# Introducing Subchromosome Representations to 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 introduces subchromosome representations to the linkage learning genetic algorithm (LLGA). The subchromosome representation is utilized for effectively lowering the number of building blocks in order to escape from the performance limit implied by the convergence time model for the linkage learning genetic algorithm. A preliminary implementation to realize subchromosome representations is developed and tested. The experimental results indicate that the proposed representation can improve the performance of the linkage learning genetic algorithm on uniformly scaled problems, and the initial implementation provides a potential way for the linkage learning genetic algorithm to incorporate prior linkage information when such knowledge exists.

## 1 Introduction

Linkage learning, which makes genetic algorithms (GAs) capable of detecting associations among genes and properly arranging these closely related genes to form building blocks, is one of the key challenges of the genetic algorithm design. In order to ensure a genetic algorithm works well, the building blocks represented on the chromosome have to be tightly linked. Otherwise, studies [1, 2] have shown that a genetic algorithm may fail to solve problems without such prior knowledge. One way to alleviate the burden of choosing an appropriate chromosome representation for genetic algorithm users is to employ the genetic linkage learning technique. Among the existing linkage learning methods, such as perturbation-based techniques [3,4], model builders [5,6,7], and linkage learners [8,9], is the linkage learning genetic algorithm (LLGA), which uses an evolvable genotypic structure capable of learning genetic linkage during the evolutionary process through its special expression mechanism.

While LLGA achieved successful linkage learning on problems with badly scaled building blocks, it was less successful on problems consisting of uniformly scaled building blocks. The convergence time model for LLGA [10] explains the difficulty faced by LLGA and indicates the performance limit of LLGA on uniformly scaled problems. This paper seeks to enhance the design of LLGA based on the time models in order to improve the performance of LLGA on uniformly scaled problems.

In particular, this paper introduces subchromosome representations to LLGA. The subchromosome representation is developed to avoid the performance limit implied by the convergence time model for LLGA. This paper presents a preliminary implementation of the proposed representation and verifies the performance improvement with empirical results. The objective of this study is to initiate a better design of LLGA that can lead to scalable genetic linkage learning.

This paper is organized as follows. The next section gives a brief review of the linkage learning genetic algorithm. Section 3 describes the subchromosome representation proposed in this paper in detail. The experiments for observing the effect of using subchromosomes and the experimental results are presented in Sect. 4. Finally, we outlined future research directions followed by conclusions.

## 2 Review of the Linkage Learning Genetic Algorithm

This section reviews key elements of the linkage learning genetic algorithm (LLGA) [11]. LLGA is capable of learning genetic linkage in the evolutionary process without the help of extra measurements and techniques. A modified version of LLGA working with *promoters* [12] is used in this study and described in this section. Readers may consult other materials [11,12] for detailed information.

### 2.1 Chromosome Representation

The LLGA's chromosome representation is mainly composed of moveable genes, non-coding segments, probabilistic expression, and promoters. Moveable genes are encoded as (*gene number*, *allele*) pairs on the LLGA chromosome, and an LLGA chromosome is considered as a circle. These genes are allowed to move around and reside anywhere in any order on the chromosome. Non-coding segments are inserted into the chromosome to create an evolvable genotype capable of learning linkage. Non-coding segments act as non-functional genes residing between functional genes to form gaps for precisely expressing genetic linkage.

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 according to 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. 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 LLGA to capture the knowledge about linkage and to prompt the evolution of linkage.

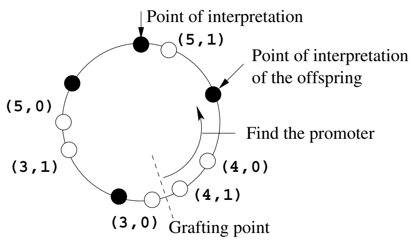

**Fig. 1.** After selecting the grafting point on the recipient, the nearest promoter *before* the grafting point is then the point of interpretation of the offspring.

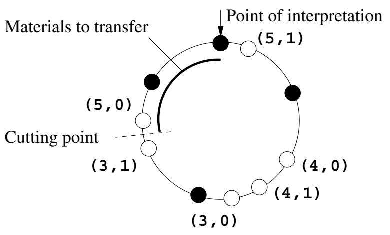

**Fig. 2.** After selecting the cutting point on the donor, the genetic material *after* the cutting point and before the current point of interpretation is transferred.

