行政院國家科學委員會專題研究計畫 成果報告

研究與發展專為無線網路系統客製化之最佳化演算架構研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 98-2221-E-009-072-
執 行 期 間 ：98 年 08 月 01 日至 99 年 07 月 31 日
執 行 單 位 ：國立交通大學資訊工程學系（所）

計 畫 主 持 人：陳穎平
共 同 主 持 人：許騰尹、陳耀宗
計畫參與人員：碩士班研究生-兼任助理人員：黃淵暐
碩士班研究生-兼任助理人員：古明哲
碩士班研究生-兼任助理人員：許庭毓
博士班研究生-兼任助理人員：林季穎
博士班研究生-兼任助理人員：李長紘

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 99 年 10 月 29 日

一、前言

演化計算 (Evolutionary computation) 為目前人工智慧領域中，一個相當重要的研究方向，藉由模擬自然界物種演化的機制，或是動植物賴以生存的法則，設計出各種最佳化演算法，適用於解決各式各樣不同種類的問題 [1, 2]。也由於其具有黑箱最佳化 (Black-box optimization) 演算架構的特性，演化計算領域中大部分的方法，皆可應用於各式各樣的最佳化問題上。因此，本計畫「研究與發展專為無線網路系統客製化之最佳化演算架構」原訂以兩年的期間，利用演化計算領域中最佳化方法極富彈性之特點，研發出可客製化之最佳化演算架構，並將之為無線網路技術中所存在的各項最佳化問題量身訂製，提供適切之最佳化工具與軟體。然而本計畫僅核定為一年期計畫，故計畫目標修訂為完成混合式變數之最佳化工具，並於計畫執行期間，進行演化計算最化佳方法論之各項相關研究。

二、研究目的

本計畫原訂之最終目標，在於研究與發展出「提供無線網路系統相關應用之最佳化服務基礎架構」，旨在以可客製化的最佳化架構，分別對無線網路應用系統中存在之數個參數型別與種類不同之最佳化議題加以適當處理與解決。

由於僅核定為一年期計畫，故修訂為著重演化計算最佳化方法論上的理論探討與特性分析。預期以延伸式精簡基因演算法為基礎架構，提出新式的泛用型最佳化演算架構，能直接適用於含有不同型態決策變數的待處理問題，包含布林變數、整數和實數。我們利用演化計算領域中之最佳化方法視目標函數 (Objective function) 為黑箱 (Black box) 的特性，開發出可適用於各種型別不同的參數最佳化問題之可客製化架構，以在將來配合各存在於無線網路應用系統中不同種類之最佳化問題，並將最佳化架構進行專為符合無線網路應用系統所需之客製化。

三、文獻探討

許多現實世界中的問題大多不像純數學問題般單純，可以直接套用公式或經過固定的計算程序來得到正確解答。這些現實問題最終仍需仰賴最佳化技術與工具的幫助，方能解決各決策變數 (Decision variable) 或稱參數 (Parameter) 的決定問題。舉凡工業設計、排程規劃、電路設計、資料壓縮、經濟學、建築學等等眾多領域，都存在著各式各樣不同的最佳化問題。譬如積體電路配置問題，對於相同的電路設計該如何配置能夠使用最小的面積，或是建築工程中，相同的建築材料該如何設計才能獲得最大的支撐力問題。這些問題常常都不難要找到一組可行解 (Feasible solution)，甚至是多組可行解，但是如果要找到問題的最佳解，通常就不是那麼地容易。不同的解在問題中有不同的結果反應，如果我們能客觀地分辨結果的優劣，就能以最佳化技術提升價值與成本的比值，以期能在各式問題中降低成本或是改善成果。。

其中，最常見的最佳化形式要屬問題的參數調整。對於想要進行最佳化處理的問題，通常需要定義一個目標函數 (Objective function)，來協助我們使用各種最佳

化技術。其中一種是使用經驗法則 (heuristic)，在可行解範圍逐步尋求最佳化。此類演算法包括基因演算法 (Genetic algorithm) [1, 2]、模擬退火演算法 (Simulated annealing) [3, 4]、螞蟻族群演算法 (Ant colony algorithm) [5]、粒子群最佳化 (Particle Swarm Optimization) [6, 7]… 等等。這類方法藉由模擬自然界的運作來達到最佳化目的。這類方法不再受限於目標函數的數學特性，可以應用於非線性、不可微分、或是不連續函數。無法用數學函數描述的問題，都可以設計模型，根據模擬得到的回饋進行最佳化演算。只要兩組解的優勝劣敗能夠被某種方式比較，甚至連不存在目標函數的問題也能適用，例如：個人化之樂音片段產生 [8]。此類演算法的可行性與實用性非常高，具有一定的求解能力，在有限時間內通常可以獲得在品質方面可被接受解，因此漸漸地被廣泛應用於現實世界問題。

四、研究方法

1. 變數型態之研究與分析

以目前現有的許多最佳化問題而論，我們依據常見的參數型態給予分類並討論分析。舉以一個小偷的背包問題為例子，分別對三種型態問題作一情境模擬。此問題設定是，小偷的背包有固定的重量限制，而現在有金、銀、銅三種不同材質的製品，其重量跟單價都不一樣，小偷該如何選擇才能在條件限制下獲得最大利益。

- 布林值 (Boolean values)

  布林變數常見的被使用在決策性變數上，已經確知有數個選項，每個選項可以用單一布林變數來表示選取或不選取。布林變數的問題通常也就是一般的排列組合問題。當小偷問題中的三種製品都只有一個時，即可用三個布林變數分別表示要帶走或不帶走情況，此即為典型的布林參數問題。

- 整數 (Integers)

  整數是處理離散資料的型態，一些對應到實體個數的參數問題常常就必須用整數來表示。若小偷的背包問題中，三種製品分別都有一個以上之數量，則可以用三個整數參數來記錄，構成整數參數最佳化的問題。

- 實數 (Real numbers)

  現實世界的工程問題大多是運作在實數域上，因此實數參數也就是最常被使用的型態，通常我們可以用實數向量來表示一組問題解。因為實數的連續性，除了在特定的問題類型之下 (例如：線性規劃問題或是可以實數近似之最佳化問題)，實數最佳解的搜尋經常比布林與整數型態的解還來得困難許多。假想在小偷的背包問題中，如果小偷有工具可以對三種製品做切割動作，那此問題就必須使用實數參數來表示帶走某種製品的數量，此問題則轉為實數最佳化問題。

上述三種問題情境，如同大部份最佳化問題一樣，都只有針對單一型態參數來對應問題中的決策變數，但仍不可否認地，有部份最佳化問題需要同時解決不同型態的參數。例如，當小偷問題中的金製品只有一個，銀製品有數個，而銅製品又可以被切割時，我們面對的最佳化問題即為一個混合型態決策變數的最佳化問題。作為研究計畫第一步，我們首先要針對混合型態參數的最佳化問題分析與瞭解問題特性，作為往後發展基礎，方有可能針對各類領域的最佳化問題提供量身訂製的最佳化架構服務。

2. 最佳化演算法架構

泛用型最佳化演算法的設計，包含了兩個最主要的部份，分別是將延伸式精簡基因演算法擴充至整數參數與實數參數。2.1 描述如何修改邊際乘積機率模型和最小描述長度原則機制，使整數參數也能適用。2.2 則是在整數架構中，再加上由本實驗團隊所開發之連續域離散化技巧「隨選分割」 (Split-on-domain) 來處理實數問題。

2.1 整數延伸式精簡基因演算法 (iECGA)

在整數延伸式精簡基因演算法的架構中，我們首先定義整數的範圍可以從 1 到 u，然後使用整數向量取代原本的二進位位元字串作為個體基因的表示方式。為了方便和原本的延伸式精簡基因演算法做比較，我們通常將整數範圍定為 2 的冪次方數。也就是 $d = u\text{-}l = 2^n$，則此範圍內的整數，都可以用長度為 n 的二進位字串來表示。在延伸式精簡基因演算法中，邊際乘積機率模型針對某特定群組進行元樣式的統計。舉例來說 $s = [1, 3, 4]$ 是某一基因群組，而 $|s| = 3$則是群組大小，邊際乘積機率模型的計算即如表格 1 所示，總共有 $2^{|s|}$ 個可能樣式。

表格 1: 邊際乘積機率模型之二進位範例

目前族群	樣式	次數
00110	0*00*	0
01010	0*01*	2
01110	0*10*	1
01100	0*11*	2
00010	1*00*	1
10001	1*01*	0
	1*10*	0
	1*11*	0

表格 2: 邊際乘積機率模型之整數範例

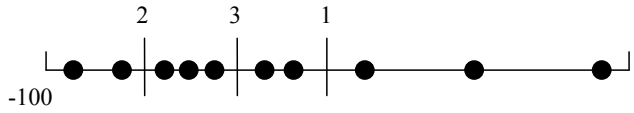
目前族群	樣式	次數
3472	0*0*	0
1624	0*1*	1
0314	0*2*	0
6715	...	.
4360	...	.
7164	7*6*	1
	7*7*	0

而整數延伸式精簡基因演算法修改邊際乘積機率模型統計對象為整數參數，對於同樣一組群組 $s = [1, 3, 4]$，若整數範圍 $d = u\text{-}l$，則總共要對 $d^{|s|}$ 種不同樣式作出現次數統計，如表格 2所示。除了修改邊際乘積機率模型以符合整數特性之外，模型複雜度估計的運算公式也必須加以修改。整數延伸式精簡基因演算法將二進位布林參數樣式的兩種情形擴展到整數範圍的 d 種情形，因此 Model Complexity 公式修正為公式 (1)。而 Compressed Population Complexity和其他部分的機制都和原延伸式精簡基因演算法相同。

$$\text{Model Complexity} = \log_2 N \sum_{i=1}^{m} d^{s_i} \tag{1}$$

2.2 實數延伸式精簡基因演算法

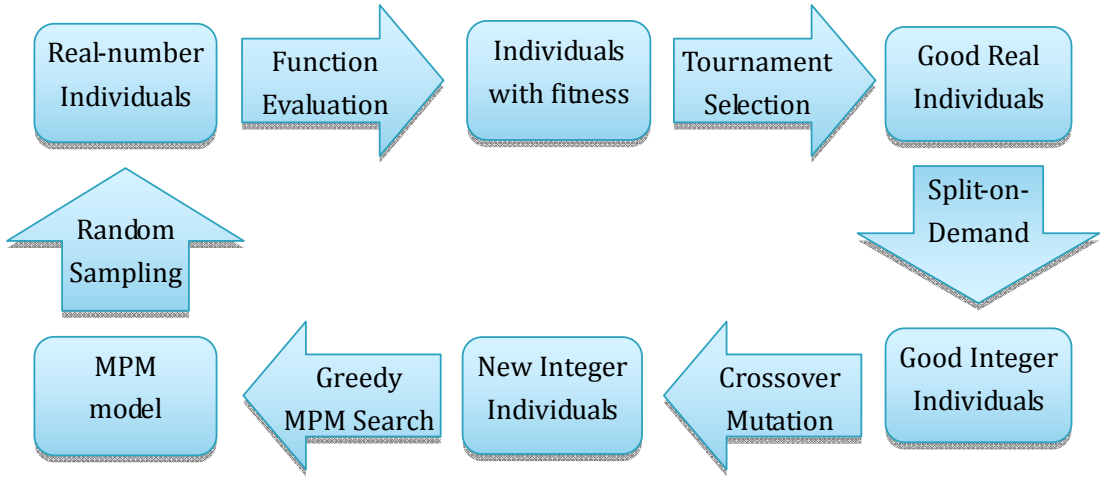
在討論實數延伸式精簡基因演算法之前，必須先介紹本實驗團隊過去所成功發展的連續值域適應性之離散化演算法「隨選分割」(Split-on-demand, SoD)。此演算法將一連續值域分割成數個區間，使得每個區間內的搜尋個體數目小於 N*λ，其中N為族群大小、λ為分割比率，可以用來平衡全域搜尋 (Global search) 跟區域搜尋 (Local search) 的強度與比重，也就是試圖在探索 (Exploration) 和利用 (Exploitation) 間找到適當的平衡。圖表 1 顯示一組隨選分割的範例。經過隨選分割處理，可以技巧性地將實數離散化為整數。



圖表 1: 隨選分割範例

延伸式精簡基因演算法原本是設計處理二進位資料的方法，為了能夠處理實數參數，我們將隨選分割機制整合在整數延伸式精簡基因演算法的流程中。因此實數延伸式精簡基因演算法架構中的個體分別有實數向量和整數向量兩種基因態，在目標函數的評估運算和存活個

體的選擇階段，個體以實數向量表示。接下來利用隨選分割將實數在該值域中，離散化為後編碼為整數向量進行整數延伸式精簡基因演算法之重組程序，重組完的子代個體為整數，之後經過隨機取樣，重新轉換回實數個體。圖表 2 為實數延伸式精簡基因演算法之流程圖。



圖表2: 實數延伸式精簡基因演算法架構流程

2.3 泛用型最佳化技術

本計畫所提出之創新泛用型最佳化技術，即奠基於原本的二進位延伸式精簡基因演算法，及本實驗室過去所開發的整數延伸式精簡基因演算法和實數延伸式精簡基因演算法。我們將以延伸式精簡基因演算法為最底層之最佳化引擎，而將同一個問題中各種不同的決策變數，加以適當地型別轉換，經由最佳化引擎處理後，再回復其原始型態。由過去發展最佳化技術之相關成功經驗得知，我們應可順利同時進行不同型態之決策變數的最佳化工作，並開發出優異創新之泛用型最佳化架構。

五、 結果與討論

本計畫原擬以兩年期間，進行「專為無線網路系統客製化之最佳化演算架構」之研究、探討、與發展工作。最終之預期成果，為開發出一套可客製化之泛用型最佳化架構，以提供各工程暨科學領域問題之最佳化服務。並且，將此最佳化架構針對無線網路系統，客製化成為量身訂製之最佳化基礎服務與工具，以處理無線網路應用系統內，各種不同之最佳化問題。然而如前所述，本計畫被核定為一年期，故完成之項目為原訂之第一年主題「可客製化之泛用型最佳化架構的設計與發展」。在此主題中，以演化計算方法論為基礎，配合數項創新技術，設計並發展出新的可客製化之泛用型最佳化演算架構，能直接適用於含有不同型數策變數之最佳化問題。已完成之具體工作項目如下：

- 參與人員獲得以下之訓練：
  - 培養研究生分工合作之能力；
  - 訓練參與人員研究、統合與論文寫作能力；
  - 統整研究成果並發表學術論文；
  - 學習實作系統之實務經驗；
  - 強化參與人員之資料分析、演化計算、機械學習、數值分析與最佳化技術等相關技能。
- 設計最佳化演算架構: 提出可適用於含有各種不同型態決策變數之問題的新型最佳化技術，以因應真實世界狀況中高度複雜之工程問題與困難。
- 實作最佳化計算架構: 將所提出之技術，實作為獨立的最佳化工具與服務，以供本計畫之相關人員，甚至是其他研究領域之人員分享與使用。已完成之原始程式碼，可由此網址下載：

  http://nclab.tw/SM/2010/01/

- 撰寫報告並投稿論文。基於國科會之補助，本實驗室發表了以下的相關論文：
  - 期刊論文：
    - Chuang, C.-Y., & Chen, Y.-p. (2010). Sensibility of linkage information and effectiveness of estimated distributions. *Evolutionary Computation, 18*(4). doi: 10.1162/EVCO_a_00010. (SCI).
    - Chen, Y.-p., & Jiang, P. (2010). Analysis on the facet of particle interaction in particle swarm optimization. *Theoretical Computer Science, 411*(21), 2101–2115. doi: 10.1016/j.tcs.2010.03.003. (SCI, EI).
  - 會議論文：
    - Huang Y.-w. & Chen, Y.-p. (2010). Detecting General Problem Structures with Inductive Linkage Identification. In *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010)*. (Accepted).
    - Lin J.-H. & Chen, Y.-p. (2010). XCS with Bit Masks. In *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010)*. (Accepted).
    - Chen, Y.-p. (2010). Estimation of distribution algorithms: Basic ideas and future directions. In *Proceedings of World Automation Congress 2010 (WAC 2010)* (pp. IFMIP–152). (Invited).
    - Chen, C.-M., Chen, Y.-p., Shen, T.-C., & Zao, J. (2010). On the optimization of degree distributions in LT codes with covariance matrix adaptation evolution strategy. In *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010)* (pp. 3531–3538). doi: 10.1109/CEC.2010.5586202. (EI).

- Chen, C.-M., Chen, Y.-p., Shen, T.-C., & Zao, J. (2010). Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition. In *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010)* (pp. 3635–3642). doi: 10.1109/CEC.2010.5586340. (EI).

參考文獻

[1] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass.: Addison-Wesley Pub. Co., 1989.

[2] D. E. Goldberg, *The design of innovation : lessons from and for competent genetic algorithms*. Boston: Kluwer Academic Publishers, 2002.

[3] S. Kirkpatrick*, et al.*, "Optimization by Simulated Annealing," *Science,* vol. 220, pp. 671-680, 1983.

[4] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications,* vol. 45, pp. 41-51, 1985.

[5] M. Dorigo*, et al.*, "Ant algorithms for discrete optimization," *Artificial Life,* vol. 5, pp. 137-172, 1999.

[6] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the Sixth International Symposium on Micromachine and Human Science,* pp. 39-43, 1995.

[7] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proceedings of IEEE International Conference on Neural Networks,* pp. 1942-1948, 1995.

[8] T. Y. Fu*, et al.*, "Evolutionary interactive music composition," *GECCO 2006: Genetic and Evolutionary Computation Conference, Vol 1 and 2,* pp. 1863-1864, 2006.

附錄

期刊論文：

1. Chuang, C.-Y., & Chen, Y.-p. (2010). Sensibility of linkage information and effectiveness of estimated distributions. *Evolutionary Computation, 18*(4). doi: 10.1162/EVCO_a_00010. (SCI).

2. Chen, Y.-p., & Jiang, P. (2010). Analysis on the facet of particle interaction in particle swarm optimization. *Theoretical Computer Science, 411*(21), 2101–2115. doi: 10.1016/j.tcs.2010.03.003. (SCI, EI).

會議論文：

3. Huang Y.-w. & Chen, Y.-p. (2010). Detecting General Problem Structures with Inductive Linkage Identification. In *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010)*. (Accepted).

4. Lin J.-H. & Chen, Y.-p. (2010). XCS with Bit Masks. In *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010)*. (Accepted).

5. Chen, Y.-p. (2010). Estimation of distribution algorithms: Basic ideas and future directions. In *Proceedings of World Automation Congress 2010 (WAC 2010)* (pp. IFMIP–152). (Invited).

6. Chen, C.-M., Chen, Y.-p., Shen, T.-C., & Zao, J. (2010). On the optimization of degree distributions in LT codes with covariance matrix adaptation evolution strategy. In *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010)* (pp. 3531 – 3538). doi: 10.1109/CEC.2010.5586202. (EI).

7. Chen, C.-M., Chen, Y.-p., Shen, T.-C., & Zao, J. (2010). Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition. In *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010)* (pp. 3635–3642). doi: 10.1109/CEC.2010.5586340. (EI).

Chuang, Chung-Yao and Chen, Ying-ping. xxxx. Sensibility of Linkage Information and Effectiveness of Estimated Distributions. *Evolutionary Computation*, uncorrected proof.

Sensibility of Linkage Information and Effectiveness of Estimated Distributions

Chung-Yao Chuang cychuang@nclab.tw
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
Ying-ping Chen* ypchen@nclab.tw
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Abstract

The probabilistic model building performed by estimation of distribution algorithms (EDAs) enables these methods to use advanced techniques of statistics and machine learning for automatic discovery of problem structures. However, in some situations, it may not be possible to completely and accurately identify the whole problem structure by probabilistic modeling due to certain inherent properties of the given problem. In this work, we illustrate one possible cause of such situations with problems consisting of structures with unequal fitness contributions. Based on the illustrative example, we introduce a notion that the estimated probabilistic models should be inspected to reveal the effective search directions, and further propose a general approach which utilizes a reserved set of solutions to examine the built model for likely inaccurate fragments. Furthermore, the proposed approach is implemented on the extended compact genetic algorithm (ECGA) and experiments are performed on several sets of additively separable problems with different scaling setups. The results indicate that the proposed method can significantly assist ECGA to handle problems comprising structures of disparate fitness contributions and therefore may potentially help EDAs in general to overcome those situations in which the entire problem structure cannot be recognized properly due to the temporal delay of emergence of some promising partial solutions.

Keywords

Sensible linkage, effective distribution, linkage sensibility, probabilistic model, model pruning, estimation of distribution algorithm, extended compact genetic algorithm, evolutionary computation.

1 Introduction

Estimation of distribution algorithms (EDAs; Mühlenbein and Paaß, 1996; Larrañaga and Lozano, 2001; Pelikan, Goldberg, et al., 2002) are a class of evolutionary algorithms that replace the traditional variation operators, such as mutation and crossover, by building a probabilistic model on promising solutions and sampling the built model to generate new candidate solutions. Using probabilistic models for exploration enables these methods to automatically capture the likely structure of promising solutions and exploit the identified problem regularities to facilitate further search. It is presumed that EDAs can detect the structure of the problem by recognizing the regularities within the promising solutions. However, for certain problems, EDAs are unable to identify the

*To whom correspondence should be addressed.

entire structure of the problem at a given time because the set of selected solutions on which the probabilistic model is built contains insufficient information regarding some parts of the problem and renders EDAs incapable of processing these parts accurately.

This paper starts by observing the evolutionary process of an EDA when dealing with an exponentially scaled problem, and recognizing that the population on which the probabilistic model is built does not necessarily contain sufficient information for all problem structures to be detected completely and accurately. Based on this observation, this study proposes a general concept that estimated probabilistic models should be inspected to reveal the effective search directions, and we provide a practical approach that utilizes a reserved set of solutions to examine the built model for the fragments that may be inconsistent with the actual problem structure. Furthermore, the proposed approach is implemented on the extended compact genetic algorithm (ECGA; Harik, 1999) and experimented on several sets of additively separable problems with different scaling difficulties (Goldberg, 2002) to demonstrate the applicability.

The following section briefly reviews the research topics concerning this study. Section 3 then demonstrates the interaction between the scaling difficulty and probabilistic model building performed by EDAs. More specifically, we will investigate how the scaling difficulty shadows the ability of EDAs to recognize problem structures and causes inaccurate processing on the part of some solutions. Accordingly, a general approach will be proposed in Section 4 to resolve this issue and enforce accurate processing during the optimization process. In Section 5, an implementation of the proposed approach on the extended compact genetic algorithm will be detailed. Section 6 presents the empirical results, followed by discussion and analysis in Section 7. Finally, Section 8 concludes the paper.

2 Background

Genetic algorithms (GAs; Holland, 1992; Goldberg, 1989) are search techniques loosely based on the paradigm of natural evolution, in which species of creatures tend to adapt to their living environments through mutation and inheritance of useful traits. Genetic algorithms mimic this mechanism by introducing artificial selections and genetic operators to discover and recombine partial solutions. By properly growing and mixing promising partial solutions, which are often referred to as building blocks (BBs; Goldberg, 2002), GAs are capable of efficiently solving a host of problems. The ability to implicitly process a large number of partial solutions has been recognized as an important source of the computational power of GAs. According to the Schema theorem (Holland, 1992), short, low-order, and highly fit subsolutions increase their share in the final combined solution. Further, as stated in the building block hypothesis (Goldberg, 1989), GAs implicitly decompose a problem into subproblems by processing building blocks. This decompositional bias is a good strategy for tackling many real-world problems, because real-world problems can oftentimes be reliably solved by combining the pieces of promising solutions in the form of problem decomposition.

However, proper growth and mixing of building blocks are not always achieved. GAs in the simplest form employ fixed representations and problem-independent recombination operators, which often breaks promising partial solutions while performing crossovers. This can cause crucial building blocks to vanish, thus leading to a convergence to local optima. In order to overcome this building block disruption problem, various techniques have been proposed. In this study, we focus on one line of effort often called the estimation of distribution algorithm (EDA; Mühlenbein and Paaß, 1996;

Larrañaga and Lozano, 2001; Pelikan, Goldberg, et al., 2002). These methods construct probabilistic models of promising solutions and utilize the built models to generate new solutions. Ideally, by detecting dependencies among variables through probabilistic modeling, these approaches can capture the structure of the problem and thus avoid the disruption of identified partial solutions. Early EDAs, such as population-based incremental learning (PBIL; Baluja, 1994) and the compact genetic algorithm (cGA; Harik et al., 1999), assume no interaction between decision variables, that is, decision variables are assumed to be independent of each other. Subsequent studies progressed from capturing pairwise interactions, such as mutual-information-maximizing input clustering (MIMIC; De Bonet et al., 1997), Baluja's dependency tree approach (Baluja and Davies, 1997), and the bivariate marginal distribution algorithm (BMDA; Pelikan and Mühlenbein, 1999), to modeling multivariate interactions, such as the extended compact genetic algorithm (ECGA; Harik, 1999), the Bayesian optimization algorithm (BOA; Pelikan et al., 1999), the estimation of Bayesian network algorithm (EBNA; Etxeberria and Larrañaga, 1999), the factorized distribution algorithm (FDA; Mühlenbein and Mahnig, 1999), and the learning version of FDA (LFDA; Mühlenbein and Höns, 2005). Along this line of research, questions arose naturally regarding the ability of EDAs to solve problems and the probabilistic models employed to learn the problem structures. Early studies recognized that solving problems composed of higher order building blocks is not expected to be accomplished by using just any probability density structure. Bosman and Thierens (1999) demonstrated that even when the set of variables forming a building block is linked and expressed by the best possible MIMIC-like chain structure, directly sampling that chain to generate new solutions is not a good strategy for reliable optimization. More recently, Echegoyen et al. (2007) compared the behavior of EBNA with approximate and exact Bayesian network learning. In another vein, Hauschild et al. (2007) analyzed the structure and complexity of learned probabilistic models and attempted to facilitate the model building process by incorporating the knowledge acquired from previous models (Hauschild et al., 2008).

Another topic relevant to this study is the impact of disparate *scale* among different building blocks on the behavior and performance of the evolutionary algorithms. It is commonly observed that building blocks with higher marginal fitness contributions—salient building blocks—converge before those with lower marginal fitness contributions. This sequential convergence behavior is referred to as domino convergence (Thierens et al., 1998). In real-world applications, it is often the case that some parts of the problem are more prominent and contribute more to the fitness than other parts.[1] Such a situation can pose two types of difficulties. Firstly, because the processing on the population is statistical in nature, building block scaling can cause inaccurate processing of less fit building blocks (Goldberg et al., 1992; Goldberg and Rudnick, 1991). The second difficulty arises because the lower fitness of a building block generally causes it to be processed at a later time compared to those of higher fitness. This delay on timeline can cause the building block to converge under random pressure, instead of proper selective pressure. Previous studies on this topic include the explicit role of scale in a systematic experimental setting (Goldberg et al., 1990), a theoretical model

[1]The reader may note that this statement cannot be formally proved nor disproved because we do not know nor even have a way to estimate the distribution of all real-world problems. However, this intuition can be better articulated by the explanation provided in Goldberg (2002): differences in scale are likely to be common across the space of likely problems, that is, the chance that we encounter differences in scale may be much larger than encountering equivalence in scale.

on the convergence behavior of exponentially scaled problems (Thierens et al., 1998), an extension of that model to building blocks more than one variable long (Lobo et al., 2000), and a convergence model of linkage learning genetic algorithms (LLGAs; Harik, 1997) on problems with different scaling setups (Chen and Goldberg, 2005).

Although the aforementioned scaling difficulty exists in a number of problems and degrades the performance of many evolutionary algorithms (EAs), there are scant investigations concerning the behavior of EDAs in the presence of scaling difficulties. Therefore, this study attempts to explore how the scaling difficulty affects EDAs, and proposes a practical countermeasure to assist EDAs on problems with different scalings. Specifically, we propose the notion that the estimated probabilistic models should be examined to enforce accurate processing of building blocks and prevent random drift from taking place. In the remainder of this paper, our approach will be demonstrated and evaluated on the test problems constructed by concatenating several trap functions. A k-bit trap function is a function of unitation[2] which can be expressed as

$$f_{\text{trap}_k}(s_1 s_2 \cdots s_k) = \begin{cases} k, & \text{if } u = k \\ k - 1 - u, & \text{otherwise} \end{cases},$$

where u is the number of ones in the binary string $s_1 s_2 \cdots s_k$. The trap functions were used pervasively in the studies concerning EDAs and other evolutionary algorithms because they provide well-defined structures among variables, and the ability to recognize intervariable relationships is essential to solve the problems consisting of traps (Deb and Goldberg, 1993, 1994).

3 Linkage Sensibility

The ability of EDAs to handle the building block disruption problem comes primarily from the explicit modeling of selected promising solutions using probabilistic models. The model construction algorithms, though they differ in their representative power, capture the likely structures of good solutions by processing the population-wise statistics collected from the selected solutions. By reasoning the dependencies among different parts of the problem and the possible formations of good solutions, reliable mixing and growing of building blocks can be achieved. As noted by Harik (1999), learning a good probability distribution is equivalent to learning linkage, where linkage refers to the dependencies among variables. Bosman and Thierens (1999) further recognized that in order to achieve reliable optimization, linkage information should be utilized in a way such that each corresponding building block can be identified and used as a whole.

In most studies on EDAs, it is presumed that EDAs can detect linkage and recognize building blocks according to the information contained in the set of selected solutions. However, in this study, we argue that in some situations, accurate and complete linkage information cannot be acquired by distribution estimation because the selected set of solutions on which the model is built contains insufficient information on the lower fitness parts of the problem. For example, consider a 16-bit maximization problem

[2] A function in which the function value depends only on the number of ones in the binary input string.

Table 1: Marginal product models built by ECGA when solving an exponentially scaled problem. Each group of variables represents a marginal model in which a marginal distribution resides. The converged variables are crossed out.

Generation	Marginal product model
1	$[\boldsymbol{s_1\ s_2\ s_3\ s_4}]\ [s_5\ s_{10}\ s_{16}]\ [s_6\ s_7]\ [s_8\ s_9\ s_{12}]\ [s_{11}\ s_{14}\ s_{15}]\ [s_{13}]$
2	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ [\boldsymbol{s_5\ s_6\ s_7\ s_8}]\ [s_9\ s_{13}\ s_{16}]\ [s_{10}\ s_{14}\ s_{15}]\ [s_{11}s_{12}]$
3	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ [\boldsymbol{s_9\ s_{10}\ s_{11}\ s_{12}}]\ [s_{13}\ s_{16}]\ [s_{14}\ s_{15}]$
4	$\cancel{[s_1]}\ \cancel{[s_2]}\ \cancel{[s_3]}\ \cancel{[s_4]}\ \cancel{[s_5]}\ \cancel{[s_6]}\ \cancel{[s_7]}\ \cancel{[s_8]}\ \cancel{[s_9]}\ \cancel{[s_{10}]}\ \cancel{[s_{11}]}\ \cancel{[s_{12}]}\ [\boldsymbol{s_{13}\ s_{14}\ s_{15}\ s_{16}}]$

formed by concatenating four 4-bit trap functions as subproblems,

$$f(s_1 s_2 \cdots s_{16}) = \sum_{i=0}^{3} (5^{3-i} f_{\text{trap}_4}(s_{4i+1} s_{4i+2} s_{4i+3} s_{4i+4})) \ ,$$

where $s_1 s_2 \cdots s_{16}$ is a solution string. Note that in contrast to other studies of EDAs, in which the test problems are scaled uniformly, that is, the subproblems are of equal fitness, in this problem, each elementary trap function is scaled exponentially. This scaling is an abstraction for problems of distinguishable prominence or solving priority among the constitutive subproblems. Suppose that we choose ECGA (Harik, 1999), which uses a class of multivariate probabilistic models called marginal product models (MPMs), to tackle this problem.[3] By observing subsequent generations of the optimization process, a series of models built by ECGA can be obtained like those listed in Table 1. In this table, the variables enclosed by the same pair of brackets are considered dependent and are modeled jointly. Each group of variables represents a marginal model in which a marginal distribution resides, and the converged variables are crossed out.[4]

It can be observed that the models shown in Table 1 are only partially correct in each generation. More specifically, in each generation, only the most fit building block on which the population has not converged is correctly modeled. This is due to the fact that some part of the problem contributes much more than all others combined. If one part of the problem is worth more than the others, then this part of the solution solely determines the chance regarding whether or not the solution will be selected. As a consequence, only the most fit building block can provide sufficient information to be modeled correctly, since the model searching is performed based on the selected solutions. The remaining parts of the model are primarily the result of low fitness partial solutions "hitchhiking" on the more fit building blocks.

From the above example, we can see that not all building blocks can be detected from a given set of selected solutions by probabilistic model building. Model building algorithms cannot "see" the entire structure of the problem from the selected set of solutions because the disparate scale among different building blocks prevents complete linkage information from being included in the selected population. In this work, we will refer to this concept as *linkage sensibility* and those problem structures that can be identified properly using the given set of solutions are called *sensible linkage*. Based on this notion, we reexamine EDAs on the building block disruption problem. It is clear

[3]See Section 5.1 for a more detailed description of ECGA and marginal product models.

[4]The convergence of a variable is defined as all solutions in the population possessing the same value for that variable, that is, no further changes for that variable will occur.

that the disruption problem still exists in the insensible portion of the problem because that part of the problem cannot be modeled properly. Although the above example is an extreme case of scaling, in that each subproblem is exponentially scaled, in real-world problems, it is often the case that the constitutive subproblems are weighted significantly differently, which implies that the linkage might be only partially sensible. In addition to the building block disruption problem, the random drift of the less salient parts of the problem mentioned in Section 2 further worsens the situation. These situations and issues are usually handled by increasing population size when EDAs are adopted. However, we may gain a new way to deal with these situations if it is possible to distinguish a sensible linkage from an insensible linkage.

4 Effective Distributions

The idea of sensible linkage can be closely mapped into another notion called *effective distributions*. By effective distributions, we mean that by sampling these distributions, the solution quality can be reliably advanced. Hence, the crucial criteria for effective distributions are the consistency with building blocks and the provision of good directions for further search. If it is possible to extract effective marginal distributions from the built probabilistic model, we can perform partial sampling using only these marginal distributions, and leave the remaining parts of the solutions unchanged. Thus, the diversity is maintained and we are free from the building block disruption and random drift problems. For instance, returning to the earlier 16-bit optimization problem, if it is possible to identify those partial models that are built on the sensible linkage like $[s_1\ s_2\ s_3\ s_4]$ in the first generation and $[s_5\ s_6\ s_7\ s_8]$ in the second generation, we can sample only the corresponding marginal distributions which are, in this case, effective. That is, in the first generation, for each solution string, we resample only $s_1s_2s_3s_4$ according to the marginal distribution and keep $s_5s_6 \cdots s_{16}$ unchanged. In the second generation, we resample only s_1 to s_8 according to the marginal distributions and keep $s_9s_{10} \cdots s_{16}$ with the same values (note that $s_1s_2s_3s_4$ are converged). In this way, we do not have to resort to increasing the population size to deal with the problems caused by the disparate building block scaling.

The above thoughts leave us one complication: the identification of effective distributions. However, the direct identification of effective distributions may be a difficult if not impossible task. It may be wise to adopt a complementary approach—to identify those marginal distributions that are *not* likely to be effective. If there is a way to identify the ineffective distributions, we can bypass them and use only the rest of the probabilistic model, and thus approximate the result of knowing effective distributions. Our idea is that we can split the entire population into two subpopulations, use only one of the subpopulations for building the probabilistic model, and utilize the other subpopulation to collect some statistics for possible indications of ineffectiveness of certain marginal distributions in the probabilistic model built on the first subpopulation. That is, with some appropriate heuristics or criteria, we can prune the likely ineffective portions of the model.

In the next section, our implementation in ECGA of the proposed concept will be detailed. More specifically, a judging criterion will be proposed to detect the likely ineffective marginal distributions of a given marginal product model.

5 ECGA with Model Pruning

This section starts with a brief review of the (ECGA; Harik, 1999). Based on the idea of detecting the inconsistency of statistics gathered from two subpopulations of the

Table 2: An example of a marginal product model that defines a probability distribution over four variables. The variables enclosed in the same brackets are modeled jointly, and each variable subset is considered independent of the other variable subsets.

$[s_1]$	$[s_2\ s_4]$	$[s_3]$
$P(s_1 = 0) = 0.4$	$P(s_2 = 0, s_4 = 0) = 0.2$	$P(s_3 = 0) = 0.5$
$P(s_1 = 1) = 0.6$	$P(s_2 = 0, s_4 = 1) = 0.1$	$P(s_3 = 1) = 0.5$
	$P(s_2 = 1, s_4 = 0) = 0.1$	
	$P(s_2 = 1, s_4 = 1) = 0.6$	

same source, a mechanism is devised to identify the possibly ineffective parts of the built probabilistic model. Finally, an optimization algorithm incorporating the proposed technique is described in detail.

5.1 Extended Compact Genetic Algorithm

ECGA uses a product of marginal distributions on a partition of the variables. This kind of probability distribution belongs to a class of probabilistic models known as marginal product models (MPMs). In this kind of model, subsets of variables can be modeled jointly, and each subset is considered independent of other subsets. In this work, the conventional notation is adopted that variable subsets are enclosed in brackets. Table 2 presents an example of MPM defined over four variables: s_1, s_2, s_3, and s_4. In this example, s_2 and s_4 are modeled jointly and each of the three variable subsets ($[s_1]$, $[s_2\ s_4]$, and $[s_3]$) is considered independent of the other subsets. For instance, the probability that this MPM generates a sample $s_1s_2s_3s_4 = 0101$ is calculated as follows,

$$\begin{aligned} P(s_1s_2s_3s_4 = 0101) &= P(s_1 = 0) \times P(s_2 = 1, s_4 = 1) \times P(s_3 = 0) \\ &= 0.4 \times 0.6 \times 0.5\ . \end{aligned}$$

In fact, as its name suggests, a marginal product model represents a distribution that is a "product" of the marginal distributions defined over variable subsets.

In ECGA, both the structure and the parameters of the model are searched and optimized in a greedy fashion to fit the statistics of the selected set of promising solutions. The measure of a good MPM is quantified based on the minimum description length (MDL) principle (Rissanen, 1978), which states that any regularity in a given set of data can be used to compress that data, and the success of a model in capturing those regularities can be measured by the cost of expressing the model and the length of the data compressed according to the model. The MDL principle thus penalizes both inaccurate and complex models, thereby leading to a descriptive yet not overly complicated distribution. Specifically, the search measure is the MPM complexity which is quantified as the sum of model complexity, C_m, and compressed population complexity, C_p. The greedy MPM search first considers all variables as independent and each of them forms a separate variable subset. In each iteration, the greedy search merges two variable subsets that yield the greatest reduction in $C_m + C_p$. This process continues until there is no further merge that can decrease the combined complexity.

The model complexity, C_m, quantifies the model representation in terms of the number of bits required to store all the marginal distributions. Suppose that the given problem is of length ℓ with binary encoding, and the variables are partitioned into m

subsets each of size k_i, $i = 1 \ldots m$, such that $\ell = \sum_{i=1}^{m} k_i$. Then the marginal distribution corresponding to the ith variable subset requires $2^{k_i} - 1$ frequency counts to be completely specified. Taking into account that each frequency count is of length $\log_2(n+1)$ bits, where n is the population size, the model complexity, C_m, can be defined as

$$C_m = \log_2(n+1) \sum_{i=1}^{m} \left(2^{k_i} - 1\right).$$

The compressed population complexity, C_p, quantifies the suitability of the model in terms of the number of bits required to store the entire selected population (the set of promising solutions picked by the selection operator) under an ideal compression scheme. The compression scheme is based on the partition of the variables. Each subset of the variables specifies an independent "compression block" on which the corresponding partial solutions are optimally compressed. Theoretically, the optimal compression method encodes a message of probability p_i using $-\log_2 p_i$ bits. Thus, taking into account all possible messages, the expected length of a compressed message is $\sum_i -p_i \log_2 p_i$ bits, which is optimal. In information theory (Cover and Thomas, 1991), the quantity $-\log_2 p_i$ is called the *information* of that message and $\sum_i -p_i \log_2 p_i$ is called the *entropy* of the corresponding distribution. Based on information theory, the compressed population complexity, C_p, can be derived as

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} \log_2 p_{ij},$$

where p_{ij} is the frequency of the jth possible partial solution to the ith variable subset observed in the selected population.

Note that in the calculation of C_p, it is assumed that the jth possible partial solution to the ith variable subset is encoded using $-\log_2 p_{ij}$ bits. This assumption is fundamental to our technique of identifying the likely ineffective marginal distributions. More precisely, the information of the partial solutions, $-\log_2 p_{ij}$, is a good indicator of inconsistency of statistics gathered from two separate subpopulations.

5.2 Model Pruning

Our technique of identifying the possibly ineffective fragments of a marginal product model is based on the notion that ECGA uses compression performance to quantify the suitability of a probabilistic model for a given set of solutions. The degree of compression is a quite representative metric to the fitness of modeling, because all good compression methods are based on capturing and utilizing the relationships among data (Grünwald, 2007). Thus, if the compression scheme of the MPM built on one set of solutions is incapable of compressing another set of solutions produced under the same condition,[5] then we can speculate that some of the constitutive marginal models observed in the first set of solutions are likely inconsistent with the distribution of the corresponding partial solutions observed in the second set of solutions. Such inconsistency can be seen

[5]For example, if all individuals are produced by sampling the same probabilistic model and selected using the same selection technique under the same pressure.

as a disagreement on the direction of further search. However, under the premise that these two sets of solutions are produced under the same condition, they are supposed to reveal similar directions of further search. Thus, we can reasonably speculate that proper selection pressures were not applied on these partial solutions (causing them to drift toward two different directions), and the true linkage structures on these parts of the problem is not sensible under this condition. Recalling our definition in Section 4, an effective distribution should be capable of providing good direction for further search and consistent with the linkage structure. Thus, if the abovementioned inconsistency is found, we can expect that with a high probability,[6] the inconsistent marginal models are ineffective. Based on the reasoning, we can perform a systematic checking on the given MPM for the likely ineffective portions.

Suppose that the population of solutions, P, is split into two subpopulations S and T. The model searching is performed on S', the set of promising solutions selected from S. Then we can use the statistics collected from T', the set of solutions selected from T, to examine the built probabilistic model, M. Since each marginal model functions independently, they can be inspected separately. Recall the former description that a variable subset, which specifies a marginal model, is viewed as a "compression block" that encodes each possible partial solution according to the marginal distribution. The jth possible partial solution to the ith variable subset is encoded using $-\log_2 p_{ij}$ bits, where p_{ij} is the frequency of the jth possible partial solution to the ith variable subset observed in S'. Assuming that the given problem is of length ℓ with binary encoding, and there are m variable subsets with each of size k_i, $i = 1 \ldots m$, in the built model M, for the ith marginal model, $i = 1 \ldots m$, we can check whether or not

$$\sum_{j=1}^{2^{k_i}} q_{ij}(-\log_2 p_{ij}) > k_i,$$

where q_{ij} is the frequency of the jth possible partial solution to the ith variable subset collected from T'. If the inequality holds, then the compression scheme employed in the ith marginal model is not a good one for compressing the corresponding partial solutions in T' because it encodes a k_i-bit partial solution to a bit string with an expected length of more than k_i bits. Based on the earlier reasoning, such a condition indicates that the marginal model is likely ineffective because T' does not agree on this part of the model. Otherwise, the scheme should be able to compress the partial solutions in T'.

Further explained from a machine learning perspective (Mitchell, 1997), a good model should generalize well to unseen instances. Otherwise, it captures coincidental regularities among the training data or what it has observed. If model building is performed on the portion where linkage is not sensible from the given set of solutions, it will "overfit" these partial solutions (i.e., take on hitchhikers) that were not subject to proper selection pressures. Consequently, the regularities captured by this part of modeling tend to be inconsistent with the true problem structure. Furthermore, the partial solutions that were not subject to proper selection pressure appear to be random, and such a situation brings about the phenomenon of random drift mentioned in Section 2. By its nature, drift is random, and two different subpopulations tend to drift in two different directions. Thus, we can use the statistical inconsistency between S' and

[6] Because the solutions are generated probabilistically, we cannot be absolutely sure.

Algorithm 1 ECGA with Model Pruning

```
Initialize a population P with n solutions of length ℓ.
while the stopping criteria are not met do
    Evaluate the solutions in P.
    Divide P into two subpopulations S and T at random.
    S' ← apply t-wise tournament selection on S.
    T' ← apply t-wise tournament selection on T.
    M ← build the MPM on S' with greedy search.
    M' ← prune M based on the inconsistency with T'.
    for each remaining marginal distribution D in M' do
        for each solution s = s_1 s_2 ··· s_ℓ in P do
            Change the values in s partially by sampling D.
        end for
    end for
end while
```


T' to locate the possible drift portions of the solutions and identify the likely ineffective parts within the whole model. By removing these likely ineffective parts, we can forge a partial but more effective model.

An issue in practice concerning the calculation of the inequality is that sometimes one or more possible partial solutions are absent in the set of selected solutions, leaving $-\log_2 p_{ij}$ undefined because $p_{ij} = 0$. In the present work, we handle this practical problem by assigning a very small value, smaller than $1/n$, to the p_{ij}'s that are zero and normalizing them such that p_{ij}'s sum to 1 (i.e., $\sum_j p_{ij} = 1$).

5.3 Integration

In this section, the optimization process incorporating ECGA and the proposed technique is described. This combination helps ECGA to achieve better performance when a disparate scale exists among different parts of the problem.

The procedure is presented in Algorithm 1. This process starts with initializing a population of solutions. After initialization, the solutions are evaluated, and then the entire population is randomly split into two subpopulations. Selection operations are performed on the two subpopulations separately with the same operator and selection pressure. Model building is performed on one of the subpopulations. The other subpopulation is used to prune the built model using the technique described previously. Finally, all solutions in the population are altered by sampling the remaining marginal distributions, which are considered effective, in the pruned model. These steps are repeated until the stopping criteria are satisfied.

A prominent difference between the above process and the regular EDAs is that the sampling might not include all variables. As introduced in Section 4, the existing solutions are altered by sampling only the marginal distributions surviving the model pruning process. Thus, a solution string might not be entirely modified in an iteration. This technique hence avoids random drift and inaccurate processing of low-fitness building blocks by postponing the processing until sufficient linkage information is available. Similar to the concept proposed by Bosman and Thierens (1999) that linkage information estimated from the selected solutions has to be utilized to recognize

Table 3: Marginal product models before and after pruning when solving a 16-bit exponentially scaled problem with the proposed approach.

Generation		Marginal product model (before and after pruning)
1	Before	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_{13}\ s_{16}]\ [s_6\ s_7\ s_{12}]\ [s_8\ s_{11}]\ [s_9\ s_{10}]\ [s_{14}\ s_{15}]$
	After	$[s_1\ s_2\ s_3\ s_4]$
2	Before	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [s_5\ s_6\ s_7\ s_8]\ [s_9\ s_{14}]\ [s_{10}\ s_{15}]\ [s_{11}\ s_{13}\ s_{16}]\ [s_{12}]$
	After	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [s_5\ s_6\ s_7\ s_8]$
3	Before	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\cancel{s_5}]\ [\cancel{s_6}]\ [\cancel{s_7}]\ [\cancel{s_8}]\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}]\ [s_{15}\ s_{16}]$
	After	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\cancel{s_5}]\ [\cancel{s_6}]\ [\cancel{s_7}]\ [\cancel{s_8}]\ [s_9\ s_{10}\ s_{11}\ s_{12}]$
4	Before	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\cancel{s_5}]\ [\cancel{s_6}]\ [\cancel{s_7}]\ [\cancel{s_8}]\ [\cancel{s_9}]\ [\cancel{s_{10}}]\ [\cancel{s_{11}}]\ [\cancel{s_{12}}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
	After	$[\cancel{s_1}]\ [\cancel{s_2}]\ [\cancel{s_3}]\ [\cancel{s_4}]\ [\cancel{s_5}]\ [\cancel{s_6}]\ [\cancel{s_7}]\ [\cancel{s_8}]\ [\cancel{s_9}]\ [\cancel{s_{10}}]\ [\cancel{s_{11}}]\ [\cancel{s_{12}}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$

building blocks, we further address that the validity of the linkage information should be confirmed beforehand. In this way, better performance in terms of function evaluations can be achieved if a disparate scale exists among different parts of the problem.

In order to confirm that the proposed method meets its design purpose, Table 3 lists the models before and after pruning when the earlier exponentially scaled problem is solved by Algorithm 1. It can be seen that the proposed approach appropriately removes the ineffective parts during each stage of the optimization process. In order to further illustrate the behavior and effect of the proposed approach, the algorithm is applied to another problem with a different scaling called *overloaded scaling*[7]

$$f(s_1 s_2 \cdots s_{16}) = \sum_{i=0}^{1} f_{\text{trap}_4}(s_{4i+1}s_{4i+2}s_{4i+3}s_{4i+4}) + \sum_{i=2}^{3} \frac{1}{5} f_{\text{trap}_4}(s_{4i+1}s_{4i+2}s_{4i+3}s_{4i+4}) \ ,$$

where $s_1 s_2 \cdots s_{16}$ is a solution string. The overloaded cases are those with two scales, where some subproblems are at the high level and the rest are at the low one. The models before and after pruning when such a problem is solved are shown in Table 4. It can be observed that the proposed method works as expected in splitting the solving process according to the scaling structure. The two subproblems of higher fitness are handled first, and the two subproblems of lower fitness are solved later.

6 Experiments

The experiments are designed to reveal the behavior of the proposed approach in handling sets of problems with different scaling difficulties. Because ECGA is limited in handling overlapped building blocks, we use only test problems that are additively separable. In this study, three bounding models of scalings (Goldberg, 2002) are considered: exponential, power law, and uniform. While the uniform and exponential cases

[7] As mentioned by Goldberg (2002), the word "overloaded" is a reference to the application of this idea in the early messy GA work (Goldberg et al., 1990), where such distributions were used to try to overload or overwhelm the ability of the messy GA to keep all building blocks present through all phases of the process.

Table 4: Marginal product models before and after pruning when solving a 16-bit problem of the overloaded scaling with the proposed approach.

Generation	Marginal product model (before and after pruning)	
1	Before	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]\ [s_9\ s_{16}]\ [s_{10}\ s_{14}\ s_{15}]\ [s_{11}\ s_{13}]\ [s_{12}]$
	After	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]$
2	Before	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]\ [s_9\ s_{13}\ s_{14}]\ [s_{10}\ s_{12}]\ [s_{11}\ s_{15}]\ [s_{16}]$
	After	$[s_1\ s_2\ s_3\ s_4]\ [s_5\ s_6\ s_7\ s_8]$
3	Before	$[s_1]\ [s_2]\ [s_3]\ [s_4]\ [s_5]\ [s_6]\ [s_7]\ [s_8]\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
	After	$[s_1]\ [s_2]\ [s_3]\ [s_4]\ [s_5]\ [s_6]\ [s_7]\ [s_8]\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
4	Before	$[s_1]\ [s_2]\ [s_3]\ [s_4]\ [s_5]\ [s_6]\ [s_7]\ [s_8]\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$
	After	$[s_1]\ [s_2]\ [s_3]\ [s_4]\ [s_5]\ [s_6]\ [s_7]\ [s_8]\ [s_9\ s_{10}\ s_{11}\ s_{12}]\ [s_{13}\ s_{14}\ s_{15}\ s_{16}]$

bound the scaling performance of an algorithm at two extremes, the power law cases enable us to see the behavior in between. Based on the different scalings, three sets of test functions are constructed using f_{trap_k} as the elemental function:

$$\text{Exponential: } \sum_{i=0}^{m-1}(k+1)^i f_{\text{trap}_k}(s_{k\times i+1}s_{k\times i+2}\cdots s_{k\times i+k})$$

$$\text{Power law: } \sum_{i=0}^{m-1}(i+1)^3 f_{\text{trap}_k}(s_{k\times i+1}s_{k\times i+2}\cdots s_{k\times i+k})$$

$$\text{Uniform: } \sum_{i=0}^{m-1} f_{\text{trap}_k}(s_{k\times i+1}s_{k\times i+2}\cdots s_{k\times i+k})$$

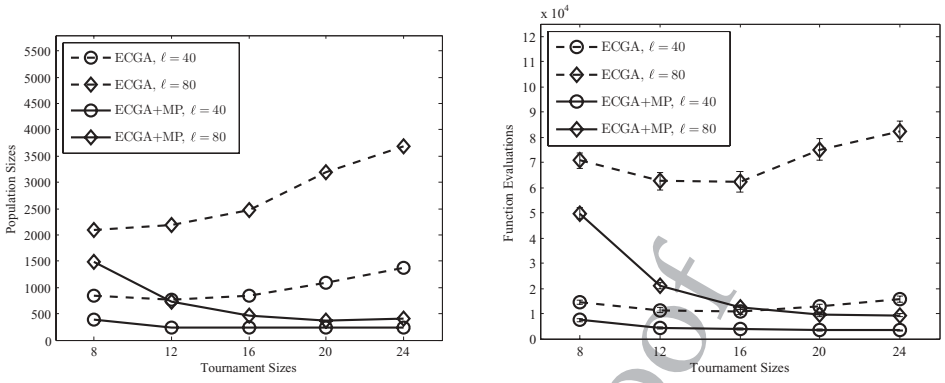
By adopting different scaling setups, we can compare the original ECGA with our approach under different degrees of linkage sensibilities. By varying k and m, we can observe the behavior of the proposed method with respect to different problem and subproblem sizes in a controlled manner. Furthermore, various selection pressures are also taken into consideration to make a more thorough observation.

The purpose of the following experiments is to understand the impact of the proposed method on the *computational resource* (population size and function evaluations) required to solve a problem. Thus, we do not use solution quality as a measure of comparison but treat it as a minimum requirement. More precisely, we use a bisection method (Sastry, 2001) to bound the minimum population size capable of achieving reliable convergence to the optimum. Of course, solution quality can be an important indicator for evaluating a newly invented approach. However, the primary goal of this study is to design a more *economic* approach for solving problems, and the experiments are designed to evaluate the ability of the proposed approach in this aspect.

