# A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms

**Ying-ping Chen**
**Tian-Li Yu**
**Kumara Sastry**
**David E. Goldberg**

NCLab Report No. NCL-TR-2007009
April 2007

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
`http://nclab.tw/`

# A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms

Ying-ping Chen[1], Tian-Li Yu[2], Kumara Sastry[3], and David E. Goldberg[3]

[1]Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
ypchen@nclab.tw

[2]Department of Electrical Engineering
National Taiwan University
Taipei 106, Taiwan
tianliyu@cc.ee.ntu.edu.tw

[3]Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{ksastry, deg}@uiuc.edu

April 13, 2007

**Abstract**

This paper reviews and summarizes existing linkage learning techniques for genetic and evolutionary algorithms in the literature. It first introduces the definition of linkage in both biological systems and genetic algorithms. Then, it discusses the importance for genetic and evolutionary algorithms to be capable of learning linkage, which is referred to as the relationship between decision variables. Existing linkage learning methods proposed in the literature are reviewed according to different facets of genetic and evolutionary algorithms, including the means to distinguish between good linkage and bad linkage, the methods to express or represent linkage, and the ways to store linkage information. Studies related to these linkage learning methods and techniques are also investigated in this survey.

## 1 Introduction

Genetic and evolutionary algorithms have been broadly and successfully applied to solving problems in numerous domains since they were proposed by Holland [1, 2]. As the scale and complexity of problems handled by genetic and evolutionary algorithms increase, researchers begin to realize that for practical use, certain crucial mechanisms have to be integrated into the framework of evolutionary computation. Among these crucial mechanisms suggested by practitioners is the ability to learn linkage, referred to as the relationship between variables. In the past few decades, there has been growing recognition that effective genetic and evolutionary computation demands understanding of linkage in order to tackle complicated, large scale problems [2, 3]. Studies have shown that easy problems can be solved by any ordinary genetic and evolutionary algorithms, but when harder problems are considered, scalability has been elusive. As indicated by the results presented in the literature [4, 5], even separable problems could be exponentially hard if the knowledge of the variable groups were not available.

In order to resolve the issue which is raised because the knowledge of the relationship between variables is unavailable, a variety of linkage learning techniques have been proposed and developed to handle the *linkage problem*, which refers to the need of good building-block linkage. These linkage learning techniques are so diverse, sophisticated, and highly integrated with the genetic algorithms that it is a difficult task to review all of them from a simple, unified, and straightforward point of view. Furthermore, given the importance of linkage learning in genetic and evolutionary algorithms and the amount of the effort made in this area, an up-to-date global overview of existing linkage learning techniques is needed not only for reviewing the current status of this field but also for revealing the potential future direction of research. As a consequence, a comprehensive survey is in order to serve as a milestone for the progress of research on linkage learning.

The purpose of this survey is to provide different facetwise views of existing linkage learning techniques as well as to gather the growing literature under a uniform classification. In particular, the paper reviews existing linkage learning techniques according to following different facets of genetic and evolutionary algorithms:

- the means to distinguish between good linkage and bad linkage;
- the methods to express or represent linkage;
- the ways to store linkage information.

Moreover, research which are precursors or closely related to these linkage learning techniques are also investigated.

The next section gives the definition of linkage in both biological systems and genetic algorithms. It also discusses the importance for genetic algorithms to learn linkage such that the coding traps can be avoided. Sections 3, 4, and 5 review existing linkage learning techniques according to the different viewpoints mentioned above. Related research are included in Section 6. Finally, Section 7 summarizes and concludes this paper.

## 2 Linkage: Definition and Importance

This section first introduces the definition of linkage in both fields of biology and evolutionary computation. Then, the need to employ the techniques for learning linkage when applying a genetic algorithm to solve problems is presented.

### 2.1 What Is Linkage?

The genetic algorithm is a powerful search methodology inspired by natural evolution. It imitates the procreation process and operates on the principle of the survival of the fittest. Therefore, understanding the bond and resemblance between the (natural) biology system and the (artificial) genetic and evolutionary algorithm may be helpful to realize the role and importance of learning linkage.

In biological systems, *linkage* refers to the level of association in inheritance of two or more non-allelic genes that is higher than to be expected from independent assortment [6]. During meiosis, *crossover* events might occur between strands of the chromosome that genetic materials are recombined as shown in Figure 1. Therefore, if two genes are closer to each other on a chromosome, there is a higher probability that they will be inherited by the offspring together. Genes are said to be *linked* when they reside on the same chromosome, and the distance between each other determines the level of their linkage. Figure 2 gives an illustrative example of different genetic linkage between two genes. The closer together a set of genes is on a chromosome; the more probable it will not be split by chromosomal crossover during meiosis.

When applying genetic algorithms, we usually use strings of characters drawn from a finite alphabets as chromosomes and genetic operators to manipulate these artificial chromosomes. Holland [2] suggested that genetic operators which can learn linkage information for recombining alleles might be necessary for genetic and evolutionary algorithms to succeed. Many well known and widely employed crossover operators, including one-point crossover and two-point

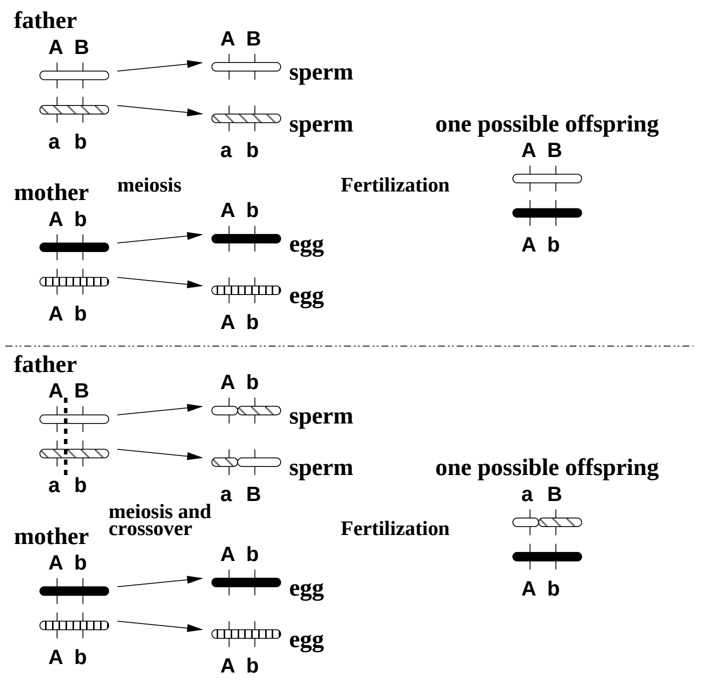

Figure 1: Meiosis and crossover. The upper part shows meiosis without crossover, and the lower part shows a crossover event occurs during meiosis.

