一、前言

演化計算 (Evolutionary computation) 為人工智慧 (Artificial intelligence) 領域中，一個相當重要的研究方向，藉由學習和模仿自然界的演化機制，發展出許多可應用在最佳化、搜尋、設計等問題的演算法。尤其是演化計算具有黑箱最佳化的特性，演算法極具調整彈性，可以面對各式各樣的最佳化問題 [1, 2]。在本計畫中，吾人欲持續對演化計算方法進行更加深入的研究，並將此一具高度可客製化的最佳化工具，應用在通信傳輸技術中的最佳化問題，加以檢驗此最佳化工具之效能並獲得高品質解。以期能利用演化計算領域中最佳化方法極富彈性之特點，在面對多樣的最佳化目標時，快速地提供決策選項。於後續章節中，我們將詳述為期兩年之本計畫之研究動機、相關研究背景與研究目的。

二、研究目的

本實驗室在前一年度國科會計畫 (NSC 98-2221-E-009-072) 中以延伸式精簡基因演算法 (Extended compact genetic algorithm, ECGA) 為基礎，嘗試提出新的混合型態參數的最佳化架構。經過一整年的研究，本研究團隊在該目標已獲得了數項成果。本計畫首先延續相關開發經驗，進一步地研究偵測決策變數間關係之技術，以期能達成在萃取出變數間關聯性的資訊後，提升最佳化方法效能之目的。除此以外，吾人亦著重於演化計算方法之基本理論探討，以期強化演化計算方法論之理論基礎，深入演化計算之核心機制與數學層級之思考範圍，以試圖改變部分研究學者認為演化計算方法缺乏理論依據之看法。於更加了解最佳化演算法之行為特性後，再以通信傳輸技術中的 LT 編碼 (LT code) 做為最佳化標的，以所開發之最佳化方法處理 LT 編碼中採用的編碼密度機率模型 (Degree distribution) 的最佳化問題，以期除了驗證演算方法的有效性之外，並可提供 LT 編碼之使用者針對不同之應用情境以將編碼密度機率模型客製化、最佳化之契機。

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。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。

然而，針對演化計算中之基因演算法而言，在過去的文獻中，相關學者已曾指出，基因演算法在解編碼不適當 (亦即有相依關係的變數未能被安排在一起) 的情形下，最佳化效能將極其低落 [9]。因此，基因演算法之主要改進方向之一，就在於鏈結學習 (Linkage learning)，亦即偵測變數間之相依性，並利用該資訊輔以各種方式，包括動態調整解編碼、設計特殊演算子… 等，來增進基因演算法之最佳化效能 [10]。此外，自從 No-free-lunch 定理 [11] 提出後，「泛用型」最佳化演算法的存在即在理論層面遭到質疑；同時，現今對於最佳化演算方法的核心機制在理論層面的認識不足，亦是目前演化計算領域無法長足發展的主因之一。是故，在本計畫的第一年度中，吾人對於這幾個問題進行深入的探討與研究。

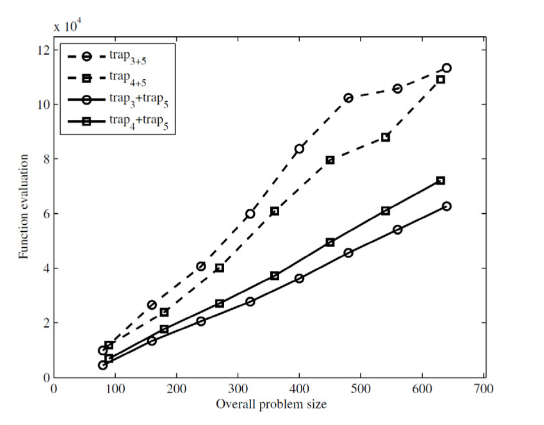
在應用層面，本計畫於第二年度內針對 LT 編碼中採用的編碼密度機率模型進行最佳化研究，並且也提出了 LT 編碼方法的改進方式，創造出能夠針對不同應用情境客製化的可能性。LT 編碼目前已被以基本元件的形式採用於許多重要的 rateless 編碼框架中，因此，增進 LT 編碼本身的效能是一件極為重要的事。為了 LT 編碼的效能，許多研究提出了 LT 編碼部分元件的改進方式。[12, 13] 針對 LT 編碼中的解碼演算法加以改進，使用不同的機制以還原來源資料。[14] 則將亂數產生器置換為混沌 (chaos) 數列產生演算法，以提供 LT 編碼做為亂數使用。