The use of *promoters*, which were called *start expression genes*, was proposed [12] in LLGA to handle separation inadequacy and to improve nucleation potential. Promoters are special non-functional elements on the chromosome. While in LLGA without promoters, all genes and non-coding segments can be the points of interpretation of the child created by crossover, only promoters can be the points of interpretation in LLGA with promoters.

## 2.2 Exchange Crossover

The exchange crossover operator is another key mechanism to make LLGA capable of learning genetic linkage. It is defined on a pair of chromosomes. One of the two chromosomes is the *donor*, and the other is the *recipient*. 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 of the offspring.

In LLGA with promoters, although the grafting point can still be any genes or non-coding segments, the point of interpretation of the offspring is no longer the grafting point. Instead, the new point of interpretation is the nearest promoter *before* the grafting point on the chromosome. After the grafting point is randomly chosen, the first promoter in front of the grafting point is the point of interpretation of the offspring. The genetic material is then transferred in the following order: (1) the segment between the promoter and the grafting point, (2) the segment chosen from the donor, and (3) the rest of the recipient. Figure 1 shows how promoters work, the black filled circles are promoters of the chromosome. Exchange crossover in LLGA with promoters selects only one cutting point at random. The other cutting point is always the element (either functional or non-functional) just before the point of interpretation of the donor. Figure 2 shows the genetic materials to be transferred during a crossover event.

## 2.3 Linkage Learning Mechanisms

With the integration of PE and exchange crossover, LLGA is capable of solving difficult problems without prior knowledge of good linkage. Traditional GAs have been shown to perform poorly on difficult problems [1,2] without such knowledge. To better decompose and understand the behavior of LLGA, two key mechanisms of linkage learning, *linkage skew* and *linkage shift*, have been identified and analyzed [11]. Both mechanisms make the building block's linkage tighter. With these two mechanisms, the linkage of building blocks can evolve, and tightly linked building blocks are formed during the process.

**Quantifying Linkage.** For studying the linkage learning process, a proposed definition for quantifying linkage [11] is adopted. The linkage is the sum of the square of the inter-gene distances of a building block, considering the chromosome to be a circle of circumference 1. The definition is appropriate in that the linkage specifies a measure directly proportional to the probability for a building block to be preserved under exchange crossover.

**Linkage skew.** Linkage skew, the first linkage learning mechanism [11], 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 an inferior 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.

**Linkage shift.** Linkage shift is the second linkage learning mechanism [11]. 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 exchange crossover. Compared to linkage skew, linkage shift gets linkage of building blocks in each individual higher.

## 2.4 Time Models

LLGA has been studied on problems containing multiple building blocks in two forms—the uniformly scaled problem and the exponentially scaled problem—not only because of their prevalence in the literature but also because they are abstract versions of many decomposable problems [13]. Uniformly scaled problems resemble those with subproblems of equal importance, and exponentially scaled problems represent those with subproblems of distinguishable importance. As reported previously [11], when the building blocks of a problem are exponentially scaled, LLGA can solve the problem in a linear time function of the number of building blocks. However, when the building blocks are uniformly scaled, LLGA either needs a population size that grows exponentially with the problem size or takes exponential time to converge. In order to explain LLGA's seemingly inconsistent behavior, the following time models were previously proposed to understand how LLGA works.

**Tightness Time.** Tightness time was proposed [14] to model the time for learning genetic linkage of a single building block of order-$k$. By extending linkage skew and linkage shift, tightness time was expressed as

$$t'_\ell(\epsilon) = \frac{k^2}{2c_s} \log \frac{\epsilon}{\epsilon_0} \; , \tag{1}$$

where $\epsilon = 1 - \lambda$, $\lambda$ is the given linkage, $k$ is the order of the building block, $c$ and $c_s$ are constants. The model shows that tightness time is proportional to the square of the order, $k$, and to the logarithm of the desired linkage.

**Convergence Time.** Based on the tightness time model for a single building block, a convergence time model for LLGA was developed [10] to model the LLGA convergence on multiple uniformly scaled building blocks. First, the sequential behavior of LLGA, which indicates that LLGA works on both uniformly scaled and exponentially scaled building blocks one by one, was identified. Then, the *first-building-block model*, which assumes that the convergence time is an accumulation of the time to tighten the first building block, was proposed to model the sequential behavior. Considering the effect and interaction of coexisting uniformly scaled building blocks by using the probability of linkage learning events, the connection between tightness time and the sequential behavior were established. By integrating these models, the LLGA convergence time model for certain desired linkage can be presented as

$$t_{\mathrm{C}}(m, \epsilon) = \left( \frac{k^2(k+1)}{2c_s\sqrt{2\pi}} \log \frac{\epsilon}{\epsilon_0} \right) \sum_{i=1}^{m} \frac{2^i}{i\sqrt{i}} + t_{\mathrm{C}0} \; , \tag{2}$$

where $c_s$ and $t_{\mathrm{C}0}$ are constants, $m$ is the number of uniformly scaled building blocks, $k$ is the order of a building block, $\epsilon = 1 - \lambda$, and $\lambda$ is the desired linkage.