6.1 Effect of Selection Pressure

This section describes the experiments designed for observing the effect of selection pressure on both the original ECGA and the ECGA combined with the proposed approach. The purpose of these experiments is twofold.

- First, we want to determine the range of selection pressure with which the proposed approach works as we designed. Appropriate selection pressure is quite



(a) Population Sizes (b) Function Evaluations

Figure 1: Empirical results of the proposed method and original ECGA on 40- and 80-bit ($k = 4$, $m = 10$ and 20) exponential scaled problems. Five tournament sizes ranging from 8 to 24 were used to observe the behavior of the algorithms under different selection pressures.

important to the proper functioning of our approach because the pruning mechanism is designed according to the statistical inconsistencies between the two subpopulations.

- Second, because the proposed approach will be compared with the original ECGA in the subsequent experiments, in order to make a fair and meaningful comparison, the selection pressure must be set to an appropriate value for the original ECGA to work under good conditions.

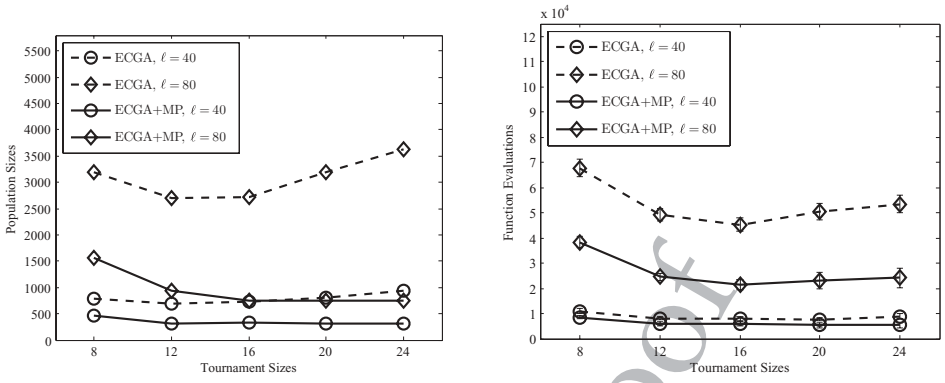
6.1.1 Experimental Settings

Because tournament selection is adopted, the selection pressure is altered by changing the tournament size. We consider tournament sizes ranging from 8 to 24, and the problem instances used to make the observations are of length 40 bits and 80 bits with 4-bit trap functions as subproblems ($k = 4$, $m = 10$ and 20, respectively).

For simplicity, the splitting of population is performed in the way that the two resulting subpopulations are disjoint and of equal size. The stopping criterion is set such that a run is terminated when all solutions in the population converge to the same fitness value. For each tournament size, the minimum required population size is determined by a bisection method (Sastry, 2001) such that on average, $m - 1$ building blocks converge to the correct values in 50 runs for each of the two problem instances.

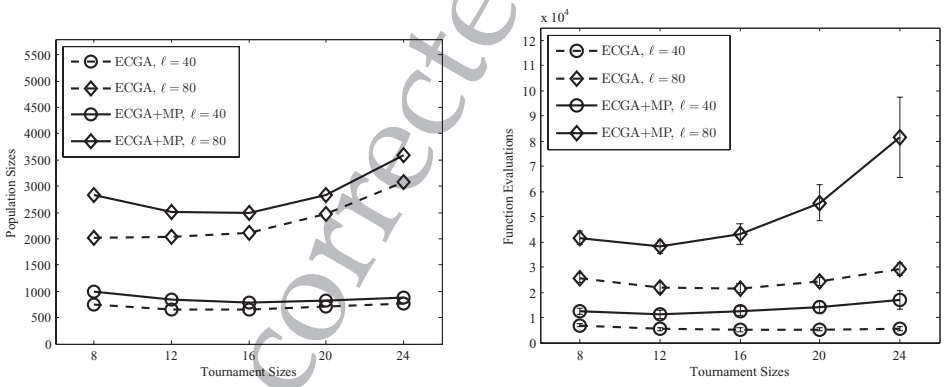
6.1.2 Results and Observations

The results for exponential, power law, and uniformly scaled problems are presented in Figures 1, 2, and 3, respectively. It can be observed from Figures 1(b), 2(b), and 3(b) that for all three scalings, the original ECGA works best (in terms of the number of function evaluations) under tournament size 12 or 16. Based on that, we will use these two tournament sizes in the following sets of experiments to ensure that the improvement of our approach over the original ECGA is not a result of improper selection pressure. In fact, we also performed experiments using a tournament size of 4, of which the results



(a) Population Sizes (b) Function Evaluations

Figure 2: Empirical results of the proposed method and original ECGA on 40- and 80-bit ($k = 4, m = 10$ and 20) power law scaled problems. Five tournament sizes ranging from 8 to 24 were used to observe the behavior of the algorithms under different selection pressures.



(a) Population Sizes (b) Function Evaluations

Figure 3: Empirical results of the proposed method and original ECGA on 40- and 80-bit ($k = 4, m = 10$ and 20) uniformly scaled problems. Five tournament sizes ranging from 8 to 24 were used to observe the behavior of the algorithms under different selection pressures.

are listed in Table 5. This demonstrates that adopting a lower selection pressure does not yield better performance for ECGA or for our approach.

The results of these experiments give some insights into the pruning mechanism. It can be observed that the appropriateness of a particular selection pressure is related to the linkage sensibility of the problem at hand. This property could cause inconvenience in choosing selection pressure for the algorithm because when dealing with black box optimization, we usually do not have any information about the problem at hand. Fortunately, Figures 1(b), 2(b), and 3(b) also suggest that under tournament sizes ranging

Table 5: Empirical results of the proposed method and original ECGA using a tournament size of 4. Experiments were conducted on 40- and 80-bit problems formed by concatenating 4-bit trap functions with three different scalings. The symbols ℓ, n, and f_{ev} denote problem size, population size, and function evaluations, respectively.

		ℓ	n	f_{ev}	std. of f_{ev}
Exponential	ECGA	40	1,719	44,487.72	2,682.02
		80	3,748	187,549.92	5,912.06
	ECGA+MP	40	1,405	37,373.00	2,027.11
		80	4,221	210,881.16	8,568.54
Power law	ECGA	40	1,604	32,946.16	2,105.37
		80	5,507	163,557.90	6,017.21
	ECGA+MP	40	1,248	27,755.52	1,929.44
		80	4,361	141,034.74	5,884.63
Uniform	ECGA	40	1,346	17,228.80	1,489.44
		80	3,479	58,308.04	3,411.61
	ECGA+MP	40	2,181	30,446.76	2,411.81
		80	5,598	100,540.08	5,535.96

from 8 to 16, our approach works better than the original ECGA in the exponential and power law scaled cases. Under this range of tournament sizes (8 to 16), the behavior of the proposed approach in uniformly scaled cases is relatively stable compared to that under higher selection pressure. This observation demonstrates that for a broad range of selection pressure, the improvement obtained by using the pruning mechanism can be expected in cases of limited linkage sensibility, while in cases for which linkage information is completely sensible, the overhead is relatively stable.

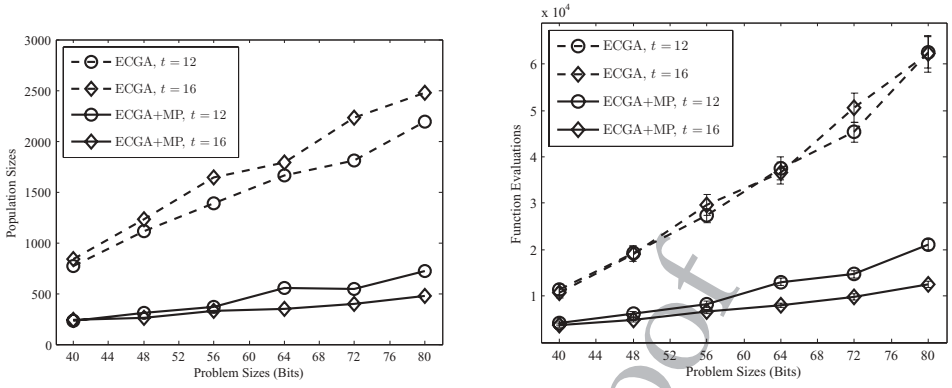
6.2 Impact on Population Requirement with Increasing *m*

This section describes experiments designed to reveal the behavior of the proposed approach when the number of subproblems within a problem is growing (i.e., increasing m with fixed k). In order to illustrate the effectiveness and benefit of adopting the pruning mechanism and to estimate the overhead when it is not needed, the proposed approach will be compared with the original ECGA on three sets of problems with different scaling setups.

6.2.1 Experimental Settings

The problem instances used in this set of experiments are composed of 4-bit trap functions and ranging from 40 to 80 bits ($k = 4$, $m = 10 \ldots 20$). Two selection pressures are adopted by setting tournament size t to 12 and 16. The reason for using these two tournament sizes is because our approach is compared with the original ECGA, which seems to perform better with $t = 12$ or $t = 16$ according to the previous set of experiments. Otherwise, a question might arise as to whether or not the inferior performance of the original ECGA under some scaling difficulties comes from the inappropriate setting of selection pressure.

As in the previous experiment, the splitting of population is also performed in the way that the two resulting subpopulations are disjoint and of equal size. The stopping criterion is set such that a run is terminated when all solutions in the population converge to the same fitness value. For each problem instance, the minimum required



(a) Population Sizes (b) Function Evaluations

Figure 4: Empirical results of the proposed method compared to the original ECGA on exponential scaled problems with tournament sizes $t = 12$ and $t = 16$. Problem sizes ranging from 40 to 80 bits ($k = 4$, $m = 10 \ldots 20$) were used to observe the performance of the algorithms.

population size is determined by a bisection method such that on average, $m - 1$ building blocks converge to the correct values in 50 runs.

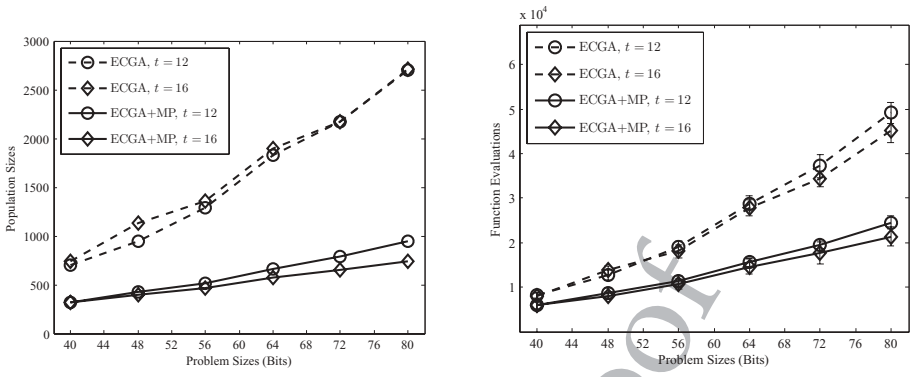
6.2.2 Results and Observations

The empirical results for exponentially scaled problems are shown in Figure 4. The minimum population sizes required by the proposed method are much smaller than the sizes needed by the original ECGA, and grow at a relatively slow rate. The same situation is also observed in the function evaluations for which our approach performed remarkably well. This improvement can be explained by the previous discussion on random drift and linkage sensibility presented in earlier sections. If simultaneous detection and processing of all building blocks cannot be achieved, additional costs have to be paid for the inaccurate processing and random drift of subsolutions. By adopting the pruning mechanism, we can save these costs by detecting possibly ineffective partial models and postponing the changes on them until accurate processing can be made.

Figure 5 shows the results for power law scaled problems. The results of the minimum population sizes are similar to those obtained in the previous set of experiments. The proposed method still uses fewer function evaluations, but the differences are reduced. This is because the linkage sensibility of the power law scaled problems is less limited compared to that of the exponential scaled problems.

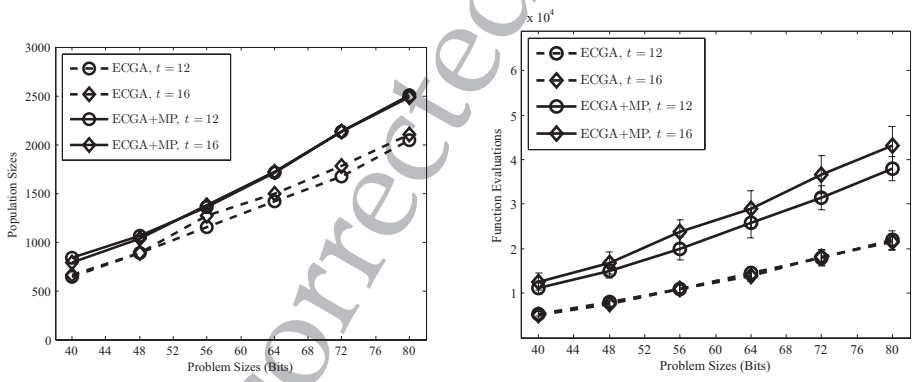
The empirical results for uniformly scaled problems are presented in Figure 6. As expected, the proposed method requires larger population sizes than which was needed by the original ECGA. Due to the fact that for uniformly scaled problems, the model building process can correctly identify all building blocks, the verification on the built model may just be useless and wasteful. The results also suggest that the function evaluations used by the proposed method are about twice as the number of what was needed by the original ECGA.

In order to support the significance of the observations, we have also performed Welch's t-test on the results. For each problem size, a t-test of the null hypothesis that the



(a) Population Sizes (b) Function Evaluations

Figure 5: Empirical results of the proposed method compared to the original ECGA on power law scaled problems with tournament sizes $t = 12$ and $t = 16$. Problem sizes ranging from 40 to 80 bits ($k = 4$, $m = 10 \ldots 20$) were used to observe the performance of the algorithms.



(a) Population Sizes (b) Function Evaluations

Figure 6: Empirical results of the proposed method compared to the original ECGA on uniformly scaled problems with tournament sizes $t = 12$ and $t = 16$. Problem sizes ranging from 40 to 80 bits ($k = 4$, $m = 10 \ldots 20$) were used to observe the performance of the algorithms.

number of function evaluations spent by ECGA and the number of function evaluations spent by the proposed approach are with equal means against the alternative that the means are not equal was performed. The significance level was set to 5%, and the respective statistics are listed in Table 6. The resulting statistics suggest that the outcomes of the proposed approach are significantly different from those of the original ECGA for all three scaling setups.

6.3 Impact on Population Requirement with Varying *k*

This section describes the experiments that accompany the previous ones to further demonstrate the performance of the proposed approach. The experiments were designed

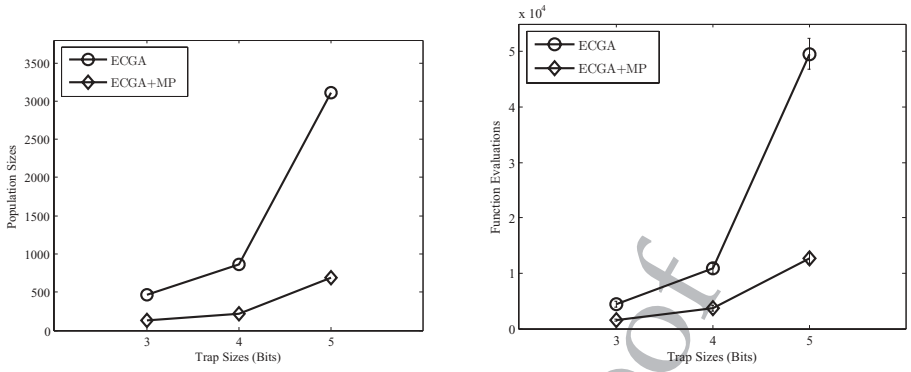
Table 6: Welch's t-test on empirical results presented in Figures 4, 5, and 6. The null hypothesis is that the number of function evaluations spent by ECGA and the number of function evaluations spent by ECGA-MP with equal means against the alternative that the means are not equal. The first three rows indicate whether the null hypothesis is rejected, the p-value, and the t-statistics from the tests, respectively. The last row lists whether the number of average function evaluations needed by ECGA-MP is smaller ($<$) or larger ($>$) than the number needed by the original ECGA.

Problem size	40	48	56	64	72	80
(a) Exponential scaled cases with tournament size 12						
Reject null	True	True	True	True	True	True
p-value	4.409×10^{-48}	2.137×10^{-60}	2.031×10^{-69}	1.102×10^{-56}	2.053×10^{-70}	5.563×10^{-63}
t-statistics	41.9194	64.0567	82.6409	66.3387	97.2660	82.4819
Comparison	$<$	$<$	$<$	$<$	$<$	$<$
(b) Exponential scaled cases with tournament size 16						
Reject null	True	True	True	True	True	True
p-value	1.458×10^{-48}	6.409×10^{-53}	2.149×10^{-54}	1.847×10^{-58}	4.341×10^{-60}	8.518×10^{-58}
t-statistics	40.7834	60.9651	71.7845	81.9204	89.1136	87.8952
Comparison	$<$	$<$	$<$	$<$	$<$	$<$
(c) Power law scaled cases with tournament size 12						
Reject null	True	True	True	True	True	True
p-value	1.094×10^{-25}	1.208×10^{-33}	5.515×10^{-48}	4.294×10^{-61}	6.05×10^{-53}	1.608×10^{-72}
t-statistics	14.5298	18.4542	29.4004	48.5933	44.0576	63.2243
Comparison	$<$	$<$	$<$	$<$	$<$	$<$
(d) Power law scaled cases with tournament size 16						
Reject null	True	True	True	True	True	True
p-value	1.582×10^{-22}	6.032×10^{-50}	1.047×10^{-47}	1.717×10^{-59}	3.383×10^{-56}	7.91×10^{-69}
t-statistics	12.7581	30.2641	28.5023	37.5145	38.8386	49.2693
Comparison	$<$	$<$	$<$	$<$	$<$	$<$
(e) Uniformly scaled cases with tournament size 12						
Reject null	True	True	True	True	True	True
p-value	3.356×10^{-35}	1.23×10^{-39}	4.264×10^{-33}	3.399×10^{-33}	4.006×10^{-45}	4.903×10^{-53}
t-statistic	–25.4683	–26.7928	–23.7365	–22.9905	–29.0505	–33.4524
Comparison	$>$	$>$	$>$	$>$	$>$	$>$
(f) Uniformly scaled cases with tournament size 16						
Reject null	True	True	True	True	True	True
p-value	1.126×10^{-34}	1.776×10^{-33}	8.802×10^{-40}	6.649×10^{-32}	3.684×10^{-38}	4.062×10^{-44}
t-statistic	–25.7066	–25.9356	–31.5460	–25.0263	–28.6037	–34.0405
Comparison	$>$	$>$	$>$	$>$	$>$	$>$

to observe the behavior of the proposed approach when the size of the constitutive subproblem changes (i.e., varying k while fixing m). As in the previous set of experiments, the original ECGA will also be tested for comparison.

6.3.1 Experimental Settings

In contrast to the previous set of experiments, we use trap functions of different sizes to form our test problems. While the size of constituting subproblem varies, the number of the subproblems remains fixed. The problem instances are constructed by concatenating



(a) Population Sizes (b) Function Evaluations

Figure 7: Empirical results of the proposed method compared to the original ECGA for exponential scaled problems composed of subproblems of sizes 3, 4, and 5 ($k = 3, 4$, and 5). In this experiment, tournament size $t = 16$ was used and the number of subfunctions forming the test problems was fixed at 10 (i.e., $m = 10$).

10 trap functions of size 3, 4, or 5 ($k = 3, 4$, or 5, $m = 10$). Tournament size $t = 16$ is used in this set of experiments.

As in the previous experiments, the splitting of the population is also performed so that the two resulting subpopulations are disjoint and of equal size. The stopping criterion is set such that a run is terminated when all solutions in the population converge to the same fitness value. For each problem instance, the minimum required population size is determined by a bisection method such that on average, $m - 1$ building blocks converge to the correct values in 50 runs.

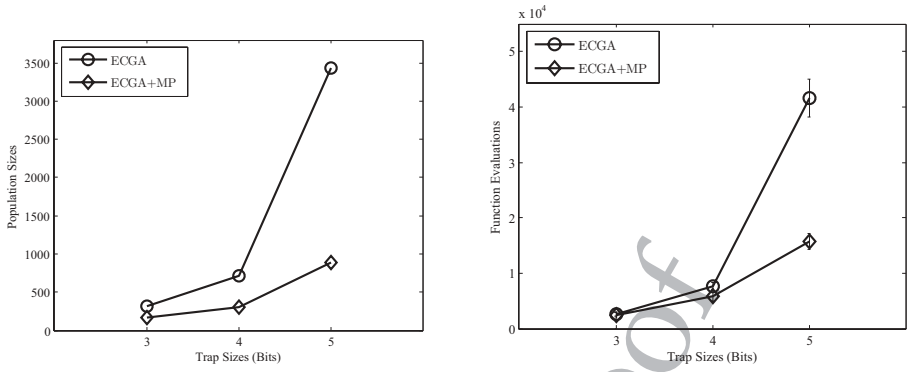
6.3.2 Results and Observations

The results for exponential and power law scaled problems are presented in Figures 7 and 8, respectively. It can be observed that for these three different subproblem sizes, the proposed approach uses smaller population sizes and fewer function evaluations to solve the test problems. Furthermore, the degree of improvement over the original ECGA seems to increase with the size of the constituting subproblems. As can be seen in the problems composed of 5-bit trap functions, the pruning mechanism achieves great savings in function evaluations compared to the original ECGA.

On the other hand, for the uniformly scaled problems, our approach still requires larger population sizes than what was needed by the original ECGA. This result is no surprise, as it can be conjectured that in solving uniformly scaled problems, the verification on the built model may be useless and wasteful. A further observation is that these results seem to be consistent with what we observed in the previous set of experiments in which the function evaluations used by the proposed method are about twice the number needed by the original ECGA.

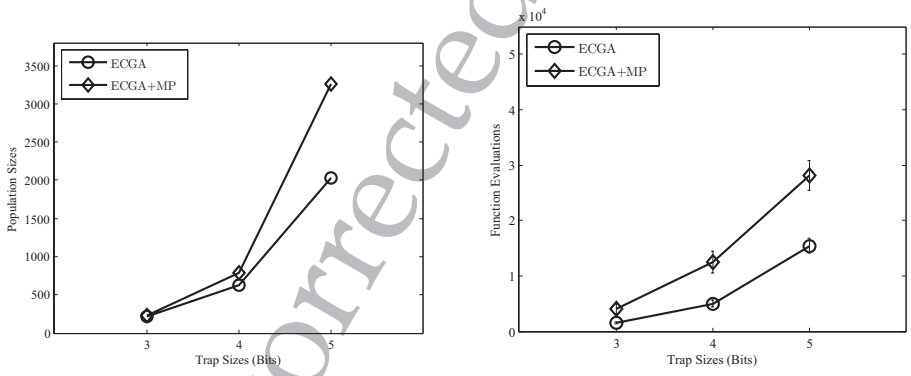
6.4 Building versus Verifying

This section describes the sets of experiments on the proposed method to reveal the change in performance when different splitting ratios of the two subpopulations are



(a) Population Sizes (b) Function Evaluations

Figure 8: Empirical results of the proposed method compared to the original ECGA for power law scaled problems composed of subproblems of sizes 3, 4, and 5 ($k = 3$, 4, and 5). In this experiment, tournament size $t = 16$ was used and the number of subfunctions forming the test problems was fixed at 10 (i.e., $m = 10$).



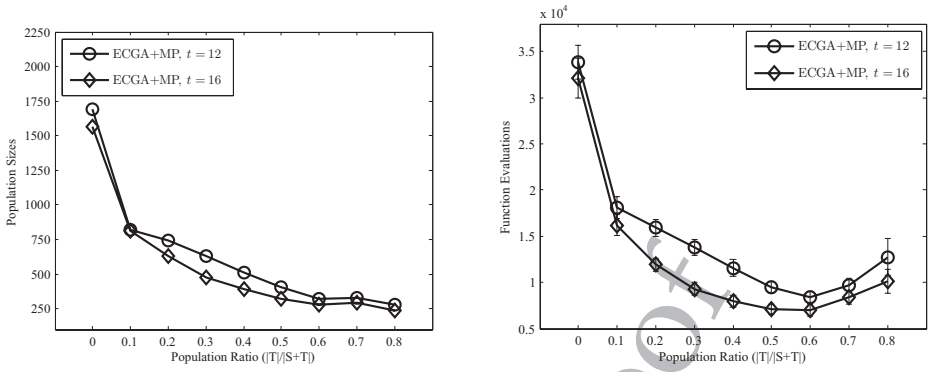
(a) Population Sizes (b) Function Evaluations

Figure 9: Empirical results of the proposed method compared to the original ECGA for uniformly scaled problems composed of subproblems of sizes 3, 4, and 5 ($k = 3$, 4, and 5). In this experiment, tournament size $t = 16$ was used and the number of subfunctions forming the test problems was fixed at 10 (i.e., $m = 10$).

adopted. It presents the experimental results to illustrate the behavior under different scalings. The purpose for performing these experiments is twofold:

- First, we would like to observe how the splitting ratio is related to the scaling or linkage sensibility of a problem.
- Second, we wish to empirically study the change in performance obtained from decreasing or increasing the proportion of population for checking the model.

It is important in practice to spend function evaluations wisely. Since using too large a proportion of the population for pruning may result in a waste of resources, it should



(a) Population Sizes (b) Function Evaluations

Figure 10: Empirical results of the proposed method for a 60-bit exponential scaled problem with different splitting ratios between the two subpopulations. The splitting ratio ($|T|/|S+T|$) ranging from 0.0 (ECGA without pruning) to 0.8 was used to observe the change in performance of the proposed approach.

be estimated to what degree the expense on checking the built model yields savings, and how the scaling of the problem is related to this matter.

6.4.1 Experimental Settings

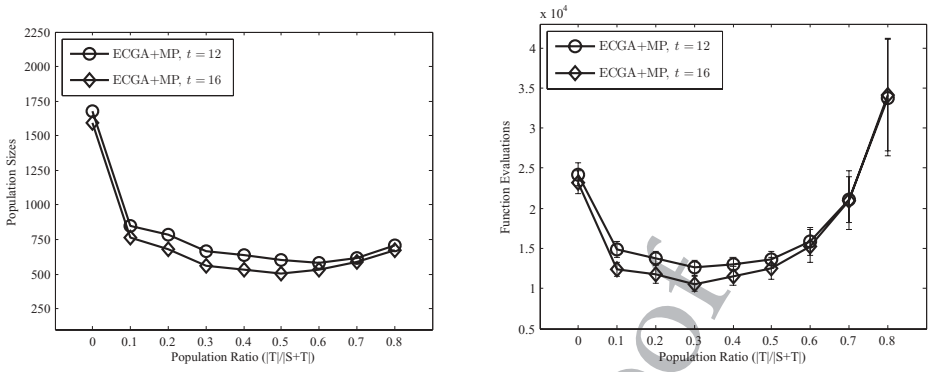
The problem instances used in this set of experiments were of 60 bits formed by concatenating 4-bit trap functions ($k = 4$, $m = 15$). The splitting ratio ($|T|/|S+T|$) ranged from 0.0 to 0.8. The ratio 0.0 represents the result of running the original ECGA (without pruning), which serves as a baseline. Two selection pressures were adopted by setting tournament size t to 12 and 16.

As in the previous experiments, the stopping criterion is set such that a run is terminated when all solutions converge to the same fitness value. For each splitting ratio, the minimum required population size was determined by a bisection method such that on average, $m - 1$ building blocks converge to the correct values in 50 runs.

6.4.2 Results and Observations

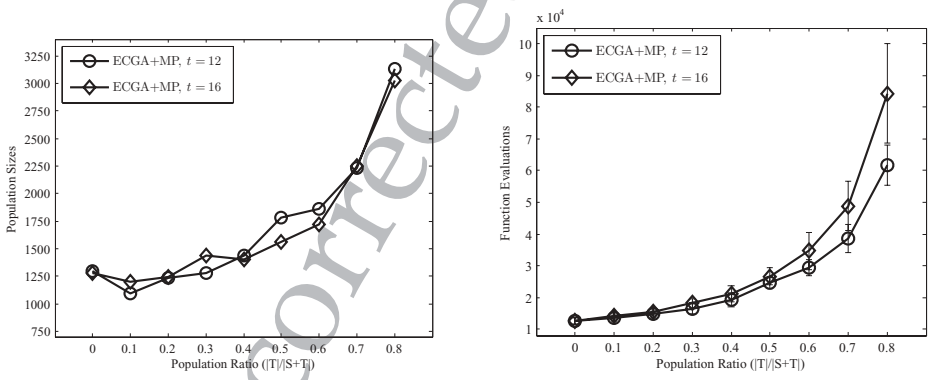
The empirical results for exponential scaled problems are presented in Figure 10. For both tournament sizes, the required population size decreases as the splitting ratio increases. However, the number of generations increases with the splitting ratio. The combined effect is that the minimum required function evaluation is obtained when the splitting ratio is 0.6, and the required function evaluation grows when the splitting ratio either increases or decreases.

Figure 11 shows the results for power law scaled problems. In contrast to the previous case, the required population size does not strictly decrease with the increment of the splitting ratio. The population size first decreases as the splitting ratio grows and then hits a turning point at 0.5 ($t = 16$) or 0.6 ($t = 12$). Similar to the exponential scaled case, the number of generations increases with the splitting ratio. The combined effect is that the number of function evaluations first decreases and then increases. For both tournament sizes, the minimum is obtained when the splitting ratio = 0.3.



(a) Population Sizes (b) Function Evaluations

Figure 11: Empirical results of the proposed method for a 60-bit power law scaled problem with different splitting ratios between the two subpopulations. The splitting ratio ($|T|/|S+T|$) ranging from 0.0 (ECGA without pruning) to 0.8 was used to observe the change in performance of the proposed approach.



(a) Population Sizes (b) Function Evaluations

Figure 12: Empirical results of the proposed method for a 60-bit uniformly scaled problem with different splitting ratios between the two subpopulations. The splitting ratio ($|T|/|S+T|$) ranging from 0.0 (ECGA without pruning) to 0.8 was used to observe the change in performance of the proposed approach.

Figure 12 shows the results for uniformly scaled problems. As expected, Figures 12(a) and 12(b) both share a common pattern in which the population size and the number of function evaluations increase with the splitting ratio. This is because in the uniformly scaled case, the linkage is always completely sensible, and there is no need to verify or prune the built probabilistic model.

These experimental results demonstrate that under different scaling setups, the behavior of the proposed approach corresponding to the splitting ratio varies differently. The empirical results suggest that if the given problem is evidently with distinguishable prominence among the constituting subproblems, using higher splitting ratios will yield

better performance. Lower ratios are more suitable if the problem at hand is composed of subproblems with roughly equal salience.

Another insight provided by this set of experiments is that reducing the size of the proportion of population spent on the pruning mechanism can considerably improve the performance. As shown in Figures 10(b) and 11(b), compared to the original ECGA (splitting ratio = 0.0 in the figures), significant performance gain can be obtained by using a mere 10% of the population to validate the built model. On the other hand, Figure 12(b) also demonstrates that using this small percentage of population on the pruning mechanism will not bring serious overhead for the overall performance.

6.5 Splitting Ratio versus Subproblem Size

This section describes the experiments extending the previous set of experiments for observing the interaction between the splitting ratio and the performance. The focus of this set of experiments is to study the effect of different splitting ratios when the size of the constitutive subproblem changes (i.e., varying k while fixing m). Our main purpose is to see whether the result of adopting a particular splitting ratio changes significantly when the complexity of the problem varies. Furthermore, we want to empirically examine whether or not the improvement of using just 10% of the population to validate the built model is still prominent for different sizes of the constitutive subproblems.

6.5.1 Experimental Settings

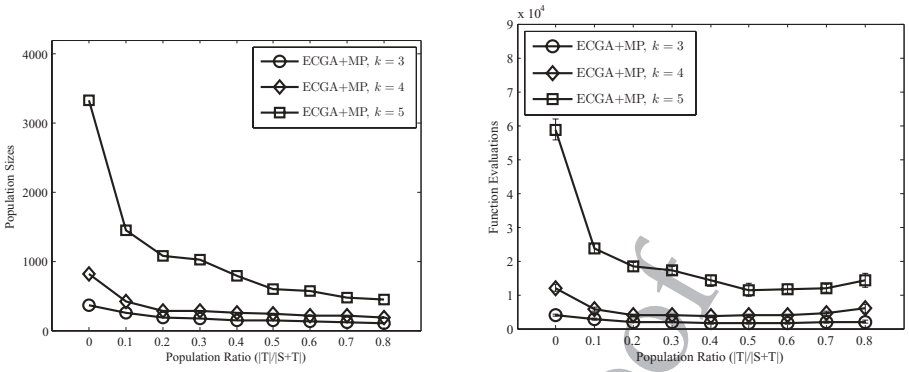
In this set of experiments, we use trap functions of different sizes to construct our test problems. While the size of constitutive subproblems varies, the number of the subproblems forming the test problems remains the same. The problem instances are built by concatenating 10 trap functions of sizes 3, 4, or 5 ($k = 3$, 4, or 5, $m = 10$). Tournament size $t = 16$ is adopted in this set of experiments.

As in the previous set of experiments, the splitting ratio ($|T|/|S + T|$) ranges from 0.0 to 0.8. The ratio 0.0 represents the result of running the original ECGA (without pruning), which serves as a baseline. The stopping criterion is set such that a run is terminated when all solutions in the population converge to the same fitness value. For each splitting ratio, the minimum required population size was determined by a bisection method such that on average, $m - 1$ building blocks converge to the correct values in 50 independent runs.

6.5.2 Results and Observations

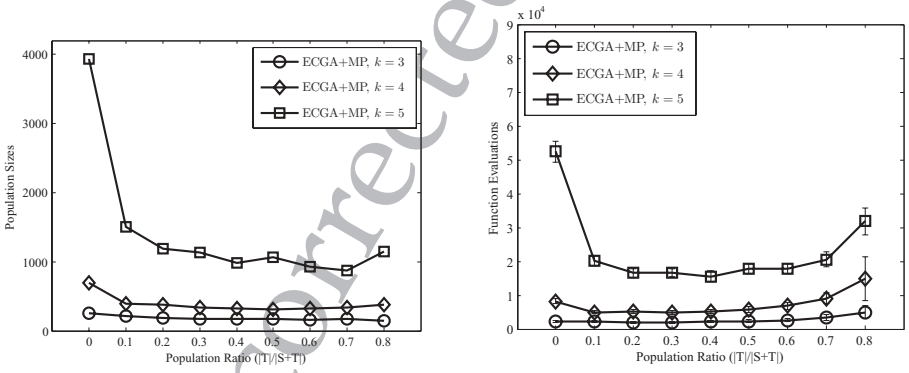
The results for exponential, power law, and uniformly scaled problems are presented in Figures 13, 14, and 15, respectively. We can see that the result of adopting a particular splitting ratios does not change significantly relative to other splitting ratios for all three subproblem sizes. It can also be observed that several kinds of behavior similar to what we have seen in the previous experiments are presented in these results. For the uniformly scaled problems, the results presented in Figure 15(b) shows a similar pattern to what is observed in the previous set of experiments in which the number of function evaluations increased with the splitting ratio. In addition, similar to the results from the previous set of experiments, we can see that using a small percentage (10%) of population on the pruning mechanism does not bring serious overhead to the overall performance for all three subproblem sizes.

On the other hand, for exponential and power law scaled problems, the greatest improvements are obtained when using 10% of the population to validate the built



(a) Population Sizes (b) Function Evaluations

Figure 13: Empirical results of the proposed method using different splitting ratios ($|T|/|S+T|$) for exponential scaled problems composed of subproblems of sizes 3, 4, or 5 ($k = 3$, 4, or 5). In this experiment, tournament size $t = 16$ was used and the number of subfunctions forming the test problems was fixed at 10 (i.e., $m = 10$)



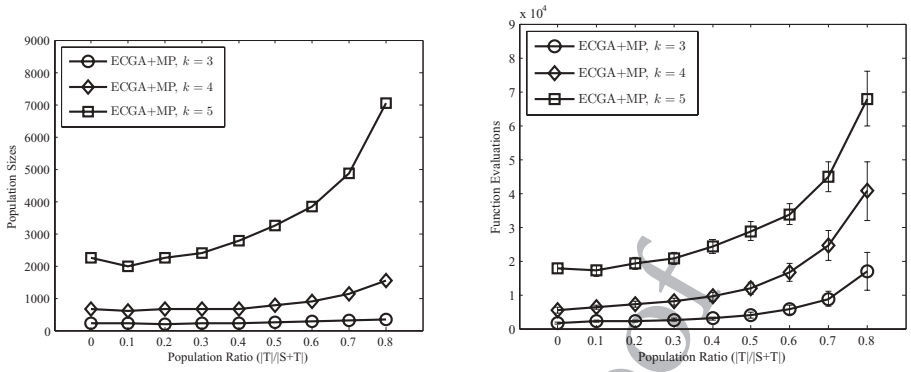
(a) Population Sizes (b) Function Evaluations

Figure 14: Empirical results of the proposed method using different splitting ratios ($|T|/|S+T|$) for power law scaled problems composed of subproblems of sizes 3, 4, or 5 ($k = 3$, 4, or 5). In this experiment, tournament size $t = 16$ was used and the number of subfunctions forming the test problems was fixed at 10 (i.e., $m = 10$)

model. Furthermore, similar to what we have observed in the experiments described in Section 6.3, the degree of improvement over the original ECGA (splitting ratio = 0.0) increases with the size of the constitutive subproblem.

7 Discussion

We utilized the existence of disparate scales in problems to create a controlled experimental environment in order to study the situation in which complete, accurate linkage information may or may not be available for the estimation of distribution algorithms.



(a) Population Sizes (b) Function Evaluations

Figure 15: Empirical results of the proposed method using different splitting ratios ($|T|/|S+T|$) for uniformly scaled problems composed of subproblems of sizes 3, 4, or 5 ($k = 3, 4$, and 5). In this experiment, tournament size $t = 16$ was used and the number of subfunctions forming the test problems was fixed at 10 (i.e., $m = 10$)

According to the obtained results shown in Figures 4(b) and 5(b), the proposed approach does improve the original ECGA for the test problems where disparate scales exist among building blocks. In this section, we discuss some interesting aspects of the proposed approach and possible extensions of this work.

7.1 Overhead in Uniformly Scaled Problems

The empirical results presented in Figure 6(b) show that for the uniformly scaled cases, the proposed approach uses nearly twice as many function evaluations as the original ECGA does. We speculate that this double expenditure is a general property of the proposed approach when dealing with uniformly scaled problems.

This speculation can be explained through a reverse thinking on a hypothetical situation described as follows. Suppose that given a uniformly scaled problem, the original ECGA with appropriate selection pressure needs a population of size n to handle that problem properly. Now, consider adopting the proposed approach to handle the same problem. If we use a population of size $2n$, then in our algorithm, the entire population will be divided into two subpopulations of size n, assuming that the splitting of population is disjoint and of equal size. If the original ECGA is capable of detecting the accurate problem structure with a population of size n, then in our algorithm, a subpopulation of size n will also do the job. In the ideal case, there will be no statistical inconsistency between the built model and the set of promising solutions selected from the second subpopulation. As a result, we waste half of the population for the use of pruning which causes the extra cost compared to the original ECGA.

In order to support the inference, we performed an experiment based on the scenario just described. Table 7 lists some of the empirical results obtained from the experiments described in Section 6.2. This table shows that for 40-bit and 80-bit uniformly scaled problems formed by concatenating 4-bit trap functions, the original ECGA needs populations of sizes 646 and 2,042, respectively, to solve the given problem. Based on these results, we used population sizes that are twice that to run our approach. The results are

Table 7: Empirical results of the original ECGA using tournament size 12. Experiments were conducted on 40-bit and 80-bit uniformly scaled problems formed by concatenating 4-bit trap functions. The symbols ℓ, n, g, and f_{ev} denote problem size, population size, generation, and function evaluations, respectively.

	ℓ	n	g	$SD\ g$	f_{ev}	$SD\ f_{ev}$
ECGA	40	646	8.36	0.92	5,400.56	594.65
	80	2042	10.72	1.01	21,890.24	2,064.38

Table 8: Empirical results of the proposed approach using a tournament size of 12. Experiments were conducted on 40-bit and 80-bit uniformly scaled problems formed by concatenating 4-bit trap functions. The symbols ℓ, $2n$, g, and f_{ev} denote problem size, twice of the population size required by the original ECGA, generation, and function evaluations, respectively.

	ℓ	$2n$	g	$SD\ g$	f_{ev}	$SD\ f_{ev}$
ECGA+MP	40	1,292	9.24	0.89	11,938.08	1,154.42
	80	4,084	10.58	0.70	43,208.72	2,868.90

listed in Table 8. It can be observed that the function evaluations spent by the proposed approach for 40-bit and 80-bit problems are about twice the amount of the original ECGA needed in each case.

Although the inference together with the empirical validation can serve as an intuitive explanation, it cannot fully explain the results presented in Section 6.2. As illustrated in Figure 6(a), the minimum population sizes needed by the proposed method is not exactly twice that required by the original ECGA. In fact, the numbers are much lower than twice what is needed by the original ECGA. On the other hand, our approach uses more generations compared to the original ECGA because the subpopulation for model building was not sufficiently large for all problem structures to be detected properly in the beginning of the process. In this situation, the processing was slowed down because the pruning mechanism removed certain parts of the model exhibiting statistical inconsistencies. As a consequence, the originally expected simultaneous processing of building blocks was not fully achieved and delay of convergence occurred. Nevertheless, spending more generations seems to yield an equivalent use of function evaluations as the hypothetical case described above. We think that the pruning mechanism introduces an additional interaction between population size and generations. Further empirical or theoretical studies are needed to investigate such an interaction.

7.2 A Deeper Look at the Pruning Criterion

This section provides a more detailed elaboration on the adequacy of the proposed pruning metric. To start this discussion, let

$$\lambda_i = \sum_{j=1}^{2^{k_i}} q_{ij}(-\log_2 p_{ij})$$

which is the quantity to be examined by the pruning criterion (i.e., whether $\lambda_i \geq k_i$). Based on λ_i, we can reformulate the issue of adequacy more concisely as "is it possible

that p_i is not effective but $\lambda_i \leq k_i$ or p_i is effective but $\lambda_i \geq k_i$?" To elaborate on this, we have to separate the discussion into two cases:

1. p_i is not effective and $\lambda_i \leq k_i$, and
2. p_i is effective and $\lambda_i \geq k_i$.

For the first case, if p_i is not effective and its ineffectiveness is caused by drift, it is possible that $\lambda_i \leq k_i$ if the set of solutions on which q_i is estimated also drifts in the same direction. However, by its nature, drift is random, and two different sets of solutions tend to drift in two different directions. Thus, we can expect the chances of this situation to be small and our empirical results also support this conjecture.

For the second case, if p_i is effective, that is, it provides a good direction for further search, then the (sub)solutions on which p_i is estimated must be subjected to proper selection pressure. Based on the premise that these two sets of solutions are produced under the same conditions, we can expect that the (sub)solutions on which q_i is estimated should also be subjected to the same pressure. In this case, if these two sets of solutions are produced under the same conditions and the selection pressure is properly applied (i.e., no drifting), it would be unreasonable to see inconsistencies between p_i and q_i (i.e., $\lambda_i \geq k_i$.) However, the above discussion is based on the assumption that the population size is sufficiently large. If the population size if not sufficiently large, inconsistencies tend to be observed because there are too few samples to reveal the true statistical property.

Using the above discussion, we can further analyze what would happen if we use more than one set of solutions to prune the built model. This kind of techniques is used frequently in machine learning research to assess the performance of a learning algorithm, in which multiple reserved subsets of testing instances are examined. Extending from the above discussion, let P be the probability of the above case 1 (p_i is not effective and $\lambda_i \leq k_i$) and use r sets of solutions for validating the built model. Then the probability that we cannot detect the ineffectiveness of a marginal model will be P^r, for it is tested independently on r different sets, which is smaller than the probability of using only one set of solutions (i.e., P). However, in this paper, we focused on the baseline behavior of the proposed approach, since we know that employing a larger r should yield better performance and should also incur higher costs.

7.3 Pruning Network-Based Probabilistic Models

In this work, we have introduced a technique to prune a given marginal product model based on the statistics collected from a reserved set of solutions. It is possible to extend the fundamental idea and concept to design pruning mechanisms for other EDAs. For example, consider the EDAs that use network-based probabilistic models with the Bayesian information criterion (BIC; Schwarz, 1978) as the model scoring metrics, such as EBNA (Etxeberria and Larrañaga, 1999) and a variant of BOA (Pelikan et al., 2001). In the binary case, BIC assigns a given network structure B of ℓ variables a score

$$\begin{aligned} S(B) &= \sum_{i=1}^{\ell} \left(-n \times H(X_i|\Pi_i) - 2^{|\Pi_i|} \frac{\log_2 n}{2} \right) \\ &= -\sum_{i=1}^{\ell} n \times H(X_i|\Pi_i) - \sum_{i=1}^{\ell} 2^{|\Pi_i|} \frac{\log_2 n}{2} \ , \end{aligned}$$

where X_i, $i = 1 \ldots \ell$, are variables, $H(X_i|\Pi_i)$ is the conditional entropy of X_i given its parent Π_i in the network, and n is the population size. The conditional entropy $H(X_i|\Pi_i)$ is given by

$$H(X_i|\Pi_i) = -\sum_{x_i,\pi_i} p(x_i,\pi_i)\log_2 p(x_i|\pi_i),$$

where $p(x_i,\pi_i)$ is the probability of instances with $X_i = x_i$, $\Pi_i = \pi_i$, and $p(x_i|\pi_i)$ is the conditional probability of instances with $X_i = x_i$ given that $\Pi_i = \pi_i$.

The term $\sum_{i=1}^{\ell} n \times H(X_i|\Pi_i)$ provides the same functionality as the compressed population complexity (C_p) in ECGA because $H(X_i|\Pi_i)$ denotes the average number of bits required to store a value of X_i with compression given the information of Π_i. Thus, we can check whether or not variable X_i should be pruned away by using the following inequality

$$-\sum_{x_i,\pi_i} q(x_i,\pi_i)\log_2 p(x_i|\pi_i) > 1,$$

where $q(x_i,\pi_i)$ is the frequency of $X_i = x_i$, and $\Pi_i = \pi_i$ is observed in the set of solutions selected from the reserved subpopulation. Using the idea described in Section 5.2, if this inequality holds, X_i should be removed because it encodes a one-bit partial solution to a bit string with an expected length of more than one bit.

However, despite the similarities in ideas, some technical complications remain to be overcome before we can finish the design of a pruning mechanism for network-based probabilistic models. For instance, what if a variable which we intend to prune is a parent node of some other variables? In summary, pruning network-based probabilistic models is potentially feasible, but requires further investigation.

8 Summary and Conclusions

This paper reviewed previous studies on EDAs and scaling difficulties. It then illustrated how the scaling difficulty shadows the EDA ability in recognizing building blocks. Following that, a notion called *linkage sensibility* was introduced to describe the observation, and we used the term *sensible linkage* to refer to the problem structures that can be extracted by inspecting only the set of selected solutions. Based on this concept, we briefly defined the effectiveness of distributions estimated by probabilistic model building and proposed a general approach to achieve more effective modeling. Finally, an implementation of the proposed approach on ECGA was introduced and experiments were done using several test functions of different scaling difficulties. In this section, we briefly summarize the major results derived from this work and outline the possible future extensions of this research.

8.1 Contributions

In this work, we have shown that the underlying facilities for EDAs to solve problems efficiently and reliably do not work as expected when the problem at hand is composed of subparts of unequal fitness contributions. More specifically, under this situation, the model built from the selected solutions cannot fully reflect the true problem structures. Although there are previous studies and discussion on the parameter selection (Pelikan et al., 2002; Lima and Lobo, 2004; Pelikan and Lin, 2004; Yu et al., 2007), selection

mechanisms (Lima et al., 2007, 2008), and model building algorithms (Echegoyen et al., 2007) related to the model accuracy, we consider the conditions discussed in this paper to be more fundamental and closer to the problem nature than those other factors. This is because in our discussion, the condition that suppresses modeling accuracy is embedded in the problem inherently. For some situations, we can reasonably fine-tune algorithmic parameters or select between alternative model building approaches, however, in general we do not have a way to remove a property (e.g., scaling) that exists inherently within the problem to improve modeling accuracy.

Alongside the modeling inaccuracy is the phenomenon of random drift. In a finite population, the selection process can cause convergence to some subsolutions for reasons other than the fitness contribution of these subsolutions. The converged subsolutions might be hitchhikers that appear with other high quality building blocks in selected solutions, or just a result of stochastic errors of sampling due to small population accumulated over generations. As demonstrated in the earlier sections with problems having disparate scalings among subparts, a problem property (e.g., scaling) can cause drift in population as well as making some parts of the problem structure undetectable to the model building process. This situation is usually resolved by increasing the population size to maintain diversity in response to the possible drift. In contrast, our approach handles this situation by relating these two co-occurring events and by using a pruning mechanism to avoid building models on, and sampling from, the possible drift portions. In this way, we effectively save the cost that we originally have to pay for the maintenance of diversity by using larger populations.

Empirical results show that our approach improves the original ECGA in cases where disparate scales exist among constitutive subproblems and in the uniformly scaled problems (i.e., all the constitutive subproblems have the same fitness contribution), the overhead of using the proposed pruning mechanism is about the amount of function evaluations spent by the original ECGA. The experimental results further suggest that this constant overhead in uniformly scaled cases is not affected by the size of the subfunction (i.e., k) forming the problem, and the improvement in nonuniformly scaled cases seems to increase with the size of the problem. Moreover, we also demonstrated through experiments that in the nonuniformly scaled cases, a small proportion (10%) of population spent on the pruning mechanism can greatly reduce the amount of required function evaluations compared to that spent by the ECGA without pruning.

The experiments with different scaling setups also led to another consideration that whether uniformly or near-uniformly scaled problems adopted by many previous studies are suitable to fully test the performance of an algorithm designed for solving black box optimization problems. In our humble opinion, presuming a black box optimization problem to be handled that is uniformly scaled is too strong an assumption, because there will be no information to confirm this assumption prior to the application of the algorithm. Thus, we believe that in order to generalize beyond the assumption that all subproblems are uniformly scaled, the constant-time overhead for solving the uniformly scaled cases is a reasonable tradeoff.

In addition, several efficiency enhancement techniques for EDAs (Sastry and Goldberg, 2004; Sastry et al., 2004, 2005, 2006; Lima et al., 2005, 2006) rely on the structure information delivered from the probabilistic models. Their good functioning crucially depends on the structural accuracy of the built models. Thus, it is conceivable that if the built model does not properly capture the true structure of the underlying problem, the model-based enhancement mechanism will not fully work as expected. Furthermore, as we demonstrated in this paper, the condition that hinders the model building

algorithm from constructing models that truly reflect the problem structure may be an inherent property of the underlying problem (e.g., different scales among constitutive subproblems). Thus, we think that adapting pruning mechanisms will provide a more appropriate circumstance for the model-based enhancement techniques to work.

8.2 Future Work

In this paper, we demonstrated a pruning mechanism design and its integration into ECGA. It may also serve as a basis for developing other techniques for more efficient and robust optimization. Some possible extensions of this work are outlined as follows.

First of all, the immediate direction is to design pruning mechanisms for other EDAs. As illustrated in Section 7.3, we can extend the pruning metric described in this paper to handle network-based models with a Bayesian information criterion. However, a pruning mechanism for network-based models requires more than that. We also need to consider the possible disruption of variable dependencies after pruning a particular variable. The simplest solution is to consider only those variables that are not depended upon by other variables as possible candidates for pruning. However, the validity of such an approach requires further investigation. A more promising yet more sophisticated approach is to first identify the tightly related components (e.g. cliques or strongly connected subgraphs) in the model, and then process each component as a unit which is similar to how we process the marginal product models in this work.

Another direction for future research is to assist efficiency enhancement techniques that use the information contained in the built model. As described previously in Section 8.1, some model-based efficiency enhancement techniques for EDAs crucially rely on the structural accuracy of the probabilistic models. However, most of those studies implicitly assume the information contained in the given population is sufficient for learning accurate model structures. As demonstrated in the previous sections by nonuniformly scaled problems, this assumption does not always hold. From this perspective, incorporating pruning mechanisms to preprocess the built model for these enhancement techniques is a promising direction for designing more robust approaches.

From an abstract point of view, this work also demonstrates an instance of a new class of techniques operating on built models to control, adapt, or regulate the optimization process. Another example based on this viewpoint is the termination criterion proposed by Ocenasek (2006) which uses an entropy-based measurement to evaluate the built model for detecting an appropriate stopping point. According to the information collected in the model, we can gain better control over the process compared to the conventional evolutionary algorithms. Such an idea may be carried over to other designs of EDAs so that more robust and efficient optimization can be realized.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Baluja, S., and Davies, S. (1997). Using optimal dependency-trees for combinational optimization. In *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*, pp. 30–38.

Bosman, P. A. N., and Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, Vol. 1, pp. 60–67.

Chen, Y.-p., and Goldberg, D. E. (2005). Convergence time for the linkage learning genetic algorithm. *Evolutionary Computation*, 13(3):279–302.

Cover, T. M., and Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley-Interscience.

De Bonet, J., Isbell, C., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems*, Vol. 9, pp. 424–430.

Deb, K., and Goldberg, D. E. (1993). Analyzing deception in trap functions. In *Foundations of Genetic Algorithms 2*, pp. 93–108.

Deb, K., and Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10(4):385–408.

Echegoyen, C., Lozano, J. A., Santana, R., and Larrañaga, P. (2007). Exact Bayesian network learning in estimation of distribution algorithms. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 1051–1058.