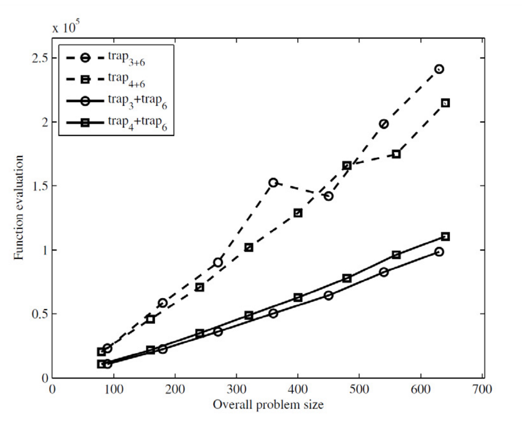
除了這些研究以外，更多的相關研究者則聚焦在設計編碼密度機率模型上，以期得到比已被證明在來源資料符號數趨向無限大時非常接近最佳解的 Robust soliton distribution 能提供 LT 編碼更佳效能的編碼密度機率模型。於是，這類研究 [15, 16] 專注於處理來源資料符號數較少的情境上，即便這些符號數小於 30 的情境事實上目前可以使用高斯消除法來處理 [17, 18]。為了要能對需使用較多、但距離無限大很遠的 LT 編碼情境最佳化，[19] 首先提出了使用經驗法則來進行編碼密度機率模型的最佳化動作，並測試了符號數為 100 的情況。然而，就實務層面及需求 (例如，即時多媒體資料傳輸、音訊與視訊串流等) 來看，符號數約在數面到數萬之間，才是亟需研究的區段。此乃由於這種數量級的符號數仍然距離無限大很遠，Robust soliton distribution 無法幫得上忙，但卻又多到非常難以尋得可提供較佳 LT 編碼效能的編碼密度機率模型。包含本實驗室及合作研究者在內，過去已有數項以演化計算方法來對編碼密度機率模型最佳化的研究 [20-22]，故在本計畫中，我們在第二年度中更進一步地將前一年度內所得之成果，應用於此重要的最佳化的問題上，並且亦試圖改進 LT 編碼原設計機制。

