Tightness Time for the Linkage Learning Genetic Algorithm

Ying-ping Chen[1] and David E. Goldberg[2]

[1] Department of Computer Science and Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
ypchen@illigal.ge.uiuc.edu
[2] Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
deg@uiuc.edu

Abstract. This paper develops a model for *tightness time*, linkage learning time for a single building block, in the linkage learning genetic algorithm (LLGA). First, the existing models for both linkage learning mechanisms, *linkage skew* and *linkage shift*, are extended and investigated. Then, the tightness time model is derived and proposed based on the extended linkage learning mechanism models. Experimental results are also presented in this study to verify the extended models for linkage learning mechanisms and the proposed model for tightness time.

1 Introduction

Linkage learning, one of the fundamental challenges of the research and development of genetic algorithms (GAs), has often been either ignored or overlooked in the field of evolutionary computation. In spite of Holland's call for the evolution of tight linkage in his historical publication of *Adaptation in Natural and Artificial Systems* [1], there have been relatively few efforts made on the subject of linkage learning and evolution until the past decade. Thierens and Goldberg [2,3] showed that simple genetic algorithms fail to solve hard problems without tight linkage and analyzed several possible but unsuccessful techniques to overcome the linkage learning difficulty. Recognizing the importance of linkage evolution in powerful and general evolutionary processes, linkage-related operators and mechanisms were therefore developed.

Among the ways to achieve tight linkage or its equivalent such as perturbation techniques, model builders, and linkage learners is the linkage learning genetic algorithm (LLGA), which uses (*gene number*, *allele*) style coding scheme with non-coding segments [4,5,6] to create an evolvable genotypic structure that makes GAs capable of learning tight linkage of building blocks through the special expression mechanism, *probabilistic expression* (PE) [7,8]. Compared to the perturbation-based algorithms, the LLGA does not employ extra linkage detection procedures to gain knowledge of linkage. Compared to the model builders,

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 837–849, 2003.

the LLGA operates in a local, distributed manner without building a global model across all individuals. Therefore, the linkage learning process is a very special and essential part of the LLGA, and this paper seeks to better understand the LLGA's linkage learning mechanisms.

In particular, this paper examines the current theoretical analysis of the linkage learning mechanisms [9,7]. Models for both linkage learning mechanisms, *linkage skew* and *linkage shift*, are refined and extended. The theoretical results are confirmed with experiments. Moreover, a model for *tightness time*, linkage learning time for a single building block, is therefore proposed and empirically verified. Artificial evolutionary systems usually create fitness associated with a greedy choice of the best alleles before linkage has evolved. Understanding the race between allelic convergence and linkage convergence is critical to designing LLGAs that work well. In particular, understanding tightness time is critical to getting time scale right, especially in systems like the LLGA.

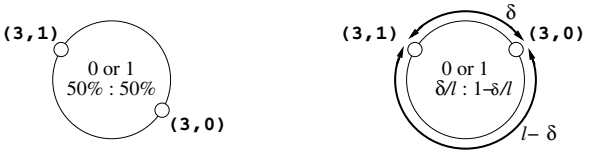
This paper is organized as follows. The following section gives a short survey of competent GAs and a brief review of the LLGA. Section 3 extends the existing theoretical analysis of both linkage learning mechanisms, and Sect. 4 verifies the models with experimental results. Section 5 proposes the tightness time model based on the extended mechanism models. Finally, the conclusions of this paper are drawn in Sect. 6.

2 Brief Review of Competent GAs and the LLGA

Current linkage detection, adaptation, or learning techniques can be roughly classified into three categories. The first category is based on perturbation techniques. Algorithms in this category, such as messy GA (mGA) [10], fast messy GA (fmGA) [11], gene expression messy GA (GEMGA) [12], and linkage identification by nonlinearity check/non-monotonicity detection (LINC/LIMD) [13, 14], perturb chromosomes in some particular way and detect linkage among genes by observing the difference of fitness. After determining the relationship among genes, building-block (BB) preserving recombination operators are then employed to create promising solutions.

