages 281–290. North-Holland, Amsterdam. 7

Yao, Q., Zhu, X., and Kuo, W. (2011). Heuristics for component assignment problems based on the birnbaum importance. *IIE Transactions*, page in press. 87

Zhu, X. and Kuo, W. (2008). Comments on "a hierarchy of importance indices". *IEEE Transaction on Reliability*, 57:529–531. 1

Zhu, X., Li, X., Yao, Q., and Chen, Y. (2011). Challenges and models in supportig logistics system design for dedicated-biomass-based bioenergy industry. *Bioresource Technology*, 102:1344–1351. 91

Zuo, M. J. (1993). Reliability and component importance of a consecutive-k-out-of-n system. *Microelectronics Reliability*, 33:243–258. 9, 13, 16

Zuo, M. J. and Kuo, W. (1990). Design and performance analysis of consecutive-k-out-of-n structure. *Naval Research Logistics*, 37:203–230. 1, 6, 7, 13, 42, 50, 51, 55, 57

Zuo, M. J. and Shen, J. (1992). System reliability enhancement through heuristic design. In *Proceedings of the 11th International Conference on Offshore Mechanics and Arctic Engineering (OMAE'92)*, volume II, pages 301–304, New York. ASME. 1, 4, 7, 42, 55, 58

Vita

Qingzhu Yao received a Bachelor of Science in Applied Mathematics from Tianjin University, Tianjin, China in 2002. He obtained a Master of Science in Computational Mathematics from Nankai University, Tianjin, China in 2005. He is currently a Ph.D. candidate in Industrial Engineering and a M.S. candidate in Statistics at University of Tennessee, Knoxville, TN. His research interests include reliability optimization in system design and operations research.