Etxeberria, R., and Larrañaga, P. (1999). Global optimization using Bayesian networks. In A. O. Rodriguez, M. S. Ortiz, and R. S. Hermida (Eds.), *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pp. 332–339.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6(4):333–362.

Goldberg, D. E., Deb, K., and Korb, B. (1990). Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4(4):415–444.

Goldberg, D. E., and Rudnick, M. (1991). Genetic algorithms and the variance of fitness. *Complex Systems*, 5(3):265–278.

Grünwald, P. (2007). *The minimum description length principle*. Cambridge, MA: MIT Press.

Harik, G. R. (1997). *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. PhD thesis, University of Michigan, Ann Arbor.

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

Harik, G. R., Lobo, F. G., and Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297.

Hauschild, M., Pelikan, M., Lima, C. F., and Sastry, K. (2007). Analyzing probabilistic models in hierarchical BOA on traps and spin glasses. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2007)*, pp. 523–530.

Hauschild, M. W., Pelikan, M., Sastry, K., and Goldberg, D. E. (2008). Using previous models to bias structural learning in the hierarchical BOA. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2008)*, pp. 415–422.

Holland, J. H. (1992). *Adaptation in natural and artificial systems*. Cambridge, MA: MIT Press.

Larrañaga, P., and Lozano, J. A. (2001). *Estimation of distribution algorithms: A new tool for evolutionary computation*, Vol. 2 of *Genetic Algorithms and Evolutionary Computation*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Lima, C. F., and Lobo, F. G. (2004). Parameter-less optimization with the extended compact genetic algorithm and iterated local search. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pp. 1328–1339.

Lima, C. F., Lobo, F. G., and Pelikan, M. (2008). From mating pool distributions to model overfitting. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2008)*, pp. 431–438.

Lima, C. F., Pelikan, M., Goldberg, D. E., Lobo, F. G., Sastry, K., and Hauschild, M. (2007). Influence of selection and replacement strategies on linkage learning in BOA. In *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 1083–1090.

Lima, C. F., Pelikan, M., Sastry, K., Butz, M. V., Goldberg, D. E., and Lobo, F. G. (2006). Substructural neighborhoods for local search in the Bayesian optimization algorithm. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)*, pp. 232–241.

Lima, C. F., Sastry, K., Goldberg, D. E., and Lobo, F. G. (2005). Combining competent crossover and mutation operators: A probabilistic model building approach. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2005)*, pp. 735–742.

Lobo, F. G., Goldberg, D. E., and Pelikan, M. (2000). Time complexity of genetic algorithms on exponentially scaled problems. In D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pp. 151–158.

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill Higher Education.

Mühlenbein, H., and Höns, R. (2005). The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27.

Mühlenbein, H., and Mahnig, T. (1999). FDA: A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376.

Mühlenbein, H., and Paaß, G. (1996). From recombination of genes to the estimation of distributions. I. Binary parameters. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN IV)*, pp. 178–187.

Ocenasek, J. (2006). Entropy-based convergence measurement in discrete estimation of distribution algorithms. In J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea (Eds.), *Towards a new evolutionary computation. Advances in estimation of distribution algorithms*, Vol. 192 of *Studies in Fuzziness and Soft Computing* (pp. 39–50). Berlin: Springer.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, pp. 525–532.

Pelikan, M., Goldberg, D. E., and Lobo, F. G. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20.

Pelikan, M., Goldberg, D. E., and Sastry, K. (2001). Bayesian optimization algorithm, decision graphs, and Occam's razor. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 519–526.

Pelikan, M., and Lin, T.-K. (2004). Parameter-less hierarchical BOA. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pp. 24–35.

Pelikan, M., and Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry (Eds.), *Advances in soft somputing—Engineering design and manufacturing* (pp. 521–535). Berlin: Springer.

Pelikan, M., Sastry, K., and Goldberg, D. E. (2002). Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica*, 14:465–471.

Sastry, K. (2001). Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign. Also IlliGAL Report No. 2002004.

Sastry, K., Abbass, H. A., Goldberg, D. E., and Johnson, D. D. (2005). Sub-structural niching in estimation of distribution algorithms. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2005)*, pp. 671–678.

Sastry, K., and Goldberg, D. E. (2004). Designing competent mutation operators via probabilistic model building of neighborhoods. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, pp. 114–125.

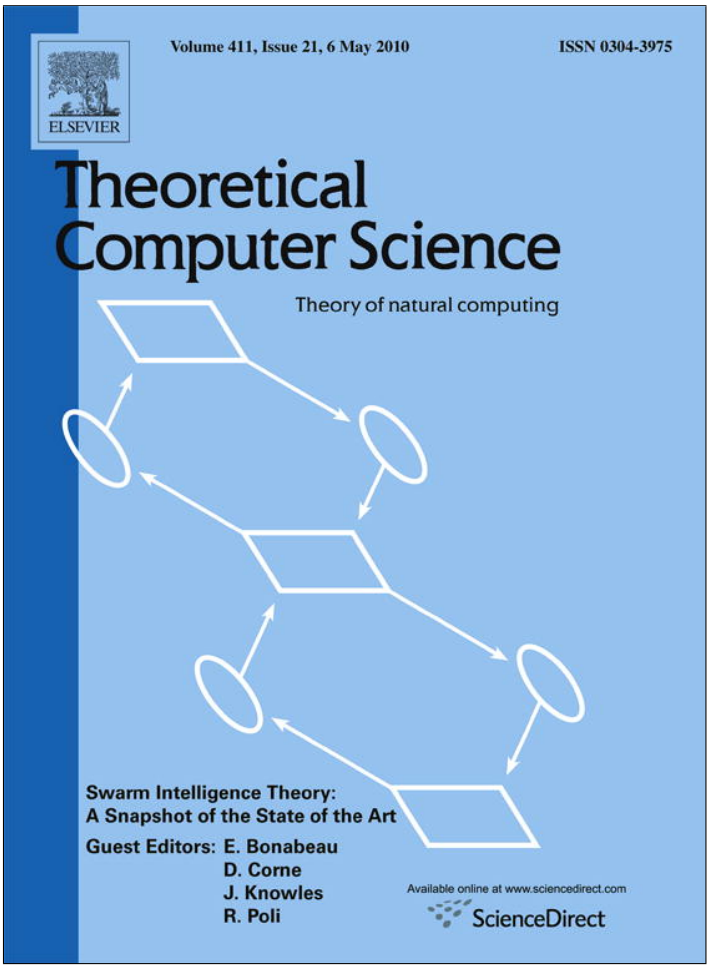
Sastry, K., Lima, C. F., and Goldberg, D. E. (2006). Evaluation relaxation using substructural information and linear estimation. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2006)*, pp. 419–426.

Sastry, K., Pelikan, M., and Goldberg, D. E. (2004). Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC 2004)*, pp. 720–727.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Thierens, D., Goldberg, D. E., and Pereira, Â. G. (1998). Domino convergence, drift and the temporal salience structure of problems. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 535–540.

Yu, T.-L., Sastry, K., Goldberg, D. E., and Pelikan, M. (2007). Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference (GECCO-2007)*, pp. 601–608.



Theoretical Computer Science 411 (2010) 2101–2115

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Analysis of particle interaction in particle swarm optimization

Ying-ping Chen *, Pei Jiang

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

ARTICLE INFO

Keywords:
Particle swarm optimization
Particle interaction
Social-only model
Statistical interpretation
Theoretical analysis
Progress rate
Convergence
Sphere function

ABSTRACT

In this paper, we analyze the behavior of particle swarm optimization (PSO) on the facet of particle interaction. We firstly propose a statistical interpretation of particle swarm optimization in order to capture the stochastic behavior of the entire swarm. Based on the statistical interpretation, we investigate the effect of particle interaction by focusing on the social-only model and derive the upper and lower bounds of the expected particle norm. Accordingly, the lower and upper bounds of the expected progress rate on the sphere function are also obtained. Furthermore, the sufficient and necessary condition for the swarm to converge is derived to demonstrate the PSO convergence caused by the effect of particle interaction.

1. Introduction

Particle swarm optimization (PSO), introduced by Kennedy and Eberhart [1] in 1995, was proposed based on an inspiration from the social behavior of insects or animals that the exchanging and sharing of information among a group of individuals benefit the group survival by improving the group capability of foraging. In the framework of PSO, the insects or animals are considered as *particles* flying through the multi-dimensional search space and searching for the optimal position. The movement of particles is affected by three factors: the inertia, personal experience (the cognitive part), and particle interaction (the social part).

Since its introduction, PSO has been empirically shown to be a very useful and effective optimization framework [2] for the easiness to implement and flexibility to use. Although PSO is widely applied in many research fields nowadays, the theoretical analysis on PSO is still quite limited. To the best of our knowledge, the first analysis was proposed by Kennedy [3]. Particle trajectories for design choices were shown. Ozcan and Mohan [4,5] assumed fixed attractors and constant coefficients to demonstrate the particle trajectory as a sinusoidal wave. With similar assumptions, Maurice and Kennedy [6] simplified PSO to a deterministic dynamical system and analyzed its stability. Such simplified, deterministic versions of PSO or similar systems, employing a single particle, fixed attractors, or constant coefficients, were analyzed by many researchers for stability, convergence, and parameter selection [7–11]. Kadirkamanathan et al. [12] and Jian et al. [13] started to consider the randomness in acceleration coefficients, but attractors were still fixed. Away from the common PSO configuration, Emara and Fattah [14] as well as Gazi and Passino [15] analyzed PSO in a continuous time setting.

Most of the existing studies do not provide analysis on the facet of particle interaction, which is definitely an essential mechanism of PSO. In this paper, under more practical assumptions, including multiple particles, unfixed attractors, and stochastic acceleration coefficients, we make the first attempt to analyze the effect of particle interaction. In particular, we consider the PSO system from a macrostate viewpoint, analyze the swarm behavior, and obtain theoretical results on the progress rate as well as the convergence criterion.

The paper is organized as follows. In Section 2, we will describe the particle swarm optimization algorithm and propose the statistical interpretation. In Section 3, we will analyze the mean positions of particles by considering the effect of particle

* Corresponding author.
E-mail addresses: ypchen@nclab.tw (Y.-p. Chen), pjiang@nclab.tw (P. Jiang).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.03.003

interaction and derive the expected progress rate of the swarm on the sphere function. Next, we will look into the variance of the particle positions and show that the swarm will converge under certain condition in Section 4. Finally, Section 5 summarizes and concludes this paper.

2. PSO and particle interaction

In this section, we will firstly describe the standard PSO algorithm and then discuss the operations of PSO step by step, followed by the proposal of our statistical interpretation.

2.1. The standard PSO algorithm

First of all, for easily making an abstraction of PSO based on statistics and probabilistic distributions, we restate the standard PSO system as the following algorithm:

Algorithm 1 (*Standard PSO*).

procedure STANDARD PSO(Objective function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$)
  Initialize a swarm of m particles
  while the stopping criterion is not satisfied **do**
    Evaluate each particle
    for particle $i, i = 1, 2, \ldots, m$ **do** ▷ Update the best positions
      if $\mathcal{F}(\mathbf{X_i}) < \mathcal{F}(\mathbf{Pb_i})$ **then**
        $\mathbf{Pb_i} \leftarrow \mathbf{X_i}$
        if $\mathcal{F}(\mathbf{Pb_i}) < \mathcal{F}(\mathbf{Nb})$ **then**
          $\mathbf{Nb} \leftarrow \mathbf{Pb_i}$
        end if
      end if
    end for
    for particle $i, i = 1, 2, \ldots, m$ **do** ▷ Generate the next generation
      $\mathbf{V_i(t+1)} \leftarrow w\mathbf{V_i(t)} + \mathbf{C_p} \otimes (\mathbf{Pb_i} - \mathbf{X_i}) + \mathbf{C_n} \otimes (\mathbf{Nb} - \mathbf{X_i})$
      $\mathbf{X_i(t+1)} \leftarrow \mathbf{X_i(t)} + \mathbf{V_i(t+1)}$
    end for
  end while
end procedure

Throughout this paper, boldface is used to distinguish vectors from scalars, and $\|\cdot\|$ denotes the L^2 norm of a vector. The notation $\otimes$ indicates component-by-component multiplication. According to Algorithm 1, we can see that a standard PSO system comprises the following two main operations regarding the information sharing and utilizing:

(1) Updating attractors: Update the personal best position, $\mathbf{Pb_i}$, found by each particle, and the neighborhood best position, $\mathbf{Nb}$, found by any member within the neighborhood. Since $\mathbf{Pb_i}$ and $\mathbf{Nb}$ exert gravity on other particles, they are referred to as *attractors* in this study.
(2) Updating particles: Update the velocities at time t by using a linear combination of the inertia, $\mathbf{V_i(t)}$, and the gravitation from the cognitive part, $\mathbf{Pb_i}$, and the social part, $\mathbf{Nb}$, respectively. w is the weight for the inertia and is usually a constant. $\mathbf{C_p}$ and $\mathbf{C_n}$ are random vectors with each component sampled from uniform distributions $U(0, c_p)$ and $U(0, c_n)$ with $c_p > 0$ and $c_n > 0$ as acceleration coefficients. The position is then assigned according to the current position with application of the updated velocity.

As we can observe, the inherent characteristics of PSO – the interactions among particles – are implemented with the shared knowledge on the best position found by neighbors. When a particle within the neighborhood locates a position of an objective value which is better than $\mathcal{F}(\mathbf{Nb})$, the other particles will make corresponding adjustments and tend to go toward that position. Therefore, the neighborhood attractor can be viewed as a channel through which each particle can emulate the others, and the update of the neighborhood attractor can be considered as a signal urging the swarm to adjust their movements in order to respond to the new discovery in the search space.

2.2. A macroscopic view of PSO

In spite of its importance, the effect of particle interaction in PSO is hardly investigated in the literature. Although there are a number of remarkable theoretical studies that bring insights into the properties and behavior of PSO conducted in the past, most of those studies are based on the assumption that the attractor is fixed, e.g., the trajectory analysis [4,5] mentioned in Section 1. Such a setting seems an inevitable path to simplify the PSO system to the extent that rigorous analysis can be done because the highly decentralized property of a particle swarm leads the system away from a unified depiction of the entire swarm. Each particle keeps its own position and memory, in the form of the inertia and the cognitive part, $\mathbf{Pb_i}$. In addition to the personal experience, the swarm also shares collective knowledge, $\mathbf{Nb}$, and any slight change in

these quantities substantially defines a new state of the whole system. The analysis on the overall behavior of a swarm is thus beyond tractable due to the complication of state transition, and the simplification of invariant attractors becomes an unpleasant but necessary means that makes a particle able to be observed independently without the interference from the other factors of the entire swarm.

As a consequence, in order to take particle interaction into consideration in a theoretical analysis, an alternative interpretation of PSO that regards the swarm as a unity becomes necessary. With this point of view, the state of a PSO system should be considered as a measurement that reflects the overall behavior and characteristics of a swarm rather than as a detailed configuration directly related to each individual particle. For this purpose, the development of statistical mechanics may be a good example to learn from, especially the employment of statistical methods to bridge the macroscopic and microscopic descriptions. Accordingly, the state of the entire swarm can be considered as the "macrostate" — an abstraction of the detailed description of particles, i.e., the "microstate." Hence in the macrostate space, the precise configuration of particles are converted into a statistical abstraction and characterization of the entire swarm.

More specifically, the exact locations of particles are no longer traced but instead modeled and expressed by using a distribution $\theta(t)$ over $\mathbb{R}^n$. The velocities on each dimension are viewed as a random vector $\mathcal{V}(t) \in \mathbb{R}^n$. To concentrate on the social behavior, i.e., particle interaction, we use the *social-only model* of PSO categorized by Kennedy [16], in which PSO works without the cognitive component, to make the system more concise. The swarm size m is considered as the number of realizations or samples of the distribution. As to the neighborhood attractor, since the geographic knowledge about the search space is embodied in the positional distribution, it can be viewed as the best observed value of the current time step. When the neighborhood attractor is determined, the social gravitation is also accordingly determined. Formally, each particle $\mathbf{P_i}$ is a random vector sampled from $\theta(t)$, and its velocity vector $\mathbf{V_i}$ is distributed as $\mathcal{V}(t)$. Since the neighborhood attractor is the best observed value, it can be defined as

$$\mathbf{P}^* := \arg\min\{\mathcal{F}(\mathbf{P_1}), \mathcal{F}(\mathbf{P_2}), \ldots, \mathcal{F}(\mathbf{P_m})\} ,$$

and each particle $\mathbf{P_i}$ updates its position to $\mathbf{P_i} + w\mathbf{V_i} + \mathbf{C} \otimes (\mathbf{P}^* - \mathbf{P_i})$. The distributions of the next time step $\theta(t+1)$ and $\mathcal{V}(t+1)$ are thus the statistical characterization, denoted as functions $\mathcal{T}_{\mathcal{P}}$ and $\mathcal{T}_{\mathcal{V}}$, of the observed values:

$$\theta(t+1) \leftarrow \mathcal{T}_{\mathcal{P}}(\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}) ;$$
$$\mathcal{V}(t+1) \leftarrow \mathcal{T}_{\mathcal{V}}(\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}; \mathbf{V_1}, \mathbf{V_2}, \ldots, \mathbf{V_m}) .$$

By considering PSO in this way, the search/optimization process is conducted through the repeated observations on the search space by realizing particles and modifying the distribution to accommodate the newly discovered results. Furthermore, going deeper into the notion of distribution, since the inertia weight w is usually a constant, $\mathcal{V}(t)$ can be considered redundant and may be removed because given two random vectors $\mathbf{X} \sim \theta(t)$ and $\mathbf{V} \sim \mathcal{V}(t)$, where the notation " $\sim$ " indicates "is distributed according to," we can simply let $\widetilde{\theta}(t)$ be the distribution of $\mathbf{X}' := \mathbf{X} + w\mathbf{V}$ that includes the effects of both the position and the velocity. Therefore, in the following, we will alter the notation θ to denote this compound distribution and parameterize it based on varied contexts.

The remaining questions would be what distribution is suitable for the description of a swarm without sacrificing too much essence of PSO and how to update the distribution as the search process proceeds. We can consider the random vector $\mathbf{X} \sim \theta(t)$, denote $\mathrm{E}[\mathbf{X}] = \boldsymbol{\mu}$, and decompose the region

$$R := \{\mathbf{y} \in \mathbb{R}^n \mid \mathrm{Prob}\{\mathbf{X} = \mathbf{y}\} > 0\}$$

into s disjoint regions $R_1, R_2, \ldots, R_s$ such that $\mathrm{Prob}\{\mathbf{X} \in R_i\} = 1/s$ for all $i \in \{1, 2, \ldots, s\}$. Each region is associated with a random variable of velocity $\mathbf{V_i} \sim \mathcal{V}(t)$. If one point $\mathbf{x_i}$ is respectively selected from each region R_i, when s is sufficiently large, the average behavior of a swarm can therefore be characterized by

$$\begin{aligned}\sum_{i=1}^{s} \frac{1}{s}(\mathbf{x_i} + \mathbf{V_i}) &= \sum_{i=1}^{s} \frac{1}{s}\mathbf{x_i} + \sum_{i=1}^{s} \frac{1}{s}\mathbf{V_i} \\ &\approx \boldsymbol{\mu} + \sum_{i=1}^{s} \frac{1}{s}\mathbf{V_i} ,\end{aligned}$$

and each component of the term $\sum_{i=1}^{s}(1/s)\mathbf{V_i}$ can be approximated with a normal distribution according to the central limit theorem. Thus, as an attempt to characterize the overall behavior of a swarm, the normal distribution should be a reasonable starting point. It is assumed that, at time t, each particle is sampled from $\mathbf{c}(\mathbf{t}) + \mathbf{Z}$, where $\mathbf{c}(\mathbf{t}) \in \mathbb{R}^n$ is the center of distribution and $\mathbf{Z} \in \mathbb{R}^n$ is a random vector of which each coordinate is distributed according to $N(0, \sigma^2)$, where $N(0, \sigma^2)$ denotes the normal distribution with zero mean and variance σ^2. In this paper, $\phi(\cdot)$ and $\Phi(\cdot)$ are used as the probability density function (pdf) and the cumulative distribution function (cdf) of the standard normal distribution, respectively. We can then reparameterize $\theta(t)$, the distribution of $\mathbf{c}(\mathbf{t}) + \mathbf{Z}$, as $\theta(\mathbf{c}(\mathbf{t}), \sigma^2)$.

The update of distributions can now be simplified into the modification of the mean and the variance. The mean is the arithmetic average of updated positions of particles, and the variance is estimated by a maximum likelihood estimation (MLE) which will be addressed later. Under such an interpretation, the PSO system can be described with the following algorithm:

Table 1
Average p-values of normality tests.

Swarm size	Normality tests		
	Shapiro–Wilk [17]	Anderson–Darling [18]	D'Agostino–Pearson [19]
10	0.3879	0.3621	0.3985
20	0.3257	0.2842	0.3393
30	0.2903	0.2518	0.2876

Algorithm 2 (*Statistical interpretation of PSO*).
procedure PSO(Objective function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$)
  Initialize θ
  while the stopping criterion is not satisfied **do**
    for $i = 1, 2, \ldots, m$ **do**
      $\mathbf{P_i} \sim \theta$
    end for
    $\mathbf{P}^* = \arg\min\{\mathcal{F}(\mathbf{P_1}), \mathcal{F}(\mathbf{P_2}), \ldots, \mathcal{F}(\mathbf{P_m})\}$
    for $i = 1, 2, \ldots, m$ **do**
      $\mathbf{P'_i} \leftarrow \mathbf{P_i} + \mathbf{C_i} \otimes (\mathbf{P}^* - \mathbf{P_i})$
    end for
    $\boldsymbol{\mu}_{\mathbf{t+1}} \leftarrow (\sum_{i=1}^{m} \mathbf{P'_i})/m$
    $\sigma^2_{t+1} \leftarrow \mathbf{MLE}(\mathbf{P'_1}, \mathbf{P'_2}, \ldots, \mathbf{P'_m})$
    $\theta \leftarrow \theta(\boldsymbol{\mu}_{\mathbf{t+1}}, \sigma^2_{t+1})$
    $t \leftarrow t + 1$
  end while
end procedure

In order to validate the utilization of normal distributions for describing swarms, we conducted three well-known normality tests: the Shapiro–Wilk test [17], the Anderson–Darling test [18], and the D'Agostino–Pearson test [19] on the social-only PSO on the sphere function. Table 1 displays the test results, which were obtained for 100 independent runs and 10 iterations in each run. The weight for the inertia is 0.73 and the acceleration coefficient is 1.49. Since all p-values of the three normality tests significantly surpass the conventional significance level 0.05, none of these tests are able to reject the null hypothesis. As a result, in this study, adopting the normal distribution as the description of swarms is an acceptable assumption.

In summary, the macrostate model transforms the detailed configuration of PSO into a corresponding stochastic representation embodied by normal distributions. As a consequence, the update of particles is simplified as the modification of the parameters of normal distributions. In each iteration, Algorithm 2 generates a swarm of particles by means of sampling from the current distribution, and thereafter, the distribution is updated according to particle interaction. In others words, a state of Algorithm 2 is a distribution, and the sampled swarm serves as a medium for state transition. In this manner, the analysis of the behavior of the entire swarm is thus reduced to the analysis of parameterized distributions. The inclusion of particle interaction into analysis supplies numerous facets of PSO typically absent in related theoretical studies, e.g., the progress rate and the influence of objective functions, because the restriction of fixed attractors makes objective functions irrelevant. Since the No-Free-Lunch theorem states that all optimization algorithms perform identically on average [20], the effectiveness of PSO can hardly be theoretically identified unless the scope of functions is specified.

In the remainder of this paper, Algorithm 2 will be the study subject and be formally investigated on the sphere function, which is commonly adopted in the theoretical analysis of evolutionary algorithms (e.g., [21]) and can be formulated as

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2,$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$.

3. Progress rate analysis

The major benefit to develop and adopt the abstraction based on probabilistic distributions of PSO is that the mathematical model can be analyzed without the assumption of fixed attractors, because particles are in essence random vectors in the search space and consequently their behavior can be described and predicted in a statistical sense. In this section, we will demonstrate how the statistical interpretation of PSO proposed in the present work facilitates the analysis of inter-particle effects and how these effects are accounted for the progress rate of a swarm. We will begin with the n-ball hitting probability.

3.1. n-ball hitting probability

Given a distribution θ over $\mathbb{R}^n$, the term *n-ball hitting probability* refers to the probability that a random vector sampled from θ that "falls" into a specific n-dimensional ball. This probability is fundamental to the sphere model, because in the sphere model the objective function is simply the squared L^2 norm, and a subset of $\mathbb{R}^n$ constructed by collecting all the vectors with their norms bounded by a specific non-negative quantity forms an n-ball located at the origin with a radius defined by that non-negative quantity. Therefore, n-ball hitting probability is equal to the probability that the norm of a random vector is less than or equal to the radius. In other words, it is essentially the cumulative distribution function (cdf) of the norm of a random vector.

Given the center of distribution at time t, $\mathbf{c}(\mathbf{t}) = (c_1, c_2, \ldots, c_n) \in \mathbb{R}^n$, we would like to calculate the probability, denoted as $B_k(\mathbf{o})$, that $\mathbf{c}(\mathbf{t}) + \mathbf{Z} \sim \theta$ is in an n-ball located at the origin with radius k, where $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n) \in \mathbb{R}^n$ is a random vector and each coordinate of $\mathbf{Z}$ is normally distributed. Since $Z_1, Z_2, \ldots, Z_n$ are independent and identically distributed (i.i.d.) random variables, $\mathbf{Z}$ is an isotropic random vector, i.e., all directions of $\mathbf{Z}$ are equally likely to occur [22]. We elaborate this property as follows. Given $Z_1, Z_2, \ldots, Z_n \sim N(0, \sigma^2)$, for all $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$,

$$\begin{aligned}
\text{Prob}\{\mathbf{c}(\mathbf{t}) + \mathbf{Z} = \mathbf{x}\} &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x_i - c_i)^2}{2\sigma^2}\right) \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(\frac{-\sum_{i=1}^{n}(x_i - c_i)^2}{2\sigma^2}\right) \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(\frac{-d(\mathbf{x}, \mathbf{c}(\mathbf{t}))^2}{2\sigma^2}\right),
\end{aligned}$$

where $d(\cdot, \cdot)$ denotes the Euclidean distance. It is obvious that the density at point $\mathbf{x}$ is determined by $d(\mathbf{x}, \mathbf{c}(\mathbf{t}))$, regardless of the direction in which $\mathbf{x}$ is relatively to $\mathbf{c}(\mathbf{t})$. Therefore, without loss of generality, we may assume that $\mathbf{c}(\mathbf{t})$ is on the first axis by conducting a coordinate transformation. Let $r := d(\mathbf{c}(\mathbf{t}), \mathbf{o}) \geq 0$. As a result, $\mathbf{c}(\mathbf{t})$ can be expressed, after the coordinate transformation, as $(r, 0, 0, \ldots, 0)$, and the distribution is denoted as $\theta(r, \sigma^2)$. Now, the n-ball hitting probability can be formally defined as follows.

Definition 1. Given an n-ball $B_k(\mathbf{o}) \in \mathbb{R}^n$ and a random vector $\mathbf{c}(\mathbf{t}) + \mathbf{Z} \sim \theta(r, \sigma^2) \in \mathbb{R}^n$, where $\mathbf{c}(\mathbf{t}) = (r, 0, 0, \ldots, 0)$ and all coordinates of $\mathbf{Z}$ are distributed according to $N(0, \sigma^2)$, the n-ball hitting probability

$$H_B(k, \theta(r, \sigma^2)) := \text{Prob}\{\mathbf{c}(\mathbf{t}) + \mathbf{Z} \in B_k(\mathbf{o})\}.$$

The analysis approach adopted in the present work is similar to that used by Beyer in 2001 [21]. The vector $\mathbf{Z}$ is decomposed into two orthogonal vectors: $Z_1\mathbf{e_1} = (Z_1, 0, 0, \ldots, 0)$ and $\mathbf{Z}' = (0, Z_2, Z_3, \ldots, Z_n)$. We can take a closer look at the n-ball hitting probability $H_B(k, \theta(r, \sigma^2))$:

$$\begin{aligned}
H_B(k, \theta(r, \sigma^2)) &= \text{Prob}\{\mathbf{c}(\mathbf{t}) + \mathbf{Z} \in B_k(\mathbf{o})\} \\
&= \text{Prob}\left\{\|(r + Z_1)\mathbf{e_1} + \mathbf{Z}'\| \leq k\right\} \\
&= \text{Prob}\left\{(r + Z_1)^2 + \|\mathbf{Z}'\|^2 \leq k^2\right\} \\
&= \text{Prob}\left\{-k - r \leq Z_1 \leq k - r,\ 0 \leq \|\mathbf{Z}'\|^2 \leq k^2 - (r + Z_1)^2\right\}.
\end{aligned}$$

The equation shows that the n-ball hitting probability is the joint distribution of Z_1 and $W := \|\mathbf{Z}'\|^2$. Since $Z_1 \sim N(0, \sigma^2)$, the probability density function can be expressed as

$$p(Z_1, x) := \text{Prob}\{Z_1 = x\} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-x^2}{2\sigma^2}\right),$$

and W is a chi-square random variable with $n' := n - 1$ degrees of freedom:

$$p(W, y) := \text{Prob}\{W = y\} = \frac{1}{\sigma^2} \frac{\left(\frac{y}{\sigma^2}\right)^{\frac{n'}{2}-1} \exp\left(\frac{-y}{2\sigma^2}\right)}{2^{\frac{n'}{2}}\Gamma\left(\frac{n'}{2}\right)}.$$

As a result, we can get

$$\begin{aligned}
H_B(k,\theta(r,\sigma^2)) &= \text{Prob}\left\{-k-r \le Z_1 \le k-r,\ 0 \le \|\mathbf{Z}'\|^2 \le k^2-(r+Z_1)^2\right\}\\
&= \int_{x=-k-r}^{k-r}\int_{y=0}^{k^2-(x+r)^2} p(Z_1,x)p(W,y)\,\mathrm{d}y\,\mathrm{d}x\\
&= \int_{x=-k-r}^{k-r} p(Z_1,x)\int_{y=0}^{k^2-(x+r)^2} \frac{1}{\sigma^2}\frac{\left(\frac{y}{\sigma^2}\right)^{\frac{n'}{2}-1}\exp\left(\frac{-y}{2\sigma^2}\right)}{2^{\frac{n'}{2}}\Gamma\left(\frac{n'}{2}\right)}\,\mathrm{d}y\,\mathrm{d}x\\
(\text{let u} := y/\sigma^2)\ &= \int_{x=-k-r}^{k-r} p(Z_1,x)\int_{u=0}^{\frac{k^2-(x+r)^2}{\sigma^2}} \frac{u^{\frac{n'}{2}-1}\exp\left(\frac{-u}{2}\right)}{2^{\frac{n'}{2}}\Gamma\left(\frac{n'}{2}\right)}\,\mathrm{d}u\,\mathrm{d}x\\
&= \int_{x=-k-r}^{k-r} p(Z_1,x)\mathcal{P}\left(\frac{n'}{2},\frac{k^2-(x+r)^2}{2\sigma^2}\right)\mathrm{d}x\,,
\end{aligned}$$

where $\mathcal{P}(\cdot)$ is the regularized Gamma function.

Remark 2. If an asymptotic approximation is desired for the n-ball hitting probability, $H_B(k,\theta(r,\sigma^2))$, we can utilize the normal approximation to the regularized Gamma function [23, chapter 7] as

$$\mathcal{P}\left(\frac{n'}{2},\frac{k^2-(x+r)^2}{2\sigma^2}\right) \approx \Phi\left(\frac{1}{\sqrt{2n'}}\left[\frac{k^2-(x+r)^2}{\sigma^2}-n'\right]\right)\,.$$

For the asymptotic approximation, when n is sufficiently large, the term $(1/\sqrt{2n'})[k^2-(x+r)^2]/\sigma^2$ vanishes. Thanks to the continuity of $\Phi(\cdot)$, we can obtain

$$\mathcal{P}\left(\frac{n'}{2},\frac{k^2-(x+r)^2}{2\sigma^2}\right) \approx \Phi\left(-\sqrt{\frac{n'}{2}}\right)\,.$$

Hence,

$$\begin{aligned}
H_B(k,\theta(r,\sigma^2)) &\approx \Phi\left(-\sqrt{\frac{n'}{2}}\right)\int_{x=-k-r}^{k-r} p(Z_1,x)\,\mathrm{d}x\\
&= \Phi\left(-\sqrt{\frac{n'}{2}}\right)\left[\Phi\left(\frac{k-r}{\sigma}\right)-\Phi\left(\frac{-k-r}{\sigma}\right)\right]\\
&= \Phi\left(-\sqrt{\frac{n'}{2}}\right)\left[\Phi\left(\frac{r+k}{\sigma}\right)-\Phi\left(\frac{r-k}{\sigma}\right)\right]\,.
\end{aligned}$$

In addition to the asymptotic properties of $H_B(k,\theta(r,\sigma^2))$, it would be helpful to derive a lower bound for $H_B(k,\theta(r,\sigma^2))$ to facilitate our analysis in the present work.

Lemma 3 (*Lower Bound for* $H_B(k,\theta(r,\sigma^2))$).

$$H_B(k,\theta(r,\sigma^2)) \ge \left[\Phi\left(\frac{r+\frac{k}{\sqrt{n}}}{\sigma}\right)-\Phi\left(\frac{r-\frac{k}{\sqrt{n}}}{\sigma}\right)\right]\left[1-2\Phi\left(\frac{-k}{\sqrt{n}\sigma}\right)\right]^{n-1}\,.$$

Proof. Let $\mathbf{Y} := \mathbf{c}(\mathbf{t})+\mathbf{Z}$, where $\mathbf{c}(\mathbf{t}) = (r,0,0,\ldots,0)$, and $\mathbf{Z} = (Z_1,Z_2,\ldots,Z_n)$. Let $\mathcal{D} := [-k/\sqrt{n}, k/\sqrt{n}]^n \subseteq R^n$. For all $\mathbf{x}\in\mathcal{D}$, because $\|\mathbf{x}\| \le \sqrt{n}\|\mathbf{x}\|_\infty \le \sqrt{n}(k/\sqrt{n}) = k$, we can know that $\mathbf{x}\in B_k(\mathbf{o})$. Hence, $\mathcal{D}\subseteq B_k(\mathbf{o})$, and

$$\begin{aligned}
\text{Prob}\{\mathbf{Y}\in B_k(\mathbf{o})\} \ge \text{Prob}\{\mathbf{Y}\in\mathcal{D}\} &= \text{Prob}\left\{-\frac{k}{\sqrt{n}}-r \le Z_1 \le \frac{k}{\sqrt{n}}-r\right\}\prod_{i=2}^{n}\text{Prob}\left\{-\frac{k}{\sqrt{n}} \le Z_i \le \frac{k}{\sqrt{n}}\right\}\\
&= \left[\Phi\left(\frac{\frac{k}{\sqrt{n}}-r}{\sigma}\right)-\Phi\left(\frac{-\frac{k}{\sqrt{n}}-r}{\sigma}\right)\right]\left[\Phi\left(\frac{\frac{k}{\sqrt{n}}}{\sigma}\right)-\Phi\left(\frac{-\frac{k}{\sqrt{n}}}{\sigma}\right)\right]^{n-1}\\
&= \left[\Phi\left(\frac{r+\frac{k}{\sqrt{n}}}{\sigma}\right)-\Phi\left(\frac{r-\frac{k}{\sqrt{n}}}{\sigma}\right)\right]\left[1-2\Phi\left(\frac{-k}{\sqrt{n}\sigma}\right)\right]^{n-1}\,. \quad \square
\end{aligned}$$

For the notational purpose, we let

$$\psi'(k) := \left[\Phi\left(\frac{r+\frac{k}{\sqrt{n}}}{\sigma}\right) - \Phi\left(\frac{r-\frac{k}{\sqrt{n}}}{\sigma}\right)\right]\left[1-2\Phi\left(\frac{-k}{\sqrt{n}\sigma}\right)\right]^{n-1},$$

and the antiderivative is defined as $\psi(k) := \int_{t=0}^{k} \psi'(t)\,\mathrm{d}t$.

Remark 4. Similarly, we can also define the n-sphere hitting density $H_S(k, \theta(r, \sigma^2))$ for random vector $\mathbf{c}(\mathbf{t}) + \mathbf{Z}$ as

$$\begin{aligned} H_S(k, \theta(r, \sigma^2)) &:= \text{Prob}\{\|\mathbf{c}(\mathbf{t}) + \mathbf{Z}\| = k\} \\ &= \text{Prob}\left\{-k - r \le Z_1 \le k - r, W = k^2 - x^2\right\} \\ &= \int_{x=-k-r}^{k-r} p(Z_1, x)p(W, k^2 - x^2)\,\mathrm{d}x. \end{aligned}$$

Therefore, the n-ball hitting probability, $H_B(k, \theta(r, \sigma^2))$, as the cumulative function of $H_S(k, \theta(r, \sigma^2))$, can be alternatively defined as

$$\int_{y=0}^{k}\int_{x=-y-r}^{y-r} p(Z_1, x)p(W, y^2 - x^2)\,\mathrm{d}x\,\mathrm{d}y.$$

However, the density function $H_S(k, \theta(r, \sigma^2))$ serves no purpose other than a definition in the following analysis. We left it as a side note for completeness without further discussion.

3.2. Expected particle norm

The entire PSO system can be decomposed into two fundamental components: (1) the update of attractors to share and exchange information among particles, and (2) the update of particle positions through the interaction between particles and attractors. Hence, as we gain understandings of the characteristics of attractors and particles, we may capture the stochastic behavior of the PSO system. More specifically, because the distance from the origin is the most important characteristic of the sphere model for its unimodality, in this section, we highlight the expected distance between particles and the global optimum. Given a probabilistic model according to which particles are distributed, we would like to know how close to the global optimum in expectation the sampled particles are. Since the global optimum is simply the origin in the sphere model, we concentrate on the L^2-norm of sampled particles. The expected norms of the attractor and of particles are examined, respectively. As the analysis proceeds, it can be shown that these two expectations influence the progress rate of PSO.

Given the center of a particle distribution $\mathbf{c}(\mathbf{t}) = (r, 0, \ldots, 0)$ and $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n)$ with $Z_1, Z_2, \ldots, Z_n \sim N(0, \sigma^2)$, suppose that there are m particles, $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}$, sampled as $\mathbf{c}(\mathbf{t}) + \mathbf{Z}$, the expected norm of particles can be defined as

$$\overline{P} := \mathrm{E}\left[\|\mathbf{c}(\mathbf{t}) + \mathbf{Z}\|\right],$$

which can be considered as the mean solution quality of the current swarm on the sphere function. The following lemma gives an upper bound for $\overline{P}$.

Lemma 5 (*Upper Bound for the Expected Particle Norm*). *If* $\mathbf{c}(\mathbf{t}) = (r, 0, 0, \ldots, 0)$ *and* $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n)$ *with* $Z_1, Z_2, \ldots, Z_n \sim N(0, \sigma^2)$, $\overline{P} \le \sqrt{r^2 + n\sigma^2}$.

Proof. For all positive random variable X, since the square root is a concave function, we have $\mathrm{E}\left[\sqrt{X}\right] \le \sqrt{\mathrm{E}[X]}$ according to Jensen's inequality. By utilizing this property, we can have the following derivation:

$$\begin{aligned} \overline{P} &= \mathrm{E}\left[\|\mathbf{c}(\mathbf{t}) + \mathbf{Z}\|\right] \\ &= \mathrm{E}\left[\sqrt{(Z_1 - r)^2 + \sum_{i=2}^{n} Z_i^2}\right] \\ &\le \sqrt{\mathrm{E}\left[(Z_1 - r)^2 + \sum_{i=2}^{n} Z_i^2\right]} \\ &= \sqrt{\mathrm{E}\left[r^2\right] - 2r\mathrm{E}\left[Z_1\right] + \sum_{i=1}^{n} \mathrm{E}\left[Z_i^2\right]} \\ &= \sqrt{r^2 + n\sigma^2}. \end{aligned}$$

Because $Z_i \sim N(0, \sigma^2)$, we have $\mathrm{E}\left[Z_i^2\right] = \sigma^2$ and $\mathrm{E}[Z_i] = 0$. An upper bound for the expected particle norm, $\overline{P}$, is therefore obtained. □

The expected particle norm describes how close on average a swarm is to the global optimum, i.e., the origin, of the sphere function. In order to capture the characteristic of the essential mechanism of PSO – particle interaction – we also need to investigate the attractor. As stated in the previous section, the attractor is the best observed value, i.e., in our case, the particle with the minimum objective value within the neighborhood in the current swarm. Under the adopted statistical interpretation of PSO, the expected minimum objective value of a swarm becomes traceable through order statistics, because particles are viewed as random vectors over $\mathbb{R}^n$.

Let $P_{(i,m)}$ denote the ith order statistic of $\|\mathbf{P_1}\|, \|\mathbf{P_2}\|, \ldots, \|\mathbf{P_m}\|$, e.g., $P_{(1,m)} = \min\{\|\mathbf{P_1}\|, \|\mathbf{P_2}\|, \ldots, \|\mathbf{P_m}\|\}$. Denoting the event $\|\mathbf{P_i}\| = x$ as $\{\|\mathbf{P_i}\| = x\}$, the density of $P_{(1,m)}$ at a non-negative real number x can be given as

$$\begin{aligned}\text{Prob}\left\{P_{(1,m)} = x\right\} &= \text{Prob}\left\{\bigcup_{i=1}^{m}\left[\{\|\mathbf{P_i}\| = x\}\bigcap\left(\bigcap_{j\in\{1,2,\ldots,m\}\setminus\{i\}}\left\{\|\mathbf{P_j}\| > x\right\}\right)\right]\right\}\\ &= \int_{x=-k-r}^{k-r}\binom{m}{1}H_S(x,\theta(r,\sigma^2))\left[1-H_B(x,\theta(r,\sigma^2))\right]^m\,\mathrm{d}x\,.\end{aligned}$$

Denoting $\mathrm{E}\left[P_{(1,m)}\right]$ as $\overline{P_{(1,m)}}$, a naive upper bound for $\overline{P_{(1,m)}}$ is derived in the following lemma.

Lemma 6. $\overline{P_{(1,m)}} \le \overline{P}$

Proof. The general upper bound for the expected ith order statistic states

$$\overline{P_{(i,m)}} \le \overline{P} + \left(\mathrm{Var}\left[\|\mathbf{c}(\mathbf{t}) + \mathbf{Z}\|\right]\right)^{\frac{1}{2}}\sqrt{\frac{i-1}{m-i+1}}\,.$$

As a result,

$$\overline{P_{(1,m)}} \le \overline{P} + \left(\mathrm{Var}\left[\|\mathbf{c}(\mathbf{t}) + \mathbf{Z}\|\right]\right)^{\frac{1}{2}}\sqrt{\frac{1-1}{m-1+1}} = \overline{P}. \quad \square$$

Lemma 6 causes no surprise. The expected minimum particle norm is obviously less than or equal to the expected norm. However, inspired by Lemma 6, we can seek another upper bound for $\overline{P_{(1,m)}}$ by definition.

Lemma 7 (*Upper Bound for* $\overline{P_{(1,m)}}$). (1)

$$\overline{P_{(1,m)}} = \int_{x=0}^{\infty}\left[1-H_B(x,\theta(r,\sigma^2))\right]^m\,\mathrm{d}x,$$

and (2)

$$\overline{P_{(1,m)}} \le \left(\lim_{h\to\infty}\left[h-\psi(h)\right]\right)^{\frac{m}{2}}.$$

Proof. (1) For any random variable X, $\mathrm{E}\left[|X|\right]^r = r\int_0^{\infty} t^{r-1}\mathrm{Prob}\left\{|X| > t\right\}\,\mathrm{d}t$ with $r > 0$ [24]. Since $P_{(1,m)}$ is a non-negative random variable, by letting $r = 1$ we have

$$\begin{aligned}\overline{P_{(1,m)}} &= \int_{x=0}^{\infty}\text{Prob}\left\{P_{(1,m)} > x\right\}\,\mathrm{d}x\\ &= \int_{x=0}^{\infty}\text{Prob}\left\{\bigcap_{i=1}^{m}\{\|\mathbf{P_i}\| > x\}\right\}\,\mathrm{d}x\\ &= \int_{x=0}^{\infty}\left[1-H_B(x,\theta(r,\sigma^2))\right]^m\,\mathrm{d}x.\end{aligned}$$

(2) Based on the result of (1), we obtain

$$\overline{P_{(1,m)}} = \int_{x=0}^{\infty}\left[1-H_B(x,\theta(r,\sigma^2))\right]^m\,\mathrm{d}x \le \int_{x=0}^{\infty}\left[1-\psi'(x)\right]^m\,\mathrm{d}x.$$

By resorting to Hölder's inequality, we can move m outside of the integration to obtain a more comprehensible bound as

$$\begin{aligned}\int_{x=0}^{\infty}\left[1-\psi'(x)\right]^m\,\mathrm{d}x &\le \left(\int_{x=0}^{\infty}\left[1-\psi'(x)\right]^2\,\mathrm{d}x\right)^{\frac{m}{2}}\\ &\le \left(\int_{x=0}^{\infty}\left[1-\psi'(x)\right]\,\mathrm{d}x\right)^{\frac{m}{2}}\\ &= \left(\lim_{h\to\infty}\left[h-\psi(h)\right]\right)^{\frac{m}{2}}.\end{aligned}$$

The last equation follows from $[h-\psi(h)]|_{h=0} = 0$. $\square$

Because this upper bound is presented in a limit form, a subsequent question would be whether or not it converges. The following theorem guarantees the convergence of the quantity.

Lemma 8. *$(\lim_{h\to\infty}[h-\psi(h)])^{\frac{m}{2}}$ is convergent.*

Proof. Denote $\int_{x=0}^{h}\left[1-\psi'(h)\right]\mathrm{d}x$ as $G(h)$. Since m is a constant, $(\lim_{h\to\infty}[h-\psi(h)])^{\frac{m}{2}}$ converges if $\lim_{h\to\infty}G(h)$ converges. $G(h)$ is incremental because $1-\psi'(x)$ is always non-negative. Thus, it is sufficient to show that $G(h)$ is bounded from above. When $h > r\sqrt{n}$,

$$\begin{aligned}
G(h) &= \int_{x=0}^{h}\left[1-\psi'(h)\right]\mathrm{d}x \\
&= \int_{x=0}^{h}\left(1-\left[\Phi\left(\frac{r+\frac{x}{\sqrt{n}}}{\sigma}\right)-\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]\left[1-2\Phi\left(\frac{-x}{\sqrt{n}\sigma}\right)\right]^{n-1}\right)\mathrm{d}x \\
&\le \int_{x=0}^{r\sqrt{n}}\mathrm{d}x+\int_{x=r\sqrt{n}}^{h}\left(1-\left[\Phi\left(\frac{r+\frac{x}{\sqrt{n}}}{\sigma}\right)-\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]\left[1-2\Phi\left(\frac{-x}{\sqrt{n}\sigma}\right)\right]^{n-1}\right)\mathrm{d}x \\
&\le r\sqrt{n}+\int_{x=r\sqrt{n}}^{h}\left(1-\left[\Phi\left(\frac{\frac{x}{\sqrt{n}}-r}{\sigma}\right)-\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]\left[1-2\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]^{n-1}\right)\mathrm{d}x \\
&= r\sqrt{n}+\int_{x=r\sqrt{n}}^{h}\left(1-\left[1-2\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]^{n}\right)\mathrm{d}x\,.
\end{aligned}$$

When $x \ge r\sqrt{n}$,

$$\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\le\frac{1}{2}.$$

By applying Bernoulli's inequality, we can get

$$\begin{aligned}
G(h) &\le r\sqrt{n}+\int_{x=r\sqrt{n}}^{h}\left(1-\left[1-2n\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]\right)\mathrm{d}x \\
&= r\sqrt{n}+2n\int_{x=r\sqrt{n}}^{h}\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\mathrm{d}x \\
&= r\sqrt{n}+2n\left[\left(-r\sqrt{n}+x\right)\Phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)-\sigma\sqrt{n}\cdot\phi\left(\frac{r-\frac{x}{\sqrt{n}}}{\sigma}\right)\right]\Bigg|_{x=r\sqrt{n}}^{x=h}.
\end{aligned}$$

The integration of the normal distribution is given in [25]. When $h\to\infty$, the term

$$\sigma\sqrt{n}\cdot\phi\left(\frac{r-\frac{h}{\sqrt{n}}}{\sigma}\right)$$

vanishes. Thus, now we only need to show

$$\lim_{h\to\infty}\left[\left(-r\sqrt{n}+h\right)\Phi\left(\frac{r-\frac{h}{\sqrt{n}}}{\sigma}\right)\right]<\infty.$$

Here we apply Mill's ratio to replace $\Phi(\cdot)$ with $\phi(\cdot)$ and get

$$\begin{aligned}
\left(-r\sqrt{n}+h\right)\Phi\left(\frac{r-\frac{h}{\sqrt{n}}}{\sigma}\right) &= \left(h-r\sqrt{n}\right)\left[1-\Phi\left(\frac{\frac{h}{\sqrt{n}}-r}{\sigma}\right)\right] \\
&\le \left(h-r\sqrt{n}\right)\cdot\phi\left(\frac{\frac{h}{\sqrt{n}}-r}{\sigma}\right)\cdot\left(\frac{\frac{h}{\sqrt{n}}-r}{\sigma}\right)^{-1} \\
&= \left(\sigma\sqrt{n}\right)\cdot\phi\left(\frac{\frac{h}{\sqrt{n}}-r}{\sigma}\right) \\
&= 0 \qquad \text{as } h\to\infty\,.
\end{aligned}$$

Therefore, $G(h)$ is bounded from above. The proof is completed. □

3.3. Lower and upper bounds for the expected progress rate

After the work was done in the previous sections, the progress rate of the social-only model PSO can now be formally investigated under the proposed statistical interpretation. The term "progress rate" was introduced by Rechenberg in 1973 [26]. As the name suggests, progress rate should be a quantity indicating how a particle swarm progresses, and hence in the present work, it is defined as the difference of the norms of the two distribution centers in successive time steps, because the distance to the optimum is the L^2 norm for the sphere function. Given the current center of distribution $\mathbf{c}(\mathbf{t}) = (r, 0, 0, \ldots, 0)$ and a random vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n)$ with $Z_1, Z_2, \ldots, Z_n \sim N(0, \sigma^2)$, the m particles $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}$ are sampled as $\mathbf{c}(\mathbf{t}) + \mathbf{Z}$. Let $P_{(i,m)}$ denote the ith order statistic of $\|\mathbf{P_1}\|, \|\mathbf{P_2}\|, \ldots, \|\mathbf{P_m}\|$. Let $\mathbf{P}^* := \arg\min\{\mathcal{F}(\mathbf{P_1}), \mathcal{F}(\mathbf{P_2}), \ldots, \mathcal{F}(\mathbf{P_m})\}$. By definition, $\|\mathbf{P}^*\| = P_{(1,m)}$. According to the update rules described in Section 2.2, the updated position $\mathbf{P_i'}$ is computed as $\mathbf{P_i'} = \mathbf{P_i} + \mathbf{C_i} \otimes (\mathbf{P}^* - \mathbf{P_i})$, where each coordinate of $\mathbf{C_i}$ is distributed according to $U(0, c)$ with c being the coefficient representing the compound effect of both the inertia weight and the acceleration coefficient of the social part. For simplicity, we still call c the acceleration coefficient in this paper because the inertia weight is usually constant. The center of distribution in the next step $\mathbf{c}(\mathbf{t}+\mathbf{1})$ is the mean of $\mathbf{P_1'}, \mathbf{P_2'}, \ldots, \mathbf{P_m'}$, i.e., $\mathbf{c}(\mathbf{t}+\mathbf{1}) = (\sum_{i=1}^{m} \mathbf{P_i'})/m$.

Definition 9. Given $\mathbf{c}(\mathbf{t}) = (r, 0, 0, \ldots, 0)$, the progress rate $\Delta_t := \|\mathbf{c}(\mathbf{t})\| - \|\mathbf{c}(\mathbf{t}+\mathbf{1})\| = r - \|\mathbf{c}(\mathbf{t}+\mathbf{1})\|$.

The following theorem shows that, when $c \leq 1/2$, the expected norm of the center of distribution in the next time step is bounded from above by a linear combination of the expected particle norm $\overline{P}$ and the expected minimum of the particle norm $\overline{P_{(1,m)}}$.