The second category uses model building techniques. Population-based incremental learning (PBIL) [15], univariate marginal distribution algorithm (UMDA) [16], compact GA (cGA) [17], extended compact GA (ECGA) [18], iterated distribution estimation algorithm (IDEA) [19], probabilistic incremental program evolution (PIPE) [20], and BOA [21] belong to this category. Model builders usually grasp the linkage or relationship among genes by building a global probabilistic model with the current population. New individuals are then derived according to the model, and the model building-utilizing process is repeated until the termination criterion is satisfied.

The final category adapts linkage through the chromosome representation and genetic operators. The linkage learning genetic algorithm (LLGA) [9,7,8] falls into this category. By making the chromosome itself capable of representing



(a) Gene 3 is expressed as 0 with probability 0.5 and 1 with probability 0.5.

(b) Gene 3 is expressed as 0 with probability δ/l and 1 with probability $1 - \delta/l$.

Fig. 1. Probability distributions of gene 3's alleles represented by PE chromosomes.

linkage, the LLGA integrates the linkage learning and utilizing into a unified operation and creates offspring with conceptually well-known genetic operators.

The LLGA is briefly reviewed in the remainder of this section. Readers who are interested in more detail should refer to other materials [9,7,8].

2.1 Chromosome Representation

The LLGA's chromosome representation is mainly composed of moveable genes, non-coding segments, and probabilistic expression. Moveable genes are encoded as (*gene number*, *allele*) pairs in the LLGA chromosome, and a LLGA chromosome is considered as a circle. These genes are allowed to move around and reside anywhere in any order in the chromosome. Non-coding segments are inserted into the chromosome to create an evolvable genotype capable of expressing linkage. Non-coding segments act as non-functional genes, which have no effect on fitness, residing between adjacent genes to generate gaps for expressing linkage precisely.

Probability expression (PE) was proposed to preserve building-block level diversity. For each gene, all possible alleles coexist in a PE chromosome at the same time. For the purpose of evaluation, a chromosome is *interpreted* with a *point of interpretation* (POI). The allele for each gene is determined by the order in which the chromosome is traversed clock-wisely from the point of interpretation. A complete string is then expressed and evaluated.

Consequently, each PE chromosome represents not just a single solution but a probability distribution over the range of possible solutions. Figure 1 shows the probability distribution over alleles of gene 3 of the chromosome. Therefore, if different points of interpretation are selected, a PE chromosome might be interpreted as different solutions. Furthermore, the probability of a PE chromosome to be expressed as a particular solution depends on the length of the non-coding segment between genes critical to that solution. It is the essential technique of the LLGA to capture the knowledge and to prompt the evolution of linkage.

2.2 Exchange Crossover

The exchange crossover operator is another key to make the LLGA able to learn linkage. It is defined on a pair of chromosomes. One of the two chromosomes is

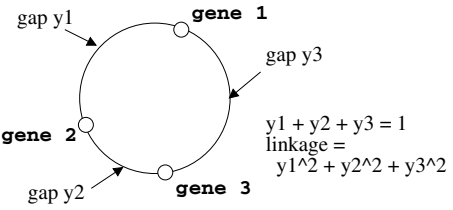


Fig. 2. Calculation for the linkage of a three-gene building block.

the *donor*, and the other is the *recipient*. The exchange crossover cuts a *random* segment of the donor, selects a grafting point on the recipient, and grafts the segment onto the recipient. The grafting point is the point of interpretation of the offspring. Starting from the point of interpretation, redundant genetic materials caused by injection are removed right after crossover to ensure the validity.

2.3 Mechanisms Making the LLGA Work

With the integration of PE and the exchange crossover operator, the LLGA is capable of solving difficult problems without prior knowledge of good linkage. Traditional GAs have been shown to perform poorly on difficult problems [2] without such knowledge. To understand the working of the LLGA, two mechanisms of linkage learning: *linkage skew* and *linkage shift* has been identified and analyzed [9]. Linkage skew occurs when an optimal building block is transferred from the donor to the recipient. Linkage shift occurs when an optimal building block resides in the recipient and survives an injection. Both linkage skew and linkage shift make the building block's linkage tighter. Thus, the linkage of building blocks can evolve during the linkage learning process, and tightly linked building blocks are formed.

3 Linkage Learning Mechanisms

In this section, we extend the existing theoretical models for both linkage learning mechanisms. First, the definition of linkage of a building block is introduced. Then, the existing models for linkage skew and linkage shift [9] are extended.