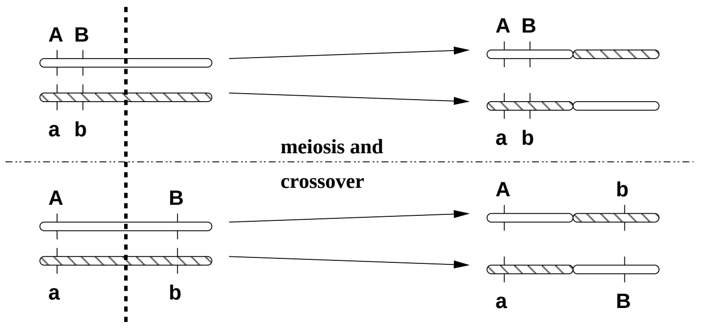

Figure 2: The different genetic linkage between two genes. The upper part shows that if the genes are closer, they are likely to maintain the allele configuration. The lower part shows that if the genes are far away from each other, it is likely for a crossover event to separate them and to change the configuration.

crossover, work under the similar situation subject to the linkage embedded in the chromosome representation as their biological counterparts do. For example, if we have a 6-bit function consisting of two independent 3-bit subfunctions, three possible coding schemes for the 6-bit chromosome are

$$C_1(A) = \mathtt{a}_0^0\ \mathtt{a}_1^0\ \mathtt{a}_2^0\ \mathtt{a}_3^1\ \mathtt{a}_4^1\ \mathtt{a}_5^1;$$

$$C_2(A) = \mathtt{a}_0^0\ \mathtt{a}_1^1\ \mathtt{a}_2^0\ \mathtt{a}_3^1\ \mathtt{a}_4^0\ \mathtt{a}_5^1;$$

$$C_3(A) = \mathtt{a}_0^0\ \mathtt{a}_1^0\ \mathtt{a}_2^1\ \mathtt{a}_3^1\ \mathtt{a}_4^1\ \mathtt{a}_5^0,$$

where $C_n(A)$ is the coding scheme $n$ for an individual $A$, and $a_i^j$ is the $i$th gene of $A$ and belongs to the $j$th subfunction.

Taking one-point crossover as an example, it is easy to see that genes belonging to the same subfunction of individuals encoded with $C_1$ are unlikely to be separated by crossover events. However, if the individuals are encoded with $C_2$, genes of the same subfunction are split almost in every crossover event. For $C_3$, genes of subfunction 0 are easily to be disconnected, while genes of subfunction 1 are likely to stay or to be transferred together.

From the viewpoint of genetic algorithms, *linkage* is used to describe and measure how close those genes that belong to a building block are on a chromosome. In addition to pointing out the linkage phenomenon, Holland [2] also suggested that the chromosome representation should adapt during the evolutionary process to avoid the potential difficulty directly caused by the coding scheme, which was identified as *coding traps*—the combination of loose linkage and deception among lower order schemata [7].

## 2.2 Linkage Learning as an Ordering Problem

Because encoding the solutions as fixed strings of characters is common in genetic algorithm practice, it is easy to see that linkage can be identified as the ordering of the loci of genes as the examples given in the previous section. Furthermore, early genetic algorithm researchers used to consider the linkage problem as an ordering problem of the chromosome representation and addressed to the same issue of building-block identification or linkage learning. That is, if a genetic algorithm is capable of rearranging the positions of genes on the fly during the evolutionary process, the responsibility of the user to choose a good coding scheme can be alleviated or even eliminated. To achieve this goal, Bagley [8] used the (*gene number*, *allele*) coding scheme to study the *inversion* operator for linkage learning by reversing the order of a chromosome segment but did not conclude in favor of the use of inversion. Frantz [9] further investigated the utility of inversion and reported that inversion was too slow and not very effective.

Goldberg and Bridges [10] analyzed the performance of a genetic algorithm with an idealized reordering operator. They showed that with an idealized reordering operator, the *coding traps*—the combination of loose linkage and deception among lower order schemata [7]—of a fixed chromosome representation can be overcome, and therefore, linkage learning can be achieved by an idealized reordering operator. This analysis was later extended to the tournament selection family, including pairwise tournament selection, $S$-ary tournament selection, and probabilistic tournament selection [11]. The upper bound of the probability to apply an idealized reordering operator found in the previous analysis on proportional selection did not exist when a tournament selection operator was used.

### 2.3 Why Is Learning Linkage Important?

These genetic algorithms either explicitly or implicitly act on an assumption of a good coding scheme which can provide tight linkage for genes of a building block on the chromosome. Goldberg, Korb, and Deb [4] conducted an experiment to demonstrate how linkage dictated the success of a simple genetic algorithm. They used an objective function composed of 10 uniformly scaled copies of an order-3 fully deceptive function [12, 13, 14, 15, 16, 17]. Three types of codings schemes were tested: tightly ordering, loosely ordering, and randomly ordering. The tightly ordering coding scheme is similar to $C_1$ described in the previous section. Genes of the same subfunction are arranged adjacent to one another on the chromosome. The loosely ordering coding scheme is like $C_2$, all genes are distributed evenly so that an overall loosest linkage can be achieved. The randomly ordering coding scheme arranges the genes according to an arbitrary order. The obtained results showed that the success of a simple genetic algorithm depends very much on the degree of linkage of building blocks. If the chromosome representation provides tight linkage, a simple genetic algorithm can solve difficult problems. Otherwise, simple genetic algorithms can easily fail. Therefore, for simple genetic algorithms, tight linkage or a good coding scheme is indeed far more important than it is usually considered.

In addition to the experiment done by Goldberg, Korb, and Deb [4], some other studies [18, 5, 19] also showed that genetic algorithms work very well if the genes belonging to the same building block are tightly linked together on the chromosome. Otherwise, if these genes spread all over the chromosome, building blocks are very hard to be created and easy to be destroyed by the recombination operator. Genetic algorithms cannot perform well under such circumstances. In practice, without prior knowledge to the problem and linkage information, it is difficult to guarantee that the coding scheme defined by the user always provides tight building blocks, although it is a key to the success of genetic algorithms.

It is clear that for simple genetic algorithms with fixed genetic operators and chromosome representations, one of the essential keys to success is a good coding scheme that puts genes belonging to the same building blocks together on the chromosome to provide tight linkage of building blocks. The linkage of building blocks dominates all kinds of building-block processing, including creation, identification, separation, preservation, and mixing. However, in the real world, it is usually difficult to know such information a priori. As a consequence, handling linkage for genetic algorithms to succeed is very important.

## 3 Unimetric Approach vs. Multimetric Approach

In this section and the following two sections, we will review existing linkage learning techniques according to different facets and aspects, including the means to distinguish between good linkage and bad linkage, the methods to express or represent linkage, and the ways to store linkage information. First, we start with classifying the linkage learning techniques based on the means employed in the algorithm to distinguish between good linkage and bad linkage in this section.

As a part of evolutionary computation, biologically inspired linkage learning techniques grow out of "fitness only" measures and try to make use of only what is provided by the problem. However, computer science and data mining approaches strive to best describe the population statistics, and therefore, artificial criteria which are not directly related to the problem are usually employed to judge the quality of the linkage configuration. The ways of thinking behind these two kinds of approaches are fundamentally different, and it is the reason we propose this classification criterion.

According to the means to distinguish between good linkage and bad linkage, we can roughly classify existing genetic and evolutionary approaches into the following two categories:

- **Unimetric approach.** A *unimetric* approach acts solely on the fitness value given by the fitness function. No extra criteria or measurements are involved for deciding whether an individual or a model is better.

- **Multimetric approach.** In contrast to unimetric approaches, a *multimetric* approach employs extra criteria or measurements other than the fitness function given by the problem for judging the quality of individuals or models.

Unimetric approaches, loosely modeled after natural environments, are believed to be more biologically plausible, while multimetric approaches are of artificial design and employ certain bias which does not come from the problem at hand to guide the search. Specifically, the reasons and motivation to propose this classification to discriminate unimetric approaches and multimetric approaches are two-fold:

1. **Biological plausibility:** One of the most important reasons to propose this classification is that we believe nature appears unimetric. Because the "fitness" of an individual in nature depends on whether or not it can adapt to its environment and survive in its environment, there is obviously no other extra measurement or criterion to enforce or guide the evolution of the species to go to certain direction, such as becoming as simple as it can be. However, given the current research results in this field that most good evolutionary approaches are multimetric ones, which utilize one or more user-defined measurements to determine the solution quality, such as preference for simpler models, we would like to separate unimetric approaches from multimetric ones and to know if there are limits to performance of unimetric methods. The theoretical results obtained on unimetric approaches might be of some significance or interests in biology, although the computational models are highly simplified.