Lemma 10. *Suppose $\mathbf{C} = (C_1, C_2, \ldots, C_n)$ is a random vector of $\mathbb{R}^n$ with i.i.d. components and $\mathbf{X}$ is a random vector of $\mathbb{R}^n$. If $\mathbf{C}$ and $\mathbf{X}$ are independent, then $E[\|\mathbf{C} \otimes \mathbf{X}\|] \leq \sqrt{\mu_2'}\, E[\|\mathbf{X}\|]$, where μ_2' is the second moment of C_i.*

Proof. For any fixed vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$,

$$\begin{aligned}
\mathrm{E}[\|\mathbf{C} \otimes \mathbf{x}\|] &= \mathrm{E}\left[\sqrt{\sum_{i=1:n} C_i^2 x_i^2}\right] \\
&\leq \sqrt{\mathrm{E}\left[\sum_{i=1:n} C_i^2 x_i^2\right]} \\
&= \sqrt{\sum_{i=1:n} \mathrm{E}\left[C_i^2\right] x_i^2} \\
&= \sqrt{\mu_2'}\, \|x\| \,.
\end{aligned}$$

Since $\mathbf{C}$ and $\mathbf{X}$ are independent, by the law of total expectation conditional on $\mathbf{X}$, this lemma is proved. □

Theorem 11 (*Upper Bound for the Expected Norm of the Next Center*). *(1) $E[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] \leq E[|1-C|]\,\overline{P} + E[|C|]\,\overline{P_{(1,m)}}$; and (2) If $c \leq 1/2$, $E[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] \leq (1-c)\overline{P} + c\overline{P_{(1,m)}}$; otherwise, $E[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] \leq [(2c^2 - 2c + 1)/2c]\overline{P} + c\overline{P_{(1,m)}}$.*

Proof. This result is derived from the triangle inequality for L^2-norm and the previous lemma:

$$\begin{aligned}
\mathrm{E}[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] &= \mathrm{E}\left[\left\|\frac{\sum_{i=1}^{m}\left[\mathbf{P_i} + \mathbf{C_i} \otimes \left(\mathbf{P}^* - \mathbf{P_i}\right)\right]}{m}\right\|\right] \\
&= \left(\frac{1}{m}\right) \mathrm{E}\left[\left\|\sum_{i=1}^{m} (\mathbf{1} - \mathbf{C_i}) \otimes \mathbf{P_i} + m\mathbf{C_i} \otimes \mathbf{P}^*\right\|\right] \\
&\leq \left(\frac{1}{m}\right)\left(\sum_{i=1}^{m} \mathrm{E}[\|(\mathbf{1} - \mathbf{C_i}) \otimes \mathbf{P_i}\|] + m\mathrm{E}\left[\|\mathbf{C_i} \otimes \mathbf{P}^*\|\right]\right) \\
&\leq \left(c^2/3 - c + 1\right)^{1/2} \overline{P} + \left(c^2/3\right)^{1/2} \overline{P_{(1,m)}}. \quad \square
\end{aligned}$$

Corollary 12 (*Lower Bound for the Progress Rate*). *$E[\Delta_t] \geq r - \left(c^2/3 - c + 1\right)^{1/2} \overline{P} - \left(c^2/3\right)^{1/2} \overline{P_{(1,m)}}$.*

After the lower bound for $\mathrm{E}[\Delta_t]$ is established in Corollary 12, the next theorem sets a lower bound for $\mathrm{E}[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|]$. An upper bound for $\mathrm{E}[\Delta_t]$ will be accordingly obtained as a corollary.

Theorem 13 (*Lower Bound for the Expected Norm of the Next Center*). *If* $c \leq 1$, $E[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] \geq r(1-\exp(-2n'[\Phi(-r/\sigma)]^m))$.

Proof. Since $\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|$ is a non-negative random variable, from Markov's inequality, we have, for any positive number a,

$$\text{Prob}\{\|\mathbf{c}(\mathbf{t}+\mathbf{1})\| > a\} \leq a^{-1}\text{E}[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] .$$

Substituting a with r,

$$r\,\text{Prob}\{\|\mathbf{c}(\mathbf{t}+\mathbf{1})\| > r\} \leq \text{E}[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] .$$

Let the jth coordinate of $\mathbf{P_i}$, $\mathbf{C_i}$, and $\mathbf{c}(\mathbf{t}+\mathbf{1})$ be P_{ij}, C_{ij}, and $c(t+1)_j$, respectively. If there exists a coordinate j such that $\min\{P_{1j}, P_{2j}, \ldots, P_{mj}\} \geq r$, then

$$\begin{aligned}
\|\mathbf{c}(\mathbf{t}+\mathbf{1})\| &\geq |c(t+1)_j| \\
&= \left| \frac{\sum_{i=1}^{m}\left[P_{ij} + C_{ij}\left(P_j^* - P_{ij}\right)\right]}{m} \right| \\
&= \frac{\sum_{i=1}^{m}\left[\left(1-C_{ij}\right)P_{ij} + C_{ij}P_j^*\right]}{m} \\
&\geq \frac{\left[\left(1-C_{ij}\right)mr + C_{ij}mr\right]}{m} \\
&= r.
\end{aligned}$$

Similarly, $\max\{P_{1j}, P_{2j}, \ldots, P_{mj}\} \leq -r$ implies $\|\mathbf{c}(\mathbf{t}+\mathbf{1})\| > r$. Let E_j^+ be the event that $\min\{P_{1j}, P_{2j}, \ldots, P_{mj}\} \geq r$ and E_j^- be the event that $\max\{P_{1j}, P_{2j}, \ldots, P_{mj}\} \leq -r$. Let $E_j := E_j^+ \bigcup E_j^-$ and $E := \bigcup_{j=1}^{m} E_j$, we have

$$\begin{aligned}
\text{Prob}\{E\} &= \text{Prob}\left\{E \bigcap E_1^+\right\} + \text{Prob}\left\{E \bigcap \left(E_1^+\right)^c\right\} \\
&\geq \text{Prob}\left\{E_1^+\right\} + \text{Prob}\left\{\left(\bigcup_{i=2}^{n} E_i\right) \bigcap \left(E_1^+\right)^c\right\} \\
&= \text{Prob}\left\{E_1^+\right\} + \left(1 - \text{Prob}\left\{E_1^+\right\}\right)\text{Prob}\left\{\bigcup_{i=2}^{n} E_i\right\} .
\end{aligned}$$

Because $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}$ are i.i.d. and for each particle all of its coordinates other than the first one are identically distributed, for all $i > 1$ the symmetry and disjointness of E_i^+ and E_i^- imply that $\text{Prob}\{E_i\} = 2\text{Prob}\left\{E_i^+\right\} = 2[1-\Phi(r/\sigma)]^m = 2[\Phi(-r/\sigma)]^m$. Let $q := 2[\Phi(-r/\sigma)]^m$ for convenience of notation. By using the inclusion–exclusion principle, we have

$$\begin{aligned}
\text{Prob}\left\{\bigcup_{i=2}^{n} E_i\right\} &= \sum_{i=1}^{n'} \binom{n'}{i} q^i (-1)^{i+1} \\
&= 1 - \sum_{i=0}^{n'} \binom{n'}{i} (-q)^i \\
&= 1 - (1-q)^{n'} \\
&\geq 1 - \exp\left(-n'q\right) .
\end{aligned}$$

As a result,

$$\begin{aligned}
\text{E}[\|\mathbf{c}(\mathbf{t}+\mathbf{1})\|] &\geq r\left(\text{Prob}\left\{E_1^+\right\} + \left(1 - \text{Prob}\left\{E_1^+\right\}\right)\left(1 - \exp\left(-n'q\right)\right)\right) \\
&\geq r\left(\text{Prob}\left\{E_1^+\right\} + 1 - \text{Prob}\left\{E_1^+\right\} - \exp\left(-n'q\right)\right) \\
&= r\left(1 - \exp\left(-2n'\left[\Phi(-r/\sigma)\right]^m\right)\right) . \quad \square
\end{aligned}$$

Corollary 14 (*Upper Bound for the Progress Rate*). *If* $c < 1$, *then* $E[\Delta_t] \leq r\exp\left(-2n'[\Phi(-r/\sigma)]^m\right)$.

With Theorems 11 and 13, we established the upper and lower bounds of the expected particle norm. Accordingly, with Corollaries 12 and 14, we derived the lower and upper bounds of the expected progress rate of a swarm in the social-only model. As aforementioned, by statistically interpreting the social-only model PSO, we can describe the "macrostate" of the particle swarm and therefore are able to analyze the stochastic behavior of PSO based on the facet of particle interaction.

4. Convergence analysis

As stated in Section 2.2, the transition from the current time step to the next time step consists of updating positions of particles, calculating the distribution center by means of the updated positions, and using the maximum likelihood estimation to calculate the distribution variance. The issues related to the centers of distributions have been addressed in Section 3. Thus, the part of variance is considered in this section. While the center of a distribution can be viewed as the indication of the average quality of the swarm at a specific time step, the variance is a direct measurement of convergence, because from the viewpoint of statistical interpretation, a swarm *converges* as the variance of the distribution reduces to zero. The word "converge" is not a unified term in the research domain of PSO [27, p. 132]. It has been used to describe the behavior of a swarm approaching the local optimum in some papers, while it simply indicates the phenomenon that a swarm of particles crowds into a specific point, sometimes called the *equilibrium*, not necessarily the local optimum, in the search space in other papers. Here in the present work, we adopt the latter definition. We concentrate on the condition under which a swarm of particles may go into a stable state. We will demonstrate that if certain condition of the relationship between the swarm size and the acceleration coefficient is satisfied, a swarm in the social-only model does converge under the mechanism of particle interaction.

Given m observed vectors $\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_m}$ that stand for the updated positions and the distribution center is denoted as $\mathbf{c(t+1)} = \bar{\mathbf{y}} := (\Sigma_{i=1}^{m} \mathbf{y_i})/m$. Let $\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m}$ be random vectors sampled from $\theta(\|\bar{\mathbf{y}}\|, \sigma_{t+1}^2)$. These vectors are n-dimensional random vectors centered at $\bar{\mathbf{y}}$, and the coordinate on each dimension is a random variable sampled from $N(0, \sigma_{t+1}^2)$, where σ_{t+1}^2 is the variance that we wish to estimate. In order to estimate the variance, the likelihood function of σ_{t+1}^2, $L(\sigma_{t+1}^2)$, can be defined as the joint probability:

$$\begin{aligned} L(\sigma_{t+1}^2) &:= \prod_{i=1}^{m} \left(\frac{1}{\sqrt{2\pi}\sigma_{t+1}}\right)^n \exp\left(\frac{-d\,(\mathbf{y_i}, \bar{\mathbf{y}})^2}{2\sigma_{t+1}^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma_{t+1}}\right)^{mn} \exp\left(\frac{-\sum_{i=1}^{m} d\,(\mathbf{y_i}, \bar{\mathbf{y}})^2}{2\sigma_{t+1}^2}\right) \\ &= K\sigma_{t+1}^{-mn} \exp\left(\frac{-R}{2\sigma_{t+1}^2}\right), \end{aligned}$$

where

$$K := \left(\frac{1}{\sqrt{2\pi}}\right)^{mn}, \qquad R := \sum_{i=1}^{m} d(\mathbf{y_i}, \bar{\mathbf{y}})^2 .$$

In order to get the σ_{t+1}^2 that maximizes $L(\sigma_{t+1}^2)$, we differentiate $L(\sigma_{t+1}^2)$ with respect to σ_{t+1}^2:

$$L'(\sigma_{t+1}^2) = -\frac{mn}{2} K \cdot \sigma_{t+1}^{-mn-2} \cdot \exp\left(\frac{-R}{2\sigma_{t+1}^2}\right) + \frac{R}{2} K \cdot \sigma_{t+1}^{-mn-4} \cdot \exp\left(\frac{-R}{2\sigma_{t+1}^2}\right) .$$

$L'(\sigma_{t+1}^2) = 0$ implies $\sigma_{t+1}^2 = R/(mn)$, and it is routine to check the maximality. Since both m and n are fixed, the only quantity needs to be examined is R, the sum of square of the distance between each updated particle and the center. Given $\mathbf{c(t)} = (r, 0, 0, \ldots, 0)$ and $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_n)$ with $Z_1, Z_2, \ldots, Z_n \sim N(0, \sigma_t^2)$, the m particles $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}$ are sampled from $\mathbf{c(t)} + \mathbf{Z}$, and the updated position is calculated as $\mathbf{P_i} + \mathbf{C_i} \otimes (\mathbf{P}^* - \mathbf{P_i})$, where $\mathbf{P}^*$ is the attractor. Since $\mathbf{c(t+1)} = \sum_{i=1}^{m} [\mathbf{P_i} + \mathbf{C_i} \otimes (\mathbf{P}^* - \mathbf{P_i})]/m$, R, as a random variable, can be defined by $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_m}$ and $\mathbf{P}^*$:

$$R = \sum_{i=1}^{m} \left\| \mathbf{P_i} + \mathbf{C_i} \otimes (\mathbf{P}^* - \mathbf{P_i}) - \frac{\sum_{j=1}^{m} \left(\mathbf{P_j} + \mathbf{C_i} \otimes (\mathbf{P}^* - \mathbf{P_j})\right)}{m} \right\|^2 .$$

Denoting $\mathbf{P_i}$'s and $\mathbf{P}^*$'s kth coordinate as P_{ik} and P_k^*, respectively, the expectation of R, $\mathrm{E}[R]$, can be derived in the following lemma:

Lemma 15. *Given the swarm size, m, and the variance of distribution at time t, $\sigma_t^2 = \sigma^2$,*

$$E\left[\sigma_{t+1}^2\right] \le \frac{(m-1)\sigma^2}{12m} \left\{ \left(5 + \frac{\sqrt{3(m-1)}}{n}\right) c^2 - 6c + 12 \right\} .$$

Proof. Defining R_j as

$$R_j := \sum_{i=1}^{m} \left\{ P_{ij} + C_{ij}\left(P_j^* - P_{ij}\right) - \frac{\sum_{k=1}^{m}\left(P_{kj} + C_{kj}(P_j^* - P_{kj})\right)}{m} \right\}^2$$

yields $R = \sum_{j=1:n} R_j$ and that we can obtain for $j > 1$,

$$\begin{aligned} \mathrm{E}\left[R_j\right] &= \left(\frac{m-1}{12m}\right)\left\{\sigma^2\left(4c^2 - 6c + 12\right) + c^2\mathrm{E}\left[\left(P_j^*\right)^2\right]\right\} \\ &\le \left(\frac{m-1}{12m}\right)\left\{\sigma^2\left(5c^2 - 6c + 12\right)\right\} . \end{aligned}$$

The last inequality follows from the fact that the independence of coordinates implies $\mathrm{E}\left[\left(P_j^*\right)^2\right] \le \sigma^2$. Moreover,

$$E[R_1] = \left(\frac{m-1}{12m}\right)\left\{\sigma^2\left(4c^2 - 6c + 12\right) + c^2\mathrm{E}\left[\left(P_1^* - r\right)^2\right]\right\} .$$

Since $\mathrm{E}\left[\left(P_1^* - r\right)^2\right]$ is less than or equal to the expected value of the extreme order statistics of $T_1^2, T_2^2, \ldots, T_m^2$, where $T_i \sim N(0, \sigma^2)$, by using the upper bound for the extreme order statistics [28],

$$\mathrm{E}\left[\left(P_1^* - r\right)^2\right] \le \sigma^2\left(1 + \sqrt{3(m-1)}\right) .$$

As a consequence,

$$\mathrm{E}\left[\sigma_{t+1}^2\right] = \mathrm{E}\left[R\right]/(mn) \le \frac{(m-1)\sigma^2}{12m}\left\{\left(5 + \frac{\sqrt{3(m-1)}}{n}\right)c^2 - 6c + 12\right\} . \quad \square$$

While Lemma 15 is under the assumption that σ_t^2 is given or more formally, the conditional expectation $\mathrm{E}\left[\sigma_{t+1}^2 | \sigma_t^2 = \sigma^2\right]$ is derived, the following theorem indicates the relationship between $\mathrm{E}\left[\sigma_t^2\right]$ and $\mathrm{E}\left[\sigma_{t+1}^2\right]$ and gives a sufficient and necessary condition that the sequence $\left\{\mathrm{E}\left[\sigma_t^2\right]\right\}$ converges to zero. Without loss of generality for the normal operation of PSO, we assume that $\mathrm{E}\left[\sigma_0^2\right] < \infty$.

Theorem 16 (*Convergence of the Expectation of Variance*). *Let* $\kappa := \sqrt{3(m-1)}/n$. *If c satisfies the condition that*

$$\frac{3 - \sqrt{9 + \frac{60+5\kappa}{m-1}}}{5+\kappa} < c < \frac{3 + \sqrt{9 + \frac{60+5\kappa}{m-1}}}{5+\kappa},$$

$\lim_{t\to\infty}\left\{E\left[\sigma_t^2\right]\right\} = 0$.

Proof. The law of total expectation and Lemma 15 imply that

$$\mathrm{E}\left[\sigma_{t+1}^2\right] \le \frac{(m-1)}{12m}\left\{\left(5 + \frac{\sqrt{3(m-1)}}{n}\right)c^2 - 6c + 12\right\}\mathrm{E}\left[\sigma_t^2\right].$$

Therefore, $\left\{\mathrm{E}\left[\sigma_t^2\right]\right\}$ is upper-bounded by the geometric sequence with the first term $\mathrm{E}\left[\sigma_0^2\right]$ and the ratio

$$\frac{(m-1)}{12m}\left\{\left(5 + \frac{\sqrt{3(m-1)}}{n}\right)c^2 - 6c + 12\right\}.$$

By solving

$$\frac{(m-1)}{12m}\left\{\left(5 + \frac{\sqrt{3(m-1)}}{n}\right)c^2 - 6c + 12\right\} < 1,$$

the theorem is proved. $\square$

Since σ_t^2 takes the value on non-negative real numbers, the convergence of sequence $\left\{\mathrm{E}\left[\sigma_t^2\right]\right\}$ implies sequence $\left\{\sigma_t^2\right\}$ converges to zero *in probability*, as shown in the following corollary.

Corollary 17 (*Convergence of Variance*). *If* $\lim_{t\to\infty}\left\{E\left[\sigma_t^2\right]\right\} = 0$, *then* $\lim_{t\to\infty}\sigma_t^2 \xrightarrow{p} 0$, *i.e., for every* $\epsilon > 0$ $\lim_{t\to\infty}\mathrm{Prob}\left\{\sigma_t^2 \ge \epsilon\right\} = 0$.

Proof. Suppose for contradiction that there exists some $\epsilon > 0$ and $\delta > 0$ such that, for all $N_0 \in \mathbb{N}$, there exists an $N(N_0) > N_0$ with Prob $\left\{\sigma^2_{N(N_0)} \geq \epsilon\right\} \geq \delta$. However, since Prob $\left\{\sigma^2_{N(N_0)} \geq \epsilon\right\} \geq \delta$ implies $\mathrm{E}\left[\sigma^2_{N(N_0)}\right] \geq \epsilon\delta$, for all $N_0 \in \mathbb{N}$, there exists an $N(N_0) > N_0$ such that $\mathrm{E}\left[\sigma^2_{N(N_0)}\right] \geq \epsilon\delta$, $\lim_{t\to\infty}\left\{\mathrm{E}\left[\sigma^2_t\right]\right\} = 0$ is contradicted. □

Theorem 16 and Corollary 17 indicate that as long as the specified condition is satisfied, a swarm will converge in probability. However, it must be noted that the acceleration coefficient, c, used in this study is the coefficient for the compound effect of both the inertia weight and the common acceleration coefficient for the neighborhood or global best position as described in Section 3.3. Therefore, further investigations are needed to gain understandings on the compound effect and clarify the relationship of these parameters such that the derived results in the present work can be applied in practice.

5. Summary and conclusions

In this study, we made the first attempt to analyze the behavior of particle swarm optimization on the facet of particle interaction. We firstly proposed a statistical interpretation of particle swarm optimization and modeled the essential PSO mechanisms with the operations on probabilistic distributions. In order to investigate the PSO behavior based on particle interaction, we focused on the social-only model of PSO, in which the personal experience of particles is ignored. From the viewpoint of macrostates, we obtained the lower and upper bounds of the expected progress rate for a swarm on the sphere function. By examining in detail the variance of the particle distribution, we further showed that under certain condition, a swarm will converge in probability due to the mechanism of particle interaction, i.e., exchanging and sharing information, which is commonly believed to be an essential mechanism of PSO but seldom theoretically analyzed in the literature.

With regard to the practical implications of this study, we demonstrated that the optimization process of PSO can be interpreted as the interplay between the attractor and the overall swarm, as shown in Theorem 11 that the expected norm of the next center is upper-bounded by a linear combination of $\overline{P}$ and $\overline{P_{(1,m)}}$ as well as that the acceleration coefficient is the weight balancing the effects of these two quantities. The major resistance in the optimization process of PSO on the sphere function is the number of dimensions, as it can be observed in Corollary 14 that the progress rate deteriorates drastically with respect to the number of dimensions. On the other hand, the swarm size is the primary factor counteracting the increasing dimensions, for the exploratory capability of the swarm is augmented in accordance with the number of particles. It is noteworthy that in a variety of theoretical studies on PSO, the effect of the objective function has been rarely taken into consideration due to the assumption of fixed attractors. By means of characterizing a swarm as a unity, the analysis of the influence of the objective function becomes possible.

With this study, we propose an alternative way to analyze particle swarm optimization from the viewpoint of macrostates instead of tracing the trajectory of each particle. The immediate follow-up work of this study includes the clarification of the compound effect of the inertia weight and the neighborhood acceleration coefficient for carrying over the theoretical results to practice and for suggesting applicable parameter settings. Moreover, tighter bounds may be derived to more accurately describe the behavior of PSO, and a complete PSO model may be considered instead of the social-only model adopted in the present work. Finally, in the long run, a unified behavioral model of PSO might be established by integrating the theoretical results from the two ends – macrostates and microstates – such that better, more robust optimization frameworks can be accordingly designed and developed.

Acknowledgements

The work was supported in part by the National Science Council of Taiwan under Grant NSC 98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of 1995 IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[2] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of 1999 IEEE Congress on Evolutionary Computation, CEC 99, 1999, pp. 1945–1950.
[3] J. Kennedy, The behavior of particles, in: Proceedings of the 7th International Conference on Evolutionary Programming, 1998, pp. 581–589.
[4] E. Ozcan, C.K. Mohan, Analysis of a simple particle swarm optimization system, Intelligent Engineering Systems Through Artificial Neural Networks 8 (1998) 253–258.
[5] E. Ozcan, C. K. Mohan, Particle swarm optimization: surfing the waves, in: Proceedings of 1999 IEEE Congress on Evolutionary Computation, CEC 99, 1999, pp. 1939–1944.
[6] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[7] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceedings of the 7th International Conference on Evolutionary Programming, 1998, pp. 591–600.
[8] F. van den Bergh, An analysis of particle swarm optimizers, Ph.D. Thesis, University of Pretoria, 2002.
[9] K. Yasuda, A. Ide, N. Iwasaki, Adaptive particle swarm optimization, in: Proceedings of 1999 IEEE International Conference on Systems, Man and Cybernetics, 2003, pp. 1554–1559.
[10] Y.-L. Zheng, L.-H. Ma, L.-Y. Zhang, J.-X. Qian, On the convergence analysis and parameter selection in particle swarm optimization, in: Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003, pp. 1802–1807.

[11] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.
[12] V. Kadirkamanathan, K. Selvarajah, P.J. Fleming, Stability analysis of the particle dynamics in particle swarm optimizer, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 245–255.
[13] M. Jiang, Y. Luo, S. Yang, Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm, Information Processing Letters 102 (2007) 8–16.
[14] H. M. Emara, H. A. A. Fattah, Continuous swarm optimization technique with stability analysis, in: Proceedings of the 2004 American Control Conference, 2004, pp. 2811–2817.
[15] V. Gazi, K. M. Passino, Stability analysis of social foraging swarms, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 34 (1) (2004) 539–555.
[16] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997, pp. 303–308.
[17] S.S. Shapiro, M.B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3/4) (1965) 591–611.
[18] T.W. Anderson, D.A. Darling, Asymptotic theory of certain "goodness of fi" criteria based on stochastic processes, Annals of Mathematical Statistics 23 (2) (1952) 193–212.
[19] R. D'Agostino, E. S. Pearson, Tests for departure from normality. Empirical results for the distributions of b_2 and $\sqrt{b_1}$, Biometrika 60 (3) (1973) 613–622.
[20] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 4 (1997) 67–82.
[21] H.G. Beyer, The Theory of Evolution Strategies, Springer, 2001.
[22] J. Jägersküpper, Algorithmic analysis of a basic evolutionary algorithm for continuous optimization, Theoretical Computer Science 379 (3) (2007) 329–347.
[23] J.K. Patel, C.B. Read, Handbook of the Normal Distribution, 2nd ed., CRC Press, 1996.
[24] Y. S. Chow, H. Teicher, Probability Theory: independence, Interchangeability, Martingales, 3rd edition, Springer, 1997.
[25] D. B. Owen, A table of normal integrals, Communications in Statistics — Simulation and Computation B.9 (1980) 389–419.
[26] I. Rechenberg, Evolutionsstrategie — Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution, Frommann-Holzboog, Stuttgart, Germany, 1973.
[27] A. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley & Sons, 2005.
[28] D. Bertsimas, K. Natarajan, C.-P. Teo, Tight bounds on expected order statistics, Probability in the Engineering and Informational Sciences 20 (4) (2006) 667–686.

Detecting General Problem Structures with Inductive Linkage Identification

Yuan-wei Huang and Ying-ping Chen

Abstract— **Genetic algorithms and the descendant methods have been deemed robust, effective, and practical for the past decades. In order to enhance the features and capabilities of genetic algorithms, tremendous effort has been invested within the research community. One of the major development trends to improve genetic algorithms is trying to extract and exploit the relationship among decision variables, such as estimation of distribution algorithms and perturbation-based methods. In this study, we make an attempt to enable a perturbation-based method, inductive linkage identification (ILI), to detect general problem structures, in which one decision variable can link to an arbitrary number of other variables. Experiments on circular problem structures composed of order-4 and order-5 trap functions are conducted. The results indicate that the proposed technique requires a population size growing logarithmically with the problem size as the original ILI does on non-overlapping building blocks as well as that the population requirement is insensitive to the problem structure consisting of similar sub-structures as long as the overall problem size is identical.**

I. Introduction

As practical optimization frameworks, genetic algorithms (GAs) have shown properties of flexibility, robustness, and easy-of-use since they were proposed [1], [2]. These methods usually get good performance when the adopted genetic operators are aware of the relationship among decision variables. Crossover operators in early genetic algorithms are likely to break promising solutions of sub-problems, which are referred to as build blocks (BBs) [3]. As a consequence, the overall performance is greatly reduced, or the problem cannot be solved [4]. In order to alleviate this issue, in recent studies, crossover operators or equivalent mechanisms that maintain the structure and diversity of building blocks have been proposed, developed, and examined. These techniques significantly increase the performance of genetic algorithms. To provide the capability of appropriately and effectively handling sub-solutions/building blocks, two key mechanisms, building-block identification and building-block exchange, have to be utilized and integrated. In this study, we focus on the mechanism of building-block identification, generalize the concept regarding the detection of building blocks, and propose the use of *inductive linkage identification* [5] to detect general problem structures.

Most of building-block/linkage identifying methods proposed and utilized in previous studies can be broadly classified into the following three categories [6]:

1) Estimation of distribution algorithms;
2) Linkage learning techniques;
3) Perturbation-based methods.

Yuan-wei Huang and Ying-ping Chen are with the Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, TAIWAN (email: {ywhuang, ypchen}@nclab.tw).

In the first category, estimation of distribution algorithms construct probabilistic models from the selected individuals of the population and describe the relationship among decision variables in a statistical way [7]. Early studies assume no interaction among variables, such as the population-based incremental learning [8] and the compact genetic algorithm [9]. Subsequent researchers use conditional probabilities to capture pairwise and/or multi-variate interactions, e.g., the mutual information maximizing input clustering [10], Baluja's dependency tree approach [11], the bivariate marginal distribution algorithm [12], the factorized distribution algorithm [13], and the Bayesian optimization algorithm [14]. Methods in this category are usually quite efficient from the traditional viewpoint of computational cost in evolutionary computation because they do not need additional fitness evaluations. Nevertheless, less salient building blocks, which contribute little to the total fitness, are less statistically significant and therefore might be ignored and undetected [15].

For the methods of the second category, building-block identification is oftentimes viewed as the (gene/variable) ordering problem. By rearranging variables during the evolutionary process, interdependent variables are put closer according to the adopted coding scheme such that these variables are less likely to be split apart by subsequent operations. In these studies, the messy genetic algorithm [4] and its more efficient descent, the fast messy genetic algorithm [16], exploit building blocks to identify linkages. Since the rearranging mechanism often acts too slow to cooperate with the selection operator, such a condition usually leads to premature convergence. The linkage learning genetic algorithm [17] performs two-point crossover on a specifically designed circular chromosome representation such that tight linkages among related variables can be formed on the chromosome and preserved during the evolutionary process.

Methods in the last category analyze the fitness difference caused by perturbing variables to identify linkages. For example, the gene expression messy genetic algorithm [18] incorporates a special genotype for pairwise relations and a function involving perturbation to find linkage sets. Linkage identification by nonlinear check [6] uses the linear summation of different and non-overlapping building blocks to detect linkages. Borrowing the idea from estimation of distribution algorithms, the dependency detection for distribution derived from the fitness difference [15] clusters variables according to the fitness difference values caused by perturbation. Because a

perturbed variable only effects the building blocks containing itself, information can be obtained on less salient building blocks from fitness difference values. However, since extra fitness evaluations are required every time a variable is perturbed, methods in this category cost more function evaluations to identify linkages, although the actual overall computational cost might be less.

From the viewpoint of extracting the problem structure and exploiting the obtained information in order to conduct optimization, estimation of distribution algorithms can be considered as approaches at the "global" end or organizing the obtained information in a "top-down" manner. Estimation of distribution algorithms assume a probabilistic model and adjust the model parameters to fit the promising solutions. On the other hand, linkage learning techniques and perturbation methods are at the "local" end and processing the information in a "bottom-up" manner. These methods implicitly or explicitly extract information out of the selected individuals and recognize the problem structure parts by parts. In this study, we aim at enhancing the global problem structure detection capability of perturbation-based methods and at blurring the line between estimation of distribution algorithms and methods in the other two categories.

In particular, we firstly extend the notion of building blocks commonly adopted in perturbation-based methods from overlapping building blocks to general problem structures. Then, a linkage identification technique, called *inductive linkage identification* [5], utilizing the ID3 decision tree [19] is modified and adopted to detect global problem structures. Experiments on the scalability and flexibility are conducted to examine the capability of the modified inductive linkage identification. The results demonstrate that the proposed technique requires a population size growing logarithmically with the problem size. The population requirement is insensitive to the problem structure consisting of similar sub-structures as long as the overall problem size is identical.

For the remainder of this paper, the background of linkage identification is briefly introduced in section II. Why and how inductive linkage identification works are reviewed with illustrative examples in section III. Experiments and results are provided and discussed in section IV, followed by the summary and conclusions given in section V.

II. Linkages, Building Blocks, and Problem Structures

De Jong et al. [20] defined the term *dependency*, which is also referred to as *linkage*, as "two variables in a problem are interdependent if the fitness contribution or optimal setting for one variable depends on the setting of the other variable." Moreover, the *order* of a problem is also stated as "the order is the largest number of variables that are interdependent." To obtain the complete information of linkages, the contribution of each possible pair of variables needs to be examined. Although it is usually an expensive work to process all possible pairs of variables, dependencies should be examined as much as possible in a reasonable time such that the employed genetic algorithm can perform well.

The Schema theorem [1] states that short, low-order, and highly fit sub-solutions increase their market shares to be combined. Furthermore, the building block hypothesis [3] implies that combining small partial solutions is essential for genetic algorithms and also consistent with human innovation. According to these observations, a problem model called the *additive decomposable function* (ADF) and written as a sum of low-order sub-functions is proposed.

Let a string of length ℓ, $\mathbf{s} = s_1 s_2 s_3 \dots s_\ell$, present a solution, where $\mathbf{s}$ is a permutation of the decision variables $\mathbf{x} = x_1 x_2 x_3 \dots x_\ell$ determined by the adopted coding scheme. The fitness function for $\mathbf{s}$ is then defined as

$$f(\mathbf{s}) = \sum_{i=1}^{m} f_i(\mathbf{s}_{\mathbf{v}_i}) \ ,$$

where m is the number of sub-functions, $f_i(\cdot)$ is the i-th sub-function, and $\mathbf{s}_{\mathbf{v}_i}$ is the solution string for $f_i(\cdot)$. For example, if $\mathbf{v}_i = (4, 2, 3, 6)$, $\mathbf{s}_{\mathbf{v}_i} = s_4 s_2 s_3 s_6$. If $f_i(\cdot)$ is also a sum of other sub-functions, it can be replaced by these sub-functions. Therefore, without loss of generality, each $f_i(\cdot)$ can be assumed a non-linear function, and the number of variables of $f_i(\cdot)$ is referred to as its order, i.e., complexity. In the ADF model, variables in the same set $\mathbf{v}_i$ are interdependent. These sets referred to as *linkage sets*, and the related term *building block* (BB) is used for the candidate solutions to the corresponding sub-functions.

For complex problems, sub-functions are oftentimes overlapping. Similar to interdependent variables, shared variables affect the respective contributions of the overlapping building blocks to the total fitness of the problem and make these building blocks interdependent. Under such a circumstance, considering the interdependent sub-functions as either a single, longer building block or separate, shorter ones become inappropriate. Reviewing previous studies on pairwise interactions, building blocks, and order-k linkage sets, researchers attempt to capture structures of certain orders. However, if the overall structure can somehow be recognized as that obtained by the model building process in estimation of distribution algorithms, perturbation-based methods should also be able to provide sufficient understandings of the problem for those linkage-aware operations.

Therefore, in this paper, we firstly generalize the concept of overlapping building blocks to the notion of the *problem structure* such that interactions among variables can be described as general as possible. The term *sub-problem* is used to describe how the overall problem structure is constructed instead of decomposed. The terms *interaction* and *linkage* are still used for the dependency between any two variables.

III. Inductive Linkage Identification

In this section, the perturbation-based method called *inductive linkage identification* (ILI) is reviewed. Firstly, a brief introduction of the ID3 decision tree is given, followed by how ILI adopts ID3 into the fitness perturbation and linkage

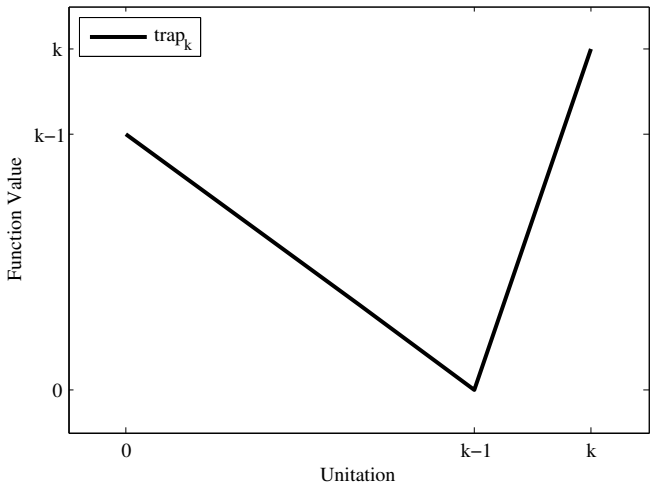


Fig. 1
A k-TRAP FUNCTION.

identification procedure. Then, we describe the modified version of ILI for detecting general problem structures. A simple example is also given for illustration.

A. ID3 for Recognizing Linkage

The ID3 decision tree construction algorithm is a supervised categorization method working on discrete data sets, in which the datum entries consist of several decision variables and each decision variable is limited to certain predefined values. ID3 aims to build a decision tree according to entropy and information gain. By eliminating the most useless variable, the training data set can be split into two subsets. One contains the datum entries using that variable, and the other does the rest. Then, the described procedure is applied to these two subsets recursively.

For the perturbation procedure, the fitness difference values, denoted as df, are obtained by subtracting the fitness value after perturbation from the original fitness value. This operation implicitly isolates the affected portions of the whole problem structure and reveals them as fitness difference values. The k-trap function [21], [22] is employed in this study as an illustrative example as well as the elementary sub-problem for composing larger problem instances:

$$trap_k(s_1 s_2 s_3 \dots s_k) = \begin{cases} k, & \text{if } u = k; \\ k - u - 1, & \text{otherwise.} \end{cases} ,$$

where u is the number of 1's in the solution string. Figure 1 shows the characteristic of a k-trap function.

With k-trap functions as elementary sub-problems, more complicated problem instances can be created following the ADF model. For example, an 8-bit function composed of a 3-trap and a 5-trap function can be defined as

$$f(s_1 s_2 s_3 \dots s_8) = trap_3(s_1 s_2 s_3) + trap_5(s_4 s_5 s_6 s_7 s_8) .$$

$\overline{s_1} s_2 s_3 \dots s_8$		f	df
111	01001	5	3
111	10100	5	3
111	01111	3	3
000	11111	7	1
000	00100	5	1
000	00001	5	1
000	10110	3	1
000	11100	3	1
001	01000	4	1
001	00011	3	1
001	00011	3	1
001	10100	3	1
010	01000	4	1
010	00100	4	1
010	01100	3	1
010	10100	3	1
010	00111	2	1
010	11011	1	1
100	00000	5	-1
100	00100	4	-1
110	11111	5	-1
110	01101	1	-1
110	01111	0	-1
110	11011	0	-1
101	10000	3	-1
101	01101	1	-1
101	11110	0	-1
011	00001	3	-3
011	00110	2	-3
011	01111	0	-3

TABLE I
RESULTS OBTAINED BY PERTURBING VARIABLE s_1.

By conducting perturbation on a binary variable s_1, the fitness difference df is obtained as

$$\begin{aligned} df &= f(s_1 s_2 s_3 \dots s_8) - f(\overline{s_1} s_2 s_3 \dots s_8) \\ &= (trap_3(s_1 s_2 s_3) + trap_5(s_4 s_5 s_6 s_7 s_8)) \\ &\quad - (trap_3(\overline{s_1} s_2 s_3) + trap_5(s_4 s_5 s_6 s_7 s_8)) \\ &= trap_3(s_1 s_2 s_3) - trap_3(\overline{s_1} s_2 s_3) . \end{aligned} \quad (1)$$

Equation (1) gives a mathematical explanation that df is only affected by the perturbed variable s_1 and those variables belonging to the same sub-problem as s_1. Table I is the example of Equation (1) and shows that permutations of s_1, s_2, and s_3 yield the identical df value.

ILI considers the distinct df values as the classification categories and each variable as the decision variable for the ID3 algorithm. By performing ID3 on the perturbed variable as the tree root, a decision tree is accordingly constructed. The internal nodes on the decision tree are then collected as a linkage set $\mathbf{V}_i$. Figure 2 shows the built decision tree corresponding to Table I, and the internal nodes s_1, s_2, and s_3 forms a linkage set.

B. Original ILI

The original ILI [5] can handle only those problem structures composed of non-overlapping sub-problems. After perturbing a variable and constructing a decision tree as shown in Figure 2, a linkage set is identified, and the used variables are removed from the variable set. The procedure of perturbation

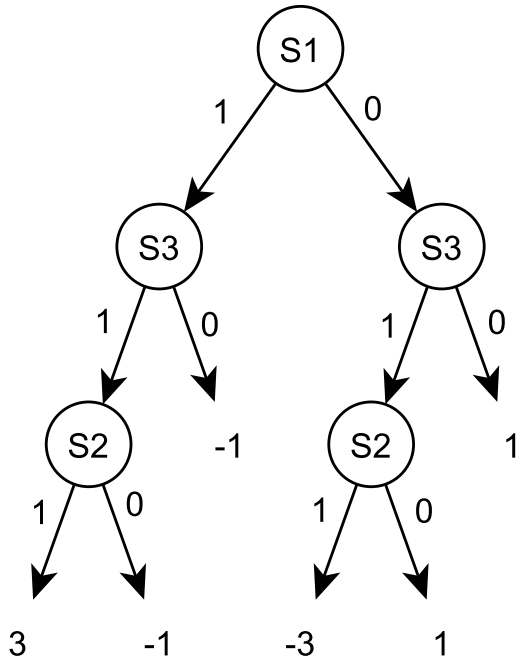
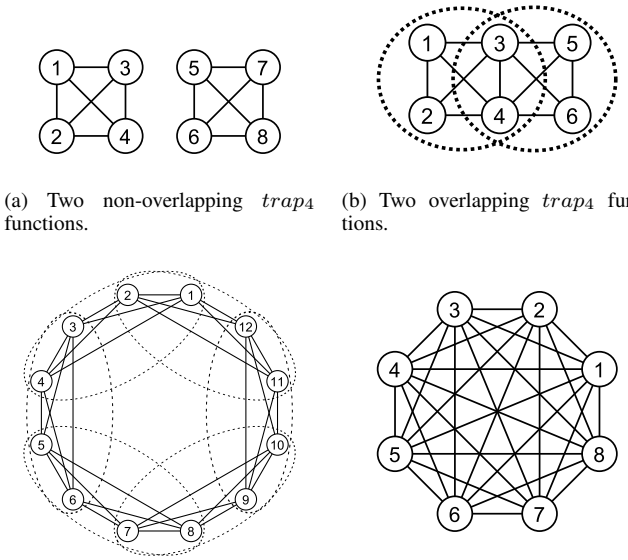


Fig. 2
THE DECISION TREE CONSTRUCTED FOR TABLE I.



(a) Two non-overlapping $trap_4$ functions.

(b) Two overlapping $trap_4$ functions.

(c) Circular structure.

(d) Complete graph.

Fig. 3
PROBLEM STRUCTURE EXAMPLES: VARIABLES OF SUB-PROBLEMS ARE CIRCLED BY DASHED ECLIPSES.

and decision tree construction is repeated on one of the uncategorized variables until all variables are categorized. Taking Table I for example, after $\mathbf{V}_1 = \{s_1, s_2, s_3\}$ is identified, ILI then perturbs variable s_4 and constructs a decision tree with $\{s_4, s_5, s_6, s_7, s_8\}$ and the next uncategorized variable. In this example, the final linkage sets are $\mathbf{V}_1 = \{s_1, s_2, s_3\}$ and $\mathbf{V}_2 = \{s_4, s_5, s_6, s_7, s_8\}$.

When there is no overlapping building blocks, experiments [23] demonstrate that the required population size grows sublinearly with the problem size while the complexity of subproblems is fixed. On the other hand, the population size requirement grows exponentially with the complexity of subproblem while the problem size is fixed. Such a result indicates that ILI is not sensitive to the overall problem size as well as the number of sub-problems but sensitive to the complexity of sub-problems.

C. Proposed Modifications on ILI

As mentioned in section II, overlapping sub-problems may form large, complicated problem structures and may be difficult or inappropriate to be identified as separate building blocks. Taking overlapping sub-problems $trap_4(s_1s_2s_3s_4)$ and $trap_4(s_3s_4s_5s_6)$ as an example, $\{s_1, s_2\}$ indirectly interacts with $\{s_5, s_6\}$ via $\{s_3, s_4\}$ since they belong to both of the sub-problems. These direct and indirect interactions do form a dependency structure of the two sub-problems. Instead of viewing them as either one building block or two, the actual structure should be found and reported to the subsequent linkage-aware operations.

In order to visualize these problem structures, a graph notation is adopted. Each variable is represented as a graph node, and direct interactions between any two variables are represented as edges between the corresponding graph nodes. E.g, Figure 3(a) shows the graph representation for two non-overlapping $trap_4$ functions because variables in the same sub-problems are interdependent, interactions among the four related variables are represented as a complete graph of four nodes. Since these two sub-problems are not overlapping, there exists no edge connecting the two separate sub-graphs. Figure 3(b) is the example for two overlapping sub-functions, $trap_4(s_1s_2s_3s_4)$ and $trap_4(s_3s_4s_5s_6)$, and shows that the shared variables interact with all other variables, while the unshared variables only interacts with the variables of the same sub-problems. Using the graph representation, complex dependency structures can be illustrated. E.g., Figure 3(c) shows a circular structure consisting of six overlapping $trap_4$ sub-problems with two shared variables between adjacent sub-problems. Figure 3(d) is the case in which each variable depends on all others to create a very complex overlapping problem structure.

To extend ILI to general problem structures composed of arbitrary overlapping sub-problems, a key modification on ILI is proposed. As described in section III-B, a variable is removed from the variable set V when it is categorized. Such an operation makes the removed variable invisible at later stages of ID3 and thus renders the linkages to other sub-problems undetectable. For example, thinking of Figure 3(b), when

Algorithm 1 Modified ILI for general problem structures.

procedure ILI(f, ℓ, n)
  Initialize a population P with n strings of length ℓ
  Evaluate the fitness of strings in P using f
  $V \leftarrow$ Shuffle$(1, 2, 3, \ldots, \ell)$
  $M_{\ell\times\ell} \leftarrow 0_{\ell\times\ell}$
  for each v in V **do**
    for each $s^i = s^i_1 s^i_2 s^i_3 \ldots s^i_l$ in P **do**
      Perturb s^i_v
      $df^i \leftarrow$ calculate the fitness difference
    end for
    Build an ID3 tree using (P, df) with v as root
    for each internal node v_j in the tree **do**
      $m_{v,j} \leftarrow 1$
      $m_{j,v} \leftarrow 1$
    end for
  end for
  Return the structure matrix M
end procedure

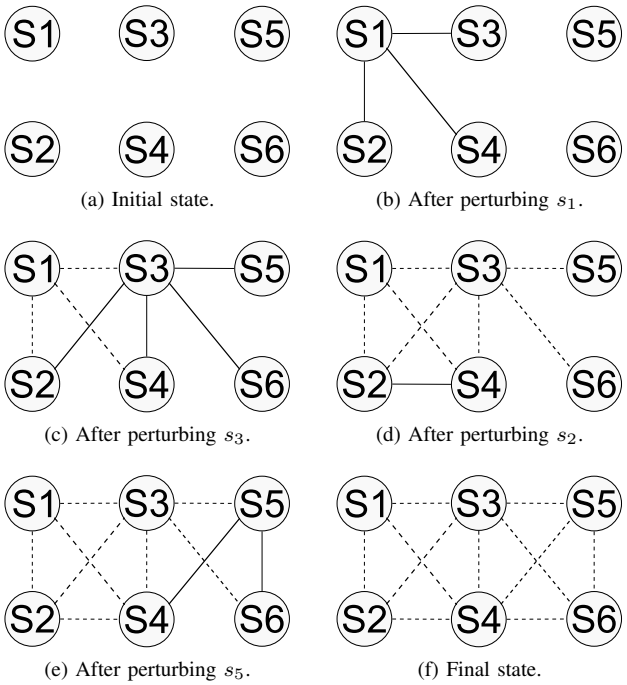
the perturbation and ID3 tree construction are performed on variable s_1, the resultant linkage set is $\{s_1, s_2, s_3, s_4\}$ and the rest elements are $\{s_5, s_6\}$ where the relations between $\{s_3, s_4\}$ and $\{s_5, s_6\}$ are lost. One of the proposed modifications is to perturb and perform ID3 on each variable s_i without removing any variable such that all variables can be examined repeatedly by ID3.

Another modification is to make ILI not directly return linkage sets corresponding to sub-problems, which are also referred to as building blocks. As aforementioned, the concept of building blocks is not very clear when sub-problems are overlapping. In order to determine the overall problem structure, a ℓ-by-ℓ matrix $M_{\ell\times\ell}$ is employed, where ℓ is the number of variables. The element $m_{i,j} = 1$ if there is a connection between variables s_i and s_j; otherwise, $m_{i,j} = 0$. In this study, we make linkages undirected. After s_i is perturbed and a linkage set containing s_j is constructed, not only $m_{i,j}$ but $m_{j,i}$ are also marked.

Algorithm 1 shows the modified inductive linkage identification procedure. For further illustration, the modified ILI is demonstrated by an example composed of two 4-trap functions with two shared variables defined as

$$f(s_1 s_2 s_3 s_4 s_5 s_6) = trap_4(s_1 s_2 s_3 s_4) + trap_4(s_3 s_4 s_5 s_6) \quad (2)$$

and shown in Figure 4. Initially, the structure matrix $M_{6\times6}$ is a zero-matrix indicating that there is no known interaction among any variables as showed in Figure 4(a). After initialization, ILI begins to perturb variables in a randomly determined order: s_1, s_3, s_2, s_5, s_4, and s_6. By perturbing and performing ID3 on variable s_1, a linkage set $\{s_1, s_2, s_3, s_4\}$ is recognized and indicates that s_1 interacts with s_2, s_3, and s_4. Figure 4(b) shows the detected partial structure. Notice that although s_2, s_3, and s_4 belong to the same sub-problem as defined in Equation (2), there is no interaction among them detected at the current iteration. Next, when s_3 is perturbed, ID3 identifies



(a) Initial state. (b) After perturbing s_1. (c) After perturbing s_3. (d) After perturbing s_2. (e) After perturbing s_5. (f) Final state.

Fig. 4
PROBLEM STRUCTURES DETECTED DURING THE ILI PROCESS. DASHED AND SOLID LINES REPRESENT KNOWN AND NEWLY DISCOVERED INTERACTIONS RESPECTIVELY.

that s_3 interacts with all other five variables since it belongs to both sub-problems as shown in Figure 4(c). The procedure is repeated on s_2, s_5, s_4, and s_6 sequentially. After all variables are proceeded, the final structure is constructed as shown in Figure 4(f).

Because there is no controlling parameter for the problem order/complexity, the obtained linkage information of the overall problem structure is unconstrained by any assumptions on the complexity of sub-problems. The only key factor in this condition regarding the correctness is whether or not the employed population is large enough for ILI to avoid getting confused by the fitness difference noise. Our preliminary experiments involving different problem structures have shown that the proposed ILI modification is able to construct correct problem structures as long as sufficiently large populations are utilized. For the purpose of gaining more understanding of the population size requirement by the modified ILI, in the next section, we design and conduct more experiments to observe the scalability and flexibility of the proposed modification.

IV. EXPERIMENTS AND RESULTS

Experiments and results on circular structures are examined in this section. Circular structures hold certain good properties for experimental control. The number of linkages increases linearly with the number of sub-problems and so does the number of nodes. These easily controlled properties enable us to concentrate on the population requirement.

The required population size is determined by a bisection method. For a given problem structure and a range of population sizes $[bound_L, bound_U]$, if the modified ILI can correctly detect the given problem structure for at least 29 times out of 30 independent runs with the population size $P_{size} = (bound_L + bound_U)/2$, we consider that P_{size} is large enough for the modified ILI to detect this problem structure and set P_{size} as the new $bound_U$ for the next iteration. Otherwise, P_{size} is too small to provide sufficient statistics, and thus, the next iteration will be conducted on interval $[P_{size}, bound_U]$. This bisection procedure repeats until the interval is smaller than 2, and the final mean value, P_{size}, is regarded as the required population size. For all the experiments in this study, the bisection process is performed for 50 interdependent trails, and the mean value and the standard deviation are calculated accordingly. Please note that in the experiments for simplicity, building blocks are arranged with consecutive variables on the chromosome, but they can actually be arbitrarily arranged because the same result will be obtained on all possible permutations of the variables.

A. Scalability on Circular Structures

In this series of experiments, the scalability of the modified ILI is examined by using the $trap_4$ and $trap_5$ functions with circular overlapping problem structures. In the experiments with $trap_4$, each sub-problem shares two variables with one of its neighbor sub-problem and the other two variables with the other neighbor. The circular overlapping structure can be described as

$$C4_n(s_1 s_2 s_3 \dots s_{2n}) = \sum_{i=1}^{n-1} trap_4(s_{2i-1} s_{2i} s_{2i+1} s_{2i+2}) + trap_4(s_{2n-1} s_{2n} s_1 s_2) \ ,$$

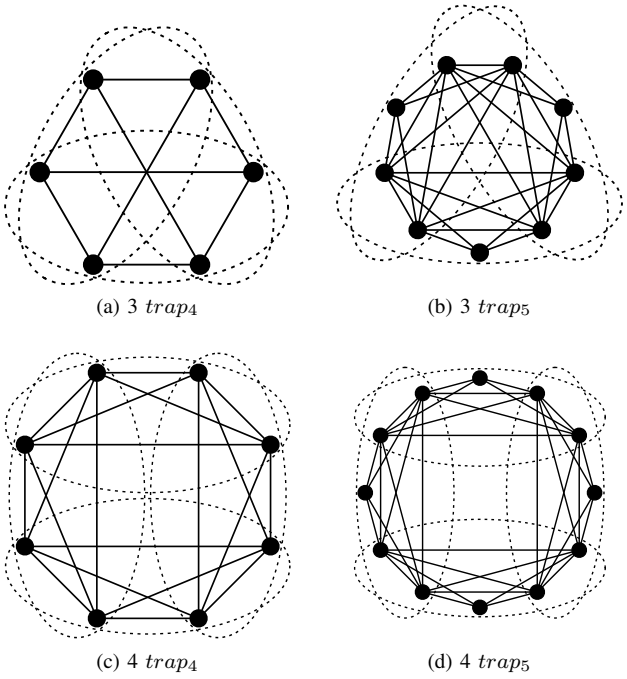
where n is the number of sub-problems and greater than 2 to form a circle. For example, $C4_3(s_1 s_2 s_3 \dots s_6) = trap_4(s_1 s_2 s_3 s_4) + trap_4(s_3 s_4 s_5 s_6) + trap_4(s_5 s_6 s_1 s_2)$ is the smallest circular problem structure for $trap_4$ under this definition as shown in Figure 5(a), and inserting one more sub-problem will form a structure shown in Figure 5(c).

The overlapping scheme for the $trap_5$ function is similar, except that each sub-problem has one unshared variable. For example, the minimal circular structure of 3 $trap_5$ functions, shown in Figure 5(b), can be put as

$$\begin{aligned} C5_3(s_1 s_2 s_3 \dots s_9) \quad &= trap_5(s_1 s_2 s_3 s_4 s_5) \\ &+ trap_5(s_4 s_5 s_6 s_7 s_8) \\ &+ trap_5(s_7 s_8 s_9 s_1 s_2) \ , \end{aligned}$$

where s_3, s_6, and s_9 are unshared.

The experimental results of $C4_n$ and $C5_n$ are shown in Figure 6. The results demonstrate that the modified ILI is capable of correctly detect linkages among variables even when the problem size gets large. The first observation is that the results can be well fitted by using logarithmic curves. Such a phenomenon implies that the required population size grows logarithmically with respect to the number of sub-problems and indicates the modified ILI is quite efficient and scalable on these problem structures in the experiments. The population size growth rate is similar to that required by the original ILI on non-overlapping building blocks as given in the literature [23]. Secondly, since the growth of required population sizes can be well fitted with logarithmic curves for both $trap_4$ and $trap_5$ functions, for the problems composed of traps, the modified ILI should require a population size growing logarithmically with the problem size.



(a) 3 $trap_4$ (b) 3 $trap_5$ (c) 4 $trap_4$ (d) 4 $trap_5$

Fig. 5
THE MINIMAL CIRCULAR PROBLEM STRUCTURES.