3.1 Quantifying Linkage

In order to show the linkage learning and evolutionary process of the LLGA, linkage of building blocks has to be quantified. In this paper, we employ a proposed definition [9], which is the sum of the square of its inter-gene distances, considering the chromosome to be a circle of circumference 1. Figure 2 shows an example for calculating the linkage of a three-gene building block. The definition is appropriate in that linkage in such definition specifies a measure directly

proportional to the probability for a building block to be preserved under the exchange crossover operator. Additionally, it was also theoretically justified that for any linkage learning operator working on the same form of representation, the expected linkage of a randomly spaced order-k building block is $\frac{2}{k+1}$ [9].

3.2 Linkage Skew

Linkage skew, the first linkage learning mechanism, occurs when an optimal building block is successfully transferred from the donor onto the recipient. The conditions for an optimal building block to be transferred are (1) the optimal building block resides in the cut segment, and (2) the optimal building block gets expressed before the deceptive one does. The effect of linkage skew was found to make linkage distributions move toward higher linkages by eliminating less fit individuals. Linkage skew does not make the linkage of a building block of any particular individual tighter. Instead, it drives the whole linkage distribution to a higher state.

Let $\Lambda_t(\lambda)$ be the probability density function of the random variable λ representing the linkage of the optimal building block at generation t. The following model to describe the evolution of linkage under linkage skew only has been proposed [9]:

$$\Lambda_{t+1}(\lambda) = \frac{\lambda \Lambda_t(\lambda)}{\overline{\Lambda_t}}. \tag{1}$$

Based on (1), the linkage average at generation $t+1$ was calculated as [7]:

$$\overline{\Lambda_{t+1}} = \int_0^1 \lambda \Lambda_{t+1}(\lambda)\, d\lambda = \frac{\overline{\Lambda_t^2}}{\overline{\Lambda_t}}, \tag{2}$$

and thus

$$\overline{\Lambda_{t+1}} = \overline{\Lambda_t} + \frac{\sigma^2(\Lambda_t)}{\overline{\Lambda_t}}, \tag{3}$$

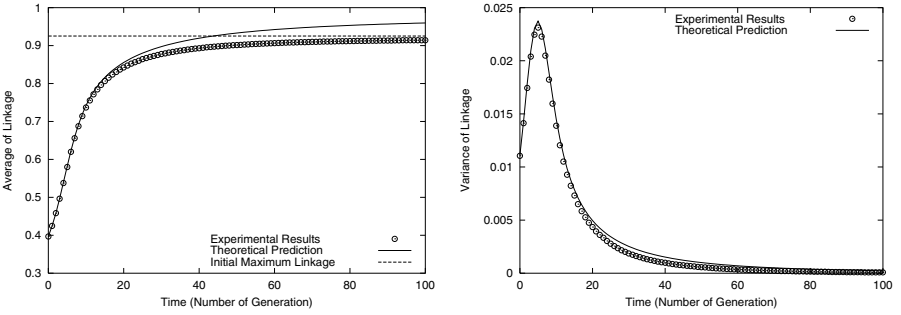
where $\sigma^2(\Lambda_t)$ is the variance in the linkage distribution at generation t.

In addition to the average (i.e. the first moment) of $\Lambda_{t+1}(\lambda)$, we can actually calculate all other moments of $\Lambda_{t+1}(\lambda)$ in the same way:

$$\overline{\Lambda_{t+1}^n} = \int_0^1 \lambda^n \Lambda_{t+1}(\lambda)\, d\lambda = \frac{\overline{\Lambda_t^{n+1}}}{\overline{\Lambda_t}}. \tag{4}$$

According to the property of a probability distribution, the moments about the origin completely characterize a probability distribution [22]. From (4), we can construct all the moments about the origin of $\Lambda_{t+1}(\lambda)$ as follows:

$$\overline{\Lambda_{t+1}^n} = \frac{\overline{\Lambda_t^{n+1}}}{\overline{\Lambda_t}} \qquad n = 1, 2, 3, \cdots \tag{5}$$



(a) The Average of Linkage. (b) The Variance of Linkage.

Fig. 3. Linkage Skew on an Order-4 Trap Building Block.

Hence, the relation between $\Lambda_t(\lambda)$ and $\Lambda_{t+1}(\lambda)$ is established.