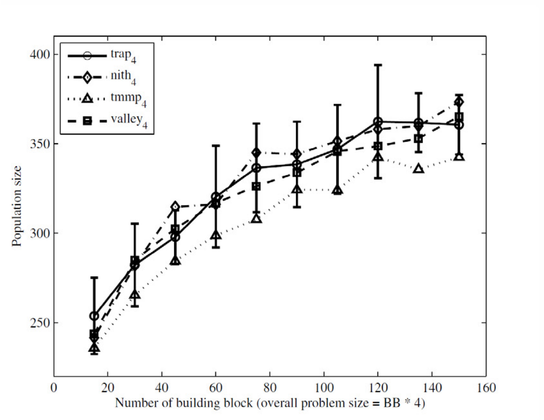
四、研究方法

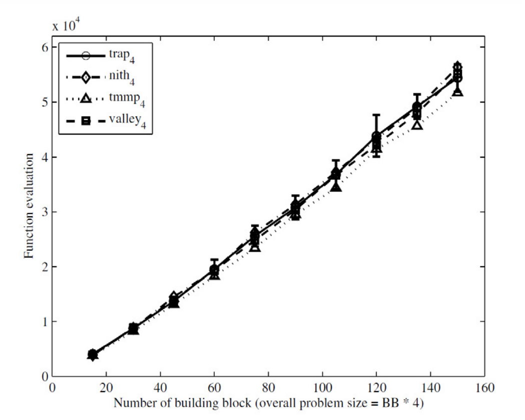
1. 針對歸納式鏈結學習法之性質探討

歸納式鏈結學習 (ILI, Inductive linkage learning) 法為本實驗室過去所提出 [23]，使用機器學習領域之 ID3 方法進行鏈結學習的技術。先前本實驗室已對此方法進行，建構基石的難度影響 [24, 25]、問題結構的影響 [26]… 等數項分析。奠基於這些成果，在本計畫的支持下，我們更進一步探討此方法在處理含有不同大小與型別的建構基石之問題時的表現，以及其所需之人口數目相對於問題大小的成長關係，如圖所示。









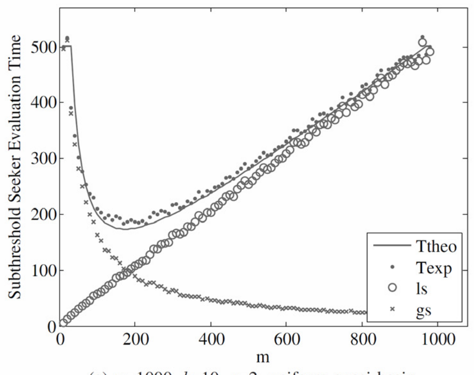
2. 解除 No-free-lunch 定理之桎梏

為了奠定泛用型方法存在之可能性及其範圍，吾人提出一個足夠廣泛的數學框架，並以理論觀點配合計算實務，來合理地解釋 No-free-lunch 定理雖然在其所定義的最佳化問題、最佳化方法等範疇內為真。但由於在實務中，絕大部分被包含在「所有問題」中的問題 (亦即目標函數) 其實完全毋須考慮，在此情形之下，No-free-lunch 所宣稱之「任何兩個最佳化方法在所有問題上的平均效能相等」在發展各項於實務中所使用的最佳化計算方法方面，並不會造成任何影響。

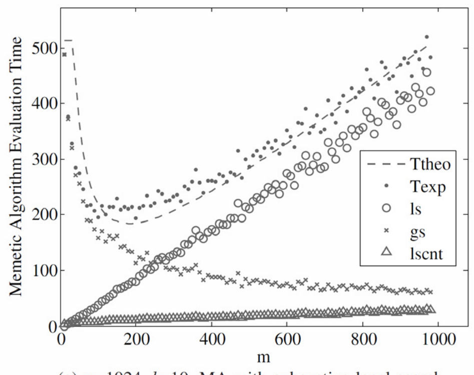
3. 建立彌因演算法核心運作機制之數學模型

彌因演算法 (Memetic algorithm) 與基因演算法的不同之處，在於彌因演算法強調以「後天學習所獲得之技能」的觀點，來思考區域搜尋 (Local search) 運算子在最佳化過程中所扮演的角色。而最佳化方法要達到優良效能，其必要條件即為所使用之廣域搜尋 (Global search) 運算子與區域搜尋運算子能合作與平衡。然而，長期以來縱有一些零星的理論探討研究，大多流於範圍狹窄或模型不實際。因此上述概念的本身雖被大部分相關學者所認同，但一直停留在觀念階段，無法落實為可進行運算的數學標的。本實驗室針對此點，建立起具體卻又不失一般性之彌因演算法數學模型，用以探討廣域搜尋機制與區域搜尋機制的相對性質與平衡之取得，並得以之做為設計出效能更佳之演算法的指導原則。

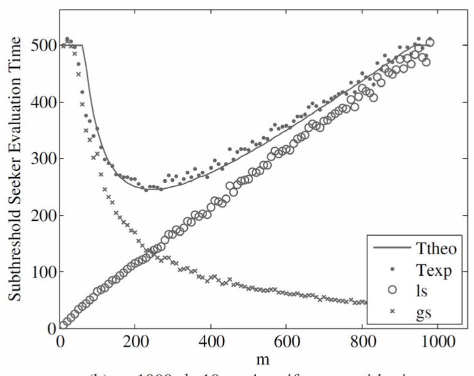
而由我們的研究結果可知，如下圖所示，廣域搜尋機制與區域搜尋機制的平衡乃奠基於搜尋資源是否平均分配於這兩種搜尋機制。這裡所謂的「平均分配」並非是以主動且直接的方式，讓廣域搜尋和區域搜尋各佔一半的搜尋資源，而是要藉由調整其採用的最佳化演算法本身所具有的參數來達成。若該演算化在某組參數設定下，能恰好在廣域搜尋行為和區域搜尋行為展現時使用各半的搜尋資源，則應是其最有效率的情境。