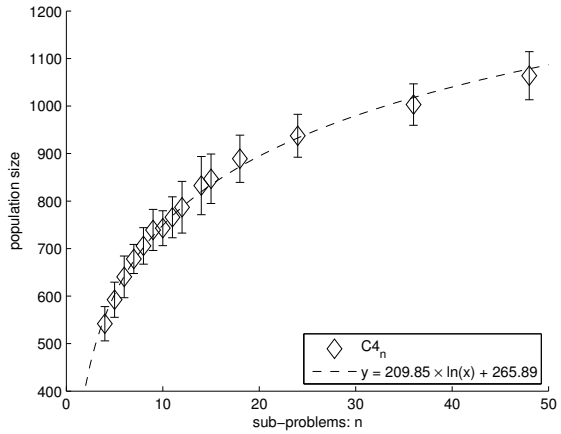
B. Insensitivity on Sub-structures

The series of experiments in this section aims to examine the capability of detecting problem structures composed of sub-structures. Separate circular problem structures form a large problem structure, and the population requirement of the modified ILI is compared to that for larger structures of the same problem size. In these experiments, circular problem structures composed of m smaller sub-structures are defined as

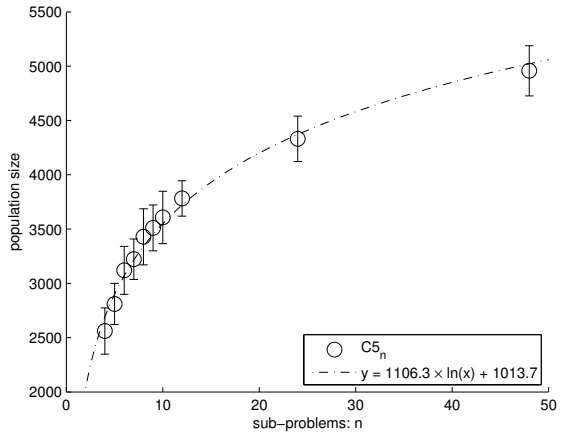
$$C4_n^m = \sum_{i=0}^{m-1} C4_n(s_{1+ni} s_{2+ni} s_{3+ni} \dots s_{2n+ni}) \ .$$

Figure 7 shows examples of circular structures composed of two and three sub-structures of five $trap_4$ sub-problems.

Figure 8 shows the experimental results of $C4_n$, $C4_n^2$, and $C4_n^3$. As shown in the figure, when the overall problem sizes are identical, the required population sizes of $C4_n^2$ and $C4_n^3$ are very close to that of $C4_n$. All the experimental results are also well fitted by the same logarithmic curve that fits the results of $C4_n$. It indicates that the modified ILI is able to



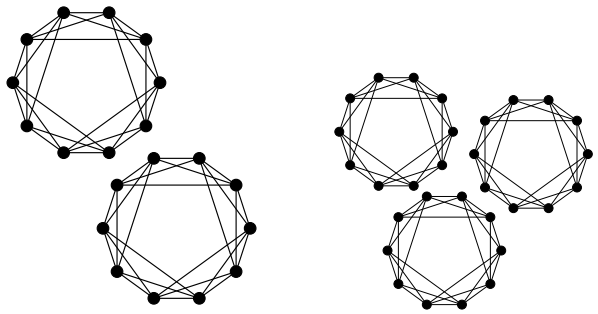
(a) $trap_4$ functions as sub-problems.



(b) $trap_5$ functions as sub-problems.

Fig. 6

THE POPULATION REQUIREMENT FOR THE MODIFIED ILI TO CORRECTLY DETECT CIRCULAR PROBLEM STRUCTURES COMPOSED OF $trap_4$ AND $trap_5$ FUNCTIONS.



(a) $C4_n^2$ (b) $C4_n^3$

Fig. 7

CIRCULAR PROBLEM STRUCTURES COMPOSED OF SEPARATE SUB-STRUCTURES, WHERE $n = 5$.

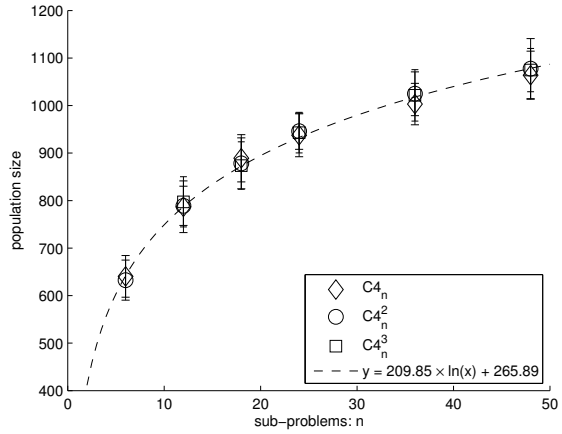


Fig. 8

CIRCULAR PROBLEM STRUCTURES COMPOSED OF ONE, TWO, AND THREE SUB-STRUCTURES WITH $trap_4$ AS THE ELEMENTARY SUB-PROBLEMS.

correctly identify the isolated as well as the interdependent parts of a large problem structure without additional cost.

V. SUMMARY AND CONCLUSIONS

In this paper, we extended the inductive linkage identification to detect general problem structures composed of overlapping sub-problems and conducted experiments by using circular overlapping structures for gaining more insights and understandings. According to the experimental observations, the proposed technique was found able to correctly detect circular problem structures and require a population size growing logarithmically with the problem size. The population requirement was observed insensitive to the problem structure consisting of similar sub-structures for the identical overall problem size.

One of the major differences between ILI and most of the other existing linkage learning methods is the absence of algorithmic parameters for the complexity of sub-problems. The proposed modification of ILI keeps this feature unchanged. Since ILI performs the task of linkage identification without assumptions on the problem structure, such as the chosen probabilistic model or the maximum degree of interactions, the relationship among variables should be extracted as authentic as possible.

Since the modified ILI is capable of detecting general problem structures, it may be applied in two ways. Firstly, by serving as a preprocessing step of genetic algorithms, the proposed techniques describes the variable dependencies with a graph such that delicately-designed genetic operators or processing mechanisms can utilize the linkage information to preserve the building blocks. Secondly, the proposed technique can be used as a tool to inspect and extract the relationship among decision variables for understanding the inner structure of the problem at hand in order to assist any further applicable operations.

ACKNOWLEDGMENTS

The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 1992, ISBN: 0-262-58111-6.

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

[3] ——, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, ser. Genetic Algorithms and Evolutionary 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] C.-Y. Chuang and Y.-p. Chen, "Linkage identification by perturbation and decision tree induction," in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 357–363.

[6] M. Munetomo and D. E. Goldberg, "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.

[7] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. binary parameters," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN IV)*, 1996, pp. 178–187.

[8] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep., 1994.

[9] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 287–297, November 1999.

[10] J. de Bonet, C. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Advances in Neural Information Processing Systems*, vol. 9, pp. 424–430, 1997.

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

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

[13] 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.

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

[15] M. Tsuji, M. Munetomo, and K. Akama, "Linkage identification by fitness difference clustering," *Evolutionary Computation*, vol. 14, no. 4, pp. 383–409, 2006.

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

[17] 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.

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

[19] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[20] E. D. De Jong, R. A. Watson, and D. Thierens, "On the complexity of hierarchical problem solving," in *Proceedings of Genetic and Evolutionary Computation Conference 2005 (GECCO-2005)*, 2005, pp. 1201–1208.

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

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

[23] C.-Y. Chuang and Y.-p. Chen, "Recognizing problem decomposition with inductive linkage identification: Population requirement vs. subproblem complexity," in *Proceedings of the Joint 4th International Conference on Soft Computing and Intelligent Systems and 9th International Symposium on advanced Intelligent Systems (SCIS & ISIS 2008)*, 2008, pp. 670–675.

XCS with Bit Masks

Jia-Huei Lin and Ying-ping Chen

***Abstract*—In this paper, a modified XCS is proposed to reduce the numbers of learned rules. XCS is a type of learning classifier systems and has been proven able to find accurate, maximal generalizations. However, XCS usually produces too many rules such that the readability of the classification model is greatly reduced. As a result, XCS users may not be able to obtain the desired knowledge or useful information from the learned rule set. In our attempt to handle this problem, a new mechanism, called *bit masks*, is devised in order to reduce the number of classification rules and therefore to improve the readability of the generated model. A series of n-bit multiplexer experiments, including 6-bit, 11-bit, and 20-bit multiplexers, to examine the performance of the proposed framework. For the problem composed of integer-typed variables, two synthetic oblique datasets, Random-Data2 and Random-Data9, are adopted to compare the performance of XCS and that of the proposed method. According to the experimental results, XCS with bit masks can perform similarly as XCS on n-bit multiplexers and generates significantly fewer rules on integer-typed problems.**

I. Introduction

Learning classifier systems (LCS) [1] are machine learning systems designed to combine reinforcement learning, evolutionary computation, and other heuristics to produce efficient adaptive systems. These rule-based machine learning algorithms originated and have evolved in the cradle of evolutionary computation and artificial intelligence. There have been a number of studies on the architecture and performance of LCS. In recent years, a simplified version of LCS, XCS [2], [3] has become one of the most important XCS variations since XCS was shown to be able to solve real-world classification problems with high accuracy. XCS is designed to evolve a representation of the best solution as well as to evolve a complete and accurate payoff map of all possible solutions for all possible problem instances. That is, XCS evolves rules that improve the ability to obtain the environmental reward and mine the environment for prediction patterns, which are expressed in the form of classifiers. The repeatedly refined prediction patterns allow the XCS system to make better decisions for consecutive actions.

However, some shortcomings still exist in XCS. For real-world applications, frequent pattern mining [4] often incurs numerous frequent item sets and rules, which much decrease the effectiveness of data mining since users have to go through a large number of mined rules in order to find useful ones. The great number of classification rules lowers the readability of the classification model in real-world applications. Since the XCS also produces numerous rules, in this study, XCS with bit masks is proposed to handle such a problem. The developed mechanism of bit masks is used to detect stable building blocks in classifiers and to prevent crossover and/or mutation operators from unnecessarily altering them. Consequently, the resultant classification model needs fewer rules than that evolved by the original XCS to achieve the same level of accuracy. A series of n-bit multiplexer experiments, including 6-bit, 11-bit, and 20-bit multiplexers, are exploited to examine the performance of the proposed method. For the integer-typed problems, two synthetic oblique datasets [5], Random-Data2 and Random-Data9, are used to compare the performance of XCS and that of the proposed method. According to the experimental results, XCS with bit masks can perform similarly as XCS on n-bit multiplexers and generates significantly fewer rules on integer-typed problems.

For the remainder of this paper, section II briefly reviews XCS. Section III introduces the representation and the algorithmic structure of XCS with bit masks. Section IV describes the experiments and provides the experimental results, followed by section V which concludes this paper.

Jia-Huei Lin and Ying-ping Chen are with the Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, TAIWAN (email: {jhlin, ypchen}@nclab.tw).

II. A Brief Review of XCS

In this paper, we firstly describe the framework of XCS, followed by an introduction of XCSI, which is an adaptation of XCS for integer-typed problems. Finally, we discuss the related work of this research.

A. XCS

XCS, introduced by Wilson in 1995 [2], is an important branch of LCS [1]. XCS has become known as one of the most reliable learning classier systems for handling data mining and machine learning problems. In this section, we give an overview of the key components of XCS, including the representation, the performance component, the reinforcement component, the discovery component, the macroclassifiers, and the covering and subsumption deletion.

1) Representation: XCS evolves a set of condition-action rules which are called the population of classifiers. The condition-action rules is the representation of the knowledge gained from the environment. Each classifier consists of five main components and several additional estimates.

- Condition: The condition part C checks if the classifier matches the environment event.
- Action: The action part A specifies the decided action when the condition matches the environment event.
- Payoff prediction: The payoff prediction p estimates the average payoff after executing the action in response to the environment event.
- Prediction error: The prediction error e estimates the average error of the payoff prediction.

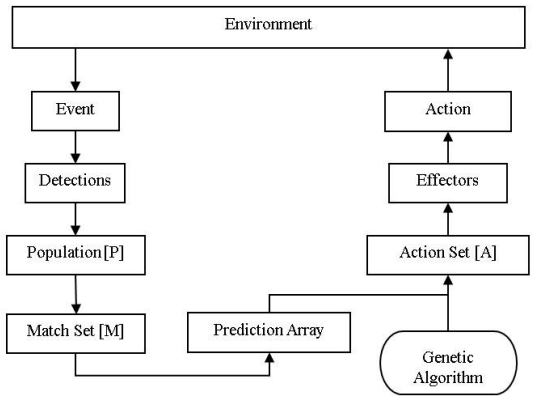


Fig. 1. The framework of XCS.

- Fitness: The fitness F reflects the scaled average relative accuracy of the classifier.

In the binary case, $C \in \{0, 1, \#\}^{\ell}$ given a problem of ℓ attributes. The symbol $\#$ represents the "don't care" condition. A defines a possible action or classification when the condition matches the environment event. p updates the results in a moving average measure of encountered payoff iteratively. Similarly, e estimates the moving average of the absolute error of the payoff prediction. F estimates the average of the accuracy of the payoff prediction of a classifier relative to other classifiers that are applicable at the same time.

2) Performance: The performance component presents the overall XCS framework, shown in Figure 1. The population of XCS starts with randomly generated classifiers or no classifier. When an event occurs, out of the whole population $[P]$, XCS forms a match set $[M]$ of classifiers which match the event. Then, the system prediction is measured for each action. The system prediction for each action is placed in the prediction array for action selection. If no classifier matches, a covering mechanism is applied to create classifiers that match each of the possible actions and place them in $[M]$. The system selects an action from the prediction array and forms an action set $[A]$. Finally, the chosen action is executed, and an environmental payoff may be returned.

3) Reinforcement: In this component, the parameters of classifiers in $[A]$ are adapted in order to achieve higher accuracy and to complete mappings of the problem space. The procedure of updating parameters is

1) The errors are updated: $\epsilon_j \leftarrow \epsilon_j + \beta(|P - p_j| - \epsilon_j)$.
2) The predictions are updated: $p_j \leftarrow p_j + \beta(|P - p_j|)$.
3) The accuracy of a classifier k_j is measured: $k_j \leftarrow 0.1 \times \exp[\ln \alpha(\epsilon_j - \epsilon_0)/\epsilon_j]$.
4) A relative accuracy $k_j^{'}$ of each classifier is determined: $k_j^{'} \leftarrow k_j / \sum k(A)$.
5) The fitness F_j are updated: $F_j \leftarrow F_j + \beta(k_j - F_j)$.

The definitions of symbols are

- α : The fall of rate in the fitness evaluation;
- β : Learning rate for updating fitness, prediction, prediction error, and action set size estimated in XCS classifiers;
- P : Environment return payoff;
- ϵ_j : Prediction error of classifier j;
- p_j : Prediction of classifier j;
- k_j : Accuracy of classifier j;
- F_j : Fitness of classifier j.

Butz et al. [6] indicated that the operation is faster for simple tasks if the prediction update comes before the error update, but this may create problems for complex tasks. Butz and Wilson [7] proposed that putting the error update before the prediction update seems to work better in complex cases.

4) Discovery: The discovery component is used to generate new classifiers. XCS executes a genetic algorithm (GA) on the current action set $[A]$ when the average time exceeds a threshold θ_{ga}. GA usually uses one-point crossover and bitwise mutation to generate new rules. Two classifiers are selected with a probability proportional to their fitness values first. After reproducing, crossing, and mutating the parent classifiers, two offspring classifiers are generated. The resulting offspring are inserted into the population $[P]$. If the size of population $[P]$ reaches the bound, a proposed method by Kovacs [8] can be adopted to determine the probability of deleting classifiers and to remove the low-fitness classifier.

5) Macroclassifiers: Macroclassifiers are a type of classifiers with the numerosity parameter num in XCS. A macroclassifier is used to speed up processing and provide a more perspicuous view of the population. Whenever XCS generates a new classifier, at the initialization step or at later stages, the population $[P]$ is scanned to examine whether the new classifier has the same condition and action as any existing macroclassifier does. If so, the new classifier is not actually inserted into the population, and the numerosity of the existing macroclassifier is incremented by one. Otherwise, the new classifier is added to $[P]$ with its own numerosity field set to one. Similarly, when a macroclassifier suffers a deletion, its numerosity is decremented by one, instead of being actually deleted. If the numerosity of a macroclassifier becomes zero, the system removes the macroclassifier from $[P]$.

6) Covering and Subsumption: Covering and subsumption are two important components of XCS. Covering is another method to introduce new classifiers into the population. When an environment event occurs and the match set does not contain all possible actions defined for the environment, the covering operation will generate classifiers to match this event for improving the accuracy. The condition of the new classifier created through covering is made to match the current event. Each attribute in the condition is mutated to "don't care" ($\#$) with a probability. Finally, the system puts the newly generated classifier into the population.

In addition to introducing new classifiers into the population, we also have to deal with rules of the same meaning in XCS. The subsumption operation is designed to make a rule that absorbs other rules if it is more general and to improve

the generalization capability of XCS. There are two forms of subsumption, GA-subsumption and Action-subsumption. In GA-subsumption, when new classifiers are generated, they are examined to see whether their conditions are subsumed by their parent classifiers or not. If the parent classifiers are more general than the new classifiers, the new classifiers are subsumed by the parents. The new classifiers will not be added to $[P]$, but the numerosity of the parent classifiers is incremented. Otherwise, the system puts the new classifiers into $[P]$. Action-subsumption is different from GA subsumption. Each action set is searched for the most general classifier R. All other classifiers in the set are compared to R to see whether R subsumes them. The subsumed classifiers are deleted from $[P]$.

7) Flow of XCS: Firstly, XCS initializes the rule set with zero reward randomly. There are four steps for the rule evaluation cycle. The steps are

1) The state of the environment is detected by detectors.
2) The system examines the condition part of each rule to determine the match set.
3) The match set will be grouped into different sets based on their own actions, and the prediction payoff for each action is calculated to determinate the chosen action.
4) Effectors implement the action in the environment, get the reward, and distribute it to the rules in the action set.

After a specified period of time, GA is executed to generate new rules and delete unfit rules in the rule discovery cycle. Wilson [3] indicated that they can find the classification rules with high accuracy with this framework.

B. XCSI

Since many problems involve integer attributes, a variation of XCS, called *XCSI* [5], for integer-typed problems is proposed. The modification in XCSI includes the presentation, mutation, covering, and subsumption. In XCSI, the presentation of the classifier condition part is changed from a string of $\{0, 1, \#\}$ to a concatenation of the interval predicates, $int_i = (l_i, u_i)$, where l_i and u_i are integers and denote the lower bound and the upper bound. A classifier matches an event x with attributes x_i if and only if $\forall x_i \; l_i \leq x_i \leq u_i$.

For the mutation operator in XCSI, Wilson indicated that the best method to mutate an allele is adding a value $\pm rand(m_0)$, where m_0 is a fixed integer, $rand$ picks an integer randomly from $(0, m_0]$, and the sign is selected equiprobably. The covering operator occurs if there is no classifier matches x. In XCSI, the new condition has components $\{l_0, u_0, \ldots, l_n, u_n\}$, where each $l_i = x_i - rand_i(r_0)$ and each $u_i = x_i + rand_i(r_0)$. r_0 is also a fixed integer and $rand_i$ picks an integer randomly from $[0, r_0]$. An interval predicate i subsumes another predicate j if $l_i \leq l_j$ and $u_i \geq l_j$. The subsumption of a classifier defined if every interval predicate in the first classifier's condition subsumes the predicate in the second classifier's condition.

III. XCS WITH BIT MASKS

As aforementioned, XCS is a promising methodology because of its versatility and capability. However, XCS is known to generate too many rules, which lower the readability of the resultant classification model. That is, the XCS user may be unable to get the needed knowledge or useful information out of the generated model.

XCS with bit masks is proposed in this study to handle such a problem. The proposed mechanism is used to detect stable building blocks in classifiers and to prevent crossover and mutation operators from unnecessarily altering these building blocks. Consequently, the resultant classification model needs fewer rules than that evolved by the original XCS to achieve the same level of accuracy.

In this section, we will firstly introduce the concept and mechanism of bit masks into XCS. Then, we discuss how bit masks are implemented in the XCS framework, followed by describing how XCS with bit masks is applied in different environments.

A. Representation

In order to introduce bit masks to XCS classifiers, the representation of XCS rules is modified to make capable of finding a set of stable building blocks composed of the attributes that should not be altered. For this purpose, a parameter, bit masks (BM), is added into the classifier representation as

$$\begin{aligned} <\text{Classifier}>::= &<\text{Condition}>:<\text{Action}>:<\text{BM}>: \\ &<\text{Payoff prediction}>:<\text{Payoff error}>: \\ &<\text{Fitness}> \end{aligned}$$

BM indicates the condition attributes that should not be altered in mutation and/or crossover operators. Rules with BM will be stabler than the standard XCS rules, and fewer classifiers will be created when the mutation and crossover operation is triggered. For example, if the rules of the condition and the action are set as Table I, attributes B and D are determined as stable building blocks in BM. Different from the standard XCS, when the mutation and crossover operation occur, the condition attributes in BM will not be altered to avoid generating redundant rules.

TABLE I
EXAMPLE OF A BIT MASK DATA SET (BM = {B, D}).

	A	B	C	D	E	Class
Event	1	0	1	0	1	2
Rule1	1	0	1	0	1	2
Rule2	#	0	1	0	1	2
Rule3	1	0	#	0	1	2
Rule4	1	0	1	0	#	2

The purpose of BM is to prevent unnecessary alteration. The rules generated by mutation and crossover operations in the standard XCS may not match the original event and some redundant rules might be created. Through the mechanism of bit masks, rules with BM can prevent stable building blocks from being altered. The collection of rules will strongly support the input event and may cover more subset of cases.

```
1: procedure FIND STABLE BUILDING BLOCKS (clset)
2:     clset: the current action set
3:     for i ← each condition attribute do
4:         isStable ← true
5:         for j ← each classifier in clset do
6:             if i of classifier j != i of input then
7:                 isStable ← false
8:             end if
9:         end for
10:        if isStable then
11:            attribute i is in a stable building block
12:        else
13:            attribute i is not in a stable building block
14:        end if
15:    end for
16: end procedure
```

Fig. 2. Find Stable Building Blocks.

B. Algorithmic Components of XCS with Bit Masks

In XCS, each rule contains one condition and one action, and the condition contains n attributes. Because of the relation between conditions and actions, attributes also have influence on actions. That is, when one attribute is changed, it may produce a different action. The connection between attributes and actions inspires the main idea of adopting bit masks. Given an environmental state, a match set will be formed in the usual way [9], and the action is chosen by the system. Once an action is chosen, the system forms an action set which consists of the classifiers in the match set advocating the chosen action. If the chosen action is the same as the environmental action, each attribute of the classifiers in the action set will be scanned. For all k, if the k-th attribute of classifiers in the action set is identical to the k-th attribute of the environmental input, the k-th attribute will be considered belonging a stable building block. The definition of variables and the pseudo code for **Find Stable Building Blocks** are shown in Figure 2.

The set of attributes, considered as a stable building block, called *bit masks* (BM). The current BM can be set to a classifier if the BM of that classifier has not been set. If the BM of a classifier has been set, it is modified by being compared with the current BM. If the current k-th attribute is also in the stable building block of the classifier, the k-th attribute will be kept. Otherwise, the k-th attribute will be removed, and the new, combined BM will be set to the classifier. The definition of variables and the pseudo code for **Set Stable Building Blocks** are shown in Figure 3.

The crossover and mutation operators in the GA component are modified for handling BMs. For mutation, the attributes in BM will not be mutated. For crossover, if the two condition attributes are in both BMs, the attributes will be fixed, and the other attributes will be exchanged to create offspring. The new classifier will be inserted into the population. If the number of classifiers is greater than that of the system events, the compensating deletion occurs as in the standard XCS.

```
1: procedure PROCEDURE SET BIT MASK(cl, BM)
2:     cl: classifier
3:     BM: BM found in the current action set
4:     cl.BM: BM of classifier cl
5:     if classifier cl has no BM then
6:         cl.BM ← BM
7:     else
8:         cl.BM ← BM ∩ cl.BM
9:     end if
10: end procedure
```

Fig. 3. Set Bit Mask

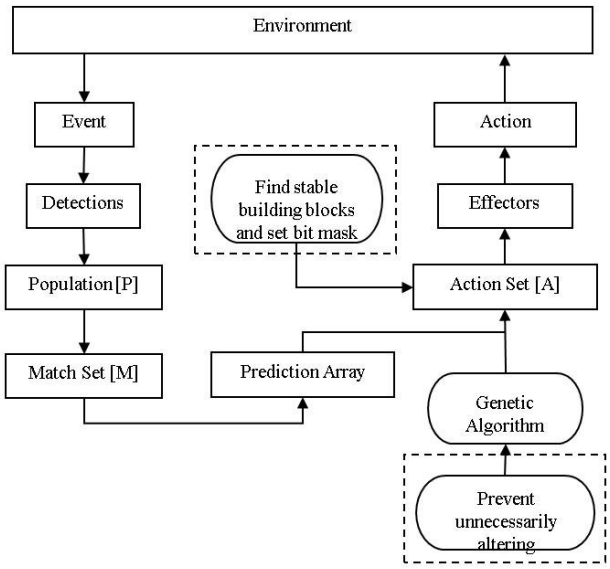


Fig. 4. Framework of XCS with bit masks.

C. Framework of XCS with Bit Masks

The overall framework of XCS with bit masks is shown in Figure 4. XCS with bit masks can be applied to problems in many domains and categories. In this paper, we focus on the bit mask mechanism and problems of classification. With the bit-mask capable representation and the corresponding operations, the flow of XCS with bit masks can be described as what follows. Firstly, we consider the data set of a classification problem as an environment. To simulate the occurrence of events, data items of the data set to classify are selected randomly or sequentially as the system input/event. The covering action on the match set is executed as the original XCS does. Thus, action sets can be formed. The BM mechanism is applied on the action set to detect stable building blocks and to record stable building blocks in classifiers. When the genetic algorithm component is triggered, the BM mechanism will prevent the attributes contained in the BM from being altered in mutation and crossover. Compared to the operations conducted in the standard XCS, the BM mechanism may result in generating fewer redundant rules.

IV. Experimental Results

We employ XCS and XCS with bit masks to handle three series of experiments and compare their performance, system errors, and population sizes. *Performance* refers to the fraction of the last 50 exploit trials that were correct. *System error* refers to the average of the absolute difference between the system prediction for the chosen action and the actual external payoff, divided by the total payoff range, which is 1000 in this study, over the last 50 exploit trials. *Population size* refers to the number of macroclassifiers. We use the XCS system implementation publicly available on the Internet [10]. The XCS system is modified to integrate with the BM mechanism. Each experiment is conducted for 200 independent runs, and the averaged statistics are reported.

A. Experimental Datasets

1) Boolean Multiplexers: Firstly, we use XCS and XCS with bit masks to tackle Boolean multiplexers of three different sizes: 6 bits, 11 bits, and 20 bits. Boolean multiplexers are defined for binary strings of length $\ell = k + 2^k$. The function value is determined by treating the first k bits as an address that indexes into the remaining 2^k bits, and the value of the indexed bit, either 0 or 1, is the function value.

2) Integer Test Functions: Secondly, we use XCS and XCS with bit masks to deal with integer datasets. The integer datasets are synthetic oblique data sets [5]. The first dataset, Random-Data2 is constructed by random vectors (x_1, x_2), with each x_i a random integer from $[1, 10]$. The outcome $o(x_1, x_2)$ for each vector is defined as

$$o(x_1, x_2) = \begin{cases} 1 & x_1 + x_2 \geq 11 \\ 0 & \text{Otherwise} \end{cases} . \quad (1)$$

An instance of Random-Data2 is composed by a vector and its outcome. The second integer dataset, Random-Data9, is constructed similar to Random-Data2 as follows. Random-Data9 has 9 dimensions, and the expression determining the outcome is defined as

$$o(\vec{x}) = \begin{cases} 1 & \sum x_i \geq 50 \\ 0 & \text{Otherwise} \end{cases} . \quad (2)$$

3) Wisconsin Breast Cancer (WBC): Finally, we use XCS and XCS with bit masks to handle a real-world problem, which is the Wisconsin Breast Cancer (WBC) database, donated to the UCI repository [11] by Prof. Olvi Mangasarian. WBC contains 699 instances collected over time by Dr. William H. Wolberg. Each instance in WBC has 9 attributes which are Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses. Each attribute has a value between 1 and 10 inclusive. Data rows look like

$$1000025, 5, 1, 1, 1, 2, 1, 3, 1, 1, 2$$
$$1017122, 8, 10, 10, 8, 7, 10, 9, 7, 1, 4$$
$$1016277, 6, 8, 8, 1, 3, 4, 3, 7, 1, 2$$

The first number is a label, the following 9 numbers are the attributes, and the last number is the class level, where 2 stands for Benign and 4 for Malignant.

TABLE II
Experimental parameters for Boolean multiplexers

	N	α	β	γ	θ_{ga}	ε	χ	μ	$P_{\#}$
6-bit	400	0.1	0.2	0.95	25	10	0.8	0.04	0.5
11-bit	800	0.1	0.2	0.95	25	10	0.8	0.04	0.5
20-bit	1600	0.1	0.2	0.95	25	10	0.8	0.04	0.5

B. Results for Boolean Multiplexers

Boolean multiplexers of three different sizes, 6-bits, 11-bits, and 20-bits, are experimented on in this series of experiments, and the parameters are listed in Table II.

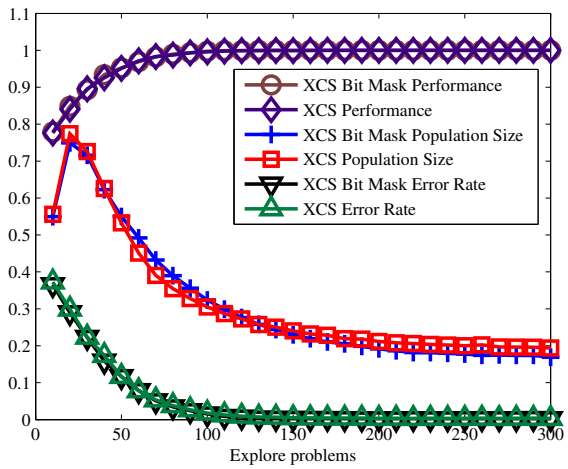
1) 6-bit Multiplexer: Figure 5(a) shows the experimental results for the 6-bit Boolean multiplexer. As we can observe, XCS gets approximately 100% performance in around 8000 exploit trails, and XCS with bit masks also gets approximately 100% performance in around 8000 exploit trails. For the system error, XCS gets approximately 0.5% system error in around 8000 exploit trails, and XCS with bit masks gets approximately 0.3% system error in around 8000 exploit trails. Finally, XCS evolves a population with 29.47 classifiers on average, and XCS with bit masks evolves a population with 25.01 classifiers on average.

We can find that XCS and XCS with bit masks can achieve similar performance and system error rate the number of exploit trails is appropriate. Hence, that XCS and XCS with bit masks have the same speed of convergence is shown, and the effect of integrating bit masks into XCS appears. XCS with bit masks can save on average 15.13% of the population size for the 6-bit multiplexer over 200 runs.

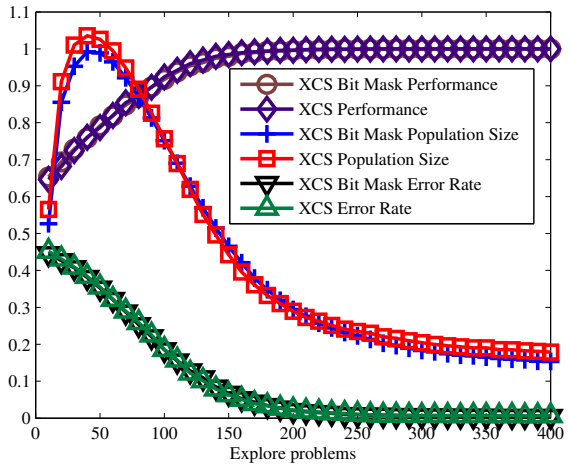
2) 11-bit Multiplexer: Figure 5(b) shows the results for the 11-bit Boolean multiplexer. From the figure, the performance of XCS reaches approximately 99% in around 9000 exploit trails, and the performance of XCS with bit masks also reaches approximately 99% in around 90000 exploit trails. The system error of XCS gets approximately 1% in around 13000 exploit trails, and the system error of XCS with bit masks gets approximately 0.8% in around 13000 exploit trails. For the population size, XCS creates 81.51 classifiers on average, and XCS with bit masks creates 74.85 classifiers on average.

From the experimental results, we can know that the outcome for the 11-bit Boolean multiplexer is similar to that for the 6-bit one. In this experiment, XCS with bit masks saves on average 8.17% of the population size over 200 runs.

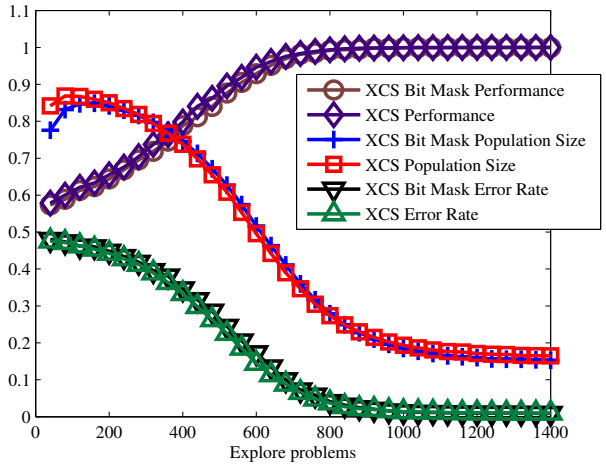
3) 20-bit Multiplexer: Figure 5(c) demonstrates the experimental results for the 20-bit Boolean multiplexer. From the result, XCS gets approximately 99% performance in around 39000 exploit trails, and XCS with bit masks also gets approximately 99% performance in around 39000 exploit trails. XCS gets approximately 1% system error in around 60500 exploit trails, and XCS with bit masks also gets approximately 1% system error in around 60500 exploit trails. For the population size, XCS evolves a population with 261.52 classifiers on average, and XCS with bit masks evolves a population with 247.67 classifiers on average. XCS with bit masks saves on average 5.30% of the population size.



(a) 6-bit multiplexer. Population sizes are divided by 150.



(b) 11-bit multiplexer. Population sizes are divided by 500.



(c) 20-bit multiplexer. Population sizes are divided by 1600.

Fig. 5. Results for Boolean multiplexers averaged over 200 runs. Explore problems are in exploit trails divided by 50.

TABLE III
EXPERIMENTAL PARAMETERS FOR INTEGER TEST FUNCTIONS

	N	α	β	γ	θ_{ga}	ε	χ	μ	$P_{\#}$
RD2	400	0.1	0.2	0.95	25	10	0.8	0.04	0.5
RD9	800	0.1	0.2	0.95	25	10	0.8	0.04	0.5

C. Results for Integer Test Functions

Two integer datasets, Random-Data2 and Random-Data9 [5], are experimented on in this series of experiments, and the parameters are listed in Table III.

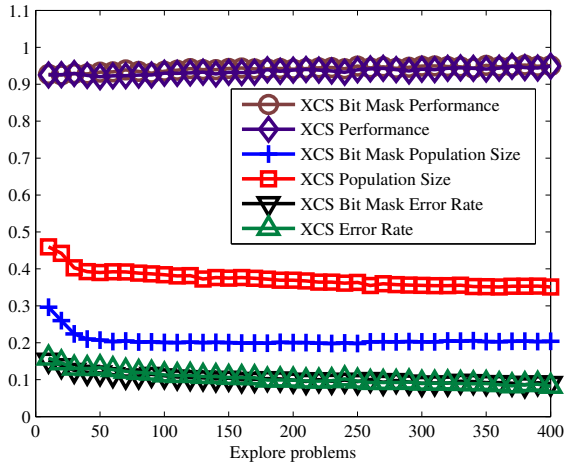
1) Random-Data2: Figure 6(a) shows the experimental results for the 2-dimensional integer dataset, Random-Data2. XCS gets approximately 94% performance in around 26000 exploit trails, and XCS with bit masks also gets approximately 94% performance in around 26000 exploit trails. For the system error, XCS gets approximately 10% system error in around 18000 exploit trails, and XCS with bit masks also gets approximately 10% system error in around 18000 exploit trails. For the population size, XCS evolves a population with 51.3 classifiers on average, and XCS with bit masks evolves a population with 32.73 classifiers on average.

Based on the experimental results for Random-Data2, we can find that XCS and XCS with bit masks provide similar performance and system error rate on handling the synthetic oblique data. However, for the population size, the effect of adopting bit masks becomes clear and significant in this experiment. As we can see, XCS with bit masks saves on average 36.20% of the population size over 200 runs.

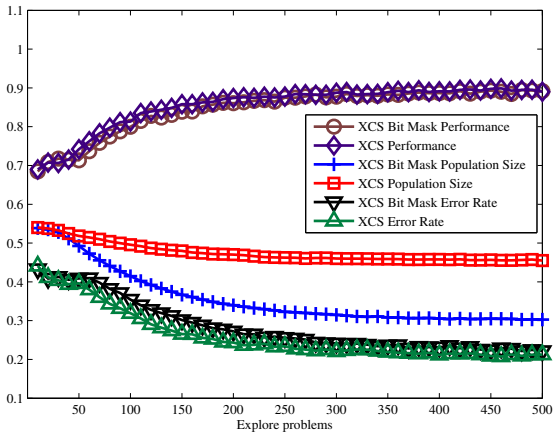
2) Random-Data9: Figure 6(b) demonstrates the experimental results for the 9-dimensional integer dataset, Random-Data9. Firstly, For the performance, XCS gets approximately 88.5% performance in around 37000 exploit trails, and XCS with bit mask gets approximately 87% performance in around 37000 exploit trails. Secondly, XCS gets approximately 23% system error in around 26000 exploit trails, and XCS with bit masks gets approximately 25% system error in 26000 exploit trails. XCS evolves a population with 542.38 classifiers on average, and XCS with bit masks evolves a population with 356.23 classifiers on average. We can observe that XCS with bit masks sacrifices little performance and saves on average 34.32% of the population size over 200 runs.

D. Results for the WBC database

Figure 7 shows the experimental results for the WBC database. XCS gets approximately 95% performance in around 7000 exploit trails, and XCS with bit masks also gets approximately 95% performance in around 7000 exploit trails. For the system error, XCS gets approximately 10% system error in around 7000 exploit trails, and XCS with bit masks also gets approximately 10% system error in around 7000 exploit trails. For the population size, XCS finally evolves a population with 271.23 classifiers on average, and XCS with bit masks finally evolves a population with 94.58 classifiers on average. Parameters of the experiment are $N = 400$, $\alpha = 0.1$,



(a) 2 dimensions. Population sizes are divided by 150.



(b) 9 dimensions. Population sizes are divided by 800.

Fig. 6. Results for integer test functions averaged over 200 runs. Explore problems are in exploit trails divided by 100.

$\beta = 0.2$,$\gamma = 0.95$, $\theta_{ga} = 25$, $\varepsilon_0 = 10$, $\chi = 0.8$, $\mu = 0.04$, and $P_{\#} = 0.5$.

Based on the experimental results for the WBC database, we can find that XCS and XCS with bit masks obtain similar performance and system error rate, while for the population size, the effect of adopting bit masks becomes remarkably significant that XCS with bit masks saves on average 65.13% of the population size over 200 runs.

In order to further justify the results on the WBC database, a tenfold cross-validation test is conducted on the WBC database. Table IV shows the results. We can observe that on average, XCS gets an accuracy of 93.20%, and XCS with bit masks gets 92.50%. The cross-validation reveals that compared to XCS, XCS with bit masks trades in an insignificant amount of performance for a significant save on the population size.

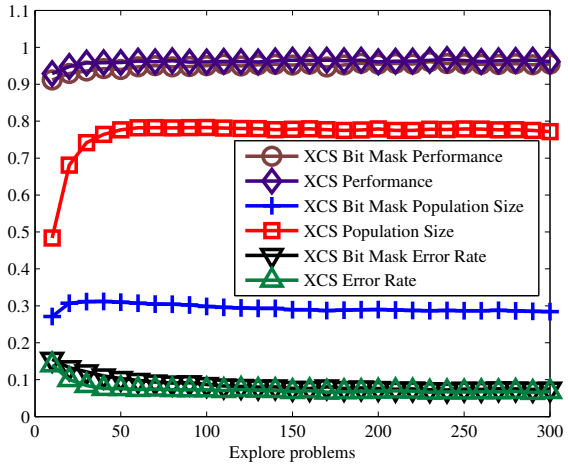


Fig. 7. Experimental results for the WBC database averaged over 200 runs. Population sizes are divided by 350. Explore problems are in exploit trails divided by 100.

TABLE IV
RESULTS OF A TENFOLD CROSS-VALIDATION ON THE WBC DATABASE.

	XCS	XCS with bit masks
#1	0.94	0.90
#2	0.94	0.91
#3	0.97	0.94
#4	0.91	0.93
#5	0.90	0.96
#6	0.96	0.91
#7	0.96	0.90
#8	0.90	0.91
#9	0.93	0.96
#10	0.91	0.93
Avg.	0.9320	0.9250

E. Discussion

Based on the experiment results presented in the previous sections, we can find two interesting points. The bit mask mechanism can help XCS to save the population size more in integer domains than it can in Boolean domains. As for the 6-bit, 11-bit, and 20-bit Boolean multiplexers, XCS with bit masks only saves less than 20% of the population size, while in integer domains, it saves more than 30%, or even more than 60% for the WBC database, of the population size.

Thus, we can know that the difference between Boolean attributes and integer attributes is quite significant in classification problems. When XCS is applied to handle Boolean multiplexers, the representation of rules is $\{0, 1, \#\}$, and it is relatively easy to for the classification system to gain knowledge from the environment by matching events and modifying rules. However, in integer domains, the rule representation is $int_i = (l_i, u_i)$, where l_i and u_i are integers, denoting the lower bound and the upper bound. A rule matches an event x if and only if $l_i \leq x_i \leq u_i$ for all attribute x_i. It is easy to see that flexibility embedded in the representation not only help to handle the high cardinality of integers but also

introduces difficulties for the classification system to learn. When integrating the bit mask mechanism into XCS, stable building blocks within rules are preserved. As a consequence, redundant rules are not necessary in this case to counter the destructive effect of crossover/mutation operators, and the number of rules can be greatly reduced.

Furthermore, it is worth noting that the bit mask mechanism save 65% rules for the WBC database but saves only 30-40% rules in the two synthetic oblique integer datasets. According to the idea of adopting bit masks in XCS, such a situation is quite reasonable. The class level of the synthetic datasets, Random-Data2 and Random-Data9, is solely determined by the sum of the integer attributes. If the sum of attributes is greater than the given threshold, the class level is set to 1 and otherwise 0. The attributes under such a class-attribute relationship actually have no particular relationship among subsets of attributes. However, in the case of the WBC database, it is believed that the class may be determined by certain combinations of attributes. Since XCS with bit masks has mechanisms to capture and model relationships among subsets of attributes, fewer rules are needed to describe the total class-attribute mapping of this problem. It is the reason why XCS with bit masks works better on the WBC database than it does on Random-Data2 and Random-Data9.

V. Conclusions

In this paper, we firstly reviewed XCS, followed by the introduction of the concept of bit masks. After integrating bit masks into XCS, we described the purpose of the mechanism of bit masks and show the framework of XCS with bit masks in detail. Finally, we implemented XCS with bit masks by modifying an existing XCS implementation provided by Martin V. Butz and conducted experiments on Boolean multiplexers, integer datasets, and a real-world problem by using both XCS and XCS with bit masks. After obtaining and observing the experimental results, two major points were discussed. In our study, XCS with bit masks performed better in integer domains than it did in Boolean domains. Since the mechanism of bit masks is to capture and model the relationship among attributes and to help XCS construct the mapping between attributes and classes, XCS with bit masks handled the real-world data better than it did the synthetic oblique datasets. The experimental results confirmed that the mechanism of bit masks can detect stable building blocks to avoid unnecessary alteration and thus deliver classification models of a slightly lower accuracy with much fewer rules.

Much work along this line of research is needed, in a variety of environments, to gain better understanding of the technique of using bit masks. In particular, bit masks do not deliver better performance in simple datasets such as Boolean multiplexers. There may be other ways to assist bit masks to improve the XCS framework. In addition to Boolean and integer datasets, real-world datasets should also be carefully examined. As for the bit mask itself, some research topics and directions, including theoretical understanding and algorithmic improvement, are waiting to be explored. Studies relating to these topics should be continuously pursued and conducted in order to develop classification systems that are not only feasible in theory but also viable in practice to further advance all the related domains and disciplines.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] J. H. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, 1975, ISBN: 0-2625-8111-6.

[2] S. W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[3] ——, "Generalization in the XCS classifier system," in *Proceedings of the Third Annual Conference on Genetic Programming (GP 98)*, 1998, pp. 665–674.

[4] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, November 2005, ISBN: 1558609016.

[5] S. W. Wilson, "Mining oblique data with XCS," *Lecture Notes in Computer Science*, vol. 1996, pp. 158–176, 2000.

[6] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "How XCS evolves accurate classifiers," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 927–934.

[7] M. V. Butz and S. W. Wilson, "An algorithmic description of XCS," *Lecture Notes in Computer Science*, vol. 1996, pp. 253–272, 2000.

[8] T. Kovacs, "Deletion schemes for classifier systems," in *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, July 1999, pp. 329–336.

[9] S. W. Wilson, "ZCS: A zeroth level classifier system," *Evolutionary Computation*, vol. 2, no. 1, pp. 1–18, 1994.

[10] M. V. Butz, "Java implementation of XCS," 2000, ftp://ftp-illigal.ge.uiuc.edu/pub/src/XCSJava/XCSJava1.0.tar.Z.

[11] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998, http://www.ics.uci.edu/˜mlearn/MLRepository.html.

ESTIMATION OF DISTRIBUTION ALGORITHMS: BASIC IDEAS AND FUTURE DIRECTIONS

YING-PING CHEN
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
ypchen@nclab.tw

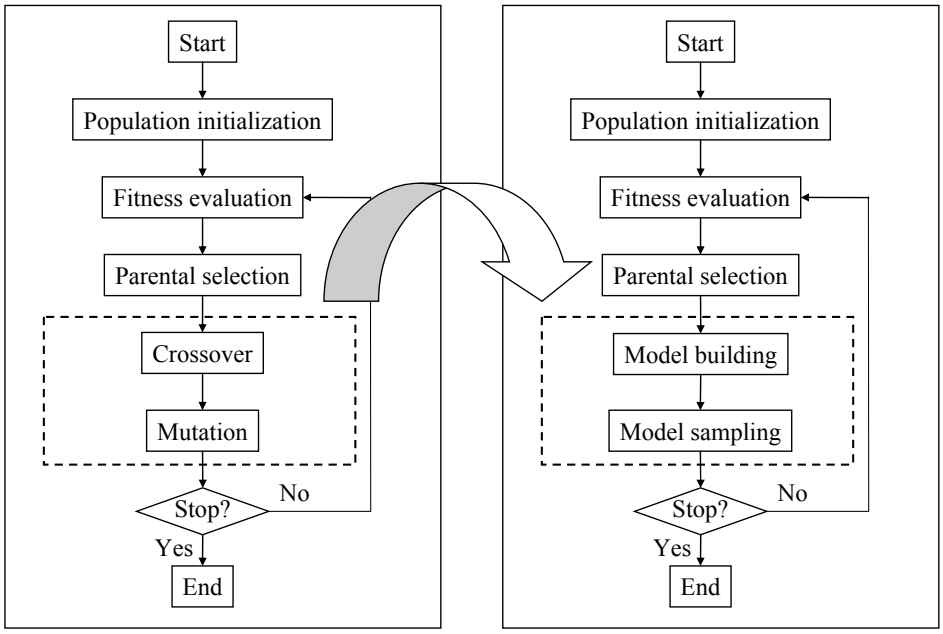
ABSTRACT—
Estimation of distribution algorithms (EDAs) are a class of evolutionary algorithms which can be regarded as abstraction of genetic algorithms (GAs) because in the design of EDAs, the population, one of the GA distinctive features, is replaced by probabilistic models/distributions. Building and sampling from the models substitute for the common genetic operators, such as crossover and mutation. Due to their excellent optimization performance, EDAs have been intensively studied and extensively applied in recent years. In order to interest more people to join the research of EDAs, this paper plays as an entry level introduction to EDAs. It starts with introducing the origination and basic ideas of EDAs, followed by presenting the current EDA frameworks, which are broadly applied in many scientific and engineering disciplines. Finally, this paper also describes some ongoing topics and potential directions in the hope that readers may get further insights into EDAs.

Key Words: Estimation of distribution algorithm, probabilistic model building genetic algorithm, global optimization, evolutionary algorithm, evolutionary computation, computational intelligence.

1. INTRODUCTION

Genetic algorithms (GAs) were proposed by Holland [1] with the inspiration of Darwinian view on the evolutionary mechanisms in nature. They were initially designed for generating classifiers in learning classifier systems as well as handling combinatorial optimization problems. Brought to the attention of many researchers by Goldberg's book [2], genetic algorithms have been widely and successfully applied to solving all kinds of search and optimization problems existing in numerous disciplines for the past decades. The proposal of genetic algorithms is remarkably intriguing because it strongly connects several seemingly not-so-related fields, such as biology, mathematical programming (optimization), artificial intelligence, etc., places itself in a unique position among these fields to stir innovations, and makes a major contribution to the creation of evolutionary computation. Similar to the progress of most scientific and engineering development, soon after its birth, the GA taskforce splits and focuses on topics of different origins and requirements. Some researchers explore potential applications of GAs, while others try to improve GA performance by incorporating natural, biological mechanisms or by advancing algorithmic designs with mathematical techniques. Among these attempts to devise better genetic algorithms or, more broadly, evolutionary algorithms, is the development of estimation of distribution algorithms.

By focusing on the performance and discarding the biological plausibility, estimation of distribution algorithms (EDAs) successfully achieve the design goal and can be viewed as abstraction of GAs because in EDAs, the population, one of the GA distinctive features, is replaced by some mathematical construction, and genetic operators are correspondingly changed to work with the adopted mathematical construction. According to the traditional GA performance indicator, function evaluations vs. solution quality, EDAs outperform GAs in most cases because the design of EDAs makes the search explicitly centralized by processing global statistics. Even if the significant computational cost of the mathematical construction is taken into consideration, the performance of EDAs is still usually superior to that of GAs. Thanks to their desirable features and properties, EDAs have been studied, improved, and broadly utilized for more than fifteen years. Given their importance in evolutionary computation and usefulness in application domains, an entry level introduction to EDAs is needed for those interested in getting familiar with and utilizing EDAs in a short time. Consequently, this paper is written to fulfill such a purpose. In particular, basic ideas, existing frameworks, and potential research directions of EDAs are briefly described. Note that this paper is not intended to be a complete survey or to provide details of EDAs. Interested readers may refer to [3, 4].



(a) Genetic Algorithm (b) Estimation of Distribution Algorithm

Figure. 1 Diagrams for a simple genetic algorithm and a basic estimation of distribution algorithm

This paper is organized as follows. Section 2 introduces the origination and basic ideas of EDAs, and section 3 presents existing EDA frameworks according the adopted mathematical construction. Section 4 describes some of the recent research issues of EDAs as the future directions, followed by section 5 which summarizes and concludes this paper.

2. BASIC IDEAS AND ORIGINATION

In this section, we will start with revisiting genetic algorithms and presenting the basic ideas of estimation of distribution algorithms, followed by a brief history of estimation of distribution algorithms.

2.1 Genetic Algorithms

Genetic algorithms are a class of evolutionary algorithms developed for conducting search and optimization by mimicking the evolutionary process in biology. GAs, use a population of solutions, called *individuals*, to gather the information regarding the search space and to implicitly process the statistics [3] in order to find the optimal solutions. Figure 1(a) shows a genetic algorithm in its simplest form. In the beginning, a solution population is initialized by random generation. Each of the individuals is evaluated by the fitness function to indicate how well it "fit" the environment, i.e., the optimization problem at hand. The individuals with better fitness have better chances to reproduce their offspring, and the parental selection process implements the idea of natural selection on the procreation side. The process to create new individuals is designed by emulating the recombination (crossover) and alteration (mutation) of genetic materials. After the next generation of individuals is created, each individual is also evaluated by the fitness function, and the GA procedure repeats until certain stop criterion is satisfied. The operation can be considered as explicitly sampling the search space and implicitly exploiting the obtained information.

2.2 Probabilistic Models vs. Populations

As we can see in Figure 1(a), the components of which the functionality is implicitly processing and exploiting the information, i.e., the individuals, in a distributed manner are identified by a dashed box. Discarding the biological plausibility, one possible algorithmic way to improve GA performance is to make the implicit mechanism explicit. In order to achieve the explicit processing of the obtained information,

probabilistic models are the chosen mathematical construction to "describe" populations. Since the process to find a probabilistic model for a given population is to estimate the probabilistic distributions on decision variables, such algorithms are called *estimation of distribution algorithms*, or sometimes, *probabilistic model building genetic algorithms* (PMBGAs). After gathering and mining the information existing in the form of individuals, the offspring individuals are then created by sampling the built probabilistic model to implement the process of information exploitation. Figure 1(b) shows an EDA in its simplest form. We can see between Figure 1(a) and Figure 1(b) that the key differences between GAs and EDAs are using probabilistic distributions to model populations and replacing genetic operators with the functionally equivalent mechanisms—probabilistic model building and sampling.

2.3 Estimation of Distribution Algorithms

There have been numerous variants of estimation of distribution algorithms proposed in the literature. Some of them adopt probabilistic models of different types or complexities, while others employ different techniques to build model. All these studies and developments on estimation of distribution algorithms started after the proposal of *population-based incremental learning* (PBIL) by Baluja [4] in 1994, while the name of "estimation of distribution algorithms" was firstly proposed by H. Mühlenbein and G. Paaß [5] in 1996. PBIL uses a probability vector to replace the population. Slightly different from most existing EDAs in which the probabilistic model is built from scratch at every generation as shown in Figure 1(b), PBIL retains some memory or experience of which the weight can be adjusted by the user. If the weight is set to zero, PBIL becomes a commonly structured EDA which is exactly the *univariate marginal distribution algorithm* (UMDA) proposed by Mühlenbein in [6] 1997. Early studies on EDAs began with simple probabilistic models of which the decision variables of optimization problems were assumed independent of each other. More and more complicated probabilistic models were used in the follow-up work along this line, which will be discussed in the following section.