2. **Technological motivations:** In addition to the biological viewpoints, there are also technological motivations to classify existing linkage learning techniques into unimetric approaches and multimetric approaches. For most multimetric methods, the algorithmic operations are serial in design, while unimetric methods are oftentimes easy to parallelize. The multimetric algorithms usually require access to all or a large part of the individuals in the population at the same time. This kind of requirement removes potential parallel advantages because it either incurs a high communication cost due to the necessary information exchange or demands a completely connected network topology to lower the communication latency. Therefore, it may be a foreseeable bottleneck when handling problems of a large number of variables. On the other hand, although many unimetric methods, such as the linkage learning genetic algorithm, do not perform as well as multimetric ones, they oftentimes use pairwise operators or operators that operate on only a few individuals. Hence, they are relatively easy to parallelize, and a wide range of parallelization methods are applicable.

According to these motivations, the means to distinguish between good linkage and bad linkage is adopted to classify existing linkage learning techniques.

For example, because all the simple genetic algorithms [2, 20, 19] and the *linkage learning genetic algorithm* (LLGA) [21, 22, 23, 24, 25, 26, 27, 28, 29] use only fitness values to operate, they are definitely considered as unimetric approaches. Moreover, the simple genetic algorithms with *inversion* [8, 30, 31, 32, 33], *punctuation marks* [34], *masked crossover* (MX) [35], *shuffle crossover* (SHX) [36], *adaptive uniform crossover* (AUX) [37], *metabits* [38], *selective crossover* (SX) [39, 40, 41], or *linkage evolving genetic operator* (LEGO) [42, 43, 44], are also included in unimetric approaches because no extra measurements are utilized in these algorithms for comparing the solution or model quality. A more detailed introduction for the adaptive crossover operators mentioned above can be found elsewhere [45]. Furthermore, introducing non-coding segments, which was previously called *introns*, into the chromosome representation can also achieve linkage learning [46, 47, 48, 49, 50, 51, 52, 53, 54, 55], and the approaches with non-coding segments are usually unimetric. As a side note, adaptive crossover and non-segments are also widely used in genetic programming [56, 57, 58, 59, 60, 61, 62].

On the other hand, most advanced genetic algorithms today, including the *gene expression genetic algorithm* (gemGA) [63, 64, 65, 66, 67, 68, 69], the *estimation of distribution algorithms* (EDAs) [70, 71, 72], the *mutual-information-maximizing input clustering* (MIMIC) algorithm [73], the *combining optimizers with mutual information trees* (COMIT) method [74, 75, 76], the *bivariate marginal distribution algorithm* (BMDA) [77], the *Bayesian optimization algorithm* (BOA) [78, 79, 80, 81], the *factorized distribution algorithm* (FDA) [82, 72, 83, 84], the mixed $\mathbb{ID}\mathbb{E}$A [85, 86, 87, 88, 89, 90], the *extended compact genetic algorithm* (ECGA) [91, 92, 93, 94], the *extended compact genetic programming* (ECGP) [95], edge histogram based sampling algorithm (EHBSA) [96, 97], and the like, are classified as multimetric approaches because they explicitly employ extra mechanisms or measurements for discriminating between good linkage and bad linkage. In addition to the obvious classification, approaches such as the *messy genetic algorithm* (mGA) [4, 98, 99, 100], the *fast messy genetic algorithm* (fmGA) [101, 102, 103], the *ordering messy genetic algorithm* (OmeGA) [104, 105, 106, 107, 108], the *structured messy genetic algorithm* [109], and the *incremental commitment genetic algorithm* [110] are in between the two classes. The members of the messy genetic algorithm family compare individuals with the fitness value, but the use of building-block filtering indeed builds an implicit extra mechanism that prefers shorter building blocks into these genetic and evolutionary algorithms.

## 4 Physical Linkage vs. Virtual Linkage

After classifying the linkage learning techniques according to the facet of how they distinguish between good linkage and bad linkage, in this section, we discuss the aspect of the methods these algorithms use to express or represent linkage.

As the development of evolutionary computation progresses, early linkage learning schemes that were biologically inspired usually represent linkage physically with the representation, such as proximity of genes on a chromosome. When computer science and data mining techniques start to get involved in the linkage learning mechanism, linkage are quite often expressed in a virtual way, such as probabilistic models. We adopt this classification criterion because such different designs indicate the trade-off between the biological inspiration and the quest for the algorithmic improvement.

According to the methods to represent linkage, we can broadly classify existing genetic and evolutionary approaches into the following two categories:

- **Physical linkage.** A genetic and evolutionary algorithm is said to use *physical linkage* if in this algorithm, linkage emerges from physical locations of two or more genes on the chromosome.
- **Virtual linkage.** On the other hand, if a genetic and evolutionary algorithm uses graphs, groupings, matrices, pointers, or other data structures that control the subsequent pairing or clustering organization of decision variables, it is said to use *virtual linkage*.

Physical linkage is closer to biological plausibility and inspired directly by it, while virtual linkage is an engineering or computer science approach to achieve the desired effect most expeditely. In particular, similar to the reasons that were discussed in the previous section, the motivations to look into this classification are also two-fold:

1. **Biological plausibility:** Because genetic and evolutionary algorithms are search techniques based on principles of evolution, it is one of our main interests to learn from nature and to borrow useful insights, inspirations, or mechanisms from genetics or biology. Given that the natural evolution apparently proceeds via genetic operations on the genotypic structures of all creatures, genetic and evolutionary algorithms that employ the mechanisms which are close to that in nature should be recognized and emphasized. By pointing out this feature or characteristic of the genetic and evolutionary algorithms that use the mechanisms existing in biological systems, we might be able to theorize certain genetic operations in biological systems with those genetic algorithms using physical linkage, such as the messy genetic algorithm and the linkage learning genetic algorithm.

2. **Algorithmic improvement:** From a standpoint of efficient or effective computation, genetic and evolutionary algorithms using virtual linkage usually yield better performance than those using physical linkage. Together with the biological point of view, this might imply two possible situations:

   (a) Using virtual linkage in genetic algorithms can achieve a better performance. This kind of artificial systems can do better than their biological counterparts on conducting search and optimization;

   (b) The power of natural systems has not been fully understood and utilized yet. More critical and essential mechanisms existing in genetics and biology should be further examined and integrated into the algorithms to improve the performance.

   Hence, for the purpose of search and optimization, in the first situation, we should focus on developing better algorithms that employ virtual linkage, such as the *probabilistic model-building genetic algorithms* (PMBGAs) or EDAs [111, 112]. In the other situation, we should appropriately choose useful genetic mechanisms and integrate these mechanisms into the algorithms.

According to these motivations, the methods to express or represent linkage is used to classify existing linkage learning techniques in this section.

For example, all the genetic algorithms use fixed chromosome representations without any extra graph, grouping, matrix, pointer, or data structure to describe linkage in principle fall into the category of physical linkage. These algorithms include the ones using binary strings, integer strings, or real-variable strings as chromosomes as long as they use the chromosome alone for operations and evolution. Another major set of algorithms belonging to the category of physical linkage is the genetic algorithms that use the (gene number, allele) coding scheme [8, 30]. This set of genetic algorithms includes inversion [8, 30, 31, 32, 33], the messy genetic algorithm [4, 98, 99, 100], the fast messy genetic algorithm [101, 102, 103], and the linkage learning genetic algorithm [21, 22, 23, 24, 25, 26, 27, 28, 29].