After knowing the relation between $\Lambda_t(\lambda)$ and $\Lambda_{t+1}(\lambda)$, the moment generation function (mgf) defined as follows can help us a lot:

$$m_{\Lambda_t}(s) = E\left[e^{s\Lambda_t}\right] = \int_{-\infty}^{\infty} e^{s\lambda}\Lambda_t(\lambda)\, d\lambda. \tag{6}$$

Assume that the mgf of $\Lambda_t(\lambda)$ exists, it can be written as

$$\begin{aligned} m_{\Lambda_t}(s) &= E\left[e^{s\Lambda_t}\right] \\ &= E\left[1 + \Lambda_t s + \frac{(\Lambda_t s)^2}{2!} + \frac{(\Lambda_t s)^3}{3!} + \cdots\right] \\ &= 1 + \overline{\Lambda_t} s + \overline{\Lambda_t^2}\frac{s^2}{2!} + \overline{\Lambda_t^3}\frac{s^3}{3!} + \cdots \end{aligned}$$

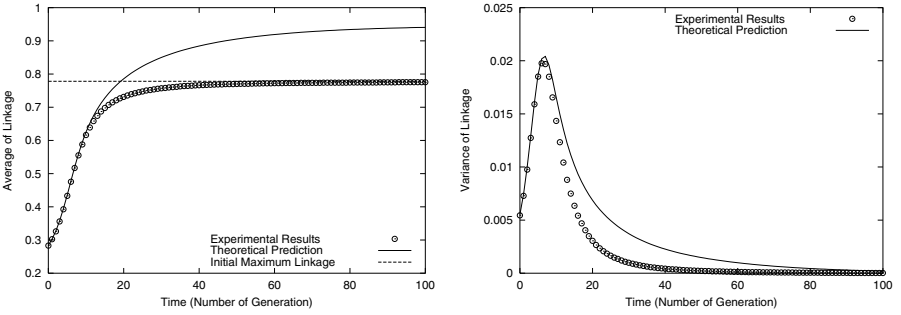
The r^{th} moment of $\Lambda_t(\lambda)$ can be obtained with

$$\overline{\Lambda_t^r} = m_{\Lambda_t}^{(r)}(0) = \left.\frac{d^r m_{\Lambda_t}(s)}{ds^r}\right|_{s=0}.$$

Given the relation between $\Lambda_t(\lambda)$ and $\Lambda_{t+1}(\lambda)$ and the property of the moment generating function, we can now get the mgf of $\Lambda_{t+1}(\lambda)$:

$$m_{\Lambda_{t+1}}(s) = m'_{\Lambda_t}\left(\frac{s}{m'_{\Lambda_t}(0)}\right). \tag{7}$$

Therefore, we have a model to calculate $\Lambda_{t+1}(\lambda)$ from $\Lambda_t(\lambda)$ when the mgf is available.



(a) The Average of Linkage. (b) The Variance of Linkage.

Fig. 4. Linkage Skew on an Order-6 Trap Building Block.

Moreover, also based on (5), we can obtain the following result:

$$\begin{aligned}\overline{\Lambda_t^n} &= \frac{\overline{\Lambda_{t-1}^{n+1}}}{\overline{\Lambda_{t-1}}} = \frac{\overline{\Lambda_{t-2}^{n+2}}/\overline{\Lambda_{t-2}}}{\overline{\Lambda_{t-2}^{2}}/\overline{\Lambda_{t-2}}} \\ &= \frac{\overline{\Lambda_{t-2}^{n+2}}}{\overline{\Lambda_{t-2}}} = \cdots \\ &= \frac{\overline{\Lambda_0^{t+n}}}{\overline{\Lambda_0}},\end{aligned} \tag{8}$$

which clearly indicates that under linkage skew, any moment of the linkage distribution at any given generation can be predicted with the information of the initial linkage distribution. The linkage learning process is solely determined by the initial distribution if there is only linkage skew working.

Finally, based on its property, linkage skew does not really tighten building blocks in any individual. It drives the linkage distribution to a higher place by propagating tight building blocks among individuals. Apparently, the linkage cannot exceed the maximum linkage in the initial population. The evolution of linkage described by (7) is therefore bounded by the initial maximum linkage.