## 2.5 Limit to LLGA's Competence

Although the proposed LLGA convergence time model [10] describes the way LLGA works on uniformly scaled problems and explains LLGA's inconsistent behavior on problems composed of building blocks of different scalings, it also reveals a critical limit to the competence in LLGA that the time for LLGA to solve uniformly scaled problems grows exponentially with the number of building blocks. Because the parameters involved in the convergence time model are the properties of the problem to solve, little guidance can be obtained from the model for setting the existing algorithmic parameters of LLGA. Therefore, instead

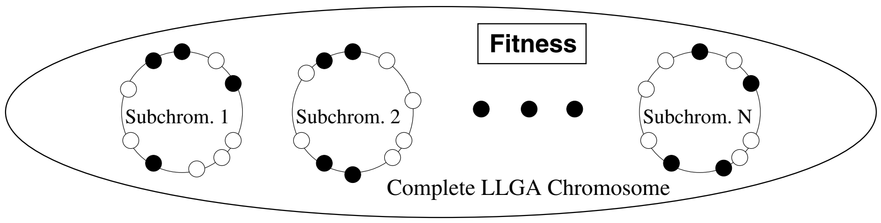

**Fig. 3.** The structure of a subchromosome is identical to that of an LLGA chromosome. Each subchromosome contains moveable genes, non-coding segments, as well as promoters and is interpreted with probabilistic expression. The union of all subchromosomes belonging to one individual forms a complete LLGA chromosome. The fitness corresponding to the solution obtained from interpreting the complete chromosome is considered the fitness of each subchromosome.

of adjusting those algorithmic parameters, another way to improve LLGA's performance on uniformly scaled problems has to be taken. This paper seeks a new design to enhance LLGA based on the insight provided by the convergence time model and takes an initial step to realize the design.

## 3 Subchromosome Representations

According to the LLGA convergence time model described by Equation (2), one possible way to enhance LLGA's performance on uniformly scaled problems is to modify the chromosome representation used by LLGA such that the number of building blocks, $m$, is effectively lowered at run time. Thus, the exponential growth of convergence time can be reduced. This section introduces the subchromosome representation to the linkage learning genetic algorithm. The subchromosome representation is first described in detail, and then, the exchange crossover operator for handling subchromosome representations is discussed.

### 3.1 Chromosome Representation

The subchromosome representation in LLGA separates a LLGA chromosome into several parts, called *subchromosomes*. The structure of a subchromosome is identical to that of an LLGA chromosome. Like an LLGA chromosome described in Sect. 2, a subchromosome contains moveable genes, non-coding segments, as well as promoters and is interpreted with probabilistic expression. The union of all subchromosomes belonging to one individual forms a complete LLGA chromosome. In subchromosome representations, there is no separate fitness measurement for each subchromosome. The fitness that corresponds to the solution obtained from interpreting the complete chromosome is used by all subchromosomes. Figure 3 shows an LLGA chromosome consisting of subchromosomes.

The goal of subchromosome representations is to create a flexible encoding mechanism that makes LLGA chromosomes capable of grouping closely related