Furthermore, because probabilistic models are employed to represent linkage of variables in PMBGAs and EDAs, the category of virtual linkage includes all PMBGAs and EDAs [70, 71, 72, 111, 112], such as the mutual-information-maximizing input clustering algorithm [73], the combining optimizers with mutual information trees method [74, 75, 76], the bivariate marginal distribution algorithm [77], the Bayesian optimization algorithm [78, 79, 80, 81], the factorized distribution algorithm [82, 72, 83, 84], the mixed IDEA [85, 86, 87, 88, 89, 90], and the extended compact genetic algorithm [91, 92, 93, 94]. It also contains the probabilistic inference framework for modeling crossover operators [113, 114, 115], such as *general linkage crossover* (GLinX) and *adaptive linkage crossover* (ALinX), and the linkless self-distancing GA [116].

## 5 Distributed Model vs. Centralized Model

The last facet of the genetic and evolutionary algorithm we explore in this work for classifying the linkage learning techniques is the ways for these approaches to store linkage information. For the biologically inspired linkage learning schemes, the evolved linkage models tend to be distributed in each individual, which are similar to those observed in nature. However, in order to facilitate the computational process, the linkage models generated by the methods utilizing computer science and data mining approaches are usually centralized as global models. To gain further insights into the nature and property of linkage, we propose this criterion to classify existing linkage learning methods.

Based on the ways to store linkage information, we can divide existing genetic and evolutionary approaches into the following two categories:

- **Distributed Model.** If a genetic and evolutionary algorithm has no centralized storage of linkage information and maintains the genetic-linkage model in a distributed manner, we call such a genetic algorithm a *distributed-model* approach.
- **Centralized Model.** In contrast to distributed-model approaches, a *centralized-model* approach utilizes a centralized storage of linkage information, such as a global probabilistic vector or dependency table, to handle and process linkage.

Similar to the unimetric approach, distributed-model approaches are also loosely modeled after evolutionary conditions in nature and more biologically plausible, while centralized-model approaches are developed to achieve the maximum information exchange and to obtain the desired results. The reasons to propose this classification to show the difference between distributed-model approaches and centralized-mode approaches are presented as follows:

1. **Biological plausibility:** Once more, we propose this classification in order to put an emphasis on the similarities as well as the dissimilarities between the genetic algorithms and the biological systems. Apparently, there exists no centralized genetic-linkage model in nature. Genotypes are distributed on all creatures or individuals. As described in the previous sections, genetic algorithms fall in the category of distributed model might serve as highly simplified computation models which can give insight of the way nature or evolution works.
2. **Computational motivations:** On the other hand, based on the classification, centralized-model approaches should be expected to have better performance when executing computation, such as search or optimization, because by centralizing the genetic-linkage model, genetic-linkage information existing in the population gets well mixed and exchanged in very little time compared to that in a distributed-model approach. Therefore, centralized-model approaches have such an edge to outperform distributed-model. However, this advantage might also be a disadvantage for centralized-model approaches. Centralized-model approaches are serial in nature, and they are very hard to parallelize. Distributed-model approaches are parallel by design. Thus, distributed-model approaches might have better *scalability* when handling large-scale problems.

According to these reasons, the ways to store linkage information is adopted to classify the linkage learning techniques.

For example, simple genetic algorithms are distributed-model approaches because any information existing in the population is stored in a distributed manner over the individuals. The linkage learning genetic algorithm [21, 22, 23, 24, 25, 26, 27, 28, 29], the messy genetic algorithm [4, 98, 99, 100], the fast messy genetic algorithm [101, 102, 103], and the *gene expression messy genetic algorithm* (gemGA) [63, 64, 65, 66, 67, 68, 69] also belong to this category for the same reason. Moreover, the linkage identification procedures proposed in the literature, including the *linkage identification by nonlinearity check* (LINC) [117, 118], the *Identifying composability using group perturbation* (gLINC) [119], the *linkage identification by non-monotonicity detection* (LIMD) [120, 121], the *linkage identification based on epistasis measures* (LIEM) [122, 123, 124], the *linkage identification with epistasis measure considering monotonicity conditions* (LIEM$^2$) [125], the *Linkage identification by nonlinearity check for real-coded genetic algorithms* (LINC-R) [126], and the *Dependency detection for distribution derived from df* (DDDDD or D$^5$) [127, 128, 129] as well as the *collective learning genetic algorithm* (CLGA) [130, 131] are in this class.

Furthermore, similar to the category of virtual linkage, the centralized-model approaches include most PMBGAs and EDAs [70, 71, 72, 111, 112], such as the mutual-information-maximizing input clustering algorithm [73], the combining optimizers with mutual information trees method [74, 75, 76], the bivariate marginal distribution algorithm [77], the Bayesian optimization algorithm [78, 79, 80, 81], the factorized distribution algorithm [82, 72, 83, 84], the mixed I$\mathbb{DEA}$ [85, 86, 87, 88, 89, 90], and the extended compact genetic algorithm [91, 92, 93, 94], and the like. The probabilistic inference framework for modeling crossover operators [113, 114, 115], such as the general linkage crossover and the adaptive linkage crossover, the *dependency structure matrix driven genetic algorithm* (DSMGA) [132, 133, 134], and the linkless self-distancing genetic algorithm [116], are also considered as centralized-model approaches.

## 6 Related Research

In this section, research related to the linkage learning techniques classified in the previous sections of this paper are presented. These mechanisms, operators, or theoretical frameworks might be applied in genetic and evolutionary algorithms to learn linkage in the future or give a better understanding of linkage learning in theory.

First of all, based on the idea of using the inversion operator with the (gene number, allele) coding scheme, permutation-based operators or methods can potentially be utilized for learning linkage. These operators and methods include *partially mapped crossover* (PMX) [135], *order crossover* (OX) [136], *cycle crossover* (CX) [136], *edge recombination* (ER) [137], *enhanced edge recombination* (EER) [138], *uniform ordering crossover* (UOX) [139], *relative ordering crossover* (ROX) [140], and the *random keys* [141]. With the (gene number, allele)-style coding or other appropriate permutation coding schemes, these genetic operators might help genetic and evolutionary algorithms to achieve linkage learning.

Many linkage learning techniques presented in the previous sections employ certain kinds of grouping or clustering methodologies in order to identify building blocks. For tackling the clustering problem, Falkenauer [142, 143] proposed the *grouping genetic algorithm* (GGA) specifically for solving clustering problems. GGA uses a specially designed chromosome representation and the grouping crossover operator such that clustering problem can be naturally handled. Although GGA has no linkage learning mechanism in the context of this survey, potentially, GGA can be employed as a linkage group identifying method for learning linkage. Because of its nature, GGA has been applied to grouping-oriented problems, including the bin packing problem [144, 145, 146], the equal pile problem [147], and other real-world problems [148].

Other than methods and operators, theoretical research regarding linkage can be found in the literature. Heckendorn and Alden proposed a series of theories on identifying linkage via limited probing [149, 150]. Prügel-Bennett [151] presented a statistical framework to model the linkage dynamics of a genetic algorithm with ranking selection, two-point crossover, and mutation on the Onemax problem. Auto-correlation and cross-correlation among genes were utilized to construct the linkage dynamics. Analyses of applying a reordering operator with different selection schemes on a GA-hard problem were also provided elsewhere [10, 11]. An idealized reordering operator and the genetic algorithm were modeled and analyzed with a set of difference equations. For studying the inversion operator, [152] proposed the use of problem generators to observe the probability for inversion. Finally, previous surveys related to linkage and linkage learning are available in the literature [153, 154].