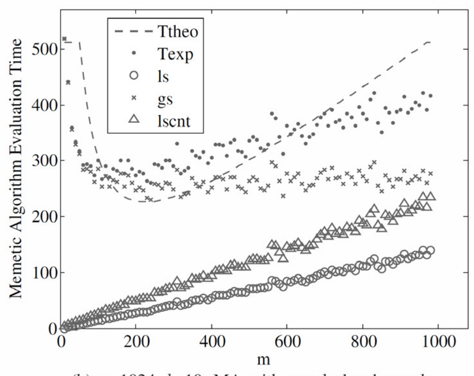
(a) n=1000, b=10, s=2, uniform quasi-basin



(a) n=1024, b=10, MA with exhaustive local search.



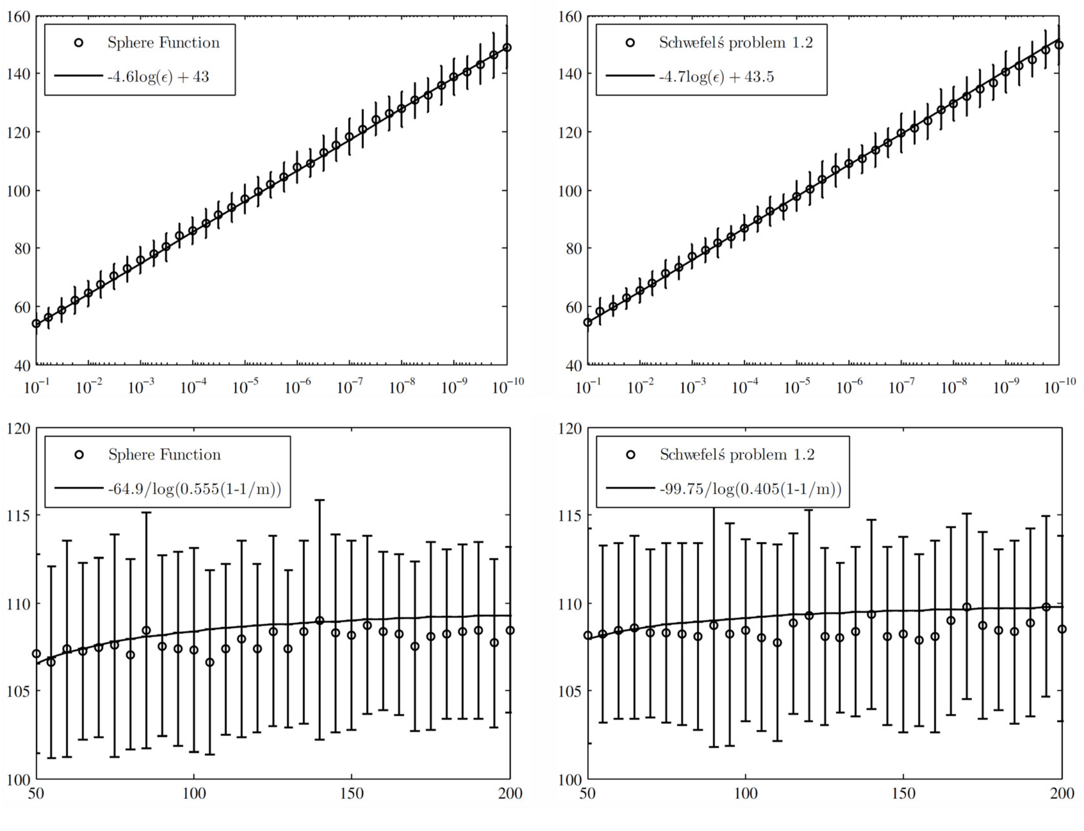
(b) n=1000, b=10, s=4, uniform quasi-basin



(b) n=1024, b=10, MA with greedy local search.