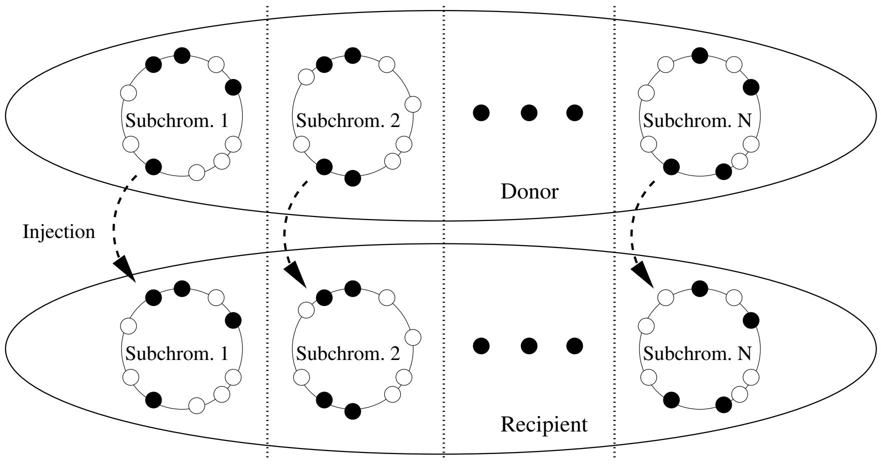

**Fig. 4.** For a pair of subchromosomes, the exchange crossover operator works as it does on conventional LLGA chromosomes. The operator cuts the genetic materials from the subchromosome of the donor and injects them into the corresponding subchromosome of the recipient. The transferred genetic materials are determined at random for each pair of subchromosomes.

building blocks to form higher-level building blocks in addition to moving genes together on the chromosome to form the first-level building blocks. Similar to linkage learning, the process of forming higher-level building blocks should be integrated with the evolutionary and problem-solving process. The subchromosomes of an LLGA chromosome shown in Fig. 3 are building blocks of the second level. The subchromosome representation can be designed to hierarchically express building blocks of even higher levels, such as the third level, and so on.

However, as an initial step to realize this representation scheme and as a pilot study of the effect of using subchromosomes in LLGA, only subchromosomes of the second level are implemented and examined in this paper. Moreover, the groups of building blocks are pre-defined, and genes on each subchromosome do not migrate to other subchromosomes. Within a subchromosome, genes and non-coding segments are still randomly distributed in initialization as they are on an LLGA chromosome without subchromosomes.

## 3.2 Exchange Crossover

Due to the adoption of the new representation, the exchange crossover operator is modified to handle subchromosomes. Since in this paper, the subchromosomes are pre-defined and do not exchange genetic materials with one another as discussed in the previous section, for simplicity, after determining the donor and the recipient, the exchange crossover operator works on subchromosomes one by one. For a pair of subchromosomes, one from the donor and the other from the recipient, exchange crossover works as it does on conventional LLGA chromosomes as described in Sect. 2. It cuts the genetic materials from the subchromosome of the donor and injects them into the corresponding subchromosome of the recipient. The transferred genetic materials are determined at random for each pair of subchromosomes. Figure 4 shows how the modified exchange crossover operator works on a pair of LLGA chromosomes consisting of subchromosomes.

## 4 Experiments

The experiments to observe the effect of using the subchromosome representation in LLGA are presented in this section. First, the parameter settings of the experiments are described in detail. Then, the experimental results are shown in the remainder of this section.

### 4.1 Parameter Settings

In this paper, trap functions [15] are used for examining the effect of adopting subchromosome representations in LLGA because trap functions provide decent linkage structures among variables, and good linkage is necessary for solving problems consisting of traps. The experiments in this study were done for order-4 traps. An order-$k$ trap function can be described by

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

where $u$ is the number of ones in the bitstring. In order to simulate the infinite-length chromosome, we let one order-4 building block embedded in 250 genes, including functional and non-functional genes. For example, for five order-4 building blocks, the 20 genes are embedded in a 1250-gene chromosome with 1230 non-functional elements. Table 1 lists all experiments conducted in this paper. The total number of building blocks in one experiment is $n_{\mathrm{bb}}$ (the number of building blocks per subchromosome) $\times$ $n_{\mathrm{s}}$ (the number of of subchromosomes). From 2 to 8 building blocks per subchromosome, all conditions for the total number of building blocks less than or equal to 60 are included in the experiments.

The gambler's ruin model [16] is utilized in the present work for population sizing, which can be approximated with the following formula:

$$\text{population size } n = -2^{k-1} \ln(\alpha) \frac{\sigma_{bb}\sqrt{\pi(m-1)}}{d} , \tag{3}$$

where $k$ is the order of building blocks, $\alpha$ is the failure probability, $\sigma_{bb}$ is the standard deviation of the fitness of a building block, $m$ is the total number of building blocks, and $d$ is the signal, which is adjusted for tournament size $s = 3$ with the equation [16]

$$d' = d + \Phi^{-1}\left(\frac{1}{s}\right)\sigma_{bb} ,$$

where $\Phi^{-1}(1/s)$ is the ordinate of a unit normal distribution.