If the problem domain knowledge is available for creating appropriate chromosome representations or designing suitable genetic operators, research can also be found in the literature to incorporate the priori knowledge in the genetic and evolutionary algorithms. Bui and Moon [155] proposed the *Hyperplane Synthesis* procedure, which employs the depth-first-search (DFS) and the breadth-first-search (BFS) tree traversal algorithms on the graph representation of the problem for defining good chromosome representations [156, 157, 158]. The proposed DFS/BFS gene arrangement procedure has been successfully applied to a variety of problems, including the traveling salesman problem (TSP) [159], graph partitioning [160], circuit ratio-cut partitioning [161], and VLSI circuit partitioner [162, 163]. In additional to creating an appropriate chromosome encoding scheme, *natural crossover* was proposed [164] for problems that have strong geographical linkage. Natural crossover has been used to optimize the artificial neural networks [165], the vehicle routing problem [166], the fixed channel assignment problem [167], and TSP [168, 169] as well. Similar to natural crossover, *Voronoi quantized crossover* (VQX) was proposed to solve TSP [170] and the sequential ordering problem [171]. Instead of using free curves, VQX uses the concept of Voronoi diagrams to swap the geographical regions in order to preserve the geographical linkage within the underlying problem. A more complete survey on chromosomal structures that exploit topological linkage can be found elsewhere [172].

## 7 Conclusions

As pointed out by Holland and verified by a number of studies, learning linkage is essential to the success of genetic and evolutionary algorithms if the prior knowledge to the problem is not available for designing a chromosome representation that provides good building block linkage. Recognizing the importance of solving the linkage problem, many linkage learning techniques have been proposed in the literature to tackle the linkage problem. These methods adopt a variety of mechanisms for linkage detecting, learning, and utilization. In this paper, we reviewed these linkage learning techniques from three different aspects: (1) the means to distinguish between good linkage and bad linkage; (2) the methods to express or represent linkage; (3) the ways to store linkage information. Research closely related these linkage learning techniques were also included.

In addition to the classification proposed in this paper, according to the time line on which the techniques included in this paper were proposed, we can observe two directions: (1) using the simple chromosome representation with the extra information about linkage groups; (2) using the complex model builders to capture linkage in probabilistic models. On the one hand, fixed representations are easier for genetic operators to manipulate. As long as the linkage groups are flexible enough to express the interaction among genes of the problem, using a simple representation with flexible linkage groups may be a good choice between cost and effectiveness. On the other hand, if the problem is too complicated for a simple representation, those complex model builders may be the only way to solve such difficult problems.

The research field of genetic and evolutionary computation is deeply inspired by nature, biology, and evolution. Every technique or methodology proposed in this field serves the following purposes: achieving excellent computational performance and/or gaining better understandings of nature. Integrating the concept of genetic linkage into evolutionary algorithms creates the research branch of linkage learning methodologies as well as leads us to investigate the applicability of observed phenomena in biology to computation. Overall, from nature, we may learn to develop general computational frameworks which can handle a broad rage of problems, and from the development of these frameworks, perhaps we can also further human knowledge to nature, biology, and evolution.

## Acknowledgments

This work was partially sponsored by the National Science Council of Taiwan under grants NSC-95-2221-E-009-092 and NSC-95-2627-B-009-001 as well as by the MOE ATU Program. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

This work was also sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, and the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

## References

[1] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, 1973.

[2] ——, *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press, 1975, ISBN: 0-262-58111-6.

[3] D. E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, ser. Genetic Algorithms and Evoluationary Computation. Kluwer Academic Publishers, June 2002, vol. 7, ISBN: 1-4020-7098-5.

[4] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1989.

[5] D. E. Goldberg, K. Deb, and D. Thierens, "Toward a better understanding of mixing in genetic algorithms," *Journal of the Society of Instrument and Control Engineers*, vol. 32, no. 1, pp. 10–16, 1993.

[6] D. L. Hartl and E. W. Jones, *Genetics: principles and analysis*, 4th ed. Sudbury, MA: Jones and Bartlett Publishers, January 1998, ISBN: 0-7637-0489-X.

[7] D. E. Goldberg, "Simple genetic algorithms and the minimal, deceptive problem," in *Genetic Algorithms and Simulated Annealing*, D. L., Ed. Los Altos, CA: Morgan Kaufmann Publishers, 1987, ch. 6, pp. 74–88, ISBN: 0-2730-8771-1.

[8] J. D. Bagley, "The behavior of adaptive systems which employ genetic and correlation algorithms," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1967, (University Microfilms No. 68-7556).

[9] D. R. Frantz, "Nonlinearities in genetic adaptive search," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1972, (University Microfilms No. 73-11116).

[10] D. E. Goldberg and C. L. Bridges, "An analysis of a reordering operator on a GA-hard problem," *Biological Cybernetics*, vol. 62, pp. 397–405, 1990.

[11] Y.-p. Chen and D. E. Goldberg, "An analysis of a reordering operator with tournament selection on a GA-hard problem," *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 825–836, 2003.

[12] D. H. Ackley, *A connectionist machine for genetic hill climbing.* Boston: Kluwer Academic, 1987.

[13] D. E. Goldberg, "Genetic algorithms and Walsh functions: Part I, a gentle introduction," *Complex Systems*, vol. 3, no. 2, pp. 129–152, 1989.

[14] ——, "Genetic algorithms and Walsh functions: Part II, deception and its analysis," *Complex Systems*, vol. 3, no. 2, pp. 153–171, 1989.

[15] K. Deb and D. E. Goldberg, "Analyzing deception in trap functions," *Foundations of Genetic Algorithms 2*, pp. 93–108, 1993.

[16] K. Deb, J. Horn, and D. E. Goldberg, "Multimodal deceptive functions," *Complex Systems*, vol. 7, no. 2, pp. 131–153, 1993.

[17] K. Deb and D. E. Goldberg, "Sufficient conditions for deceptive and easy binary functions," *Annals of Mathematics and Artificial Intelligence*, vol. 10, pp. 385–408, 1994.

[18] D. Thierens, "Analysis and design of genetic algorithms," Ph.D. dissertation, Katholieke Universiteit Leuven, Leuven, Belgium, 1995.

[19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley Publishing Co., January 1989, ISBN: 0-201-15767-5.

[20] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1975, (University Microfilms No. 76-9381).

[21] G. R. Harik and D. E. Goldberg, "Learning linkage," *Foundations of Genetic Algorithms 4*, pp. 247–262, 1996.

[22] G. R. Harik, "Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1997.

[23] F. G. Lobo, K. Deb, D. E. Goldberg, G. R. Harik, and L. Wang, "Compressed introns in a linkage learning genetic algorithm," in *Proceedings of the Third Annual Conference on Genetic Programming (GP 98).* University of Wisconsin, Madison, WI: Morgan Kaufmann, August 1998, pp. 551–558.

[24] F. G. Lobo, G. R. Harik, and D. E. Goldberg, "Linkage learning genetic algorithm in C++," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 98010, 1998.

[25] G. R. Harik and D. E. Goldberg, "Learning linkage through probabilistic expression," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 295–310, June 2000.

[26] Y.-p. Chen and D. E. Goldberg, “Introducing start expression genes to the linkage learning genetic algorithm,” *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, pp. 351–360, 2002.

[27] ——, “Tightness time for the linkage learning genetic algorithm,” *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 837–849, 2003.

[28] ——, “Convergence time for the linkage learning genetic algorithm,” *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, 2004.

[29] Y.-p. Chen, “Extending the scalability of linkage learning genetic algorithms: Theory and practice,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 2004.

[30] R. S. Rosenberg, “Simulation of genetic populations with biochemical properties,” Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1967, (University Microfilms No. 67-17836).

[31] P. J. Kennedy and T. R. Osborn, “A double-stranded encoding scheme with inversion operator for genetic algorithms,” *Proceedings of Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, pp. 398–407, 2001.

[32] O. T. Sehitoglu and G. Ü̈çoluk, “A building block favoring reordering method for gene positions in genetic algorithms,” *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 571–575, 2003.