3.3 Linkage Shift

Linkage shift is the second linkage learning mechanism [9]. It occurs when an optimal building block resides in the recipient and survives a crossover event. For the optimal building block to survive, there cannot be any gene contributing to a deceptive building block transferred. Linkage shift gets the linkage of a building block in an individual higher with deletion of duplicate genetic material caused by injection of the exchange crossover. Compared to linkage skew, linkage shift gets linkage of building blocks in each individual higher.

For linkage shift, the following recurrence equation was used [9] to depict the effect of the second mechanism on building blocks that survive crossover:

$$\overline{\lambda_0(t+1)} = \lambda_0(t) + (1-\lambda_0(t))\frac{2}{(k+2)(k+3)}, \tag{9}$$

for an order-k building block. Tracking only the average of linkage, we can approximately rewrite (9) as

$$\overline{\Lambda_{t+1}} = \overline{\Lambda_t} + (1-\overline{\Lambda_t})\frac{2}{(k+2)(k+3)}. \tag{10}$$

Given a fixed k, let $c = \frac{2}{(k+2)(k+3)}$, we can get the following recurrence relation:

$$\begin{aligned} \overline{\Lambda_{t+1}} &= \overline{\Lambda_t} + c(1-\overline{\Lambda_t}) \\ &= \overline{\Lambda_t}\,(1-c) + c. \end{aligned} \tag{11}$$

By solving the recurrence relation, the new linkage shift model is obtained as

$$\overline{\Lambda_t} = 1 - (1-\overline{\Lambda_0})(1-c)^t. \tag{12}$$

Therefore, the rate of linkage learning is mainly determined by the linkage average of the initial linkage distribution. Besides, the higher order the building block is, the longer it takes to evolve to some specific level of linkage.

4 Experimental Results

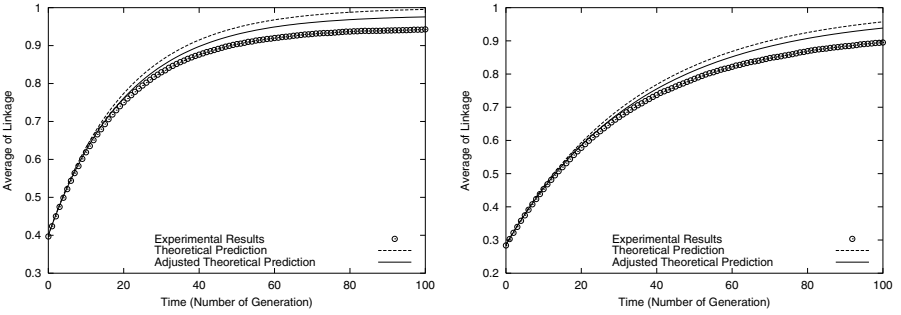
The experimental results to verify the extended linkage learning mechanism models are presented in this section. First, the parameter settings of the experiments are described. Then, experimental results for both models are shown.

4.1 Parameter Settings

In this paper, we use trap functions [23] to verify the theoretical model. We basically use similar experiments presented elsewhere [9], but the experiments in this study were done for both order-4 and order-6 traps. An order-k trap function used in this study can be described by

$$\text{trap}_k(u) = \begin{cases} u & u = k \\ k-1-u & \text{otherwise} \end{cases},$$

where u is the number of ones in the bitstring. In order to simulate the infinite-length chromosome, we let the functional genes occupy only one percent of the chromosome. That is, for the order-4 building block, the 4 genes are embedded in a 400-gene chromosome with 396 nonfunctional genes; for the order-6 building block, the 6 genes are embedded in a 600-gene chromosome with 594 nonfunctional genes. The population size are 5000 in both cases. All results in this section are averaged over 50 independent runs.



(a) Order-4 Trap Building Block. (b) Order-6 Trap Building Block.

Fig. 5. The Average of Linkage under Linkage Shift.

4.2 Linkage Skew

The extended linkage skew model can predict all moments at any given generation. For illustration purpose, we show only the prediction of the average (the first moment) and the variance (the second moment minus the square of the average) in figures.

Figures 3 and 4 show the experimental results compared to the theoretical prediction. The theoretical prediction was made based on (8). As shown in the figures, the experimental results agree with the prediction, and the extended linkage skew model is confirmed experimentally.