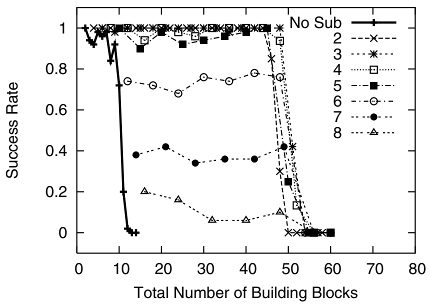

**Fig. 5.** Results of the experiments with less than or equal to 60 building blocks. The number of building blocks distributed on each subchromosome varies from 2 to 8. "No Sub" indicates LLGA without the subchromosome representation.

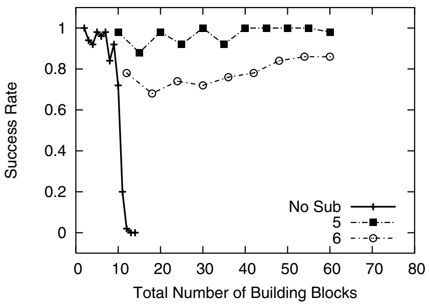

**Fig. 6.** Instead of every pair, only one pair of subchromosomes is randomly chosen for applying exchange crossover to reduce building block disruption. The results indicate that the representation works in the range of these experiments.

Other parameters are set as follows. The crossover rate is 1.0 such that the crossover event always happens. The maximum number of generation is 100,000. The number of promoters on each subchromosome is set to $2m$, where $m$ is the number of building blocks on the subchromosome. Finally, each experiment was repeated with 50 independent runs.

## 4.2 Experimental Results

For each experiment listed in Table 1, the success rate is calculated according to the results obtained in the 50 independent runs. In this paper, a success is determined by the solution quality. The solution quality is the ratio between the number of correctly solved building blocks in the end of the run and that of the total building blocks in the trial. For example, if in a particular run for solving 20 building blocks, 12 building blocks are correctly solved, the solution quality

**Table 1.** All experiments conducted in this paper. From 2 to 8 building blocks per subchromosome, all conditions for the total number of building blocks less than or equal to 60 are included in the experiments of this study.

| BBs per Subchromosome ($n_{\text{bb}}$) | N umber of Subchromosomes ($n_{\text{S}}$) |
|---|---|
| 2 | 2, 3, 4, 5, 6, . . . , 26, 27, 28, 29, 30 |
| 3 | 2, 3, 4, 5, 6, . . . , 16, 17, 18, 19, 20 |
| 4 | 2, 3, 4, 5, 6, . . . , 11, 12, 13, 14, 15 |
| 5 | 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 |
| 6 | 2, 3, 4, 5, 6, 7, 8, 9, 10 |
| 7 | 2, 3, 4, 5, 6, 7, 8 |
| 8 | 2, 3, 4, 5, 6, 7 |

of this run is 0.6. If the final solution quality of a run is equal to or greater than 0.9, the run is recorded as a success. The success rate is therefore the ratio between the number of success trials and that of the total runs.

Figure 5 gives the success rates of all the experiments with the total number of building blocks less than or equal to 60 as listed in Table 1. The number of building blocks distributed on each subchromosome varies from 2 to 8. The results for each number of building blocks on subchromosomes are shown in different line-point styles. As shown in Fig. 5, utilizing subchromosome representations in LLGA can significantly improve the performance of LLGA on the uniformly scaled problems. Compared to the results reported elsewhere [12], LLGA with the subchromosome representation can solve uniformly scaled problems about five times larger in terms of the number of building blocks than that can be solved by LLGA without subchromosomes.

Figure 5 also shows that the limit for LLGA with the subchromosome representation to solve uniform scaled problems in terms of the total number of building blocks seems to be around 50. Even with different numbers of building blocks distributed on subchromosomes, no success trial was found among all the experiments with totally 60 building blocks. However, the procedure to apply exchange crossover on subchromosomes in these experiments causes serious building block disruption, as it does in LLGA without subchromosomes. While the original design of LLGA prevents us from lowering the disruption rate and maintaining the mixing rate at the same time, LLGA with the subchromosome representation provides us a viable way to appropriately adjust the probability for applying the operator. Therefore, the crossover operator is slightly modified as follows. Instead of every pair, only one pair of subchromosomes is now randomly chosen for applying exchange crossover to reduce building block disruption. The previous experiments were repeated for $n_{\mathrm{bb}} = 5$ and 6 to check the effect of adjusting the probability. The results are shown in Fig. 6 and indicate that the representation works in the range of these experiments.