3. EXISTING EDA FRAMEWORKS

This section will introduce some popular EDA frameworks proposed in the literature and widely used in both research and practice. Since probabilistic models are the key component in EDAs, we will introduce the EDAs employing simple models first and then those adopting complex models. Although the EDAs that adopt complex models usually provide excellent performance, one must keep in mind that complex models themselves may induce spurious variable relationships. If such spurious relationships become an obstacle which prevents the EDA from solving problems, EDAs with simpler models, i.e., more suitable for the problem structure, should be used to obtain better performance. Because the frameworks described in this section will be only a fraction of all existing EDAs, interested readers should consult other materials [7, 8].

3.1 All Variables Are Considered Independent

The simplest, reasonable probabilistic model to work with EDAs is assuming that no interaction exists between variables. EDAs employing such a model estimate the probabilistic distributions of values in different ways, including PBIL [4], UMDA [6], and the *compact genetic algorithm* (cGA) [9]. These EDAs work very well on problems composed of building blocks of order one and may encounter difficulties when facing problems consisting of longer, misleading building blocks.

3.2 Interactions between Two Variables Are Considered

In order to take into account the interactions between variables, probabilistic models considering pairwise interactions are intuitive choices. The *mutual information maximization for input clustering* (MIMIC) [10] algorithm assumes that the pairs of interacting variables are chained by their relationships, while the *combining optimizers with mutual information trees* (COMIT) [11] algorithm models the all the pairwise relationships with a dependency tree. The *bivariate marginal distribution algorithm* (BMDA) [12] further considers that all the pairwise relationships can be modeled with several independent dependency trees, i.e., a forest.

3.3 Interactions among More Than Two Variables Are Considered

Finally, the probabilistic models considering multivariate dependencies are adopted in EDAs. As a rule of thumb, EDAs with more general, complicated probabilistic models are able to handle more difficult

problems as long as the adopted models do not induce harmful spurious dependencies. The *extended compact genetic algorithm* (ECGA) clusters variables into separate linkage groups and considers the joint distribution for each group. With the help of human experts, the *factorized distribution algorithm* (FDA) utilizes a fixed model as the problem structure and provides excellent, theoretically proven performance. Adopting Bayesian networks as the probabilistic model, the Bayesian optimization algorithm (BOA) [13] and the estimation of Bayesian networks algorithm (EBNA) [14] uses different criteria to judge the quality of candidate Bayesian networks.

4. ISSUES AND FUTURE DIRECTIONS

In this section, we will describe several important research issues and potential future directions of EDAs. Because the design of EDAs is based on the properties and characteristics of probabilistic models, knowing the intrinsically embedded limitations and reducing the computational cost are no doubt essential. Moreover, obtaining information by examining the built models and hybridizing EDAs with techniques of other origins are promising research directions. Please note that the materials included in this section are far from complete. Many other topics worth pursuing are available in the recent literature.

4.1 Can Models Be Misleading Or Always Partially Meaningful?

Since probabilistic models are used in EDAs as tools for optimization, an obvious question rises: Is it possible that we build an appropriate probabilistic model according to a given population, while the built model leads us away from the optimal solution? This question is about the intrinsic properties of the problems that we want to solve by using EDAs. If some problems upon which the probabilistic model built correctly is actually misleading, EDAs, no matter what kinds of probabilistic models are adopted, will not be able to handle these problems. Coffin and Smith [15, 16] investigated whether the parity functions are such deal breakers. Furthermore, Chen and Yu [17] theorized the difficulty of probabilistic model building with mathematical formalization and obtained certain theoretical results. Another question regarding problem intrinsic properties is: Is it possible that, for certain problems, the built model is always partially meaningful? Chuang and Chen [18, 19] demonstrated that the problems composed of disparate importance weights might render EDAs building partially correct models at any time. In addition to proposing the concepts of *linkage sensibility* and *effective distributions*, they provided a technique to work with ECGA.

4.2 Can Models Be Built More Easily?

The main computational cost of EDAs is apparently caused by building probabilistic models. Research along this line is always active and important. To know EDAs better, Chen et al. [20] analyzed the average time complexity of EDAs. Techniques that can build models more efficiently were proposed by Ding et al. [21], Echegoyen et al. [22], and Iclănzan et al [23]. For BMDA, probability model migration [24] and aggregation [25] were proposed to be used in a parallel configuration. For BOA, in order to reduce the model building cost, previously built Bayesian networks, were utilized to predict next network structures [26, 27] or were viewed as a prototype for incremental changes [28].

4.3 Can Models Provide Useful Information?

After building and using the probabilistic models, it seems wasteful to put the models aside. As a consequence, looking into the built probabilistic models to collect useful information is worth trying. In addition to getting information for help building the subsequent models as aforementioned [26, 27], Santana et al. [29] tried to conduct data mining on the built probabilistic models, and Echegoyen et al. [30] investigated the interaction as well as relationship between the optimization problem and the probabilistic model via analyzing the probability to the optimal solutions.

4.4 Can EDAs Be Hybridized with Other Techniques?

A common feature of evolutionary algorithms is their flexibility to work or to interface with all kinds of methods from other realms. EDAs are no exception. In order to enhance EDAs for different purposes, a host of mechanisms, methodologies, and frameworks have been integrated, including multi-objective optimization [31, 32], niching [33], adaptive variance scaling [34], Spearman's rank correlation index [35], particle swarm optimization [36], etc.

5. CONCLUSIONS

In this paper, estimation of distribution algorithms (EDAs) as a popular class of evolutionary algorithms have been reviewed. EDAs can be regarded as abstraction of genetic algorithms (GAs) because in EDAs, the population, one of the GA distinctive features, is replaced by probabilistic models, and the common genetic operators, e.g., crossover, mutation, etc., are replaced by building and sampling from the adopted probabilistic model. By pursuing optimization performance instead of insisting on biological plausibility, EDAs successfully accomplish their design goal and become more and more popular in recent years. This paper was written with the intention to provide an entry level introduction to EDAs for researchers and practitioners who are in need and interested in knowing and using EDAs in a short time. Basic ideas, existing frameworks, and potential research directions of EDAs were briefly described in the hope that more and more taskforces will join the research as well as applications of EDAs.

ACKNOWLEDGMENTS

The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The author is grateful to the National Center for High-performance Computing for computer time and facilities.

REFERENCES

1. J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.
2. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Co., 1989.
3. D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Systems*, vol. 6, no. 4, 1992, pp. 333-362.
4. S. Baluja, *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.
5. H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," *Proceedings of the Fourth Parallel Problem Solving from Nature (PPSN IV)*, 1996, pp. 178-187.
6. H. Mühlenbein, "The Equation for Response to Selection and Its Use for Prediction," *Evolutionary Computation*, vol. 5, no. 3, 1997, pp. 303-346.
7. 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, 2002, pp. 5-20.
8. P. Larrañaga and J. A. Lozano, eds., Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2002.
9. G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *Proceedings of the International Conference on Evolutionary Computation (ICEC'98)*, 1998, pp. 523-528.
10. J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Advances in Neural Information Processing Systems*, vol. 9, 1997, pp. 424.
11. S. Baluja and S. Davies, "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," *Proceedings of the 14the International Conference on Machine Learning*, 1997, pp. 30-38.
12. M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," *Advances in Soft Computing—Engineering Design and Manufacturing*, 1999, pp. 521-535.
13. M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, 1999, pp. 525-532.
14. R. Etxeberria and P. Larrañaga, "Global optimization with Bayesian networks," *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, 1999, pp. 332-339.
15. D. J. Coffin and R. E. Smith, "The Limitations of Distribution Sampling for Linkage Learning," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 364-369.
16. D. J. Coffin and R. E. Smith, "Why Is Parity Hard for Estimation of Distribution Algorithms," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007 (GECCO-2007)*, 2007, pp. 624.

17. S.-C. Chen and T.-L. Yu, "Difficulty of Linkage Learning in Estimation of Distribution Algorithms," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO-2009)*, 2009, pp. 397-404.
18. C.-Y. Chuang and Y.-p. Chen, "On the Effectiveness of Distributions Estimated by Probabilistic Model Building," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2008 (GECCO-2008)*, 2008, pp. 391-398.
19. C.-Y. Chuang and Y.-p. Chen, "Sensibility of Linkage Information and Effectiveness of Estimated Distributions," *Evolutionary Computation*, 2010. (Accepted).
20. T. Chen, K. Tang, G. Chen, and X. Yao, "On the Analysis of Average Time Complexity of Estimation of Distribution Algorithms," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 453-460.
21. N. Ding, J. Xu, S. Zhou, and Z. Sun, "Reducing Computational Complexity of Estimating Multivariate Histogram-Based Probabilistic Model," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 111-118.
22. C. Echegoyen, J. A. Lozano, R. Santana, and P. Larrañaga, "Exact Bayesian Network Learning in Estimation of Distribution Algorithms," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 1051-1058.
23. D. Iclănzan, D. Dumitrescu, and B. Hirsbrunner, "Correlation Guided Model Building," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO-2009)*, 2009, pp. 421-428.
24. J. Jaroš and J. Schwarz, "Parallel BMDA with Probability Model Migration," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 1059-1066.
25. J. Jaroš and J. Schwarz, "Parallel BMDA with an Aggregation of Probability Models," *Proceedings of 2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, 2009, pp. 1683-1690.
26. M. Hauschild, M. Pelikan, K. Sastry, and D. E. Goldberg, "Using Previous Models to Bias Structural Learning in the Hierarchical BOA," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2008 (GECCO-2008)*, 2008, pp. 415-422.
27. M. W. Hauschild and M. Pelikan, "Intelligent Bias of Network Structures in the Hierarchical BOA," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO-2009)*, 2009, pp. 413-420.
28. M. Pelikan, K. Sastry, and D. E. Goldberg, "iBOA: The Incremental Bayesian Optimization Algorithm," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2008 (GECCO-2008)*, 2008, pp. 455-462.
29. R. Santana, C. Bielza, J. A. Lozano, and P. Larrañaga, "Mining Probabilistic Models Learned by EDAs in the Optimization of Multi-objective Problems," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO-2009)*, 2009, pp. 445-452.
30. C. Echegoyen, A. Mendiburu, R. Santana, and J. A. Lozano, "Analyzing the Probability of the Optimum in EDAs Based on Bayesian Networks," *Proceedings of 2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, 2009, pp. 1652-1659.
31. M. Laumanns and J. Ocenasek, "Bayesian Optimization Algorithms for Multi-objective Optimization," *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, 2002, pp. 298-307.
32. Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, 2008, pp. 41-63.
33. H. Handa, "Estimation of Distribution Algorithms with Niche Separation Mechanism," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 110-126.
34. P. A. N. Bosman and D. Thierens, "Adaptive Variance Scaling in Continuous Multi-Objective Estimation-of-Distribution Algorithms," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007 (GECCO-2007)*, 2007, pp. 500-507.
35. A. H. Aguirre, E. V. Diharce, and S. B. Moreno, "An Estimation Distribution Algorithm with the Spearman's Rank Correlation Index," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2008 (GECCO-2008)*, 2008, pp. 469-470.
36. C. W. Ahn and H.-T. Kim, "Estimation of Particle Swarm Distribution Algorithms: Bringing Together the Strengths of PSO and EDAs," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO-2009)*, 2009, pp. 1817-1818.

On the Optimization of Degree Distributions in LT Code with Covariance Matrix Adaptation Evolution Strategy

Chih-Ming Chen, *Student Member, IEEE*, Ying-ping Chen, *Member, IEEE*,
Tzu-Ching Shen, and John K. Zao, *Senior Member, IEEE*

***Abstract*— Luby Transform code (LT code) has been a popular and practical technique in the field of channel coding since its proposal. One of the key components of LT code is a degree distribution which is used to determine the relationship between source data and codewords. Luby in his proposal suggested two general methods to construct feasible degree distributions. Such general designs work appropriately in typical situations but not optimally in most cases. To explore the full potential of LT code, in this work, we make the first attempt to introduce evolutionary algorithms to optimize the degree distribution in LT code. Degree distributions are encoded as real-valued vectors and evaluated by numerical simulation of LT code. For applications of different natures, two objectives are implemented to search good degree distributions with different decoding behavior. Compared with the original design, the experimental results are quite promising and demonstrate that the degree distribution can be customized for different purposes. In addition to manually adjusting the degree distribution as the common practice, the work presented in this paper provides an efficient alternative approach to use and adapt LT code for both practitioners and researchers.**

I. Introduction

Digital fountain code [1] is a popular class of erasure code in the field of communication. The concept of fountain code was first introduced by Byers et al. [2] in 1998. Firstly, source data are divided into several pieces with an identical length. The length of each piece can be any bits or even several bytes. Sender generates encoding packets, or called encoding symbols when the packet length is one bit, by some particular encoding operation. The encoding and sending procedure may repeat independently and unlimitedly. Infinite encoding packets are sent out continuously like a fountain, which is an important property of fountain code called *rateless*. If a receiver is interested in receiving the data, it can receive the packet flow at any time and collect the packets in any combination. Once sufficient packets, of which the amount is usually slightly more than that of the source data, are obtained, the source data can be fully recovered. During the process, no further communication is required between sender and receiver. Encoding information can be embedded in each packet. As a result, digital fountain code is especially useful in broadcast or other situations in which back channels are unavailable. Moreover, because source data can be reconstructed no matter which packets are received, fountain code is also considered reliable to handle the problem of packet loss.

Chih-Ming Chen, Ying-ping Chen, Tzu-Ching Shen, and John K. Zao are with the Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, TAIWAN (email: ccming@nclab.tw, ypchen@nclab.tw, Stecko.cs97g@nctu.edu.tw, jkzao@cs.nctu.edu.tw).

Luby Transform code (LT code) [3] proposed by Luby in 2002 is the first practical framework of fountain code. A novel coding mechanism based on a specifically designed degree distribution is proposed in the introduction of LT code. The performance of LT code totally depends on the adopted degree distribution. In his proposal, Luby deigned general methods to construct an appropriate degree distribution to be used in LT code, and the degree distribution was named *soliton distribution*. Via theoretical analysis, the feasibility of soliton distribution was proven in the literature [4]. Recently, researchers started to optimize the degree distribution in order to improve the performance of LT code [5], [6], but the obtained improvement is quite limited. In these studies, only the parameters of soliton distribution were tuned and considered as decision variables, while in the present work, we directly consider the degree distribution itself as our decision variables.

Based on LT code, an improved framework call *Raptor codes* [7], [8] was proposed by Shokrollahi. Shokrollahi integrated LT code with a pre-coding layer. Compared with pure LT code, the design of Raptor codes requires a degree distribution, called *weakened LT*, with some very different behavior and properties. Several instances were given in [9] for certain particular sizes of source symbols, but there are no existing guidelines regarding how to construct suitable degree distributions for other sizes. In this regard, we demonstrate the use of optimization techniques proposed in evolutionary computation for generating degree distributions of different, desired properties.

In this paper, according to our limited knowledge, we make the first attempt to utilize evolutionary computation techniques to optimize the degree distribution for LT code and demonstrate the feasibility of customizing degree distributions for different purposes. Particularly, we adopt the covariance matrix adaptation evolution strategy (CMA-ES) [10] to directly optimize degree distributions for two goals: reducing the overhead and lowering the failure rate. The experimental results are remarkably promising and show that significantly reduced overheads and lower failure rates can be achieved for LT code with the obtained degree distribution for a wide range of source symbol sizes.

The remainder of this paper is organized as follows. Section II describes the detailed operations of LT code, including the coding process and soliton distribution proposed by Luby. Section III introduces the evolutionary algorithm used in this paper. Experiments and results are given in section IV. Finally, section V concludes this paper.

978-1-4244-8126-2/10/$26.00

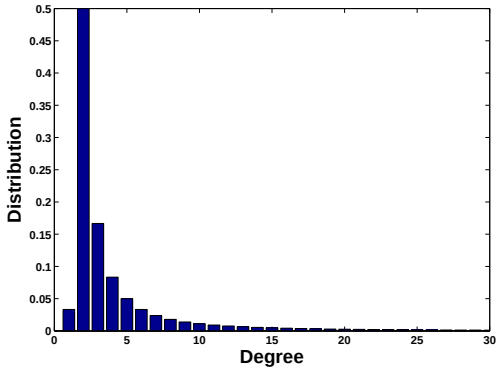
II. LT CODE

Luby introduced a new fountain code framework and gave the detail of coding operation in 2002 [3]. Similar to other fountain codes, source symbols are randomly chosen to be encoded into codewords (encoding symbols). The encoding operation is achieved by a simple boolean operator, *XOR*. The relation between source data and encoding symbols can be modeled as a sparse bipartite graph. A critical change in LT code is to decide the degree of each vertex in the bipartite graph with a probability distribution. The connectivity can be recorded as a encoding matrix and each column represents an encoding symbol. Originally, k source symbols can be fully decoding by Gaussian elimination if there exist k linearly independent columns. However, Gaussian elimination is prohibitively expensive for its computational complexity of $\mathcal{O}(k^3)$. Therefore, the belief propagation (BP) algorithm [11] is introduced to replace the expensive Gaussian elimination in the LT decoding phase. Overhead of coding is used to trade computing time because belief propagation is more efficient but more encoding symbols are needed for successful decoding. Moreover, the performance of LT code is very sensitive to the degree distribution. A good degree distribution is necessary to co-operate with belief propagation. Luby suggested soliton distributions for LT framework in his proposal of LT code. According to the mathematical verification, the properties of soliton distribution have been confirmed. In this section, details of coding operations and soliton distributions are described.

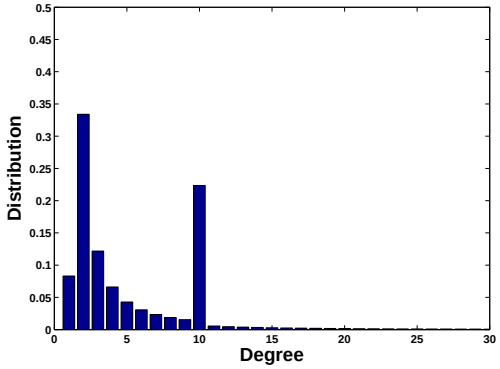
A. Encoding and decoding

Given the source data, we suppose that the source data can be cut into k source symbols with the same length of ℓ bits. Before every codeword is generated, a degree d is chosen at random according to the adopted degree distribution $\rho(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \rho(d) = 1$. The degree d decides the how many distinct source symbols will be chosen to compose an encoding symbol. d source symbols, called *neighbors*, are chosen uniformly randomly and accumulated by XOR. In the design of LT code, random numbers play an essential role during the encoding process. The approach employed by LT code for a sender to inform receivers of all encoding information is achieved by synchronizing a random number generator with a specified random number seed.

At the receiver side, when K encoding symbols were arrived which is usually slightly larger than k, belief propagation is used to reconstruct the source data step by step. All encoding symbols are initially covered in the beginning. For the first step, all encoding symbols with only one neighbor can be directly released to recover their unique neighbor. When a source symbol has been recovered but not processed, it is called a *ripple* and will be stored in a queue. At each subsequent step, ripples are popped as a processing target one by one. A ripple is removed from all encoding symbols which have it as neighbor. If an encoding symbols has only one remaining neighbor after the removing, the releasing action repeats and may produce new ripples to maintain a stable size of the queue. Maintaining the size of the ripple queue is important because the decoding process fails when the ripple queue is empty and some source symbols remain uncovered. In other words, more encoding symbols are required in the decoding process. Ideally, the process succeeds if all source symbols are recovered at the end of the decoding process.



(a) Ideal soliton distribution



(b) Robust soliton distribution

Fig. 1
EXAMPLE OF SOLITON DISTRIBUTIONS (K = 30)

B. Soliton distribution

The behavior of LT code is completely determined by the degree distribution, $\rho(d)$, and the number of encoding symbols received, K, by receiver. The overhead $\varepsilon = K/k$ denotes the performance of LT code, and ε depends on a given degree distribution. Based on his theoretical analysis, Luby proposed the ideal soliton distribution of which the overhead is 1, the best performance, in the ideal case.

Ideal soliton distribution $\rho(d)$:

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for} \quad d = 1 \\ \frac{1}{d(d-1)} & \text{for} \quad d = 2, 3, \ldots, k \end{cases} . \tag{1}$$

Ideal soliton distribution guarantees that all the release probabilities are identical to $1/k$ at each subsequent step. Hence, there is exactly one expected ripple generated at each processing step when the encoding symbol size is k. After k processing step, the source data can be ideally recovered. Fig. 1(a) shows an example of ideal soliton distribution for $k = 30$.

However, ideal soliton distribution works poorly in practice. Belief propagation may be suspended by a small variance of the stochastic encoding/decoding situation in which no ripple exists, because the expected ripple size is only one at any moment. According to the theory of random walk, the probability with which a random walk of length k deviates from its mean by more than $\ln(k/\delta)\sqrt{k}$ is at most δ. It is a baseline of ripple sizes which must be maintained to complete the decoding process. Hence, in the same paper by Luby, a modified version called *robust soliton distribution*, $\mu(d)$, was also proposed.

Robust soliton distribution $\mu(d)$:

$$R = c \cdot \ln(k/\delta)\sqrt{k}$$

$$\tau(d) = \begin{cases} R/ik & \text{for} \quad d = 1, \ldots, k/R - 1 \\ R\ln(R/\delta)/k & \text{for} \quad d = k/R \\ 0 & \text{for} \quad d = k/R + 1, \ldots, k \end{cases} . \quad (2)$$

$$\begin{aligned} \beta &= \sum_{d=1}^{k} (\rho(d) + \tau(d)) \\ \mu(d) &= \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k \end{aligned} \quad (3)$$

c and δ are two parameters for tuning robust soliton distribution. c controls the mean of the degree distribution. Smaller values of c increase the probability of low degrees and larger ones decrease it. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripple size as described. Fig. 1(b) is an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.1$. Robust soliton distribution can ensure that only $K = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoding symbols are required to recover the source data with a successful probability at least 1-δ.

Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound if k is infinite. However, in practice, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT code will not exactly match the mathematical analysis, especially when k is small. Furthermore, robust soliton distribution is a general purpose design. It provides a convenient way to construct a distribution works well but not optimally. In this work, we try to customize the degree distribution by using optimization tools proposed in the field of evolutionary computation.

III. Optimization Method

Evolution strategies (ES) are a major branch of evolutionary computation and have been developed since early 1960s. The key idea of ES is to evolve strategic parameters as well as decision variables. ES is well-known to be quite capable of dealing with continuous optimization problems. One of the simplest ES is (1+1)-ES where only one child is produced by Gaussian mutation to compete with its parent in each generation, and the other is (1, 1)-ES which is equivalent to random walk. Current general versions of ES are denoted as $(\mu^+_,\lambda)$-ES. The covariance matrix adaptation evolution strategy (CMA-ES) [10] was firstly introduced by Hansen in 1996 and is one of the most popular real-parameter optimization methods in evolutionary computation. There are some variants of CMA-ES proposed in the literature [12], [13], [14]. The search ability of CMA-ES has been theoretically analyzed and empirically verified on certain classic optimization problems, such as Ackley's function, Griewank's function, and Rastrigin's function. In CMA-ES, only a few algorithmic parameters need to be decided because CMA-ES inherits the mechanism to adapt strategic parameters during the evolutionary process. In this work, CMA-ES is utilized to optimize the degree distribution in LT framework for a wide range of k, the size of source symbols. In the remainder of this section, the way to adopt CMA-ES to handle the optimization of degree distributions are presented in detail.

A. Decision Variables

The first step to use an evolutionary algorithm is to encode the decision variables of the optimization problem. It is not difficult in this study because a degree distribution can directly form a real-number vector. In the evaluation phase, a real-number vector of arbitrary values can be interpreted as a probability distribution, i.e., a degree distribution, with normalization. Such an operation does not change the feasibility, although the problem complexity may be slightly increased. The definition of degree distributions tells us that $d \leq k$. For a specific source symbol size k, obviously the problem dimensions is at most k. However, according to the LT encoding/decoding operations, we usually do not need a non-zero probability on every single degree. Observing the soliton distributions and considering the belief propagation algorithm, there is no necessary degree except 1, which ensures the start of belief propagation. As a result, we optimize a selected subset of degrees in the present work. We choose some degrees called *tags* to form the vector $v(i)$ of decision variables according to the Fibonacci numbers smaller than half of k. A degree distribution used in this paper hence can be represented as the following formula.

Optimized degree distribution $\omega(d)$:

$$\omega(d) = \begin{cases} v(i) & d = \text{the } i\text{-th Fibonacci number}, d < k/2 \\ 0 & \text{otherwise} \end{cases} . \quad (4)$$

B. Objectives

We try to use two indicators to evaluate degree distributions for LT code in this paper. The first one is the efficiency of the LT code with the optimized degree distribution which has been discussed in section II-B. ε denotes the expected rate of overhead to transmit data. For example, $\varepsilon = 1.2$ means that in addition to the size of source data, 20% extra data are needed to recover the complete source data. This objective is to obtain some degree distribution for a specific k with the smallest ε. LT code is rateless, and the coding process depends on randomness and probability. Source data recovered by a fixed amount of encoding symbols cannot be guaranteed. Therefore,

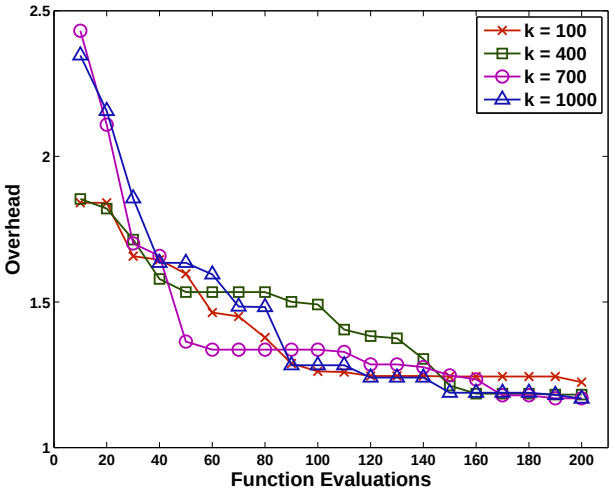


Fig. 2

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF OVERHEAD

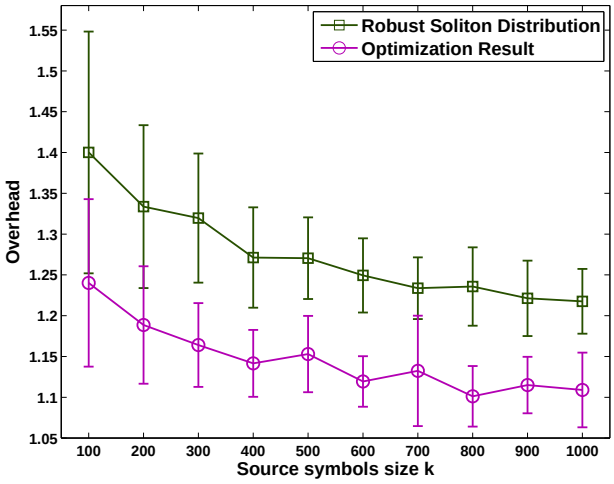


Fig. 3

AVERAGE PERFORMANCE INDICATORS ARE COMPARED BETWEEN ROBUST SOLITON DISTRIBUTION AND OPTIMIZED DEGREE DISTRIBUTIONS FOR DIFFERENT NUMBERS OF SOURCE SYMBOLS (k)

TABLE I

THE BEST INDIVIDUALS FOR THE OPTIMIZATION OF OVERHEAD

Degree	k=100	$k = 400$	$k = 400$	$k = 1000$
1	0.091397	0.116375	0.16058	0.129707
2	0.310884	0.255701	0.148543	0.266133
3	0.367223	0.34174	0.412275	0.321489
5	0.042648	0.112072	0.119163	0.077045
8	0.053247	0.071726	0.052843	0.124503
13	0.048949	0.028076	0.024701	0.000258
21	0.011876	0.013169	0.035112	0.019594
34	0.073776	0.030397	0.017738	0.033607
55	0	0.000264	0.002094	0.01543
89	0	0.01109	0.009837	0.00095
144	0	0.01939	0.002946	0.000143
233	0	0	0.014167	0.00075
377	0	0	0	0.010391

in order to evaluate ε, we provide infinite encoding symbols, in the form of a stream of encoding symbols, to simulate the decoding process until all source data are recovered. The average of required encoding symbols per simulation is the fitness value of degree distributions.

The second indicator is the amount of source symbols that cannot be recovered when a constant ratio of encoding symbols are received. In raptor codes, Low-density-parity-check (LDPC) [15] is introduced as a second layer pre-coding into LT code. LDPC is a kind of forward error correction codes. More information on LDPC can be found in [16], [17]. LDPC can fix errors of data without extra information as long as the error rate is lower than certain restriction. In such a condition, the mission of LT code is no longer to achieve full decoding. Instead, most of source symbols can be recovered with a small overhead is sufficient. For this purpose, we try to minimize the number of un-recovered source symbols given a constant overhead ε.

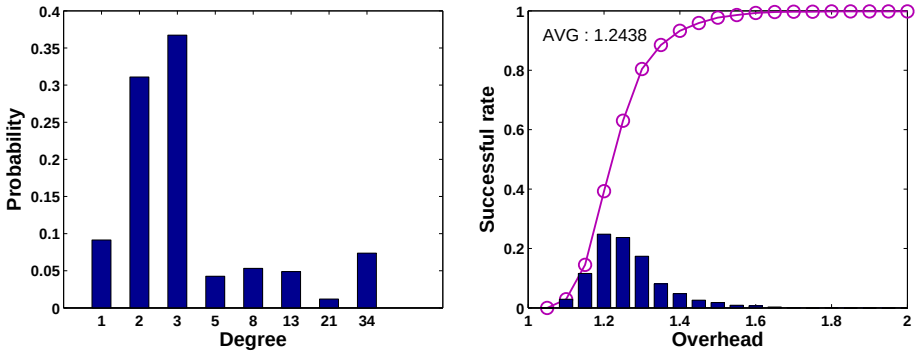
IV. EXPERIMENTS AND RESULTS

Two series of experiments are implemented for the two different objectives as described in the previous section. In each experiment, *tags* are determined by Fibonacci numbers and the specified source symbols size k. Tags are encoded as an individual, $v(i)$, and represent that only these degrees have non-zero probabilities. Initial values of tags are set as $1/|v|$ uniformly, and then CMA-ES is applied without any customization or modification. After a new individual is created, it is normalized to be a valid probability distribution and evaluated for the fitness value by simulating the LT coding process. One hundred independent runs of simulation are conducted for each function evaluation. In the first series of experiments, we minimize the expected number of encoding symbols for full decoding. In the second, the average number of source symbols that cannot be recovered for a constant $\varepsilon = 1.1$ is considered. We call the second indicator as *failure rate*. The default parameter settings given in the source code of CMA-ES are adopted in this study except for $\lambda = 10$.

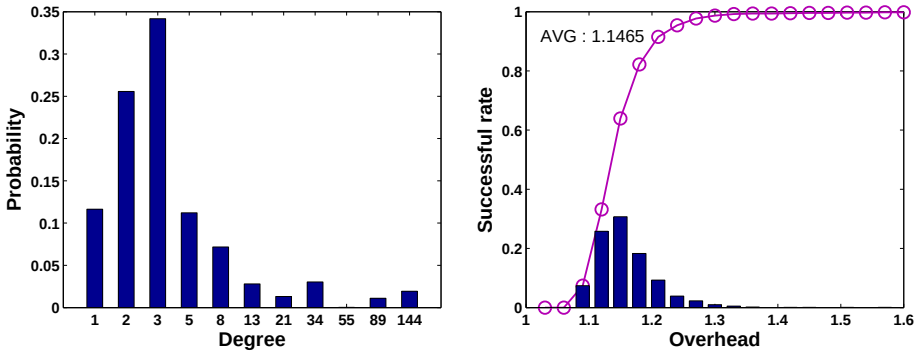
A. Overhead

In these experiments, we minimize the overhead ε for different k sizes, and the results are shown in Table I and Figs. 2–5. Fig. 2 presents the improvement during the evolutionary process. Individuals are initially uniform distributions. It is expected that overheads are quite high in the beginning and the curves descend quickly after around 100 function evaluations. Finally, the fitness almost converges after 200 function evaluations. Fig. 3 shows the comparison of ε between robust soliton distribution and the optimized distributions. The expected overhead of robust soliton distribution is given as

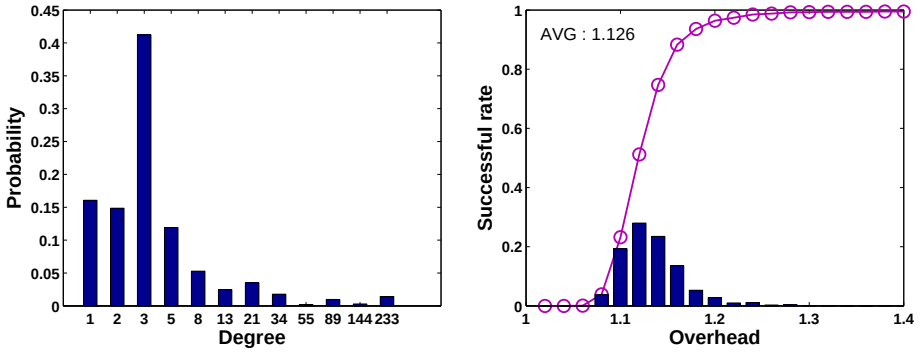
$$\frac{k + \mathcal{O}(\log^2(k/\delta)\sqrt{k})}{k} = 1 + \mathcal{O}\left(\frac{\log^2(k/\delta)}{\sqrt{k}}\right).$$



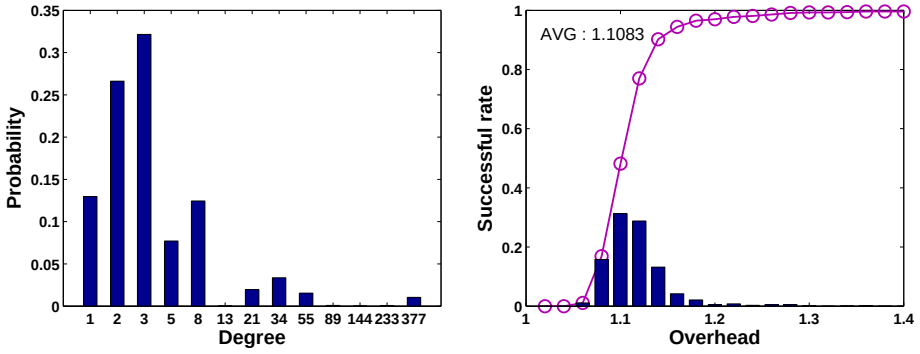
(a) $k = 100$



(b) $k = 400$



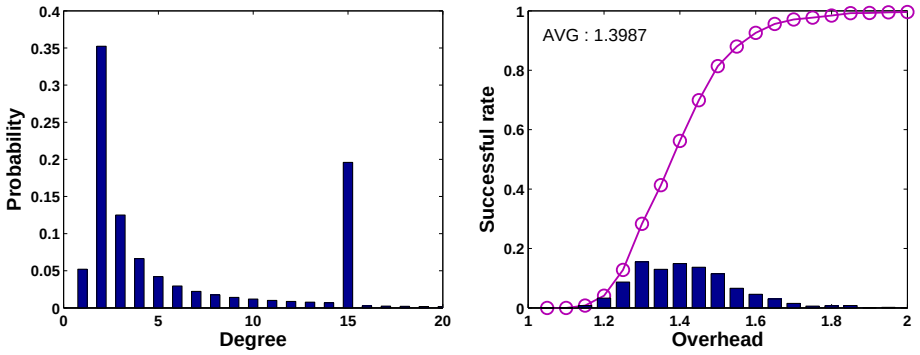
(c) $k = 700$



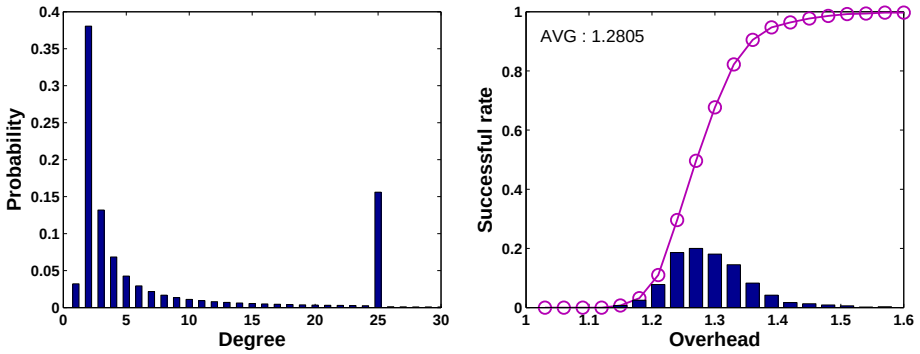
(d) $k = 1000$

Fig. 4

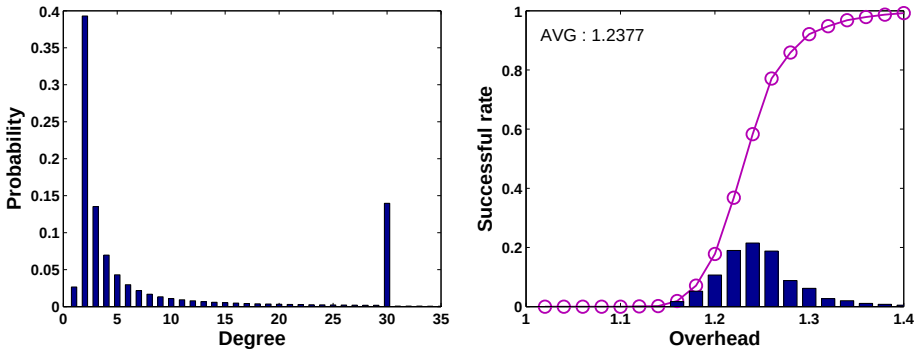
LEFT FIGURES SHOW THE OPTIMIZED DEGREE DISTRIBUTIONS. ONLY TAGS ARE PRESENTED. RIGHT FIGURES ARE THE HISTOGRAM AND ACCUMULATED CURVE OF SUCCESSFUL RATE IN 1000 INDEPENDENT SIMULATION RUNS



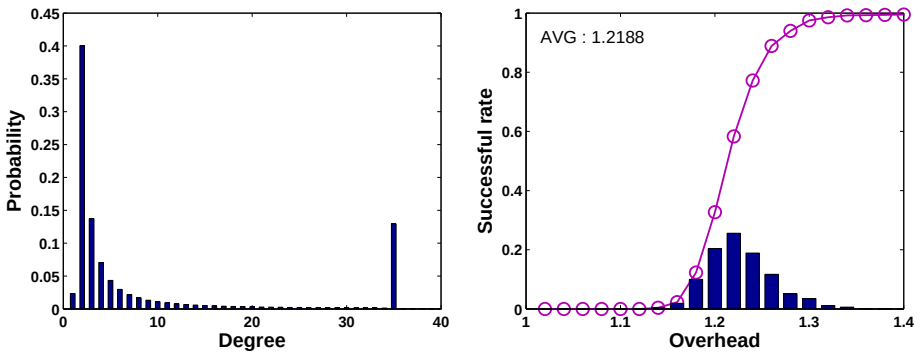
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

Fig. 5

FOR THE COMPARISON WITH SAME k'S, ROBUST SOLITON DISTRIBUTIONS AND THE CORRESPONDING PERFORMANCE INDICATORS ARE SHOWN SIMILAR TO THAT IN FIG. 4. NOTE THAT ONLY PARTS OF ROBUST SOLITON DISTRIBUTIONS ARE PLOTTED FOR CLARITY

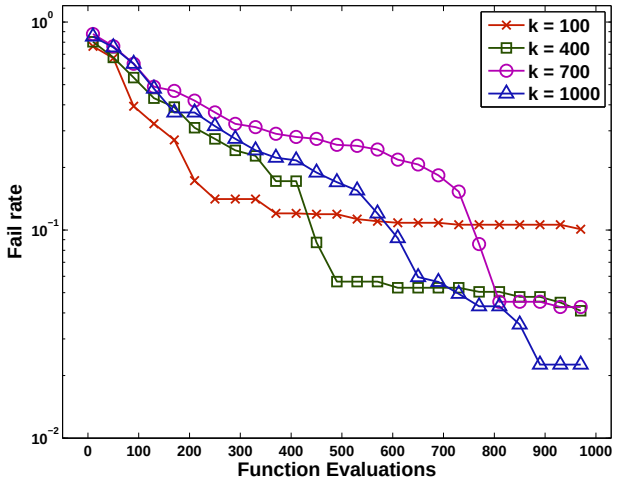


Fig. 6

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF FAILURE RATE

TABLE II

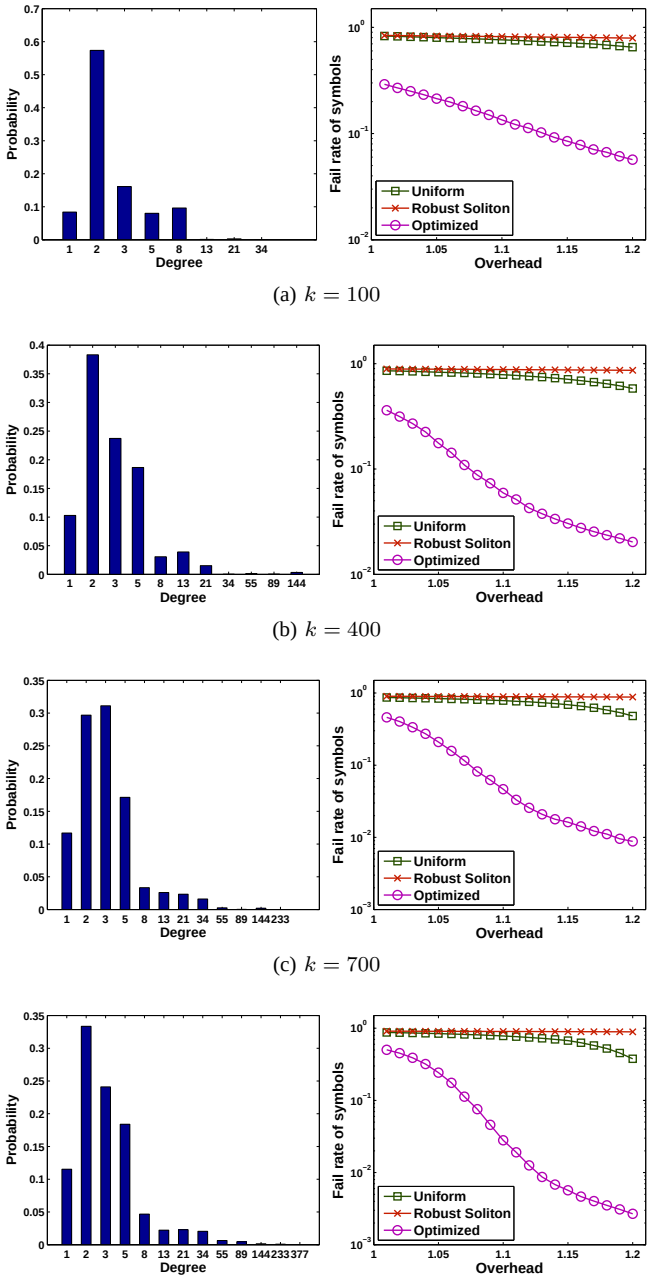
THE BEST INDIVIDUALS FOR THE OPTIMIZATION OF FAILURE RATE

Degree	k=100	$k = 400$	$k = 400$	$k = 1000$
1	0.083997	0.102892	0.116854	0.115278
2	0.573671	0.383164	0.29678	0.333564
3	0.161178	0.237312	0.31115	0.241065
5	0.08038	0.186475	0.171342	0.184027
8	0.096245	0.030706	0.033393	0.046818
13	0.001267	0.039075	0.025977	0.022223
21	0.002963	0.015193	0.023452	0.022914
34	0.000299	0.000167	0.016096	0.020526
55	0	0.001276	0.002602	0.00643
89	0	0.000303	0.000268	0.004594
144	0	0.003436	0.002072	0.001422
233	0	0	0.000015	0.000883
377	0	0	0	0.000257

The value becomes smaller when k increases, and that is why the trend of Fig. 3 shows a declination. The values of overhead are reduced at least 10% for all k's with the optimized degree distributions. Some distributions of the best individuals are given in Table I. Fig. 4 illustrates each distribution and shows the histogram of successful rate in 1000 simulation runs on the right side. Compared with similar simulation results of robust soliton distribution in Fig. 5, the improvement is quite significant.

B. Failure rate

Unlike the original LT code, we are concerned with how many source symbols can be recovered in the second set of experiments. The objective value is the average number of source symbols that cannot be recovered with a constant overhead ε. Optimization results are shown in Fig. 6. More function evaluations are needed to search for good degree distributions. The failure rate of the final results are less than 10^{-1} for all k's when $\varepsilon = 1.1$. In other words, more than 90 percent of source symbols can be recovered if extra 10 percent of encoding symbols are collected. Table II gives



(a) $k = 100$

(b) $k = 400$

(c) $k = 700$

(d) $k = 1000$

Fig. 7

THE FIGURE SHOWS THE SIGNIFICANT DIFFERENCE OF FAILURE RATE AFTER OPTIMIZATION. SIMILAR TO THAT IN FIG. 4, ONLY TAGS ARE SHOWN IN THE FIGURES

the best probability distributions found in the evolutionary process for $k = 100$, $k = 400$, $k = 700$, and $k = 1000$. The simulation results of a constant overhead are presented in Fig. 7. The red line denotes the behavior of uniform distribution, which is the initial value of optimization. Most of the source symbols remain covered except for those of which the degree is one, i.e., with probability $1/k$. The same situation happens to robust soliton distributions because the

amount of extra encoding symbols is not sufficient to complete the BP decoding process. The behavior of LT process with the optimized degree distributions is totally different and fully satisfies the requirement of weakened LT.

V. Conclusions

In this work, the first attempt to algorithmically optimize the degree distribution adopted in LT code was proposed. Evolutionary computation techniques were introduced to accomplish the optimization task. Different from the previous studies reported in the literature, each probability of degrees were directly encoded as an individual to optimize. Promising experimental results were obtained in both sets of experiments: One was to minimize the overhead, and the other was to reduce the decoding failure rate. Our experiments showed that CMA-ES was indeed capable of finding good degree distributions for different purposes without any guideline or human intervention. Compared with robust soliton distribution, the optimized overhead was decreased as least 10% for every k in the experiments. The results of failure rate minimization were also remarkably promising and able to support applications of different types and requirements.

This study creates a new research topic in which the design of degree distributions in LT code can now be algorithmic and no longer has to be manually tuning parameters of robust soliton distribution. We have empirically proved that directly manipulating the probability value for each degree is viable and worth pursuing. Given a specific k and some expected overhead, a degree distribution can be customized with existing optimization techniques. In addition, we will extend the experiments to larger k for more kinds of potential applications in the near future. The results empirically obtained by using evolutionary algorithms will be theoretically analyzed, and general guidelines, like robust soliton distribution, that are able to be customized for different goals and requirements for designing degree distributions are expected.

Acknowledgments

The authors would like to thank Martin Hornansky for fruitful discussion and conducting certain related numerical experiments. The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] D. J. C. MacKay, "Fountain codes," in *The IEE Seminar on Sparse-Graph Codes*, 2004, pp. 1–8.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998, pp. 56–67.

[3] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, 2002, p. 271.

[4] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 39.

[5] E. A. Bodine and M. K. Cheng, "Characterization of luby transform codes with small message size for low-latency decoding," in *IEEE International Conference on Communications (ICC '08)*, 2008, pp. 1195–1199.

[6] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, 2007, pp. 2576–2580.

[7] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channels," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 38.

[8] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, no. 5, pp. 2033–2051, 2006.

[9] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[10] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 312–317.

[11] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, 1982, pp. 133–136.

[12] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 1777–1784.

[13] ——, "A restart cma evolution strategy with increasing population size," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 1769–1776.

[14] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Proceedings of the Tenth International Conference on Parallel Problem Solving from Nature (PPSN X)*, 2008, pp. 296–305.

[15] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[16] D. Changyan, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1570–1579, 2002.

[17] E. Paolini and M. Chiani, "Improved low-density parity-check codes for burst erasure channels," in *IEEE International Conference on Communications (ICC '06)*, vol. 3, 2006, pp. 1183–1188.

Optimizing Degree Distributions in LT Codes by Using The Multiobjective Evolutionary Algorithm Based on Decomposition

Chih-Ming Chen, *Student Member, IEEE*, Ying-ping Chen, *Member, IEEE*,
Tzu-Ching Shen, and John K. Zao, *Senior Member, IEEE*

***Abstract*— Luby Transform code (LT code) is the first practical digital fountain code and has been widely used as basic components in many communication applications. The coding behavior of LT code is mainly decided by a probability distribution of codeword degrees. In order to customize a degree distribution for different purposes, multi-objective evolutionary algorithm is introduced to optimize degree distributions in this paper. Two critical performance indicators of LT code are considered in our experiments. Some applications hope to minimize the overhead of extra packets and some require to limit the computational cost of the coding system. To handle this problem, MOEA/D is applied to optimize two objectives simultaneously. We expect to obtain the Pareto front (PF) formed by partial optimal solutions and provide those available degree distributions to different LT code applications. Not only promising results are represented in this paper but also the behavior of LT code is thoroughly explored by optimizing the degree distribution according to multi-objectives.**

I. Introduction

Digital fountain code [1] is a popular class of erasure code in the field of communication. The concept of fountain code was introduced by Byers et al. [2] in 1998. Firstly, source data are divided into several pieces with an identical length. The length of each piece can be any number of bits or even several bytes. Sender generates encoding packets, or called *encoding symbols*, when the packet length is one bit, by certain encoding operation. The encoding procedure may repeat independently and indefinitely so infinite encoding packets are sent out continuously like a fountain, which is an important property of fountain code called *rateless*. If a receiver is interested in receiving the data, it can receive the packet flow at any time and collect the packets in any combination. Once sufficient packets, of which the amount is usually slightly more than that of the source data, are obtained, the source data can be fully recovered. During the process, no further communication is required between sender and receiver. Encoding information can be embedded in each packet. As a result, digital fountain code is especially useful in broadcast or other situations in which back channels are unavailable. Moreover, because source data can be reconstructed no matter which packets are received, fountain code is also considered reliable to handle the problem of packet loss.

Luby Transform code (LT code) [3] proposed by Luby in 2002 is the first practical framework and implementation of fountain code. A novel coding mechanism based on a specifically designed degree distribution is proposed in the introduction of LT code. The performance of LT code totally depends on the adopted degree distribution. In his proposal, Luby designed general methods to construct appropriate degree distributions to co-operate with LT code, and the degree distributions were named *soliton distribution*. Via theoretical analysis, the feasibility of soliton distribution was proven [4]. Recently, researchers started to optimize the degree distribution in order to improve the performance of LT code [5], [6], but the obtained improvement is marginal and quite limited. In these studies, only the parameters of soliton distribution were tuned and considered as decision variables, while in our present work, we directly consider the degree distribution itself as our decision variables.

In the design of LT code, redundant data and encoding computation are used to trade for the ability of forward error correction. For most applications, while the error correction ability is maintained, both costs are required to be as lower as possible, and apparently there is a trade-off among these factors. Furthermore, applications of different types and purposes have different requirements of each kind of cost. Some LT code applications which transmit data through an expensive communication channel have to reduce the data overhead. Other applications with a huge package size expect fewer executions of the encoding operator. In order to simultaneously satisfy these applications, multi-objectives are considered for optimizing the LT code degree distribution in the present work. The most important motivation of this study is to fully explore the LT coding behavior with arbitrary degree distributions and to empirically provide a proof of concept that multiple requirements on LT code can be satisfied via optimizing degree distributions with existing optimization techniques.

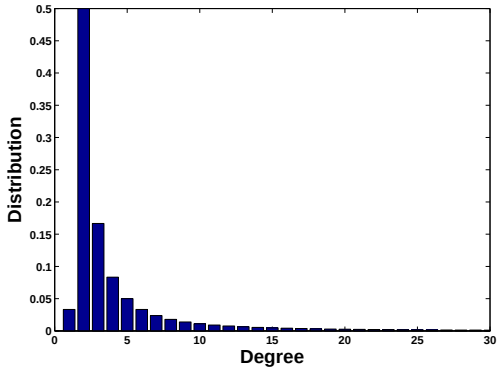
The remainder of this paper is organized as follows. Section II describes the detailed operations of LT code, including the coding process and soliton distribution. Section III introduces the background of multi-objective problems and the evolutionary algorithm used in this paper. Experiments and results are given in sections IV and V. Finally, section VI concludes this paper.

II. LT code

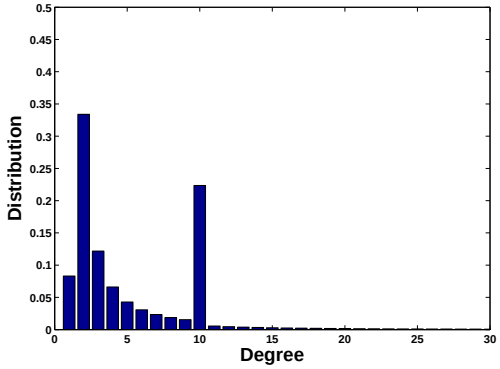
Luby introduced a practical fountain code framework and gave the details of coding operation in 2002 [3]. Similar to other fountain codes, source symbols are uniformly randomly chosen to be encoded into codewords (encoding symbols). The encoding operation is achieved by a simple boolean operator,

Chih-Ming Chen, Ying-ping Chen, Tzu-Ching Shen, and John K. Zao are with the Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, TAIWAN (email: ccming@nclab.tw, ypchen@nclab.tw, Stecko.cs97g@nctu.edu.tw, jkzao@cs.nctu.edu.tw).