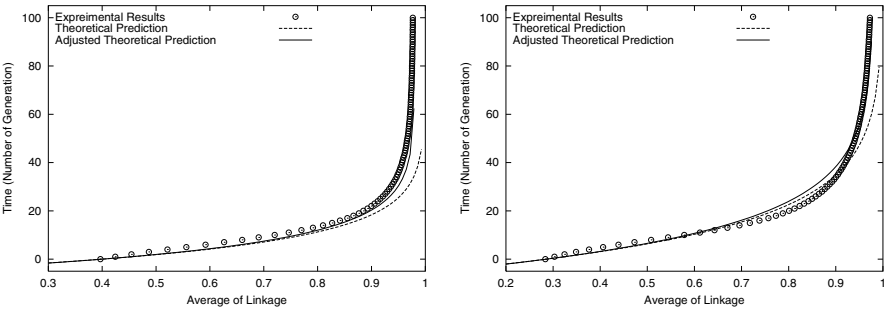
4.3 Linkage Shift

For linkage shift, we predict the average of linkage on both order-4 and order-6 traps. Figure 5 shows the experimental results. The theoretical prediction was made according to (12). We also employ the adjustment scheme [9] to reflect the difference between the infinite-length chromosome model and the real chromosomes in experiments. In this study, the maximum possible linkage in both cases is $(0.99)^2 = 0.9801$, and (12) is adjusted accordingly. As indicated in Fig. 5, the experimental results agree with the adjusted theoretical prediction, and we are now given good reason to believe that the extended model is accurate.

5 Tightness Time

5.1 The Model

Equipped with the extended models for linkage learning, we are now ready to develop the tightness time model. Based on the observation and intuition, the working relationship between linkage skew and linkage shift is as follows. Linkage shift is responsible for making the linkage of a building block in each individual



(a) Order-4 Trap Building Block. (b) Order-6 Trap Building Block.

Fig. 6. Tightness Time for a Single Building Block.

tighter; linkage skew is responsible for driving the whole distribution toward a higher place. Considering linkage shift as a refiner, linkage skew as a propagator, and that the effect of linkage skew comes pretty fast based on the experimental result, the linkage learning bottleneck is in fact linkage shift. Hence, we now start to develop the model for tightness time based on the most critical component in the framework first.

Start from (12), we can obtain

$$t = \frac{\log(1 - \overline{\Lambda_t}) - \log(1 - \overline{\Lambda_0})}{\log(1 - c)}.$$

Then, it can be rewritten as a function of linkage λ:

$$t(\lambda) = \frac{\log(1 - \lambda) - \log(1 - \overline{\Lambda_0})}{\log(1 - c)}.$$

By taking the propagation effect of linkage skew into account, a constant c_s standing for the linkage learning speed-up caused by linkage skew is added into the model. Thus, we obtain the tightness time model as follows:

$$t_\ell(\lambda) = \frac{\log(1 - \lambda) - \log(1 - \overline{\Lambda_0})}{c_s \log(1 - c)}, \tag{13}$$

where $t_\ell(\lambda)$ is the tightness time for a given linkage λ, and $c_s \approx 2$ is determined empirically.

Furthermore, given the initial linkage distribution, $\overline{\Lambda_0}$ remains constant during the whole process. For simplicity, we can define

$$\epsilon = 1 - \lambda,$$
$$\overline{\epsilon_0} = 1 - \overline{\Lambda_0}.$$

Also, $c = \frac{2}{(k+2)(k+3)} \approx \frac{2}{k^2}$ when $k \to \infty$. Therefore, (13) can be rewritten as a function of ϵ as

$$t'_\ell(\epsilon) = \frac{k^2}{2c_s} \log \frac{\epsilon}{\epsilon_0}. \tag{14}$$

Equation (14) shows that tightness time is proportional to the square of the order of building blocks. The longer the building block, the much longer the tightness time. Besides, tightness time is proportional to the logarithm of the desired linkage.

5.2 Verification

Experiments were also performed to verify the model for tightness time. Using the same parameter settings described in section 4.1, both linkage learning mechanisms work together in the experiments.