## 5 Future Work

These results are tantalizing in that parallel evolution of linkage of large number of gene groupings has been demonstrated provided the appropriate genes are associated in the same linkage group. The key challenge left is to develop mechanisms to permit or encourage the evolution of these proper associations. Different mechanisms can be imagined for this purpose, and the potential of each is briefly outlined in the following paragraphs.

**Gene migration:** Gene migration moves genes among subchromosomes within one chromosome. Proper associations can be achieved through gene migration and favored by the the evolutionary process.

**Gene duplication or redundant genes:** Redundant genes can provide higher probabilities to form correct gene groups or clusters within subchromosomes.

**Adaptive expression:** Adaptive expression can resolve the conflicts caused by gene duplication and promote those identified building blocks.

Before running off to do more mass quantities of computation, however, we should think carefully about the key lessons of this paper. In going from the limited results of Fig. 5 to the much better results of Fig. 6, we recall that the primary difference was the limited amount of mechanical disruption that was permitted in the modified subchromosome crossover. In looking back over all studies of LLGA to this point, it is clear that these procedures can only tolerate a certain amount of *rearrangement disruption.* Large amounts of fitness variance are not problematic because they can be overcome through larger populations; however, attempts to move too much material around have always caused a combinatorial overload that cannot be sorted out. In designing mechanisms to move genes around either physically or virtually, we must recognize that the overall structure can assimilate only so much movement at any one time. Attention to this should guide the design of mechanisms to realize the potential of LLGA.

## 6 Summary and Conclusions

This paper started with a brief review of the linkage learning genetic algorithm, including the chromosome representation, exchange crossover, linkage learning mechanisms, and time models. The subchromosome representation was developed and employed in the linkage learning genetic algorithm for effectively lowering the number of building blocks to escape from the limit implied by the convergence time model. An initial step to realize subchromosome representations in the linkage learning genetic algorithm was taken in this work. The preliminary experimental results of using subchromosomes in the linkage learning genetic algorithm indicated that the proposed scheme can improve the performance of the linkage learning genetic algorithm on uniformly scaled problems.

In addition to showing that the subchromosome representation helps the linkage learning genetic algorithm to solve larger uniformly scaled problems, the initial step for implementing the proposed representation in the current work also leads a possible way in making the linkage learning genetic algorithm capable of incorporating prior linkage information. With the use of subchromosomes, the distribution of genes, non-coding segments, and building blocks can be determined according to the available linkage information of the problem. In the linkage learning genetic algorithm without subchromosomes, utilizing prior linkage information is extremely difficult if not impossible. Overall, the results reveal a promising path for achieving scalable genetic linkage learning techniques.

**Acknowledgments.** The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-03-1-0129. 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 or the U.S. Government.

## References

1. Thierens, D., Goldberg, D.E.: Mixing in genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)* (1993) 38–45
2. Goldberg, D.E., Deb, K., Thierens, D.: Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers* **32** (1993) 10–16
3. Kargupta, H.: The gene expression messy genetic algorithm. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation* (1996) 814–819
4. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detectio for overlapping functions. *Evolutionary Computation* **7** (1999) 377–398
5. 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)
6. Mühlenbein, H., Mahnig, T.: FDA - a scalable evolutionary algorithm for the optimization for the optimization of additively decomposed functions. *Evolutionary Computation* **7** (1999) 353–376
7. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The bayesian optimization algorithm. *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)* (1999) 525–532
8. Levenick, J.R.: Metabits: Generic endogenous crossover control. *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)* (1995) 88–95
9. Smith, J., Fogarty, T.C.: Recombination strategy adaptation via evolution of gene linkage. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation* (1996) 826–831
10. Chen, Y.-p., Goldberg, D.E.: Convergence time for the linkage learning genetic algorithm. *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)* (2004) N/A (To appear).
11. Harik, G.R.: *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms.* PhD thesis, University of Michigan, Ann Arbor, MI (1997)
12. Chen, Y.-p., Goldberg, D.E.: Introducing start expression genes to the linkage learning genetic algorithm. *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)* (2002) 351–360
13. Goldberg, D.E.: *The Design of Innovation: Lessons from and for Competent Genetic Algorithms.* Kluwer Academic Publishers (2002)
14. Chen, Y.-p., Goldberg, D.E.: Tightness time for the linkage learning genetic algorithm. *Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003)* (2003) 837–849
15. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2* (1993) 93–108
16. Harik, G., Cantú-Paz, E., Goldberg, D.E., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation* (1997) 7–12