978-1-4244-8126-2/10/$26.00



(a) Ideal soliton distribution



(b) Robust soliton distribution

Fig. 1

EXAMPLE OF SOLITON DISTRIBUTIONS (k = 30)

XOR. The relation between source data and encoding symbols can be modeled as a sparse bipartite graph. A key design of LT code is to decide the degree of each vertex in the bipartite graph with a probability distribution. The connectivity can be recorded as an encoding matrix and each column represents an encoding symbol. Originally, k source symbols can be fully decoded by Gaussian elimination if there exist k linearly independent columns. However, Gaussian elimination is prohibitively expensive for its computational complexity of $\mathcal{O}(k^3)$. Therefore, the belief propagation (BP) algorithm [7] is introduced to replace the expensive Gaussian elimination in the LT decoding phase. Overhead of coding is used to trade computing time because belief propagation is more efficient but more encoding symbols are needed for successful decoding. Moreover, the performance of LT code is very sensitive to the degree distribution. A good degree distribution is necessary to co-operate with belief propagation. Luby suggested soliton distributions for LT framework in his proposal of LT code. According to the mathematical verification, the properties of soliton distribution have been confirmed. In this section, details of coding operations and soliton distributions are described.

A. Encoding and decoding

Given the source data, we suppose that the source data can be cut into k source symbols with the same length of ℓ bits. Before every codeword is generated, a degree d is chosen at random according to the adopted degree distribution $\rho(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \rho(d) = 1$. The degree d decides the how many distinct source symbols will be chosen to compose an encoding symbol. d source symbols, called *neighbors*, are chosen uniformly randomly and accumulated by *XOR*. In the design of LT code, random numbers play an essential role during the encoding process. The approach employed by LT code for a sender to inform receivers of all encoding information is achieved by synchronizing a random number generator with a specified random number seed.

At the receiver side, when K encoding symbols were arrived, where K is usually slightly larger than k, belief propagation is used to reconstruct the source data step by step. All encoding symbols are initially covered in the beginning. For the first step, all encoding symbols with only one neighbor can be directly released to recover their unique neighbor. When a source symbol has been recovered but not processed, it is called a *ripple* and will be stored in a queue. At each subsequent step, ripples are popped as a processing target one by one. A ripple is removed from all encoding symbols which have it as neighbor. If an encoding symbols has only one remaining neighbor after the removing, the releasing action repeats and may produce new ripples to maintain a stable size of the queue. Maintaining the size of the ripple queue is important because the decoding process fails when the ripple queue is empty and some source symbols remain covered. In other words, more encoding symbols are required in the decoding process. Ideally, the process succeeds if all source symbols are recovered at the end of the decoding process.

Both encoding and decoding, as the LT coding operations, are achieved by *XOR*. As a result, the computational complexity of LT code can be measured by how many times of *XOR* is executed. *XOR* operator is applied to build the connectivity in the conceptualized bipartite graph and to eliminate a ripple from the neighbors of codewords. It is evident that $d - 1$ *XOR* operators are necessary to generated a codeword with degree d or recover an encoding symbol. In the encoding phase, all encoding symbols are generated independently, and the computational complexity to produce codewords solely depends on the mean degree of the adopted degree distribution. In other words, the cost of each encoding symbol is decided by the mean of degree distributions. Hence, in practice, the mean degree is an important LT performance indicator since it represents the operational cost.

B. Soliton distribution

The behavior of LT code is completely determined by the degree distribution, $\rho(d)$, and the number of encoding symbols received, K, by receiver. The overhead $\varepsilon = K/k$ denotes the performance of LT code, and ε depends on a given degree distribution. Based on his theoretical analysis, Luby proposed the ideal soliton distribution of which the overhead is 1, the best performance, in the ideal case.

Ideal soliton distribution $\rho(d)$:

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for} \quad d = 1 \\ \frac{1}{d(d-1)} & \text{for} \quad d = 2, 3, \ldots, k \end{cases} . \tag{1}$$

Ideal soliton distribution guarantees that all the release probabilities are identical to $1/k$ at each subsequent step. Hence, there is *one* expected ripple generated at each processing step when the encoding symbol size is k. After k processing step, the source data can be ideally recovered. Fig. 1(a) shows an example of Ideal soliton distribution for $k = 30$.

However, ideal soliton distribution works poorly in practice. Belief propagation may be suspended by a small variance of the stochastic encoding/decoding situation in which no ripple exists, because the expected ripple size is only one at any moment. According to the theory of random walk, the probability with which a random walk of length k deviates from its mean by more than $\ln(k/\delta)\sqrt{k}$ is at most δ. It is a baseline of the ripple queue size which must be maintained to complete a decoding process. Hence, in the same paper by Luby, a modified version called *robust soliton distribution*, $\mu(d)$, was also proposed.

Robust soliton distribution $\mu(d)$:

$$R = c \cdot \ln(k/\delta)\sqrt{k}$$

$$\tau(d) = \begin{cases} R/ik & \text{for} \quad d = 1, \ldots, k/R - 1 \\ R\ln(R/\delta)/k & \text{for} \quad d = k/R \\ 0 & \text{for} \quad d = k/R + 1, \ldots, k \end{cases} . \tag{2}$$

$$\begin{aligned} \beta &= \sum_{d=1}^{k} (\rho(d) + \tau(d)) \\ \mu(d) &= \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k \end{aligned} \tag{3}$$

c and δ are two parameters for tuning robust soliton distribution. c controls the mean of the degree distribution. Smaller values of c increase the probability of low degrees, and larger ones decrease it. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripples as described. Fig. 1(b) is an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.1$. Robust soliton distribution can ensure that only $K = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoding symbols are required to recover the source data with a successful probability at least 1-δ.

Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound if k is infinite. However, in practice, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT code will not exactly match the mathematical analysis, especially when k is small. Furthermore, robust soliton distribution is a general purpose design. It provides a convenient way to construct a distribution works well but not optimally. In this work, we try to customize the degree distribution by using multi-objective optimization tools proposed in the field of evolutionary computation to simultaneously satisfy multiple performance requirements.

III. Multi-objective Problems

Multi-objective optimization problems (MOPs) are very important in real-world applications. There are two or more objectives to be considered simultaneously, and these objectives usually conflict with each other. The most intuitive approach to deal with MOPs is to transform them into single objective problems (SOPs) by using weights on the objectives and creating a weighted sum. The approach makes the problem solvable by available tools based on mathematics or heuristics for SOPs. However, such weights oftentimes cannot be predetermined, especially when the domain knowledge of the problem is unavailable. Furthermore, the best solution to the transformed single-objective problem is merely one solution on the Pareto front (PF) of the MOP. Hence, better optimization frameworks must be developed to fulfill the need of handling MOPs.

Due to the limitation of traditional mathematical methods for MOPs, more and more researchers try to solve MOPs in a direct way and to approximate the Pareto front as complete as possible. Their goal is to provide a set of solutions which are partially optimal. Many advanced multi-objective algorithms have been proposed in the literature. Some of them try to approximate the PF by using mathematical models, and others are developed based on evolutionary algorithms. A hybrid framework makes use of decomposition methods in mathematics and the optimization paradigm in evolutionary computation was proposed and called *multiobjective evolutionary algorithm based on decomposition* (MOEA/D) [8]. MOEA/D was proposed and shown to perform well on MOPs with complicated Pareto set shapes [9].

In this paper, we propose the use of MOEA/D to optimize the multiple objectives of LT code. Degree distributions significantly better than *robust soliton distribution* are expected. Moreover, exploring a complete Pareto front can help researchers to analyze the trade-off between overhead and operational cost of LT code. In the following section, we will give the formal description of MOPs and the MOEA/D framework, respectively.

A. Formal description of MOPs

In real-world applications, many problems are actually multi-objective optimization problems, and single-objective problems are special cases. A multi-objective problem can be formally stated as:

$$\begin{aligned} &\text{minimize} \quad F(x) = (f_1(x), \ldots, f_m(x)) \\ &\text{subject to} \quad \begin{cases} x \in \Omega \\ C(x) = (c_1(x), \ldots, c_t(x)) \geq 0 \end{cases} \end{aligned} , \tag{4}$$

where Ω is called the *decision space* or *variable space*, and R^m is the *objective space*. $C(x)$ represents the problem constraints and defines the feasible regions in the decision space according to problem properties [10]. $F : \Omega \rightarrow R^m$ consists of m objective functions. If Ω is a closed and connected region in R^n and all the objective functions are continuous, we call the problem a continuous MOP.

In order to consider the trade-off between objectives, the concept of *domination* between solutions is defined. Let $u = (u_1, \ldots, u_m)$, $v = (v_1, \ldots, v_m) \in R^m$ be two vectors. u is said to *dominate* v if $u_i \leq v_i$ for all $i = 1, \ldots, m$, and $u \neq v$. A point $x^* \in \Omega$ is *Pareto optimal* if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. The set of all the Pareto optimal points is called *Pareto set* (PS) and the set of all the objective vectors corresponding to the PS is called *Pareto front* (PF), where $PF = \{F(x) \in R^m | x \in PS\}$ [11].

Instead of searching for a single or just a few (Pareto) optimal solutions as in solving single-objective problems, the goal of handling multi-objective problems is to find the Pareto front as well as the Pareto set of the problem. Given the limited computational resource, including time and storage, how to provide good solutions in terms of both quality and spread is the key and challenging task for multi-objective optimization.

B. MOEA based on decomposition

One of the key ideas of MOEA/D is the use of a decomposition method to transform a MOP into a number of single-objective optimization problems. MOEA/D attempts to optimize these single-objective problems collectively and simultaneously instead of trying to directly approximate the Pareto front as many other evolutionary algorithms do because each optimal solution to these SOPs is a Pareto optimal solution to the given MOP. The collection of these optimal solutions is an approximation of the Pareto front. Weighted sum, Tchebycheff approach, boundary intersection, and other decomposition approaches can serve this purpose. In the present work, the Tchebycheff approach [11] is adopted. A single-objective optimization problem obtained by decomposing the given MOP can be represented as

$$\begin{array}{ll} \text{minimize} & g(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i | f_i(x) - z_i^*|\} \\ \text{subject to} & x \in \Omega \end{array} \quad (5)$$

where $\lambda = (\lambda_1, \ldots, \lambda_m)$ is a vector of weights, i.e., $\lambda_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. $z^* = (z_1^*, \ldots, z_m^*)$ is the reference point, i.e., $z_i^* = \min\{f_i(x) | x \in \Omega\}$ for each $i = 1, \ldots, m$.

Let $\lambda^1, \ldots, \lambda^N$ be a set of N weight vectors. If we use a large N and select the weight vectors properly, all the optimal solutions of the SOPs transformed from decomposition will well approximate the Pareto front. Moreover, we can define a neighborhood relationship for each SOP by computing Euclidean distances between weight vectors. SOPs which are considered neighbors are assumed to have similar fitness landscapes and their optimal solutions should be close in the decision space. MOEA/D exploits the information sharing among SOPs which are neighbors to accomplish the optimization task effectively and efficiently. The specification of MOEA/D is stated as follows:

- Inputs:
  - decision variables.
  - objective functions.
  - N: the number of subproblems.
  - T: the number of neighbors for each subproblem.
  - stopping criteria.
- Outputs:
  - Approximation to the *PS*: $x^1, \ldots, x^N$.
  - Approximation to the *PF*: $F(x^1), \ldots, F(x^N)$.

TABLE I
PARAMETER SETTINGS OF MOEA/D

Parameter	Value
N	50
T	10
Crossover rate	1
Mutation rate	$1/m$
Max Gen.	150

IV. EXPERIMENTS

The experiment implementation is described in this section. MOEA/D is a well-developed tool and has the characteristic of black-box optimization like other evolutionary algorithms. As described in section III-B, only input and output should be handled properly. Section IV-A shows how to encode a degree distribution into decision variables, and the objective functions are given in section IV-B. Table I lists the other algorithmic parameter settings of MOEA/D.

A. Decision variables

The first step to use an evolutionary algorithm is to encode the decision variables of the optimization problem. It is not difficult in this study because a degree distribution can directly form a real-valued vector. In the evaluation phase, a real-valued vector of arbitrary values can be interpreted as a probability distribution, i.e., a degree distribution, with normalization. Such an operation does not change the feasibility, although the problem complexity may be slightly increased. The definition of degree distributions tells us that $d \leq k$. For a specific source symbol size k, obviously the problem dimensions is at most k. However, according to the LT encoding/decoding operations, we usually do not need a non-zero probability on every single degree. Observing the soliton distribution and considering the belief propagation algorithm, there is no necessary degree except 1, which ensures the start of belief propagation. As a result, we optimize a selected subset of degrees in the present work. We choose some particular degrees, {*1,2,3,4,5,7,9,13,17,23*} to form the decision variables according to the experience. Different subsets of degrees may change the numerical results of experiments results, but the soundness of this paper will be not be affected.

B. Objectives

In this paper, degree distributions are optimized for two different objectives. The first indicator to evaluate efficiency of LT code is overhead ε. The redundancy is traded for the benefit of fountain code and those extra encoding symbols increase the cost when they are transmitted to the receiver. In most application, overhead is required to be as low as possible because the transmission is usually expensive. In

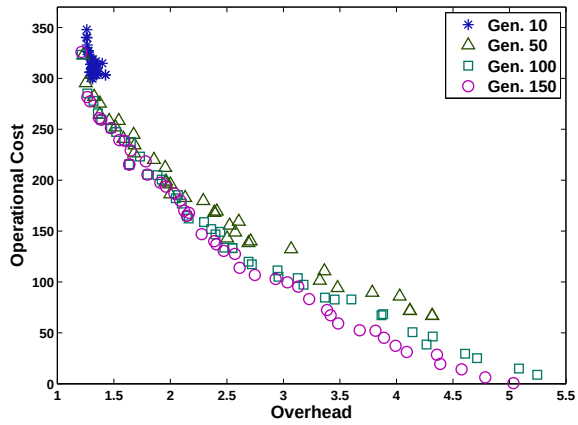


Fig. 2

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION FOR $k = 100$

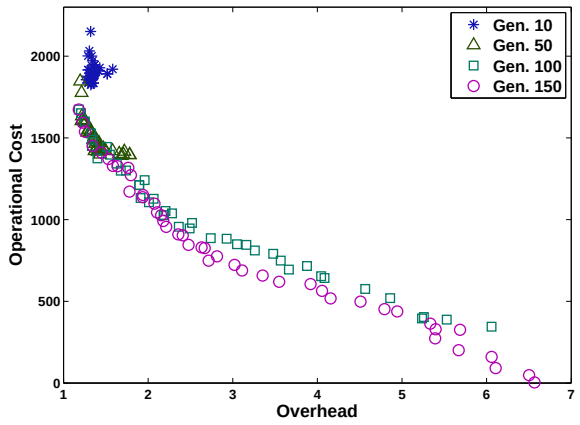


Fig. 4

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF $k = 500$

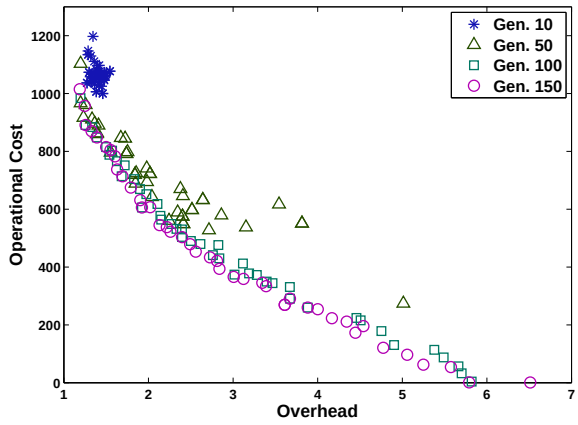


Fig. 3

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF $k = 300$

our simulation of LT code, encoding symbols are generated until source data are fully recovered. The average required codewords are calculated as the fitness. The other objective is the computational cost of the encoding and decoding process. Such an objective value can be estimated with the mean degree of degree distributions. If M_d denotes the mean value of a degree distribution, the number of how many times *XOR* is executed can denote as $(M_d - 1) * \varepsilon$. There is a trade-off between ε and M_d because when M_d is greater, fewer encoding symbols may be required, and therefore, ε is less. On the other hand, M_d is the operational cost, which is the average number of *XOR* operations that have to be executed.

V. EXPERIMENTAL RESULTS

In most multi-objective problems, there is usually a trade-off between objectives. For an n-objective problem, a solution can be represented as a point in the n-dimensional space. All points which denote the non-dominated solutions form a partial optimal set called the Pareto front. The mission of multi-objective algorithms is to approximate the Pareto front as complete as possible. In other words, solutions should well spread to provide sufficient choices to decision markers. In our experiments, overhead and operational cost of LT code are both minimized together and the minimal value of each objective is expected. Clearly, a degree distribution with the minimal operational cost has only non-zero probability on degree one because in such a case, no encoding operation is needed. The case is the pure transmission without any channel coding, and it is a special case in the LT code framework. As for the other objective, overhead has a lower bound at ratio 1. Each encoding symbol can generate a new ripple to recover a source symbol ideally such that at least k encoding symbols are required to reconstruct the original data. Different from the operational cost, such a degree distribution is not yet discovered and even its existence is not proved. Fig. 2 shows the optimization process and the final result. After 150 generations, a significant PF is represented by fifty individual points. The solution with the minimal operational cost in expectation has been found, but the best overhead is 1.2068. Several individuals are listed in Table II, where the best value of overhead and operational cost are presented in columns 2 and 3, respectively. Columns 4 and 5 give the average overhead and execution counts of *XOR* in the numerical simulation. Figs. 3 and 4 display similar results as that shown in Fig. 2 for $k = 300$ and $k = 500$. Fig. 7 presents the distribution and simulation results for each individual listed in Table II.

To our limited knowledge, there is no guideline to design a robust soliton distribution for some particular coding behaviors. In order to fairly compare our optimized results with that of robust soliton distribution, MOEA/D is also applied to optimize the parameters of robust soliton distribution, which are c and δ. The PF of the optimized robust soliton distributions is presented in Fig. 5. In the dimension of operational cost, the optimized robust soliton distributions deliver very similar results because robust soliton distributions can also become the degree distribution with only non-zero probability on degree one if some appropriate parameters are given. However,

TABLE II

OPTIMIZED ARBITRARY DEGREE DISTRIBUTIONS

Individual	Best Overhead	Best Cost	AVG. Overhead	XOR
1	4.8442	0.00042	5.1958	0.038
25	2.5608	1.29873	2.6655	407.026
35	2.0294	1.85193	2.1485	558.667
45	1.4564	2.5135	1.57211	603.742
50	1.2068	2.93541	1.2718	843.669

TABLE III

OPTIMIZED ROBUST SOLITON DISTRIBUTIONS

Individual	Best Overhead	Best Cost	AVG. Overhead	XOR
1	4.8080	0.00552	5.1323	0.377
25	3.1244	1.74314	4.1662	125.455
35	2.0708	2.76115	2.6217	324.580
45	1.5194	3.41297	1.9278	471.753
50	1.2530	6.71008	1.3097	1141.46

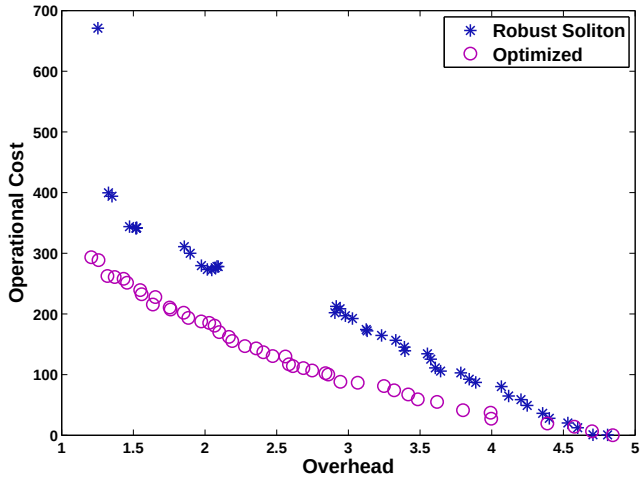


Fig. 5

COMPARISON BETWEEN THE OPTIMIZED ARBITRARY DEGREE DISTRIBUTION AND ROBUST SOLITON DISTRIBUTION

there are significant differences along the other axis. The performance is quite limited, and such a situation is caused by the fixed formula of robust soliton distribution. The figure demonstrates numerous better degree distributions that are very different from robust soliton distribution. These degree distributions can be discovered by optimization algorithms proposed in the realm of evolutionary computation.

VI. CONCLUSIONS

This paper proposed the use of multi-objective evolutionary algorithms to optimize the degree distribution in LT code. Overhead and operational cost were considered as two objectives and optimized simultaneously by using MOEA/D. The experimental results were promising and indicated that the Pareto front was well described. These results might also help researchers to better understand the behavior of LT code. For applications of different types and natures, LT code will be more efficient if choosing a specifically appropriate degree distribution is possible. Not only more choices of degree distributions are available, but also much better performance than that delivered by robust soliton distribution can be achieved, because most robust soliton distributions are dominated by the solutions discovered with MOEA/D in the experiments.

An alternation solution which designs degree distribution better than robust soliton is given in the work. While LT code is employed in real-world apparitions, the degree distribution can be customized to satisfy different requirements by using evolutionary algorithms. Fitter degree distributions will enhance the performance of those applications. Moreover, better understandings of the behavior of LT code will help the improvement of LT code. The final results show that some better distributions are beyond the model of robust soliton distribution. The theoretical analysis will also be applied to them just like the development of soliton distributions in our future work. An advanced model in which the performance is close to that of the Pareto front is in expectation.

ACKNOWLEDGMENTS

The authors would like to thank Martin Hornansky for fruitful discussion and conducting certain related numerical experiments. The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

REFERENCES

[1] D. J. C. MacKay, "Fountain codes," in *The IEE Seminar on Sparse-Graph Codes*, 2004, pp. 1–8.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998, pp. 56–67.

[3] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*, 2002, p. 271.

[4] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 39.

[5] E. A. Bodine and M. K. Cheng, "Characterization of luby transform codes with small message size for low-latency decoding," in *IEEE International Conference on Communications (ICC '08)*, 2008, pp. 1195–1199.

[6] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, 2007, pp. 2576–2580.

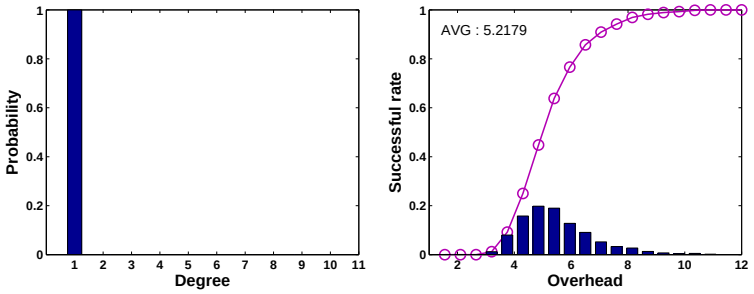
[7] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, 1982, pp. 133–136.

[8] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

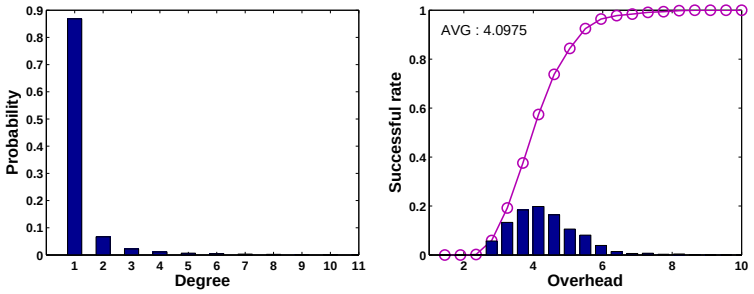
[9] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto set, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[10] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," in *First International Conference on Evolutionary Multi-Criterion Optimization*. Springer Verlag, 2001, pp. 284–298.

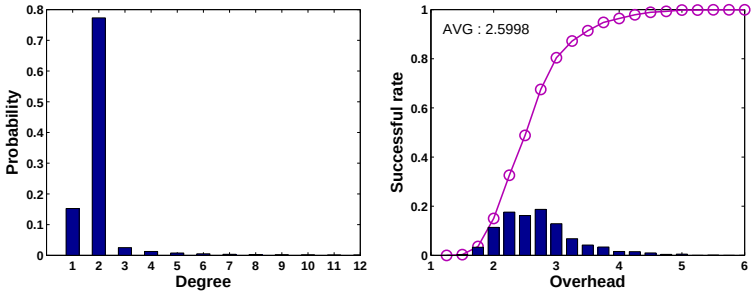
[11] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic, 1999.



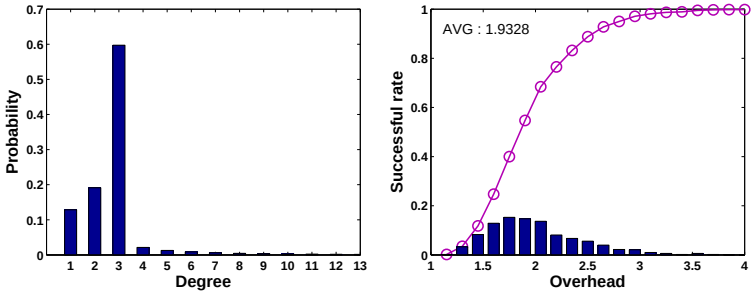
(a) Individual 1, $c = 9.72$ and $\delta = 0.00107$



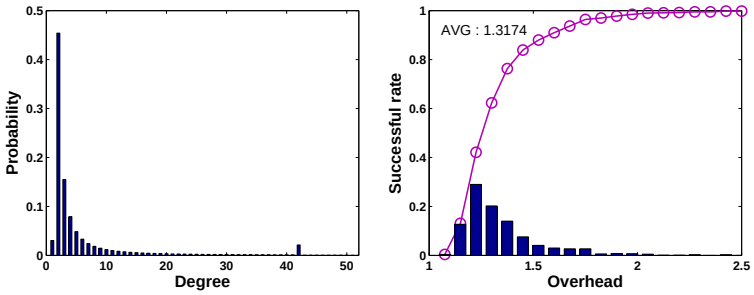
(b) Individual 25, $c = 1.634$ and $\delta = 0.185$



(c) Individual 35, $c = 2.146$ and $\delta = 0.978$



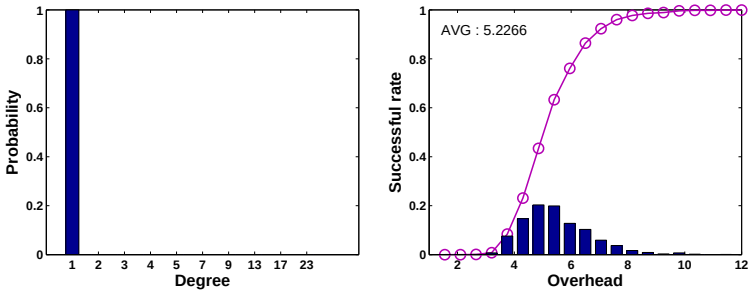
(d) Individual 45, $c = 0.96$ and $\delta = 0.601$



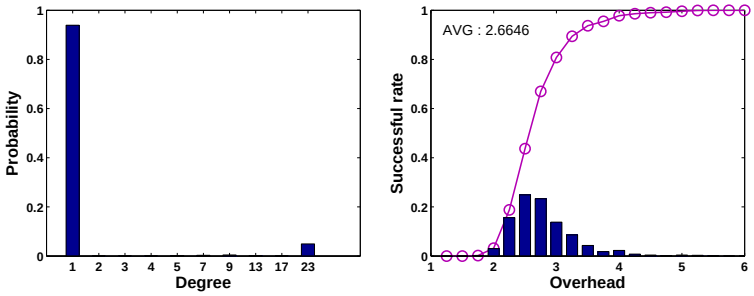
(e) Individual 50, $c = 0.0521$ and $\delta = 0.931$

Fig. 6

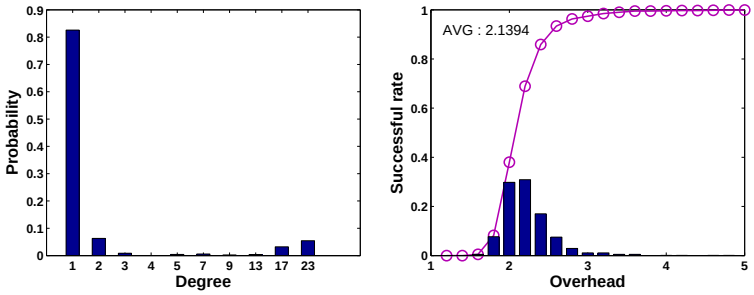
SIMULATION RESULTS OF OPTIMIZED ROBUST SOLITON DISTRIBUTIONS



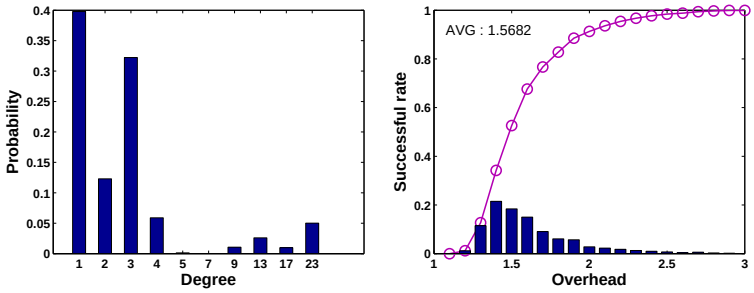
(a) Individual 1



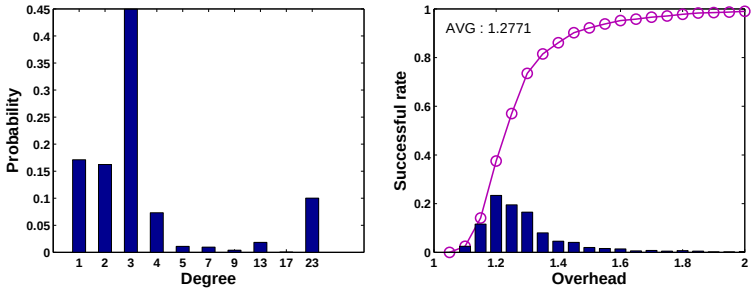
(b) Individual 25



(c) Individual 35



(d) Individual 45



(e) Individual 50

Fig. 7

SIMULATION RESULTS OF OPTIMIZED ARBITRARY DEGREE DISTRIBUTIONS

附件四

國科會補助專題研究計畫項下出席國際學術會議心得報告

日期: 99 年 8 月 17 日

計畫編號	NSC 98 － 2221 － E － 009 － 072 －		
計畫名稱	研究與發展專為無線網路系統客製化之最佳化演算架構		
出國人員 姓名	陳穎平	服務機構 及職稱	國立交通大學資訊工程系副教授
會議時間	99 年 7 月 18 日至 99 年 7 月 23 日	會議地點	Barcelona, Spain
會議名稱	(中文) 2010 IEEE 演化計算學術會議 (英文) 2010 IEEE Congress on Evolutionary Computation		
發表論文 題目	(中文) 研究以 CMA-ES 進行 LT Codes 所採用之度分佈進行最佳化 (英文) On the Optimization of Degree Distributions in LT Codes with Covariance Matrix Adaptation Evolution Strategy		

一、參加會議經過

今年 (2010) 的 IEEE 演化計算學術會議 (IEEE Congress on Evolutionary Computation) 於 7 月 18 日至 7 月 23 日在西班牙巴塞隆那 (Barcelona, Spain) 的 Centre de Convencions Internacional de Barcelona 會議中心舉行。

整個會議的議程包括多場精彩的 Plenary Talks、十數場 Tutorials、Workshops、數項演化計算方法成果競賽、以及論文之口頭報告和海報展。其內容之豐富、主題之精彩，實為演化計算界最高水準之國際學術會議，並足顯演化計算領域蓬勃發展，受到各界之重視。

二、與會心得

吾人除到場進行兩篇論文:「On the Optimization of Degree Distributions in LT Codes with Covariance Matrix Adaptation Evolution Strategy」與「Optimizing Degree Distributions in LT Codes by Using The Multiobjective Evolutionary Algorithm Based on Decomposition」之口頭報告外，亦分別聽取各相關研究論文之口頭報告，增進與交流演化計算各領域知識，獲益良多。由於此會議之水準甚高，因此，如原先所預期，大會所選出的論文品質都相當優良，口頭報告者亦或海報展示者皆盡力展示各自之成果，各研究人員都盡心盡力，做出最好的表現。

由於本實驗室的研究主題，在演化計算領域中，屬於較為理論層面之領域，因此，國內較無與相關學者從事類似研究。故在此會議與國際學者交流之過程中，獲得研究上豐碩之經驗與討論。在與國際級之相關學者切磋最先進的研究議題的同時，本實驗室研究主題的亦吸引許多研究同儕之目光，收穫豐盛。

三、考察參觀活動(無是項活動者略)

無。

四、建議

能夠參與此類國際之大型且質優的研討會，確實令人對學術研究感到前途光明。整個演化計算領域的研究主題不斷地前進、研究內容不斷地提升，而台灣學者被制度面壓迫至專心投稿於發表進度慢如牛車的學術期刊，甚少有機會直接和國際級的頂尖學者面對面談論研究議題，實為台灣學術發展之重要問題。因此，若研究之評鑑制度能放棄唯期刊論文獨尊之思維，並配合更優渥之國際學術會議

出席補助制度，方能迅速提升台灣學界在全世界的能見度、從而全面性地增進台灣的研究水準與學術國力。

五、攜回資料名稱及內容

攜回資料有研討會論文集之光碟一片。

六、其他

無。

On the Optimization of Degree Distributions in LT Code with Covariance Matrix Adaptation Evolution Strategy

Chih-Ming Chen, *Student Member, IEEE*, Ying-ping Chen, *Member, IEEE*,
Tzu-Ching Shen, and John K. Zao, *Senior Member, IEEE*

***Abstract*— Luby Transform code (LT code) has been a popular and practical technique in the field of channel coding since its proposal. One of the key components of LT code is a degree distribution which is used to determine the relationship between source data and codewords. Luby in his proposal suggested two general methods to construct feasible degree distributions. Such general designs work appropriately in typical situations but not optimally in most cases. To explore the full potential of LT code, in this work, we make the first attempt to introduce evolutionary algorithms to optimize the degree distribution in LT code. Degree distributions are encoded as real-valued vectors and evaluated by numerical simulation of LT code. For applications of different natures, two objectives are implemented to search good degree distributions with different decoding behavior. Compared with the original design, the experimental results are quite promising and demonstrate that the degree distribution can be customized for different purposes. In addition to manually adjusting the degree distribution as the common practice, the work presented in this paper provides an efficient alternative approach to use and adapt LT code for both practitioners and researchers.**

I. Introduction

Digital fountain code [1] is a popular class of erasure code in the field of communication. The concept of fountain code was first introduced by Byers et al. [2] in 1998. Firstly, source data are divided into several pieces with an identical length. The length of each piece can be any bits or even several bytes. Sender generates encoding packets, or called encoding symbols when the packet length is one bit, by some particular encoding operation. The encoding and sending procedure may repeat independently and unlimitedly. Infinite encoding packets are sent out continuously like a fountain, which is an important property of fountain code called *rateless*. If a receiver is interested in receiving the data, it can receive the packet flow at any time and collect the packets in any combination. Once sufficient packets, of which the amount is usually slightly more than that of the source data, are obtained, the source data can be fully recovered. During the process, no further communication is required between sender and receiver. Encoding information can be embedded in each packet. As a result, digital fountain code is especially useful in broadcast or other situations in which back channels are unavailable. Moreover, because source data can be reconstructed no matter which packets are received, fountain code is also considered reliable to handle the problem of packet loss.

Chih-Ming Chen, Ying-ping Chen, Tzu-Ching Shen, and John K. Zao are with the Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, TAIWAN (email: ccming@nclab.tw, ypchen@nclab.tw, Stecko.cs97g@nctu.edu.tw, jkzao@cs.nctu.edu.tw).

Luby Transform code (LT code) [3] proposed by Luby in 2002 is the first practical framework of fountain code. A novel coding mechanism based on a specifically designed degree distribution is proposed in the introduction of LT code. The performance of LT code totally depends on the adopted degree distribution. In his proposal, Luby deigned general methods to construct an appropriate degree distribution to be used in LT code, and the degree distribution was named *soliton distribution*. Via theoretical analysis, the feasibility of soliton distribution was proven in the literature [4]. Recently, researchers started to optimize the degree distribution in order to improve the performance of LT code [5], [6], but the obtained improvement is quite limited. In these studies, only the parameters of soliton distribution were tuned and considered as decision variables, while in the present work, we directly consider the degree distribution itself as our decision variables.

Based on LT code, an improved framework call *Raptor codes* [7], [8] was proposed by Shokrollahi. Shokrollahi integrated LT code with a pre-coding layer. Compared with pure LT code, the design of Raptor codes requires a degree distribution, called *weakened LT*, with some very different behavior and properties. Several instances were given in [9] for certain particular sizes of source symbols, but there are no existing guidelines regarding how to construct suitable degree distributions for other sizes. In this regard, we demonstrate the use of optimization techniques proposed in evolutionary computation for generating degree distributions of different, desired properties.

In this paper, according to our limited knowledge, we make the first attempt to utilize evolutionary computation techniques to optimize the degree distribution for LT code and demonstrate the feasibility of customizing degree distributions for different purposes. Particularly, we adopt the covariance matrix adaptation evolution strategy (CMA-ES) [10] to directly optimize degree distributions for two goals: reducing the overhead and lowering the failure rate. The experimental results are remarkably promising and show that significantly reduced overheads and lower failure rates can be achieved for LT code with the obtained degree distribution for a wide range of source symbol sizes.

The remainder of this paper is organized as follows. Section II describes the detailed operations of LT code, including the coding process and soliton distribution proposed by Luby. Section III introduces the evolutionary algorithm used in this paper. Experiments and results are given in section IV. Finally, section V concludes this paper.

978-1-4244-8126-2/10/$26.00

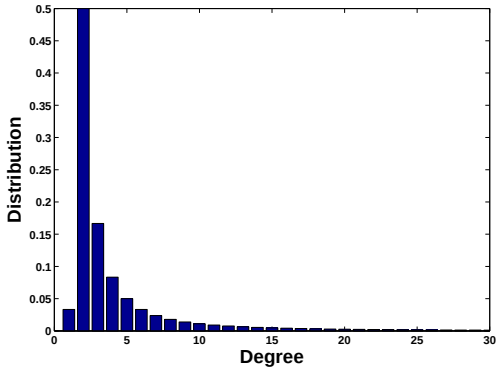
II. LT CODE

Luby introduced a new fountain code framework and gave the detail of coding operation in 2002 [3]. Similar to other fountain codes, source symbols are randomly chosen to be encoded into codewords (encoding symbols). The encoding operation is achieved by a simple boolean operator, *XOR*. The relation between source data and encoding symbols can be modeled as a sparse bipartite graph. A critical change in LT code is to decide the degree of each vertex in the bipartite graph with a probability distribution. The connectivity can be recorded as a encoding matrix and each column represents an encoding symbol. Originally, k source symbols can be fully decoding by Gaussian elimination if there exist k linearly independent columns. However, Gaussian elimination is prohibitively expensive for its computational complexity of $\mathcal{O}(k^3)$. Therefore, the belief propagation (BP) algorithm [11] is introduced to replace the expensive Gaussian elimination in the LT decoding phase. Overhead of coding is used to trade computing time because belief propagation is more efficient but more encoding symbols are needed for successful decoding. Moreover, the performance of LT code is very sensitive to the degree distribution. A good degree distribution is necessary to co-operate with belief propagation. Luby suggested soliton distributions for LT framework in his proposal of LT code. According to the mathematical verification, the properties of soliton distribution have been confirmed. In this section, details of coding operations and soliton distributions are described.

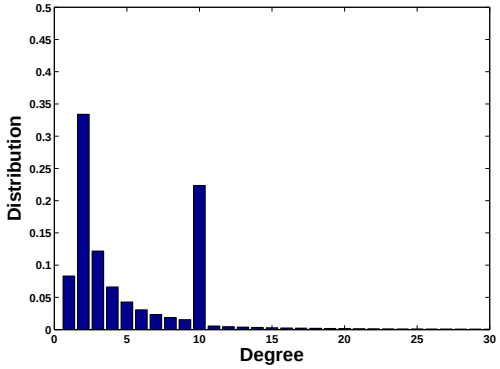
A. Encoding and decoding

Given the source data, we suppose that the source data can be cut into k source symbols with the same length of ℓ bits. Before every codeword is generated, a degree d is chosen at random according to the adopted degree distribution $\rho(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \rho(d) = 1$. The degree d decides the how many distinct source symbols will be chosen to compose an encoding symbol. d source symbols, called *neighbors*, are chosen uniformly randomly and accumulated by XOR. In the design of LT code, random numbers play an essential role during the encoding process. The approach employed by LT code for a sender to inform receivers of all encoding information is achieved by synchronizing a random number generator with a specified random number seed.

At the receiver side, when K encoding symbols were arrived which is usually slightly larger than k, belief propagation is used to reconstruct the source data step by step. All encoding symbols are initially covered in the beginning. For the first step, all encoding symbols with only one neighbor can be directly released to recover their unique neighbor. When a source symbol has been recovered but not processed, it is called a *ripple* and will be stored in a queue. At each subsequent step, ripples are popped as a processing target one by one. A ripple is removed from all encoding symbols which have it as neighbor. If an encoding symbols has only one remaining neighbor after the removing, the releasing action repeats and may produce new ripples to maintain a stable size of the queue. Maintaining the size of the ripple queue is important because the decoding process fails when the ripple queue is empty and some source symbols remain uncovered. In other words, more encoding symbols are required in the decoding process. Ideally, the process succeeds if all source symbols are recovered at the end of the decoding process.



(a) Ideal soliton distribution



(b) Robust soliton distribution

Fig. 1
EXAMPLE OF SOLITON DISTRIBUTIONS (K = 30)

B. Soliton distribution

The behavior of LT code is completely determined by the degree distribution, $\rho(d)$, and the number of encoding symbols received, K, by receiver. The overhead $\varepsilon = K/k$ denotes the performance of LT code, and ε depends on a given degree distribution. Based on his theoretical analysis, Luby proposed the ideal soliton distribution of which the overhead is 1, the best performance, in the ideal case.

Ideal soliton distribution $\rho(d)$:

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for} \quad d = 1 \\ \frac{1}{d(d-1)} & \text{for} \quad d = 2, 3, \ldots, k \end{cases} . \tag{1}$$

Ideal soliton distribution guarantees that all the release probabilities are identical to $1/k$ at each subsequent step. Hence, there is exactly one expected ripple generated at each processing step when the encoding symbol size is k. After k processing step, the source data can be ideally recovered. Fig. 1(a) shows an example of ideal soliton distribution for $k = 30$.

However, ideal soliton distribution works poorly in practice. Belief propagation may be suspended by a small variance of the stochastic encoding/decoding situation in which no ripple exists, because the expected ripple size is only one at any moment. According to the theory of random walk, the probability with which a random walk of length k deviates from its mean by more than $\ln(k/\delta)\sqrt{k}$ is at most δ. It is a baseline of ripple sizes which must be maintained to complete the decoding process. Hence, in the same paper by Luby, a modified version called *robust soliton distribution*, $\mu(d)$, was also proposed.

Robust soliton distribution $\mu(d)$:

$$R = c \cdot \ln(k/\delta)\sqrt{k}$$

$$\tau(d) = \begin{cases} R/ik & \text{for} \quad d = 1, \ldots, k/R - 1 \\ R\ln(R/\delta)/k & \text{for} \quad d = k/R \\ 0 & \text{for} \quad d = k/R + 1, \ldots, k \end{cases} . \quad (2)$$

$$\begin{aligned} \beta &= \sum_{d=1}^{k} (\rho(d) + \tau(d)) \\ \mu(d) &= \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k \end{aligned} \quad (3)$$

c and δ are two parameters for tuning robust soliton distribution. c controls the mean of the degree distribution. Smaller values of c increase the probability of low degrees and larger ones decrease it. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripple size as described. Fig. 1(b) is an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.1$. Robust soliton distribution can ensure that only $K = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoding symbols are required to recover the source data with a successful probability at least 1-δ.

Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound if k is infinite. However, in practice, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT code will not exactly match the mathematical analysis, especially when k is small. Furthermore, robust soliton distribution is a general purpose design. It provides a convenient way to construct a distribution works well but not optimally. In this work, we try to customize the degree distribution by using optimization tools proposed in the field of evolutionary computation.

III. Optimization Method

Evolution strategies (ES) are a major branch of evolutionary computation and have been developed since early 1960s. The key idea of ES is to evolve strategic parameters as well as decision variables. ES is well-known to be quite capable of dealing with continuous optimization problems. One of the simplest ES is (1+1)-ES where only one child is produced by Gaussian mutation to compete with its parent in each generation, and the other is (1, 1)-ES which is equivalent to random walk. Current general versions of ES are denoted as $(\mu^+_,\lambda)$-ES. The covariance matrix adaptation evolution strategy (CMA-ES) [10] was firstly introduced by Hansen in 1996 and is one of the most popular real-parameter optimization methods in evolutionary computation. There are some variants of CMA-ES proposed in the literature [12], [13], [14]. The search ability of CMA-ES has been theoretically analyzed and empirically verified on certain classic optimization problems, such as Ackley's function, Griewank's function, and Rastrigin's function. In CMA-ES, only a few algorithmic parameters need to be decided because CMA-ES inherits the mechanism to adapt strategic parameters during the evolutionary process. In this work, CMA-ES is utilized to optimize the degree distribution in LT framework for a wide range of k, the size of source symbols. In the remainder of this section, the way to adopt CMA-ES to handle the optimization of degree distributions are presented in detail.

A. Decision Variables

The first step to use an evolutionary algorithm is to encode the decision variables of the optimization problem. It is not difficult in this study because a degree distribution can directly form a real-number vector. In the evaluation phase, a real-number vector of arbitrary values can be interpreted as a probability distribution, i.e., a degree distribution, with normalization. Such an operation does not change the feasibility, although the problem complexity may be slightly increased. The definition of degree distributions tells us that $d \leq k$. For a specific source symbol size k, obviously the problem dimensions is at most k. However, according to the LT encoding/decoding operations, we usually do not need a non-zero probability on every single degree. Observing the soliton distributions and considering the belief propagation algorithm, there is no necessary degree except 1, which ensures the start of belief propagation. As a result, we optimize a selected subset of degrees in the present work. We choose some degrees called *tags* to form the vector $v(i)$ of decision variables according to the Fibonacci numbers smaller than half of k. A degree distribution used in this paper hence can be represented as the following formula.

Optimized degree distribution $\omega(d)$:

$$\omega(d) = \begin{cases} v(i) & d = \text{the } i\text{-th Fibonacci number}, d < k/2 \\ 0 & \text{otherwise} \end{cases} . \quad (4)$$

B. Objectives

We try to use two indicators to evaluate degree distributions for LT code in this paper. The first one is the efficiency of the LT code with the optimized degree distribution which has been discussed in section II-B. ε denotes the expected rate of overhead to transmit data. For example, $\varepsilon = 1.2$ means that in addition to the size of source data, 20% extra data are needed to recover the complete source data. This objective is to obtain some degree distribution for a specific k with the smallest ε. LT code is rateless, and the coding process depends on randomness and probability. Source data recovered by a fixed amount of encoding symbols cannot be guaranteed. Therefore,

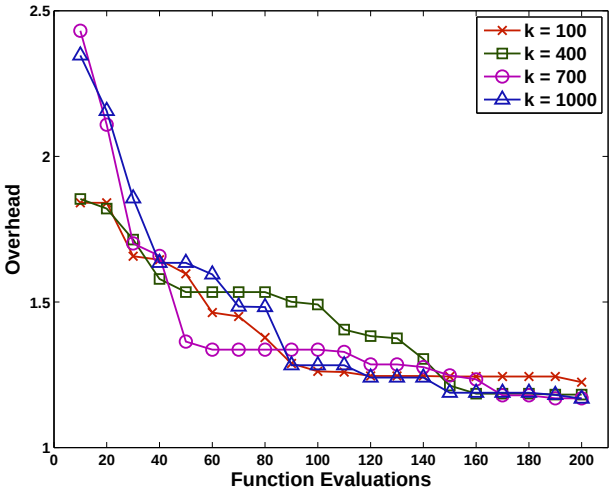


Fig. 2

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF OVERHEAD

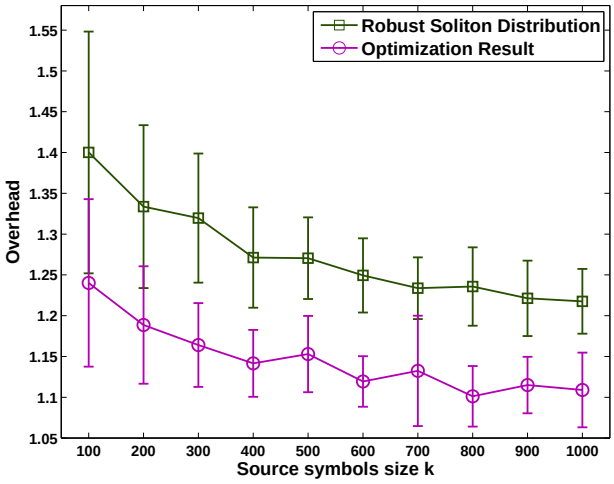


Fig. 3

AVERAGE PERFORMANCE INDICATORS ARE COMPARED BETWEEN ROBUST SOLITON DISTRIBUTION AND OPTIMIZED DEGREE DISTRIBUTIONS FOR DIFFERENT NUMBERS OF SOURCE SYMBOLS (k)

TABLE I

THE BEST INDIVIDUALS FOR THE OPTIMIZATION OF OVERHEAD

Degree	k=100	$k = 400$	$k = 400$	$k = 1000$
1	0.091397	0.116375	0.16058	0.129707
2	0.310884	0.255701	0.148543	0.266133
3	0.367223	0.34174	0.412275	0.321489
5	0.042648	0.112072	0.119163	0.077045
8	0.053247	0.071726	0.052843	0.124503
13	0.048949	0.028076	0.024701	0.000258
21	0.011876	0.013169	0.035112	0.019594
34	0.073776	0.030397	0.017738	0.033607
55	0	0.000264	0.002094	0.01543
89	0	0.01109	0.009837	0.00095
144	0	0.01939	0.002946	0.000143
233	0	0	0.014167	0.00075
377	0	0	0	0.010391

in order to evaluate ε, we provide infinite encoding symbols, in the form of a stream of encoding symbols, to simulate the decoding process until all source data are recovered. The average of required encoding symbols per simulation is the fitness value of degree distributions.

The second indicator is the amount of source symbols that cannot be recovered when a constant ratio of encoding symbols are received. In raptor codes, Low-density-parity-check (LDPC) [15] is introduced as a second layer pre-coding into LT code. LDPC is a kind of forward error correction codes. More information on LDPC can be found in [16], [17]. LDPC can fix errors of data without extra information as long as the error rate is lower than certain restriction. In such a condition, the mission of LT code is no longer to achieve full decoding. Instead, most of source symbols can be recovered with a small overhead is sufficient. For this purpose, we try to minimize the number of un-recovered source symbols given a constant overhead ε.

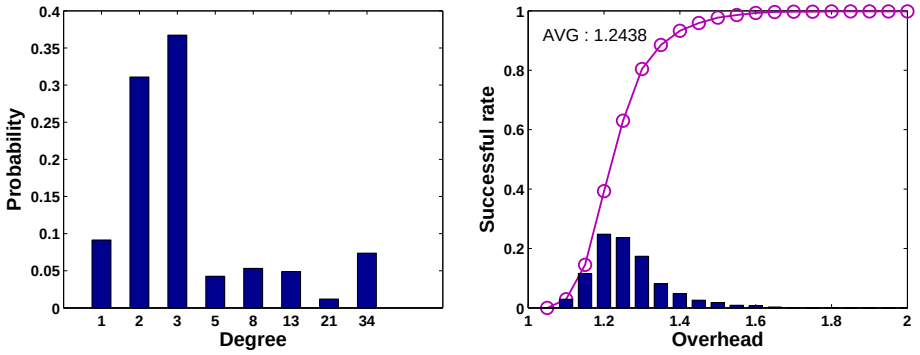
IV. EXPERIMENTS AND RESULTS

Two series of experiments are implemented for the two different objectives as described in the previous section. In each experiment, *tags* are determined by Fibonacci numbers and the specified source symbols size k. Tags are encoded as an individual, $v(i)$, and represent that only these degrees have non-zero probabilities. Initial values of tags are set as $1/|v|$ uniformly, and then CMA-ES is applied without any customization or modification. After a new individual is created, it is normalized to be a valid probability distribution and evaluated for the fitness value by simulating the LT coding process. One hundred independent runs of simulation are conducted for each function evaluation. In the first series of experiments, we minimize the expected number of encoding symbols for full decoding. In the second, the average number of source symbols that cannot be recovered for a constant $\varepsilon = 1.1$ is considered. We call the second indicator as *failure rate*. The default parameter settings given in the source code of CMA-ES are adopted in this study except for $\lambda = 10$.