[33] A. B. Simōes and C. Erensto, “Transposition versus crossover: An empirical study,” *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-1999)*, pp. 612–619, 1999.

[34] J. D. Schaffer and A. Morishima, “An adaptive crossover distribution mechanism for genetic algorithms,” *Proceedings of the Second International Conference on Genetic Algorithms (ICGA-87)*, pp. 36–40, 1987.

[35] S. J. Louis and G. J. E. Rawlins, “Designer genetic algorithms: Genetic algorithms in structure design,” *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pp. 53–60, 1991.

[36] L. J. Eshelman and D. J. Schaffer, “Productive Recombination and Propagating and Preserving Schemata,” *Foundations of Genetic Algorithms 3*, pp. 299–313, 1994.

[37] T. White and F. Oppacher, “Adaptive crossover using automata,” *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pp. 229–238, 1994.

[38] J. R. Levenick, “Metabits: Generic endogenous crossover control,” *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pp. 88–95, 1995.

[39] K. Vekaria and C. Clack, “Selective crossover in genetic algorithms: An empirical study,” *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*, pp. 438–447, 1998.

[40] ——, “Schema propagation in selective crossover,” *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, p. 268, 1999, (Late Breaking Paper).

[41] ——, “Royal road encodings and schema propagation in selective crossover,” in *Proceedings of Fourth Online World Conference on Soft Computing in Industrial Applications.* Springer-Verlag, September 1999, pp. 281–292.

[42] J. Smith and T. C. Fogarty, “An adaptive poly-parental recombination strategy,” *Proceedings of AISB-95 Workshop on Evolutionary computing*, pp. 48–61, 1995.

[43] ——, “Recombination strategy adaptation via evolution of gene linkage,” *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 826–831, 1996.

[44] J. E. Smith, “Self adaptation in evolutionary algorithms,” Ph.D. dissertation, University of the West of England, 1998.

[45] W. M. Spears, “Recombination parameters,” in *The Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. New York, NY: Oxford University Press, 1997, ch. E1.3, pp. E1.3:1–E1.3:13, ISBN: 0-7503-0895-8.

[46] J. R. Levenick, “Inserting introns improves genetic algorithm success rate: Taking a cue from biology,” *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pp. 123–127, 1991.

[47] S. Forrest and M. Mitchell, “Relative building-block fitness and the building-block hypothesis,” *Foundations of Genetic Algorithms 2*, pp. 109–126, 1993.

[48] A. S. Wu, R. K. Lindsay, and M. D. Smith, “Studies on the effect of non-coding segments on the genetic algorithm,” *Proceedings of the Sixth IEEE Conference on Tools with Artificial Intelligence*, 1994.

[49] A. S. Wu and R. K. Lindsay, “Empirical studies of the genetic algorithm with noncoding segments,” *Evolutionary Computation*, vol. 3, no. 2, pp. 121–147, 1995.

[50] ——, “A survey of intron research in genetics,” *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV)*, pp. 101–110, 1996.

[51] ——, “A comparison of the fixed and floating building block representation in the genetic algorithm,” *Evolutionary Computation*, vol. 4, no. 2, pp. 169–193, 1997.

[52] H. A. Mayer, “ptGAs—Genetic algorithms evolving noncoding segments by means of promoter/terminator sequences,” *Evolutionary Computation*, vol. 6, no. 4, pp. 361–386, 1999.

[53] D. S. Burke, K. A. De Jong, J. J. Grefenstette, C. L. Ramsey, and A. S. Wu, “Putting more genetics into genetic algorithms,” *Evolutionary Computation*, vol. 6, no. 4, pp. 387–410, 1999.

[54] C.-Y. Lee and E. K. Antonsson, “Adaptive evolvability via non-coding segment induced linkage,” *Proceedings of Genetic and Evolutionary Computation Conference 2000 (GECCO-2000)*, pp. 448–453, 2000.

[55] T. Haynes, “Collective adaptation: The exchange of coded segments,” *Evolutionary Computation*, vol. 6, no. 4, pp. 311–338, 1999.

[56] P. J. Angeline, “Two self-adaptive crossover operations for genetic programming,” in *Advances in Genetic Programming*, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, October 1996, vol. 2, ch. 5, pp. 89–109, ISBN: 0-262-01158-1.

[57] H. Iba and H. de Garis, "Extending genetic programming with recombinative guidance," in *Advances in Genetic Programming*, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, October 1996, vol. 2, ch. 4, pp. 69–88, ISBN: 0-262-01158-1.

[58] M. Wineberg and F. Oppacher, "The benefits of computing with introns," in *Proceedings of the Third Annual Conference on Genetic Programming (GP 96)*, 1996, pp. 410–415.

[59] D. Andre and A. Teller, "A study in program response and the negative effects of introns in genetic programming," in *Proceedings of the Third Annual Conference on Genetic Programming (GP 96)*, 1996, pp. 12–20.

[60] P. Nordin, F. Francone, and W. Banzhaf, "Explicitly defined introns and destructive crossover in genetic programming," in *Advances in Genetic Programming*, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, October 1996, vol. 2, ch. 6, pp. 111–134, ISBN: 0-262-01158-1.

[61] J. R. Levenick, "Swappers: Introns promote flexibility, diversity and invention," *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 361–368, 1999.

[62] H. Iba and M. Terao, "Controlling effective introns for multi-agent learning by genetic programming," *Proceedings of Genetic and Evolutionary Computation Conference 2000 (GECCO-2000)*, pp. 419–426, 2000.

[63] H. Kargupta, "The gene expression messy genetic algorithm," *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 814–819, 1996.

[64] ——, "The performance of the gene expression messy genetic algorithm on real test functions," *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 631–636, 1996.

[65] H. Kargupta and D. E. Goldberg, "SEARCH, Blackbox Optimization, and Sample Complexity," *Foundations of Genetic Algorithms 4*, pp. 291–324, 1996.

[66] H. Kargupta, "Search, computational processes in evolution, and preliminary development of the gene expression messy genetic algorithm," *Complex Systems*, vol. 11, no. 4, pp. 233–287, 1997.

[67] H. Kargupta, D. E. Goldberg, and L. Wang, "Extending the class of order-$k$ delineable problems for the gene expression messy genetic algorithm," in *Proceedings of the Second Annual Conference on Genetic Programming (GP 97)*. Stanford University, CA: Morgan Kaufmann, August 1997, pp. 364–369.

[68] H. Kargupta and S. Bandyopadhyay, "Further experimentations on the scalability of the gemGA," *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*, pp. 315–324, 1998.

[69] S. Bandyopadhyay, H. Kargupta, and G. Wang, "Revisiting the gemGA: Scalable evolutionary optimization through linkage learning." *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 603–608, 1998.

[70] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV)*, pp. 178–187, 1996.

[71] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evolutionary Computation*, vol. 5, no. 3, pp. 303–346, 1997.

[72] H. Mühlenbein, T. Mahnig, and A. Ochoa, "Schemata, distributions and graphical models in evolutionary optimization," *Journal of Heuristics*, vol. 5, pp. 215–247, 1999.

[73] J. S. D. Bonet, C. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Advances in Neural Information Processing Systems*, vol. 9, pp. 424–430, 1996.

[74] S. Baluja and S. Davies, "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 30–38, 1997.

[75] ——, "Fast probabilistic modeling for combinatorial optimization," *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, (AAAI/IAAI 98)*, pp. 469–476, 1998.

[76] S. Baluja, "Genetic algorithms and explicit search statistics," *Advances in Neural Information Processing Systems*, vol. 9, pp. 319–325, 1997.

[77] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," *Advances in Soft Computing-Engineering Design and Manufacturing*, pp. 521–535, 1999.