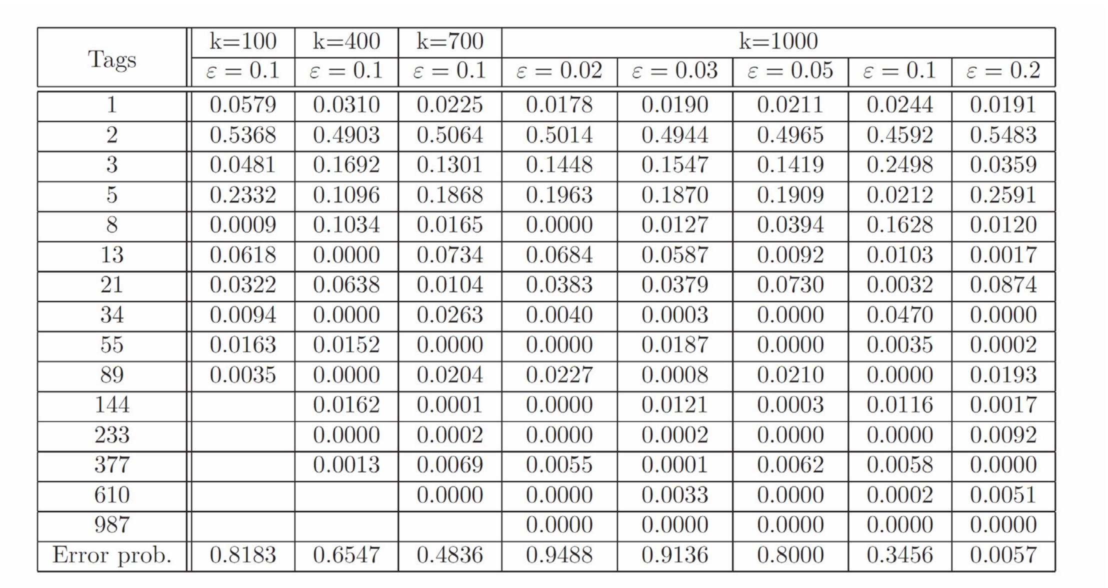
4.　分析粒子群演算法的收斂時間

粒子群演算法 (Particle swarm optimization, PSO) 由於其容易使用且效果佳的特性，目前被廣泛使用於許多的工程與科學的最佳化問題上。然後，就理論層面而言，環顧目前相關文獻，現存對於粒子群最佳化演算法之理論研究在原始構想上，都是先將粒子群的數量縮減為 1，然後認定這些 (事實上是「這個」) 粒子的移動方式為動態系統，再引用動態系統領域中所熟知的推導結果說明粒子群之收斂性。而少數奠基在相同的思維上，推廣至稍多粒子的研究中，皆假設所有的粒子之間完全獨立。雖然由於理論探討困難，高度簡化討論標的複雜度無可厚非，但討論「數量為 1」的群體，而以單一粒子最終靜止的行為說明粒子群收斂的必然性，同時還捨棄粒子群最佳化演算法中最關鍵的特性—粒子訊息交換行為。如此的理論研究，對於推進粒子群最佳化演算法的貢獻顯然極其有限。而本實驗室在這個主題中進行了重要的研究。在我們先前已發表的研究 [27] 中，首創以統計的方式來詮釋粒子群最佳化方法的運作。在我們所提出的理論框架中，不但直接考慮了由多個粒子所構成的群體，同時也一併考慮粒子間的訊息交換的關鍵機制，成為目前所有相關的理論研究中，最為貼近實際執行之粒子群最佳化演算法的理論模型。在奠基於此一理論架構上，我們進行了收斂時間的推導，並以實驗的方式獲得了如下圖所示的初步驗證。