The experimental results are shown in Figure 6. The theoretical prediction made based on (13) is also adjusted with the maximum possible linkage 0.9801. The obtained numerical data agree with our tightness time model quite well. Our hypothesis and model are therefore experimentally verified.

6 Conclusions

One of the most important issues of the design of genetic algorithms is linkage learning. Harik took Holland's call for the evolution of tight linkage seriously and developed the linkage learning genetic algorithm, which learns linkage among genes with specially designed chromosome representation and conceptually well-known genetic operators. While the LLGA performs remarkably well on badly scaled building blocks, it does not do so on uniformly scaled building blocks. This paper seeks to gain better understanding of linkage learning mechanisms of the LLGA in order to improve the capability of the LLGA.

In this paper, the current theoretical analysis of the linkage learning mechanisms are extended in several aspects and verified with experiments. Based on the extended models, a model for tightness time is proposed. Under the two linkage learning mechanisms, the evolution of linkage is basically determined by the initial linkage distribution. If the linkage learning process needs to be modified or improved, we have to seek help from other mechanisms outside the framework. Additionally, the cooperation and interaction between linkage skew and linkage shift is also theorized. It helps us to better understand the overall effect of the LLGA's linkage learning mechanisms. Finally, some important insights into the whole linkage learning process are obtained from the tightness time model.

Among existing techniques, perturbation-based algorithms and model builders get linkage information and evolve the population in two separate steps, while the LLGA gets linkage tight and alleles right at the same time. Hence, tightness time for the LLGA provides an important key to correctly handling the competition of linkage and allele on time scale. Understanding tightness time

should enable us to improve the LLGA and design better genetic algorithms as well.

More work along this line still needs to be done to understand the capabilities and limits of linkage adaptation techniques. The results shown in this paper give us more theoretical insight of the LLGA's linkage learning mechanisms and take us one step further toward scalable linkage learning.

Acknowledgments. The author would like to thank Kumara Sastry for many useful discussions and valuable comments.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US 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. Holland, J.H.: *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, MI (1975)
2. Thierens, D., Goldberg, D.E.: Mixing in genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993) 38–45
3. Thierens, D.: *Analysis and Design of Genetic Algorithms.* doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium (1995)
4. Lindsay, R.K., Wu, A.S.: Testing the robustness of the genetic algorithm on the floating building block representation. *Proceedings of the twelfth National Conference on Artificial Intelligence/eighth Innovative Applications of Artificial Intelligence Conference* (1996) 793–798
5. Wineberg, M., Oppacher, F.: The benefits of computing with introns. *Proceedings of the First Annual Conference on Genetic Programming* (1996) 410–415
6. Levenick, J.R.: Swappers: Introns promote flexibility, diversity and invention. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 361–368
7. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. Unpublished doctoral dissertation, University of Michigan, Ann Arbor (1997) (Also IlliGAL Report No. 97005).
8. Harik, G.R., Goldberg, D.E.: Learning linkage through probabilistic expression. *Computer Methods in Applied Mechanics and Engineering* **186** (2000) 295–310
9. Harik, G.R., Goldberg, D.E.: Learning linkage. *Foundations of Genetic Algorithms 4* (1996) 247–262
10. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems* **3** (1989) 493–530

11. Goldberg, D.E., Deb, K., Kargupta, H., Harik, G.: Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993) 56–64
12. Kargupta, H.: The gene expression messy genetic algorithm. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation* (1996) 814–819
13. Munetomo, M., Goldberg, D.E.: Identifying linkage groups by nonlinearity/non-monotonicity detection. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 433–440
14. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation* **7** (1999) 377–398
15. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA (1994)
16. Mühlenbein, H.: The equation for response to selection and its use for prediction. *Evolutionary Computation* **5** (1997) 303–346
17. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. *Proceedings of 1998 IEEE International Conference on Evolutionary Computation* (1998) 523–528
18. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
19. Bosman, P.A.N., Thierens, D.: Linkage information processing in distribution estimation algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 60–67
20. Salustowicz, R.P., Schmidhuber, J.: Probabilistic incremental program evolution. *Evolutionary Computation* **5** (1997) 123–141
21. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 525–532
22. Zwillinger, D., Kokoska, S.: *CRC standard probability and statistics tables and formulae*. CRC Press, Boca Raton, Florida (2000)
23. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2* (1993) 93–108