[78] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The bayesian optimization algorithm," *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 525–532, 1999.

[79] ——, "Linkage problem, distribution estimation, and bayesian networks," *Evolutionary Computation*, vol. 8, no. 3, pp. 311–341, 2000.

[80] M. Pelikan and D. E. Goldberg, "Escaping hierarchical traps with competent genetic algorithms," *Proceedings of Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, pp. 511–518, 2001.

[81] M. Pelikan, "Bayesian optimization algorithm: From single level to hierarchy," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

[82] H. Mühlenbein and T. Mahnig, "FDA - a scalable evolutionary algorithm for the optimization for the optimization of additively decomposed functions," *Evolutionary Computation*, vol. 7, no. 4, pp. 353–376, 1999.

[83] ——, "Convergence theory and applications of the factorized distribution algorithm," *Journal of Computing and Information Technology*, vol. 7, pp. 19–32, 1999.

[84] R. Santana, A. Ochoa-Rodriguez, and M. R. Soto, "The mixture of trees factorized distribution algorithm," *Proceedings of Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, pp. 543–550, 2001.

[85] P. A. Bosman and D. Thierens, "Linkage information processing in distribution estimation algorithms," *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 60–67, 1999.

[86] ——, "Continuous iterated density estimation evolutionary algorithms within the IDEA framework," in *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2000 OBUPM)*, 2000, pp. 197–200.

[87] ——, "Mixed IDEAs," Utrecht University, Utrecht, The Netherlands, Tech. Report No. UU-CS-2000-45, 2002.

[88] ——, "Advancing continuous IDEAs with mixture distributions and factorization selection metrics," in *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2001 OBUPM)*, 2001, pp. 208–202.

[89] P. A. N. Bosman and D. Thierens, "Crossing the road to efficient idEas for permutation problems," *Proceedings of Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, pp. 219–226, 2001.

[90] ——, "Permutation optimization by iterated estimation of random keys and marginal product factorizations," *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, pp. 331–340, 2002.

[91] G. R. Harik, "Linkage learning via probabilistic modeling in the ECGA," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 99010, 1999.

[92] F. G. Lobo and G. R. Harik, "Extended compact genetic algorithm in C++," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 99016, 1999.

[93] K. Sastry and D. E. Goldberg, "On extended compact genetic algorithm," *Proceedings of Genetic and Evolutionary Computation Conference 2000 (GECCO-2000)*, pp. 352–359, 2000, (Late breaking paper).

[94] K. Sastry, "Efficient cluster optimization using extended compact genetic algorithm with seeded population," in *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2001 OBUPM)*, 2001, pp. 222–225.

[95] K. Sastry and D. E. Goldberg, "Probabilistic model building and competent genetic programming," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 2003013, 2003.

[96] S. Tsutsui, M. Pelikan, and D. E. Goldberg, "Evolutionary algorithm using marginal histogram models in continuous domain," in *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2001 OBUPM)*, 2001, pp. 230–233.

[97] S. Tsutsui, "Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram," *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, pp. 224–233, 2002.

[98] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms revisited: Studies in mixed size and scale," *Complex Systems*, vol. 4, no. 4, pp. 415–444, 1990.

[99] K. Deb, "Binary and floating-point function optimization using messy genetic algorithms," Ph.D. dissertation, University of Alabama, Tuscaloosa, AL, 1991.

[100] K. Deb and D. E. Goldberg, "mGA in C: A messy genetic algorithm in C," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 91008, 1991.

[101] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik, "Rapid, accurate optimization of difficult problems using fast messy genetic algorithms," *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pp. 56–64, 1993.

[102] H. Kargupta, "SEARCH, polynomial complexity, and the fast messy genetic algorithm," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[103] L. D. Merkle, "Analysis of linkage-friendly genetic algorithms," Ph.D. dissertation, Air Force Institute of Technology, Air University, Albuquerque, New Mexico, 1996.

[104] D. Knjazew, "Application of the fast messy genetic algorithm to permutation and scheduling problems," Master's thesis, Universität Dortmund, Dortmund, Germany, 2000.

[105] ——, "Ordering messy genetic algorithm in C++," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 2000034, 2000.

[106] D. Knjazew and D. E. Goldberg, "Large-scale permutation optimization with the ordering messy genetic algorithm," *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pp. 631–640, 2000.

[107] ——, "OMEGA – ordering messy GA: Solving permutation problems with the fast messy genetic algorithm and random keys," *Proceedings of Genetic and Evolutionary Computation Conference 2000 (GECCO-2000)*, pp. 181–188, 2000.

[108] D. Knjazew, *OmeGA: A Competent Genetic Algorithm for Solving Permutation and Scheduling Problems*, ser. Genetic Algorithms and Evoluationary Computation. Kluwer Academic Publishers, January 2002, vol. 6, ISBN: 0-7923-7460-6.

[109] D. Halhal, G. A. Walters, D. A. Savic, and D. Ouazar, "Putting more genetics into genetic algorithms," *Evolutionary Computation*, vol. 7, no. 3, pp. 311–329, 1999.

[110] R. A. Watson and J. B. Pollack, "Incremental commitment in genetic algorithms," *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-1999)*, pp. 710–717, 1999.

[111] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, ser. Genetic algorithms and evolutionary computation. Boston, MA: Kluwer Academic Publishers, October 2001, vol. 2, ISBN: 0-7923-7466-5.

[112] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[113] A. A. Salman, K. Mehrotra, and C. K. Mohan, "Adaptive linkage crossover," *Proceedings of ACM Symposium on Applied Computing (SAC'98)*, pp. 338–342, 1998.

[114] ——, "Linkage crossover for genetic algorithms," *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 564–571, 1999.

[115] ——, "Linkage crossover operator," *Evolutionary Computation*, vol. 8, no. 3, pp. 341–370, 2000.

[116] W. A. Greene, "A genetic algorithm with self-distancing bits but no overt linkage," *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 367–374, 2003.

[117] M. Munetomo and D. E. Goldberg, "Identifying linkage by nonlinearity check," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 98012, 1998.

[118] ——, "Designing a genetic algorithm using the linkage identification by nonlinearity check," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 98014, 1998.

[119] D. J. Coffin and C. D. Clack, "gLINC: Identifying composability using group perturbation," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006 (GECCO-2006)*, pp. 1133–1140, 2006.

[120] M. Munetomo and D. E. Goldberg, "Identifying linkage groups by nonlinearity/non-monotonicity detection," *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 433–440, 1999.

[121] ——, "Linkage identification by non-monotonicity detection for overlapping functions," *Evolutionary Computation*, vol. 7, no. 4, pp. 377–398, 1999.

[122] M. Munetomo, "Linkage identification based on epistasis measures to realize efficient genetic algorithms," *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1332–1337, 2002.

[123] M. Munetomo, M. Tsuji, and K. Akama, "Metropolitan area network design using GA based on linkage identification with epistasis measures," *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL2002)*, pp. 652–656, 2002.

[124] M. Munetomo, N. Murao, and K. Akama, "A parallel genetic algorithm based on linkage identification," *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 1222–1233, 2003.

[125] M. Munetomo, "Linkage identification with epistasis measure considering monotonicity conditions," *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL2002)*, pp. 550–554, 2002.

[126] M. Tezuka, M. Munetomo, and K. Akama, "Linkage identification by nonlinearity check for real-coded genetic algorithms," *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO-2004)*, pp. 222–233, 2004.

[127] M. Tsuji, M. Munetomo, and K. Akama, "Modeling dependencies of loci with string classification according to fitness differences," *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO-2004)*, pp. 246–257, 2004.

[128] ——, "Population sizing of dependency detection by fitness difference classification," *Foundations of Genetic Algorithms 2005 (FOGA-2005)*, pp. 282–299, 2005.