5. 尋找對 LT 編碼最佳化時所需的替代評估函數

改進 LT 編碼最直覺的想法就是降低其所需之 overhead，因此評估函數通常是去計算 LT 編碼搭配某編碼密度機率模型的 overhead。遺憾的是目前沒有一個 closed form 可以去計算平均所需的 overhead，只能依靠大量模擬資料。若我們從另一角度來觀察 LT 編解碼行為，當收集到的 output symbols 越來越多時，LT 編碼有越高的機率可以完全解開 input symbols。這表示我們可以固定 LT 編碼接收到的 overhead，求出並最小化 LT 編碼失敗機率。在本計畫中，我們參考了兩種見於文獻的評估計算方式 [28, 29]，並使用於編碼密度機率模型的最佳化過程，進而獲得如下表所條列之高效能 LT 編碼編碼密度機率模型。



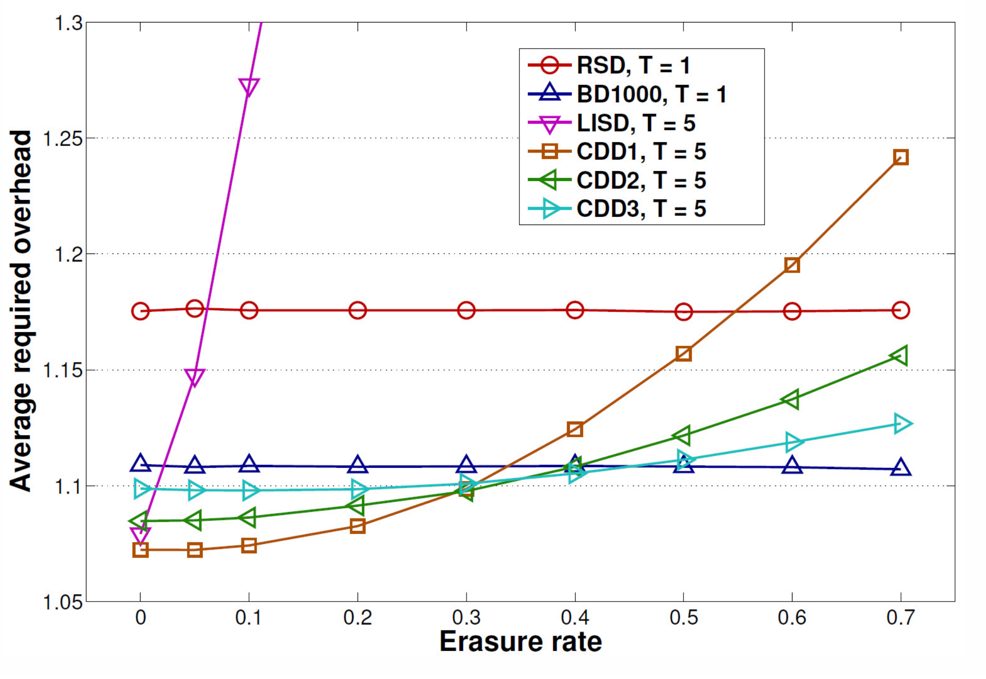
Tags	k=100	k=400	k=700	k=1000				
	$\varepsilon = 0.1$	$\varepsilon = 0.1$	$\varepsilon = 0.1$	$\varepsilon = 0.02$	$\varepsilon = 0.03$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.2$
1	0.0579	0.0310	0.0225	0.0178	0.0190	0.0211	0.0244	0.0191
2	0.5368	0.4903	0.5064	0.5014	0.4944	0.4965	0.4592	0.5483
3	0.0481	0.1692	0.1301	0.1448	0.1547	0.1419	0.2498	0.0359
5	0.2332	0.1096	0.1868	0.1963	0.1870	0.1909	0.0212	0.2591
8	0.0009	0.1034	0.0165	0.0000	0.0127	0.0394	0.1628	0.0120
13	0.0618	0.0000	0.0734	0.0684	0.0587	0.0092	0.0103	0.0017
21	0.0322	0.0638	0.0104	0.0383	0.0379	0.0730	0.0032	0.0874
34	0.0094	0.0000	0.0263	0.0040	0.0003	0.0000	0.0470	0.0000
55	0.0163	0.0152	0.0000	0.0000	0.0187	0.0000	0.0035	0.0002
89	0.0035	0.0000	0.0204	0.0227	0.0008	0.0210	0.0000	0.0193
144		0.0162	0.0001	0.0000	0.0121	0.0003	0.0116	0.0017
233		0.0000	0.0002	0.0000	0.0002	0.0000	0.0000	0.0092
377		0.0013	0.0069	0.0055	0.0001	0.0062	0.0058	0.0000
610			0.0000	0.0000	0.0033	0.0000	0.0002	0.0051
987				0.0000	0.0000	0.0000	0.0000	0.0000
Error prob.	0.8183	0.6547	0.4836	0.9488	0.9136	0.8000	0.3456	0.0057