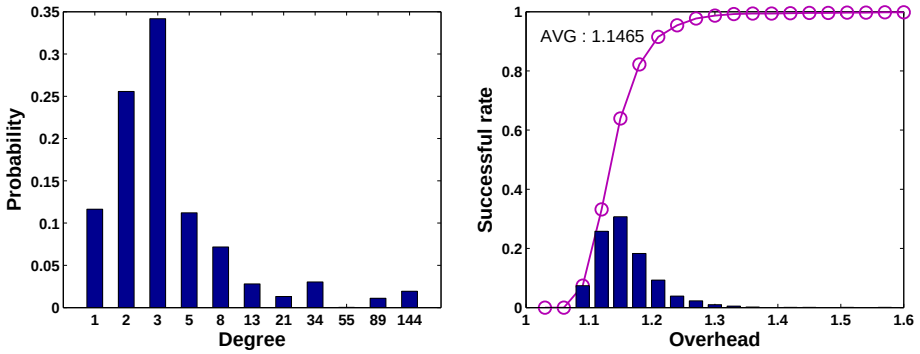
A. Overhead

In these experiments, we minimize the overhead ε for different k sizes, and the results are shown in Table I and Figs. 2–5. Fig. 2 presents the improvement during the evolutionary process. Individuals are initially uniform distributions. It is expected that overheads are quite high in the beginning and the curves descend quickly after around 100 function evaluations. Finally, the fitness almost converges after 200 function evaluations. Fig. 3 shows the comparison of ε between robust soliton distribution and the optimized distributions. The expected overhead of robust soliton distribution is given as

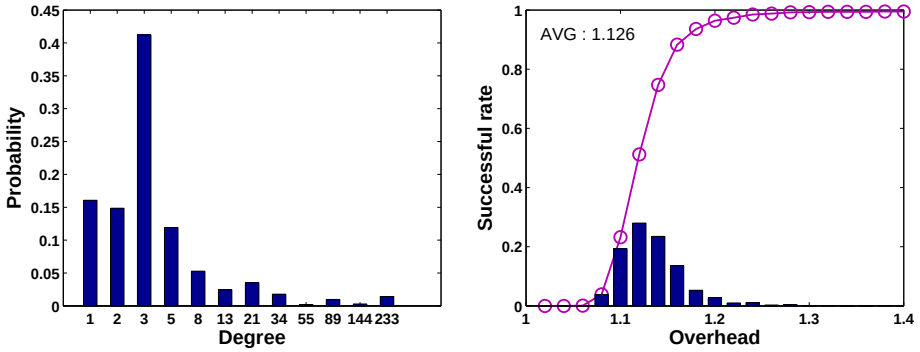
$$\frac{k + \mathcal{O}(\log^2(k/\delta)\sqrt{k})}{k} = 1 + \mathcal{O}\left(\frac{\log^2(k/\delta)}{\sqrt{k}}\right).$$



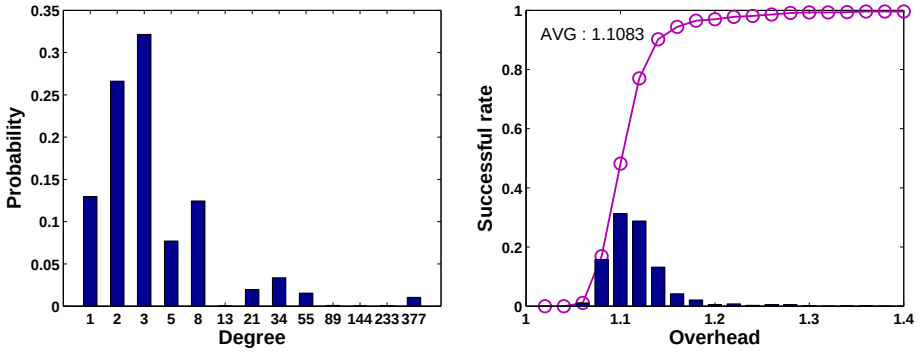
(a) $k = 100$



(b) $k = 400$



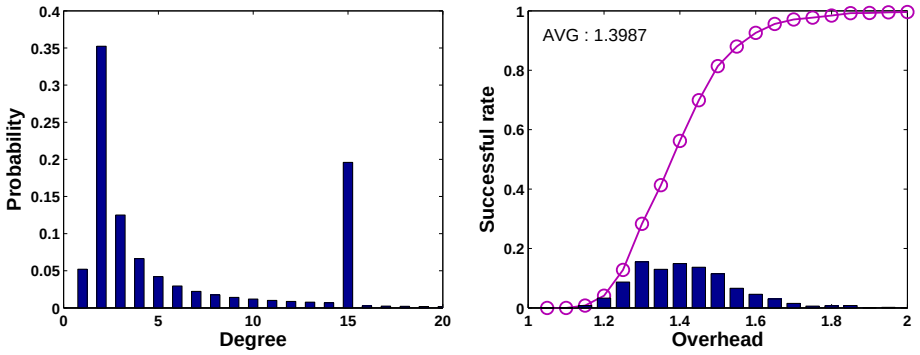
(c) $k = 700$



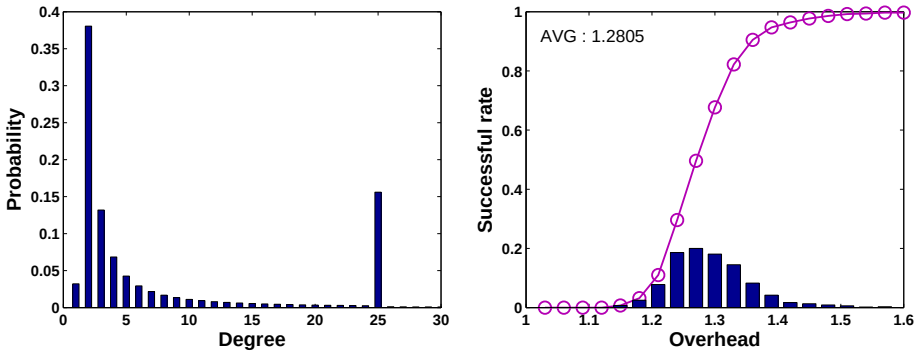
(d) $k = 1000$

Fig. 4

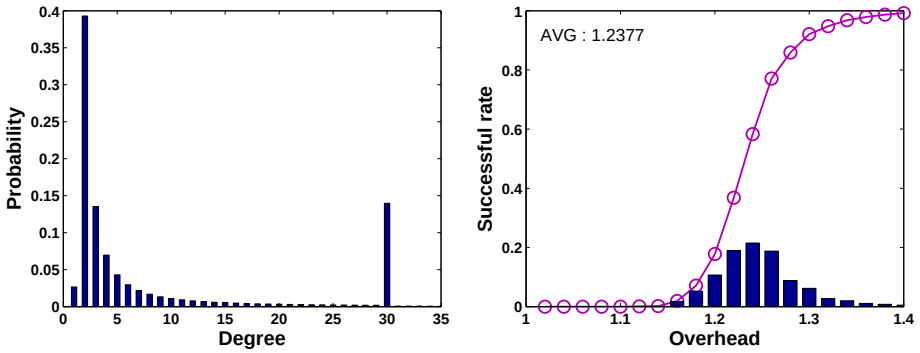
LEFT FIGURES SHOW THE OPTIMIZED DEGREE DISTRIBUTIONS. ONLY TAGS ARE PRESENTED. RIGHT FIGURES ARE THE HISTOGRAM AND ACCUMULATED CURVE OF SUCCESSFUL RATE IN 1000 INDEPENDENT SIMULATION RUNS



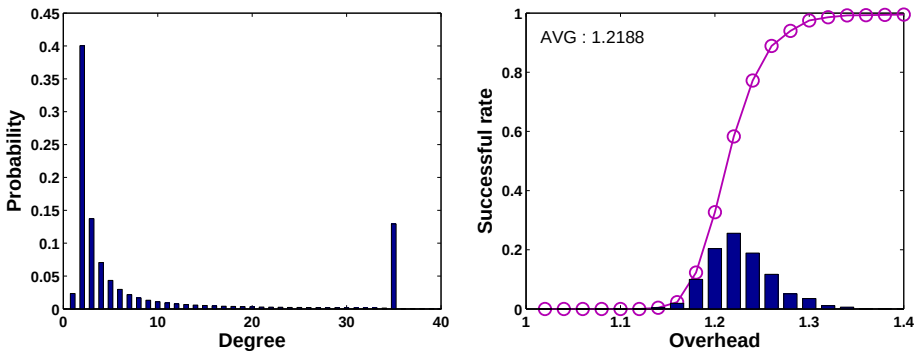
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

Fig. 5

FOR THE COMPARISON WITH SAME k'S, ROBUST SOLITON DISTRIBUTIONS AND THE CORRESPONDING PERFORMANCE INDICATORS ARE SHOWN SIMILAR TO THAT IN FIG. 4. NOTE THAT ONLY PARTS OF ROBUST SOLITON DISTRIBUTIONS ARE PLOTTED FOR CLARITY

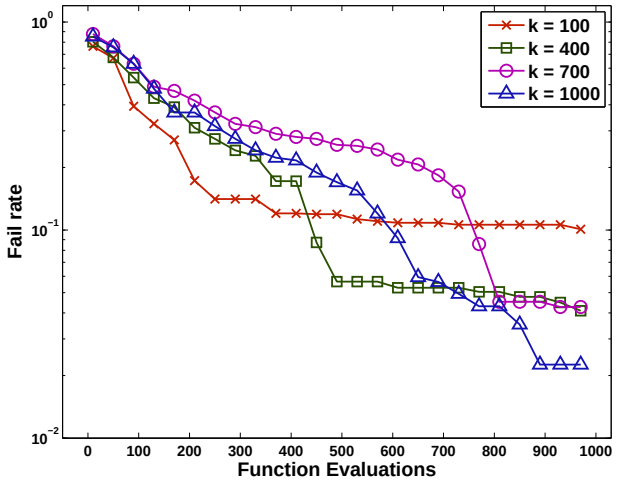


Fig. 6

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF FAILURE RATE

TABLE II

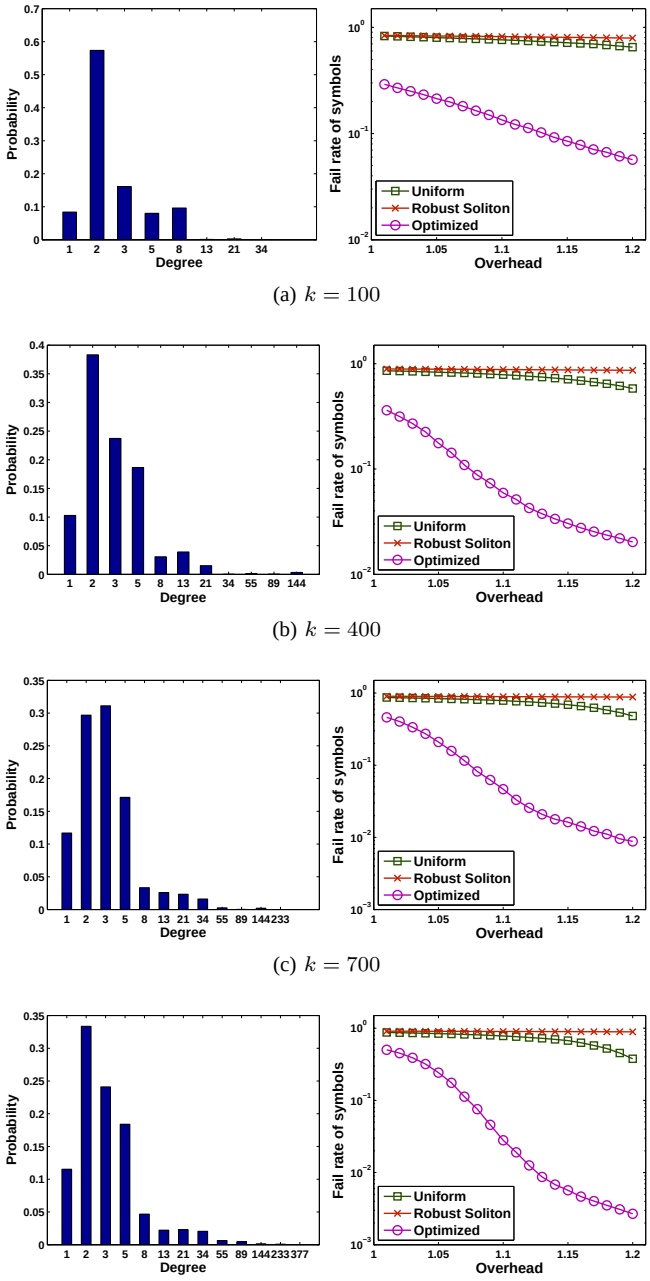
THE BEST INDIVIDUALS FOR THE OPTIMIZATION OF FAILURE RATE

Degree	k=100	$k = 400$	$k = 400$	$k = 1000$
1	0.083997	0.102892	0.116854	0.115278
2	0.573671	0.383164	0.29678	0.333564
3	0.161178	0.237312	0.31115	0.241065
5	0.08038	0.186475	0.171342	0.184027
8	0.096245	0.030706	0.033393	0.046818
13	0.001267	0.039075	0.025977	0.022223
21	0.002963	0.015193	0.023452	0.022914
34	0.000299	0.000167	0.016096	0.020526
55	0	0.001276	0.002602	0.00643
89	0	0.000303	0.000268	0.004594
144	0	0.003436	0.002072	0.001422
233	0	0	0.000015	0.000883
377	0	0	0	0.000257

The value becomes smaller when k increases, and that is why the trend of Fig. 3 shows a declination. The values of overhead are reduced at least 10% for all k's with the optimized degree distributions. Some distributions of the best individuals are given in Table I. Fig. 4 illustrates each distribution and shows the histogram of successful rate in 1000 simulation runs on the right side. Compared with similar simulation results of robust soliton distribution in Fig. 5, the improvement is quite significant.

B. Failure rate

Unlike the original LT code, we are concerned with how many source symbols can be recovered in the second set of experiments. The objective value is the average number of source symbols that cannot be recovered with a constant overhead ε. Optimization results are shown in Fig. 6. More function evaluations are needed to search for good degree distributions. The failure rate of the final results are less than 10^{-1} for all k's when $\varepsilon = 1.1$. In other words, more than 90 percent of source symbols can be recovered if extra 10 percent of encoding symbols are collected. Table II gives



(a) $k = 100$

(b) $k = 400$

(c) $k = 700$

(d) $k = 1000$

Fig. 7

THE FIGURE SHOWS THE SIGNIFICANT DIFFERENCE OF FAILURE RATE AFTER OPTIMIZATION. SIMILAR TO THAT IN FIG. 4, ONLY TAGS ARE SHOWN IN THE FIGURES

the best probability distributions found in the evolutionary process for $k = 100$, $k = 400$, $k = 700$, and $k = 1000$. The simulation results of a constant overhead are presented in Fig. 7. The red line denotes the behavior of uniform distribution, which is the initial value of optimization. Most of the source symbols remain covered except for those of which the degree is one, i.e., with probability $1/k$. The same situation happens to robust soliton distributions because the

amount of extra encoding symbols is not sufficient to complete the BP decoding process. The behavior of LT process with the optimized degree distributions is totally different and fully satisfies the requirement of weakened LT.

V. Conclusions

In this work, the first attempt to algorithmically optimize the degree distribution adopted in LT code was proposed. Evolutionary computation techniques were introduced to accomplish the optimization task. Different from the previous studies reported in the literature, each probability of degrees were directly encoded as an individual to optimize. Promising experimental results were obtained in both sets of experiments: One was to minimize the overhead, and the other was to reduce the decoding failure rate. Our experiments showed that CMA-ES was indeed capable of finding good degree distributions for different purposes without any guideline or human intervention. Compared with robust soliton distribution, the optimized overhead was decreased as least 10% for every k in the experiments. The results of failure rate minimization were also remarkably promising and able to support applications of different types and requirements.

This study creates a new research topic in which the design of degree distributions in LT code can now be algorithmic and no longer has to be manually tuning parameters of robust soliton distribution. We have empirically proved that directly manipulating the probability value for each degree is viable and worth pursuing. Given a specific k and some expected overhead, a degree distribution can be customized with existing optimization techniques. In addition, we will extend the experiments to larger k for more kinds of potential applications in the near future. The results empirically obtained by using evolutionary algorithms will be theoretically analyzed, and general guidelines, like robust soliton distribution, that are able to be customized for different goals and requirements for designing degree distributions are expected.

Acknowledgments

The authors would like to thank Martin Hornansky for fruitful discussion and conducting certain related numerical experiments. The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] D. J. C. MacKay, "Fountain codes," in *The IEE Seminar on Sparse-Graph Codes*, 2004, pp. 1–8.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998, pp. 56–67.

[3] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, 2002, p. 271.

[4] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 39.

[5] E. A. Bodine and M. K. Cheng, "Characterization of luby transform codes with small message size for low-latency decoding," in *IEEE International Conference on Communications (ICC '08)*, 2008, pp. 1195–1199.

[6] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, 2007, pp. 2576–2580.

[7] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channels," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 38.

[8] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *IEEE Transactions on Information Theory*, vol. 52, no. 5, pp. 2033–2051, 2006.

[9] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[10] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 312–317.

[11] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, 1982, pp. 133–136.

[12] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 1777–1784.

[13] ——, "A restart cma evolution strategy with increasing population size," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 1769–1776.

[14] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Proceedings of the Tenth International Conference on Parallel Problem Solving from Nature (PPSN X)*, 2008, pp. 296–305.

[15] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[16] D. Changyan, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1570–1579, 2002.

[17] E. Paolini and M. Chiani, "Improved low-density parity-check codes for burst erasure channels," in *IEEE International Conference on Communications (ICC '06)*, vol. 3, 2006, pp. 1183–1188.

Optimizing Degree Distributions in LT Codes by Using The Multiobjective Evolutionary Algorithm Based on Decomposition

Chih-Ming Chen, *Student Member, IEEE*, Ying-ping Chen, *Member, IEEE*,
Tzu-Ching Shen, and John K. Zao, *Senior Member, IEEE*

***Abstract*— Luby Transform code (LT code) is the first practical digital fountain code and has been widely used as basic components in many communication applications. The coding behavior of LT code is mainly decided by a probability distribution of codeword degrees. In order to customize a degree distribution for different purposes, multi-objective evolutionary algorithm is introduced to optimize degree distributions in this paper. Two critical performance indicators of LT code are considered in our experiments. Some applications hope to minimize the overhead of extra packets and some require to limit the computational cost of the coding system. To handle this problem, MOEA/D is applied to optimize two objectives simultaneously. We expect to obtain the Pareto front (PF) formed by partial optimal solutions and provide those available degree distributions to different LT code applications. Not only promising results are represented in this paper but also the behavior of LT code is thoroughly explored by optimizing the degree distribution according to multi-objectives.**

I. Introduction

Digital fountain code [1] is a popular class of erasure code in the field of communication. The concept of fountain code was introduced by Byers et al. [2] in 1998. Firstly, source data are divided into several pieces with an identical length. The length of each piece can be any number of bits or even several bytes. Sender generates encoding packets, or called *encoding symbols*, when the packet length is one bit, by certain encoding operation. The encoding procedure may repeat independently and indefinitely so infinite encoding packets are sent out continuously like a fountain, which is an important property of fountain code called *rateless*. If a receiver is interested in receiving the data, it can receive the packet flow at any time and collect the packets in any combination. Once sufficient packets, of which the amount is usually slightly more than that of the source data, are obtained, the source data can be fully recovered. During the process, no further communication is required between sender and receiver. Encoding information can be embedded in each packet. As a result, digital fountain code is especially useful in broadcast or other situations in which back channels are unavailable. Moreover, because source data can be reconstructed no matter which packets are received, fountain code is also considered reliable to handle the problem of packet loss.

Luby Transform code (LT code) [3] proposed by Luby in 2002 is the first practical framework and implementation of fountain code. A novel coding mechanism based on a specifically designed degree distribution is proposed in the introduction of LT code. The performance of LT code totally depends on the adopted degree distribution. In his proposal, Luby designed general methods to construct appropriate degree distributions to co-operate with LT code, and the degree distributions were named *soliton distribution*. Via theoretical analysis, the feasibility of soliton distribution was proven [4]. Recently, researchers started to optimize the degree distribution in order to improve the performance of LT code [5], [6], but the obtained improvement is marginal and quite limited. In these studies, only the parameters of soliton distribution were tuned and considered as decision variables, while in our present work, we directly consider the degree distribution itself as our decision variables.

In the design of LT code, redundant data and encoding computation are used to trade for the ability of forward error correction. For most applications, while the error correction ability is maintained, both costs are required to be as lower as possible, and apparently there is a trade-off among these factors. Furthermore, applications of different types and purposes have different requirements of each kind of cost. Some LT code applications which transmit data through an expensive communication channel have to reduce the data overhead. Other applications with a huge package size expect fewer executions of the encoding operator. In order to simultaneously satisfy these applications, multi-objectives are considered for optimizing the LT code degree distribution in the present work. The most important motivation of this study is to fully explore the LT coding behavior with arbitrary degree distributions and to empirically provide a proof of concept that multiple requirements on LT code can be satisfied via optimizing degree distributions with existing optimization techniques.

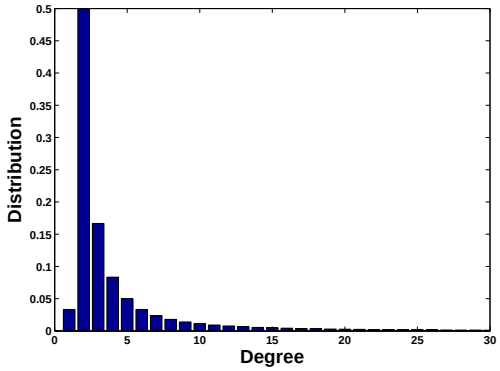
The remainder of this paper is organized as follows. Section II describes the detailed operations of LT code, including the coding process and soliton distribution. Section III introduces the background of multi-objective problems and the evolutionary algorithm used in this paper. Experiments and results are given in sections IV and V. Finally, section VI concludes this paper.

II. LT code

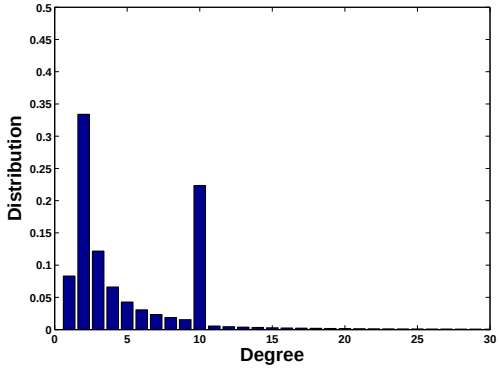
Luby introduced a practical fountain code framework and gave the details of coding operation in 2002 [3]. Similar to other fountain codes, source symbols are uniformly randomly chosen to be encoded into codewords (encoding symbols). The encoding operation is achieved by a simple boolean operator,

Chih-Ming Chen, Ying-ping Chen, Tzu-Ching Shen, and John K. Zao are with the Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, TAIWAN (email: ccming@nclab.tw, ypchen@nclab.tw, Stecko.cs97g@nctu.edu.tw, jkzao@cs.nctu.edu.tw).

978-1-4244-8126-2/10/$26.00



(a) Ideal soliton distribution



(b) Robust soliton distribution

Fig. 1

EXAMPLE OF SOLITON DISTRIBUTIONS (k = 30)

XOR. The relation between source data and encoding symbols can be modeled as a sparse bipartite graph. A key design of LT code is to decide the degree of each vertex in the bipartite graph with a probability distribution. The connectivity can be recorded as an encoding matrix and each column represents an encoding symbol. Originally, k source symbols can be fully decoded by Gaussian elimination if there exist k linearly independent columns. However, Gaussian elimination is prohibitively expensive for its computational complexity of $\mathcal{O}(k^3)$. Therefore, the belief propagation (BP) algorithm [7] is introduced to replace the expensive Gaussian elimination in the LT decoding phase. Overhead of coding is used to trade computing time because belief propagation is more efficient but more encoding symbols are needed for successful decoding. Moreover, the performance of LT code is very sensitive to the degree distribution. A good degree distribution is necessary to co-operate with belief propagation. Luby suggested soliton distributions for LT framework in his proposal of LT code. According to the mathematical verification, the properties of soliton distribution have been confirmed. In this section, details of coding operations and soliton distributions are described.

A. Encoding and decoding

Given the source data, we suppose that the source data can be cut into k source symbols with the same length of ℓ bits. Before every codeword is generated, a degree d is chosen at random according to the adopted degree distribution $\rho(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \rho(d) = 1$. The degree d decides the how many distinct source symbols will be chosen to compose an encoding symbol. d source symbols, called *neighbors*, are chosen uniformly randomly and accumulated by *XOR*. In the design of LT code, random numbers play an essential role during the encoding process. The approach employed by LT code for a sender to inform receivers of all encoding information is achieved by synchronizing a random number generator with a specified random number seed.

At the receiver side, when K encoding symbols were arrived, where K is usually slightly larger than k, belief propagation is used to reconstruct the source data step by step. All encoding symbols are initially covered in the beginning. For the first step, all encoding symbols with only one neighbor can be directly released to recover their unique neighbor. When a source symbol has been recovered but not processed, it is called a *ripple* and will be stored in a queue. At each subsequent step, ripples are popped as a processing target one by one. A ripple is removed from all encoding symbols which have it as neighbor. If an encoding symbols has only one remaining neighbor after the removing, the releasing action repeats and may produce new ripples to maintain a stable size of the queue. Maintaining the size of the ripple queue is important because the decoding process fails when the ripple queue is empty and some source symbols remain covered. In other words, more encoding symbols are required in the decoding process. Ideally, the process succeeds if all source symbols are recovered at the end of the decoding process.

Both encoding and decoding, as the LT coding operations, are achieved by *XOR*. As a result, the computational complexity of LT code can be measured by how many times of *XOR* is executed. *XOR* operator is applied to build the connectivity in the conceptualized bipartite graph and to eliminate a ripple from the neighbors of codewords. It is evident that $d - 1$ *XOR* operators are necessary to generated a codeword with degree d or recover an encoding symbol. In the encoding phase, all encoding symbols are generated independently, and the computational complexity to produce codewords solely depends on the mean degree of the adopted degree distribution. In other words, the cost of each encoding symbol is decided by the mean of degree distributions. Hence, in practice, the mean degree is an important LT performance indicator since it represents the operational cost.

B. Soliton distribution

The behavior of LT code is completely determined by the degree distribution, $\rho(d)$, and the number of encoding symbols received, K, by receiver. The overhead $\varepsilon = K/k$ denotes the performance of LT code, and ε depends on a given degree distribution. Based on his theoretical analysis, Luby proposed the ideal soliton distribution of which the overhead is 1, the best performance, in the ideal case.

Ideal soliton distribution $\rho(d)$:

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for} \quad d = 1 \\ \frac{1}{d(d-1)} & \text{for} \quad d = 2, 3, \ldots, k \end{cases} . \tag{1}$$

Ideal soliton distribution guarantees that all the release probabilities are identical to $1/k$ at each subsequent step. Hence, there is *one* expected ripple generated at each processing step when the encoding symbol size is k. After k processing step, the source data can be ideally recovered. Fig. 1(a) shows an example of Ideal soliton distribution for $k = 30$.

However, ideal soliton distribution works poorly in practice. Belief propagation may be suspended by a small variance of the stochastic encoding/decoding situation in which no ripple exists, because the expected ripple size is only one at any moment. According to the theory of random walk, the probability with which a random walk of length k deviates from its mean by more than $\ln(k/\delta)\sqrt{k}$ is at most δ. It is a baseline of the ripple queue size which must be maintained to complete a decoding process. Hence, in the same paper by Luby, a modified version called *robust soliton distribution*, $\mu(d)$, was also proposed.

Robust soliton distribution $\mu(d)$:

$$R = c \cdot \ln(k/\delta)\sqrt{k}$$

$$\tau(d) = \begin{cases} R/ik & \text{for} \quad d = 1, \ldots, k/R - 1 \\ R\ln(R/\delta)/k & \text{for} \quad d = k/R \\ 0 & \text{for} \quad d = k/R + 1, \ldots, k \end{cases} . \tag{2}$$

$$\begin{aligned} \beta &= \sum_{d=1}^{k} (\rho(d) + \tau(d)) \\ \mu(d) &= \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k \end{aligned} \tag{3}$$

c and δ are two parameters for tuning robust soliton distribution. c controls the mean of the degree distribution. Smaller values of c increase the probability of low degrees, and larger ones decrease it. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripples as described. Fig. 1(b) is an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.1$. Robust soliton distribution can ensure that only $K = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoding symbols are required to recover the source data with a successful probability at least 1-δ.

Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound if k is infinite. However, in practice, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT code will not exactly match the mathematical analysis, especially when k is small. Furthermore, robust soliton distribution is a general purpose design. It provides a convenient way to construct a distribution works well but not optimally. In this work, we try to customize the degree distribution by using multi-objective optimization tools proposed in the field of evolutionary computation to simultaneously satisfy multiple performance requirements.

III. Multi-objective Problems

Multi-objective optimization problems (MOPs) are very important in real-world applications. There are two or more objectives to be considered simultaneously, and these objectives usually conflict with each other. The most intuitive approach to deal with MOPs is to transform them into single objective problems (SOPs) by using weights on the objectives and creating a weighted sum. The approach makes the problem solvable by available tools based on mathematics or heuristics for SOPs. However, such weights oftentimes cannot be predetermined, especially when the domain knowledge of the problem is unavailable. Furthermore, the best solution to the transformed single-objective problem is merely one solution on the Pareto front (PF) of the MOP. Hence, better optimization frameworks must be developed to fulfill the need of handling MOPs.

Due to the limitation of traditional mathematical methods for MOPs, more and more researchers try to solve MOPs in a direct way and to approximate the Pareto front as complete as possible. Their goal is to provide a set of solutions which are partially optimal. Many advanced multi-objective algorithms have been proposed in the literature. Some of them try to approximate the PF by using mathematical models, and others are developed based on evolutionary algorithms. A hybrid framework makes use of decomposition methods in mathematics and the optimization paradigm in evolutionary computation was proposed and called *multiobjective evolutionary algorithm based on decomposition* (MOEA/D) [8]. MOEA/D was proposed and shown to perform well on MOPs with complicated Pareto set shapes [9].

In this paper, we propose the use of MOEA/D to optimize the multiple objectives of LT code. Degree distributions significantly better than *robust soliton distribution* are expected. Moreover, exploring a complete Pareto front can help researchers to analyze the trade-off between overhead and operational cost of LT code. In the following section, we will give the formal description of MOPs and the MOEA/D framework, respectively.

A. Formal description of MOPs

In real-world applications, many problems are actually multi-objective optimization problems, and single-objective problems are special cases. A multi-objective problem can be formally stated as:

$$\begin{aligned} &\text{minimize} \quad F(x) = (f_1(x), \ldots, f_m(x)) \\ &\text{subject to} \quad \begin{cases} x \in \Omega \\ C(x) = (c_1(x), \ldots, c_t(x)) \geq 0 \end{cases} \end{aligned} , \tag{4}$$

where Ω is called the *decision space* or *variable space*, and R^m is the *objective space*. $C(x)$ represents the problem constraints and defines the feasible regions in the decision space according to problem properties [10]. $F : \Omega \rightarrow R^m$ consists of m objective functions. If Ω is a closed and connected region in R^n and all the objective functions are continuous, we call the problem a continuous MOP.

In order to consider the trade-off between objectives, the concept of *domination* between solutions is defined. Let $u = (u_1, \ldots, u_m)$, $v = (v_1, \ldots, v_m) \in R^m$ be two vectors. u is said to *dominate* v if $u_i \leq v_i$ for all $i = 1, \ldots, m$, and $u \neq v$. A point $x^* \in \Omega$ is *Pareto optimal* if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. The set of all the Pareto optimal points is called *Pareto set* (PS) and the set of all the objective vectors corresponding to the PS is called *Pareto front* (PF), where $PF = \{F(x) \in R^m | x \in PS\}$ [11].

Instead of searching for a single or just a few (Pareto) optimal solutions as in solving single-objective problems, the goal of handling multi-objective problems is to find the Pareto front as well as the Pareto set of the problem. Given the limited computational resource, including time and storage, how to provide good solutions in terms of both quality and spread is the key and challenging task for multi-objective optimization.

B. MOEA based on decomposition

One of the key ideas of MOEA/D is the use of a decomposition method to transform a MOP into a number of single-objective optimization problems. MOEA/D attempts to optimize these single-objective problems collectively and simultaneously instead of trying to directly approximate the Pareto front as many other evolutionary algorithms do because each optimal solution to these SOPs is a Pareto optimal solution to the given MOP. The collection of these optimal solutions is an approximation of the Pareto front. Weighted sum, Tchebycheff approach, boundary intersection, and other decomposition approaches can serve this purpose. In the present work, the Tchebycheff approach [11] is adopted. A single-objective optimization problem obtained by decomposing the given MOP can be represented as

$$\begin{array}{ll} \text{minimize} & g(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i | f_i(x) - z_i^*|\} \\ \text{subject to} & x \in \Omega \end{array} \quad (5)$$

where $\lambda = (\lambda_1, \ldots, \lambda_m)$ is a vector of weights, i.e., $\lambda_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. $z^* = (z_1^*, \ldots, z_m^*)$ is the reference point, i.e., $z_i^* = \min\{f_i(x) | x \in \Omega\}$ for each $i = 1, \ldots, m$.

Let $\lambda^1, \ldots, \lambda^N$ be a set of N weight vectors. If we use a large N and select the weight vectors properly, all the optimal solutions of the SOPs transformed from decomposition will well approximate the Pareto front. Moreover, we can define a neighborhood relationship for each SOP by computing Euclidean distances between weight vectors. SOPs which are considered neighbors are assumed to have similar fitness landscapes and their optimal solutions should be close in the decision space. MOEA/D exploits the information sharing among SOPs which are neighbors to accomplish the optimization task effectively and efficiently. The specification of MOEA/D is stated as follows:

- Inputs:
  - decision variables.
  - objective functions.
  - N: the number of subproblems.
  - T: the number of neighbors for each subproblem.
  - stopping criteria.
- Outputs:
  - Approximation to the *PS*: $x^1, \ldots, x^N$.
  - Approximation to the *PF*: $F(x^1), \ldots, F(x^N)$.

TABLE I
PARAMETER SETTINGS OF MOEA/D

Parameter	Value
N	50
T	10
Crossover rate	1
Mutation rate	$1/m$
Max Gen.	150

IV. EXPERIMENTS

The experiment implementation is described in this section. MOEA/D is a well-developed tool and has the characteristic of black-box optimization like other evolutionary algorithms. As described in section III-B, only input and output should be handled properly. Section IV-A shows how to encode a degree distribution into decision variables, and the objective functions are given in section IV-B. Table I lists the other algorithmic parameter settings of MOEA/D.

A. Decision variables

The first step to use an evolutionary algorithm is to encode the decision variables of the optimization problem. It is not difficult in this study because a degree distribution can directly form a real-valued vector. In the evaluation phase, a real-valued vector of arbitrary values can be interpreted as a probability distribution, i.e., a degree distribution, with normalization. Such an operation does not change the feasibility, although the problem complexity may be slightly increased. The definition of degree distributions tells us that $d \leq k$. For a specific source symbol size k, obviously the problem dimensions is at most k. However, according to the LT encoding/decoding operations, we usually do not need a non-zero probability on every single degree. Observing the soliton distribution and considering the belief propagation algorithm, there is no necessary degree except 1, which ensures the start of belief propagation. As a result, we optimize a selected subset of degrees in the present work. We choose some particular degrees, {*1,2,3,4,5,7,9,13,17,23*} to form the decision variables according to the experience. Different subsets of degrees may change the numerical results of experiments results, but the soundness of this paper will be not be affected.

B. Objectives

In this paper, degree distributions are optimized for two different objectives. The first indicator to evaluate efficiency of LT code is overhead ε. The redundancy is traded for the benefit of fountain code and those extra encoding symbols increase the cost when they are transmitted to the receiver. In most application, overhead is required to be as low as possible because the transmission is usually expensive. In

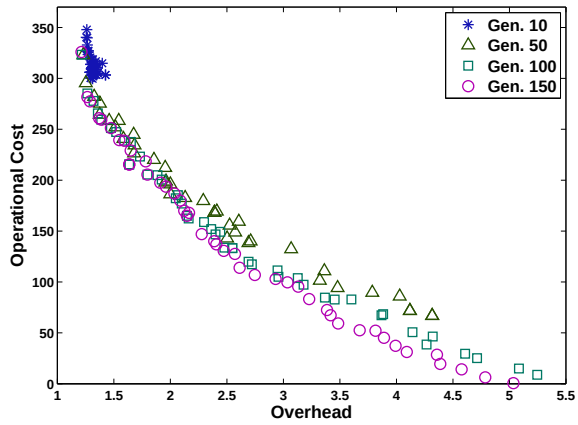


Fig. 2

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION FOR $k = 100$

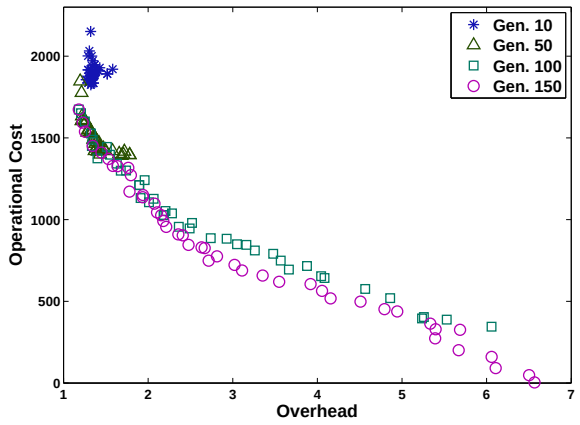


Fig. 4

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF $k = 500$

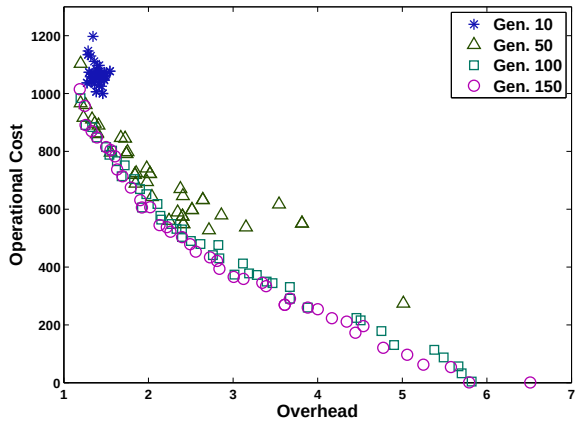


Fig. 3

EVOLUTIONARY PROCESS DURING THE OPTIMIZATION OF $k = 300$

our simulation of LT code, encoding symbols are generated until source data are fully recovered. The average required codewords are calculated as the fitness. The other objective is the computational cost of the encoding and decoding process. Such an objective value can be estimated with the mean degree of degree distributions. If M_d denotes the mean value of a degree distribution, the number of how many times *XOR* is executed can denote as $(M_d - 1) * \varepsilon$. There is a trade-off between ε and M_d because when M_d is greater, fewer encoding symbols may be required, and therefore, ε is less. On the other hand, M_d is the operational cost, which is the average number of *XOR* operations that have to be executed.

V. EXPERIMENTAL RESULTS

In most multi-objective problems, there is usually a trade-off between objectives. For an n-objective problem, a solution can be represented as a point in the n-dimensional space. All points which denote the non-dominated solutions form a partial optimal set called the Pareto front. The mission of multi-objective algorithms is to approximate the Pareto front as complete as possible. In other words, solutions should well spread to provide sufficient choices to decision markers. In our experiments, overhead and operational cost of LT code are both minimized together and the minimal value of each objective is expected. Clearly, a degree distribution with the minimal operational cost has only non-zero probability on degree one because in such a case, no encoding operation is needed. The case is the pure transmission without any channel coding, and it is a special case in the LT code framework. As for the other objective, overhead has a lower bound at ratio 1. Each encoding symbol can generate a new ripple to recover a source symbol ideally such that at least k encoding symbols are required to reconstruct the original data. Different from the operational cost, such a degree distribution is not yet discovered and even its existence is not proved. Fig. 2 shows the optimization process and the final result. After 150 generations, a significant PF is represented by fifty individual points. The solution with the minimal operational cost in expectation has been found, but the best overhead is 1.2068. Several individuals are listed in Table II, where the best value of overhead and operational cost are presented in columns 2 and 3, respectively. Columns 4 and 5 give the average overhead and execution counts of *XOR* in the numerical simulation. Figs. 3 and 4 display similar results as that shown in Fig. 2 for $k = 300$ and $k = 500$. Fig. 7 presents the distribution and simulation results for each individual listed in Table II.

To our limited knowledge, there is no guideline to design a robust soliton distribution for some particular coding behaviors. In order to fairly compare our optimized results with that of robust soliton distribution, MOEA/D is also applied to optimize the parameters of robust soliton distribution, which are c and δ. The PF of the optimized robust soliton distributions is presented in Fig. 5. In the dimension of operational cost, the optimized robust soliton distributions deliver very similar results because robust soliton distributions can also become the degree distribution with only non-zero probability on degree one if some appropriate parameters are given. However,

TABLE II

OPTIMIZED ARBITRARY DEGREE DISTRIBUTIONS

Individual	Best Overhead	Best Cost	AVG. Overhead	XOR
1	4.8442	0.00042	5.1958	0.038
25	2.5608	1.29873	2.6655	407.026
35	2.0294	1.85193	2.1485	558.667
45	1.4564	2.5135	1.57211	603.742
50	1.2068	2.93541	1.2718	843.669

TABLE III

OPTIMIZED ROBUST SOLITON DISTRIBUTIONS

Individual	Best Overhead	Best Cost	AVG. Overhead	XOR
1	4.8080	0.00552	5.1323	0.377
25	3.1244	1.74314	4.1662	125.455
35	2.0708	2.76115	2.6217	324.580
45	1.5194	3.41297	1.9278	471.753
50	1.2530	6.71008	1.3097	1141.46

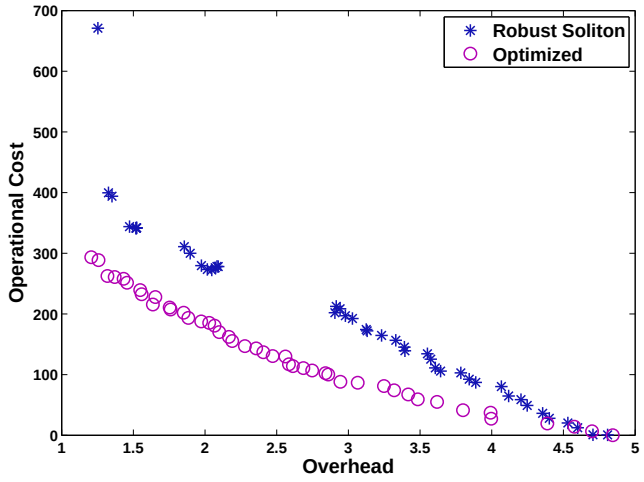


Fig. 5

COMPARISON BETWEEN THE OPTIMIZED ARBITRARY DEGREE DISTRIBUTION AND ROBUST SOLITON DISTRIBUTION

there are significant differences along the other axis. The performance is quite limited, and such a situation is caused by the fixed formula of robust soliton distribution. The figure demonstrates numerous better degree distributions that are very different from robust soliton distribution. These degree distributions can be discovered by optimization algorithms proposed in the realm of evolutionary computation.

VI. CONCLUSIONS

This paper proposed the use of multi-objective evolutionary algorithms to optimize the degree distribution in LT code. Overhead and operational cost were considered as two objectives and optimized simultaneously by using MOEA/D. The experimental results were promising and indicated that the Pareto front was well described. These results might also help researchers to better understand the behavior of LT code. For applications of different types and natures, LT code will be more efficient if choosing a specifically appropriate degree distribution is possible. Not only more choices of degree distributions are available, but also much better performance than that delivered by robust soliton distribution can be achieved, because most robust soliton distributions are dominated by the solutions discovered with MOEA/D in the experiments.

An alternation solution which designs degree distribution better than robust soliton is given in the work. While LT code is employed in real-world apparitions, the degree distribution can be customized to satisfy different requirements by using evolutionary algorithms. Fitter degree distributions will enhance the performance of those applications. Moreover, better understandings of the behavior of LT code will help the improvement of LT code. The final results show that some better distributions are beyond the model of robust soliton distribution. The theoretical analysis will also be applied to them just like the development of soliton distributions in our future work. An advanced model in which the performance is close to that of the Pareto front is in expectation.

ACKNOWLEDGMENTS

The authors would like to thank Martin Hornansky for fruitful discussion and conducting certain related numerical experiments. The work was supported in part by the National Science Council of Taiwan under Grant NSC-98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

REFERENCES

[1] D. J. C. MacKay, "Fountain codes," in *The IEE Seminar on Sparse-Graph Codes*, 2004, pp. 1–8.

[2] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998, pp. 56–67.

[3] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*, 2002, p. 271.

[4] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 39.

[5] E. A. Bodine and M. K. Cheng, "Characterization of luby transform codes with small message size for low-latency decoding," in *IEEE International Conference on Communications (ICC '08)*, 2008, pp. 1195–1199.

[6] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, 2007, pp. 2576–2580.

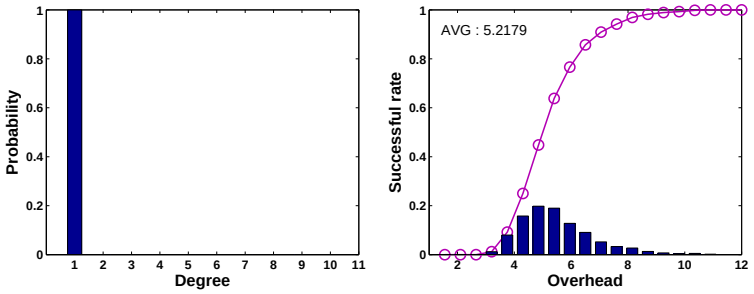
[7] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, 1982, pp. 133–136.

[8] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

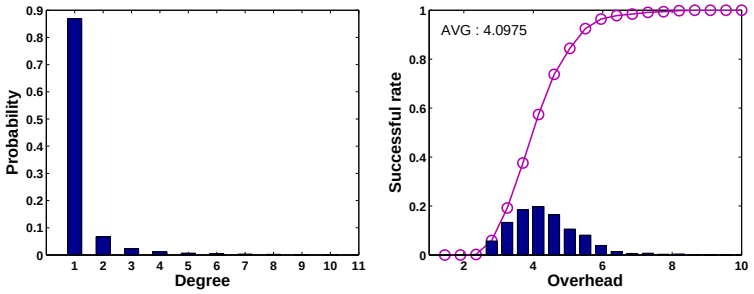
[9] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto set, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[10] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," in *First International Conference on Evolutionary Multi-Criterion Optimization*. Springer Verlag, 2001, pp. 284–298.

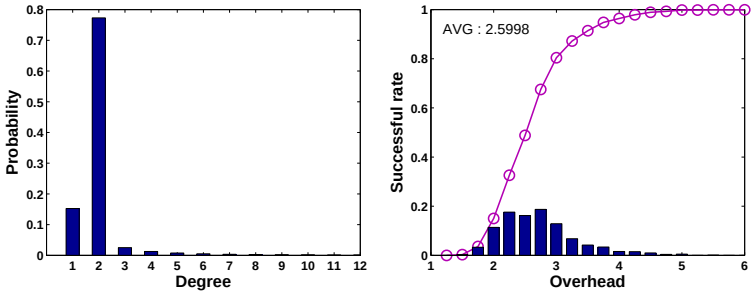
[11] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic, 1999.



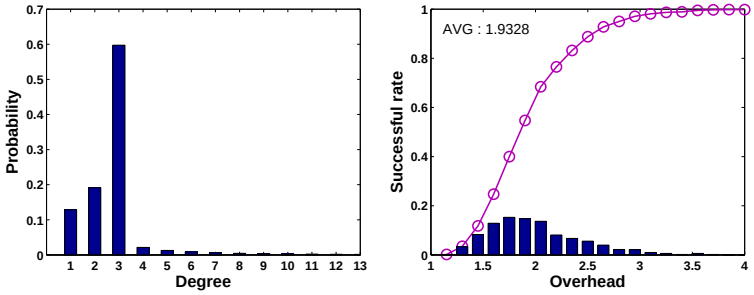
(a) Individual 1, $c = 9.72$ and $\delta = 0.00107$



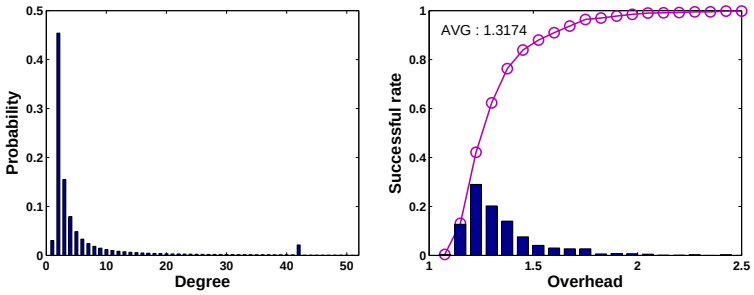
(b) Individual 25, $c = 1.634$ and $\delta = 0.185$



(c) Individual 35, $c = 2.146$ and $\delta = 0.978$



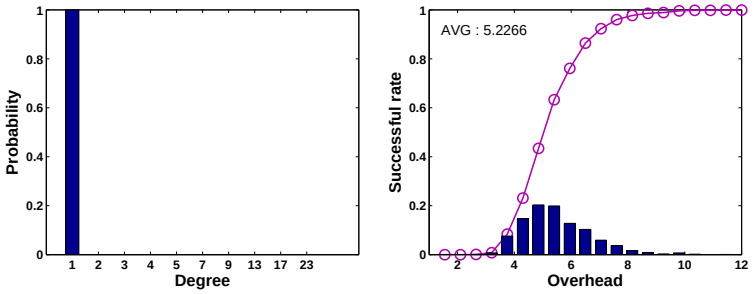
(d) Individual 45, $c = 0.96$ and $\delta = 0.601$



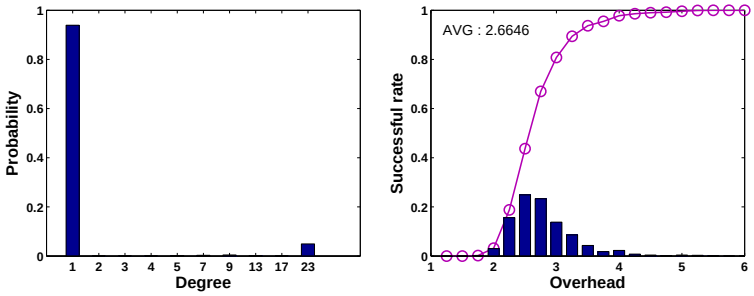
(e) Individual 50, $c = 0.0521$ and $\delta = 0.931$

Fig. 6

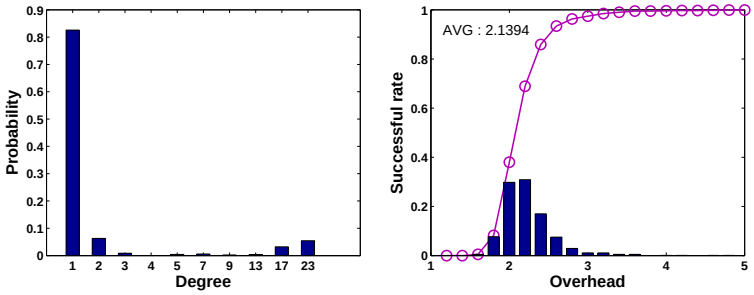
SIMULATION RESULTS OF OPTIMIZED ROBUST SOLITON DISTRIBUTIONS



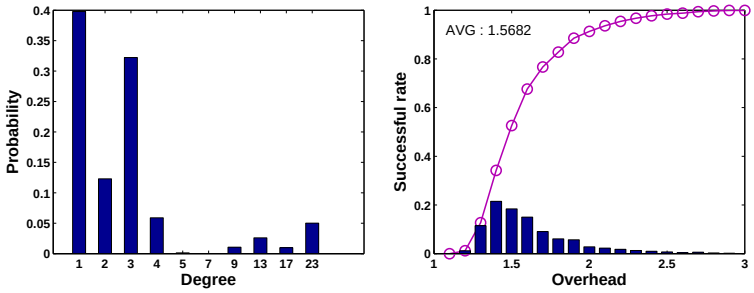
(a) Individual 1



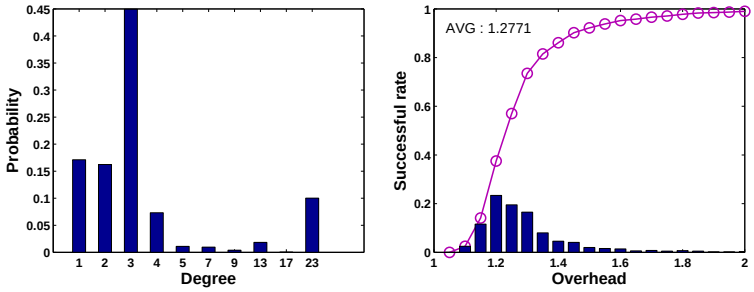
(b) Individual 25



(c) Individual 35



(d) Individual 45



(e) Individual 50

Fig. 7

SIMULATION RESULTS OF OPTIMIZED ARBITRARY DEGREE DISTRIBUTIONS

無衍生研發成果推廣資料

98 年度專題研究計畫研究成果彙整表

計畫主持人：陳穎平	計畫編號：98-2221-E-009-072-
計畫名稱：研究與發展專為無線網路系統客製化之最佳化演算架構	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	2	2	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	3	3	100%	人次	
		博士生	2	2	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	2	2	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	3	3	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	無.

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：■已發表 □未發表之文稿 □撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

本計畫為導向性之基礎研究，主要目標為兩個層次：一個實作層次在於開發混合決策變數型別之最佳化架構；另一個理論層次則在理解和探討各項演化計算方法之核心機制與工作原理。除混合決策變數型別最佳化架構已完成實作，可於將來進行測試與改進之外，對於演化計算最佳化方法之理論探討的研究成果頗為豐碩。本計畫相關之成果已發表為兩篇期刊論文與五篇會議論文，其中更包含有演化計算領域頂尖期刊之一「Evolutionary Computation」、頂尖會議之一「IEEE CEC」、以及理論電腦科學之重要期刊之一「Theoretical Computer Science」。雖本計畫暫無可直接實際應用於產業之研究成果，但本計畫成果之學術價值可由業已發表之期刊與會議論文明顯呈現，對於未來各種演化計算最佳化方法論將有一定的影響力。