[129] ——, "Linkage identification by fitness difference clustering," *Evolutionary Computation*, vol. 14, no. 4, pp. 383–409, 2006.

[130] T. P. Riopka and P. Bock, "Intelligent recombination using individual learning in a Collective Learning Genetic Algorithm," *Proceedings of Genetic and Evolutionary Computation Conference 2000 (GECCO-2000)*, pp. 104–111, 2000.

[131] T. P. Riopka, "Intelligent recombination using genotypic learning in a Collective Learning Genetic Algorithm," Ph.D. dissertation, The George Washington University, Washington, DC, 2002.

[132] T.-L. Yu, D. E. Goldberg, A. Yassine, and Y.-p. Chen, “Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm,” *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, vol. 2, pp. 1620–1621, 2003, (Poster session).

[133] ——, “Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm,” *Proceedings of Artificial Neural Networks in Engineering 2003 (ANNIE 2003)*, pp. 327–332, 2003.

[134] T.-L. Yu and D. E. Goldberg, “Dependency structure matrix analysis: Off-line utility of the dependency structure matrix genetic algorithm,” *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO-2004)*, vol. 2, pp. 367–378, 2004.

[135] D. E. Goldberg and R. Lingle, Jr., “Alleles, loci, and the traveling salesman problem,” *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pp. 154–159, 1985.

[136] L. Davis, “Applying adaptive algorithms to epistatic domains,” *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, vol. 1, pp. 162–164, 1985.

[137] D. Whitley, T. Starkweather, and D. Fuquay, “Scheduling problems and traveling salesmen: The genetic edge recombination operator,” *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pp. 133–140, 1989.

[138] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, “A comparsion of genetic sequencing operators,” *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pp. 69–76, 1991.

[139] L. Davis, “A genetic algorithms tutorial,” in *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, January 1991, pp. 1–101, ISBN: 0-442-00173-8.

[140] H. Kargupta, K. Deb, and D. E. Goldberg, “Ordering genetic algorithms and deception,” *Proceedings of the Second International Conference on Parallel Problem Solving from Nature (PPSN II)*, pp. 47–56, 1992.

[141] J. C. Bean, “Genetic algorithms and random keys for sequencing and optimization,” *ORSA Journal on Computing*, vol. 6, no. 4, pp. 154–160, 1994.

[142] E. Falkenauer, “A genetic algorithm for grouping,” *Proceedings of the Fifth International Symposium on Applied Stochastic Models and Data Analysis (ASMDA V)*, pp. 198–206, 1991.

[143] ——, “Setting new limits in bin packing with a grouping GA using reduction,” CRIF Industrial Automation, Brussels, Belgium, Technical Report RO108, 1994.

[144] E. Falkenauer and A. Delchambre, “A genetic algorithm for bin packing and line balancing,” *Proceedings of the 1992 IEEE International Conference on Robotics and Automation (RA92)*, pp. 1186–1192, 1992.

[145] E. Falkenauer, “A new representation and operators for genetic algorithms applied to grouping problems,” *Evolutionary Computation*, vol. 2, no. 2, pp. 123–144, 1994.

[146] ——, “A hybrid grouping genetic algorithm for bin packing,” *Journal of Heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[147] ——, "Solving equal piles with the grouping genetic algorithm," *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pp. 492–497, 1995.

[148] ——, "Applying genetic algorithms to real-world problems," in *Evolutionary Algorithms*, L. D. Davis, K. De Jong, M. D. Vose, and L. D. Whitley, Eds. New York: Springer, 1999, pp. 65–88, ISBN: 0-387-98826-2.

[149] R. B. Heckendorn and A. H. Wright, "Efficient linkage discovery by limited probing," *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 1003–1014, 2003.

[150] ——, "Efficient linkage discovery by limited probing," *Evolutionary Computation*, vol. 12, no. 4, pp. 517–545, 2004.

[151] A. Prügel-Bennett, "Modelling crossover-induced linkage in genetic algorithms," *IEEE Transcations on Evolutionary Computation*, vol. 5, no. 4, pp. 376–387, 2001.

[152] S. Hill and C. O'Riordan, "Inversion revisited – analysing an inversion operator using problem generators," in *Proceedings of the Analysis and Design of Representations and Operators (ADoRo) Workshop at the Genetic and Evolutionary Computation Conference (GECCO-2003 ADoRo)*, 2003, pp. 34–40.

[153] H. Kargupta and S. Bandyopadhyay, "A perspective on the foundation and evolution of the linkage learning genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 269–294, 2000.

[154] J. Smith, "On appropriate adaptation levels for the learning of gene linkage," *Genetic Programming and Evolvable Machines*, vol. 3, no. 2, pp. 129–155, 2002.

[155] T. N. Bui and B. R. Moon, "Hyperplane synthesis for genetic algorithms," *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pp. 102–109, 1993.

[156] T. N. Bui and B.-R. Moon, "Analyzing hyperplane synthesis in genetic algorithms using clustered schemata," *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pp. 108–118, 1994.

[157] T. N. Bui and B. R. Moon, "On multi-dimensional encoding/crossover," *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pp. 49–56, 1995.

[158] B.-R. Moon and C.-K. Kim, "A two-dimensional embedding of graphs for genetic algorithms," *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA-97)*, pp. 204–211, 1997.

[159] T. N. Bui and B. R. Moon, "A new genetic approach for the traveling salesman problem," *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, vol. 1, pp. 7–12, 1994.

[160] ——, "Genetic algorithm and graph partitioning," *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 841–855, 1996.

[161] T. N. Bui and B.-R. Moon, "GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 3, pp. 193–204, 1998.

[162] B.-R. Moon and C.-K. Kim, "Dynamic embedding for genetic VLSI circuit partitioning," *Engineering Applications of Artificial Intelligence*, vol. 11, pp. 67–76, 1998.

[163] B.-R. Moon, Y.-s. Lee, and C.-K. Kim, "GEORG: VLSI circuit partitioner with a new genetic algorithm framework," *Journal of Intelligent Manufacturing*, vol. 9, pp. 401–412, 1998.

[164] A. B. Kahng and B. R. Moon, "Toward more powerful recombinations," *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pp. 96–103, 1995.

[165] J.-H. Kim and B.-R. Moon, "Neuron reordering for better neuro-genetic hybrids," *Proceedings of Genetic and Evolutionary Computation Conference 2002 (GECCO-2002)*, pp. 407–414, 2002.

[166] S. Jung and B.-R. Moon, "A hybrid genetic algorithm for the vehicle routing problem with time windows," *Proceedings of Genetic and Evolutionary Computation Conference 2002 (GECCO-2002)*, pp. 1309–1316, 2002.

[167] E.-J. Park, Y.-H. Kim, and B.-R. Moon, "Genetic search for fixed channel assignment problem with limited bandwidth," *Proceedings of Genetic and Evolutionary Computation Conference 2002 (GECCO-2002)*, pp. 1172–1179, 2002.

[168] S. Jung and B.-R. Moon, "The natural crossover for the 2D Euclidean TSP," *Proceedings of Genetic and Evolutionary Computation Conference 2000 (GECCO-2000)*, pp. 1003–1010, 2000.

[169] ——, "Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidean TSP," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 557–565, 2002.

[170] D.-I. Seo and B.-R. Moon, "Voronoi quantized crossover for traveling salesman problem," *Proceedings of Genetic and Evolutionary Computation Conference 2002 (GECCO-2002)*, pp. 544–552, 2002.

[171] ——, "A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem," *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 669–680, 2003.

[172] ——, "A survey on chromosomal structures and operators for exploiting topological linkage of genes," *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)*, pp. 1357–1368, 2003.