6. 研究稀疏機率分佈的選擇

在最佳化架構中為了減輕搜尋負擔，我們採用稀疏機率分佈來取代完整機率分佈。此方法可以有效地降低搜尋空間的大小，但同時也限制了找到全域最佳解的可能。在先前的研究中，我們依靠實驗經驗，手動決定適合的 degree 來組成稀疏機率分佈中的非零項，雖然最佳化後的結果確實優於 Robust soliton distribution，但我們不曉得這些分佈是否已經逼近全域最佳解，抑或有其他的 degree 組合能夠找出更佳的稀疏機率分佈。根據文獻指出，不同 degree 的 output symbols 各自有其解碼作用。為了釐清不同 degree 對於 LT 解碼率的影響，我們考慮量化各個 degree 上的機率跟 LT 解碼率間的關係。分析這些數據有助於我們找到最適當的 degree 集合，使其所對應到的子搜尋空間能儘量逼近全域最佳解位置。

7. 改進 LT 編碼機制使其得以客製化

在噴泉碼 (Fountain codes) 的發展過程中，universal property 一直在理論層面上佔有非常重要的地位，因為它保證了無論 channel erasure rate 在何種情形下，解碼所須的 overhead 都維持不變。然而，時值今日，在實務情形中，太高的 channel erasure rate 根本不具考慮價值，因為即便料接受者確實能夠接獲資料，相較其他使用較佳 channel 的同儕使用者而言，使用經驗將大打折扣。此外，若是考慮即時多媒體傳輸的場合，只要使用稍高 erasure rate 的 channel，便可能造成整體問題，故 channel erasure rate 便成為一個設計參數，對不同的情境皆有經考慮而確定要支援的界限。但此一實務需求，以目前的 LT 編碼發展而言，完全沒有被研究者以學術問題的角度進行考慮。是故，本實驗室針對這項需求，為 LT 編碼機制引入演化計算領域中常用的 tournament selection 機制，以致 LT 編碼可依應用情境的不同，而進行客製化及最佳化調整，不必拘泥於維持 universal property。所得結果如下圖所示，噴泉碼之採用者可以情境之不同，客製出 CDD1, CDD2, CDD3 等因應不同應用情境狀況之需求。相對於具有 universal property 特性的編碼方法，如 LT 編碼，就僅能使用此圖中之水平線的編碼性質，如圖中之紅色水平線即為 Robust soliton distribution 配合 LT 編碼所能提供的效能指標。



五、結果與討論

本計畫於兩年期間，進行偵測決策變數間關係技術之深入研究、探討演化計算方法之理論根基，並且針對 LT 編碼問題進行研究，包括編碼密度機率模型的最佳化以及改進 LT 編碼機制等。在這些主題中，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；
  - 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。
- 深入探討歸納式鏈結學習法之性質: 包括其對於含有不同大小與型別的建構基石之問題的表現，以及建立數學模型以顯示並預測歸納式鏈結學習法所需之人口數目相對於問題大小的成長關係。
- 奠定泛用型方法存在之可能性及其範圍: 提出數學框架以理論觀點配合計算實務以合理地解釋 No-free-lunch 定理雖然在其定義的範圍內為真，但並不會對於發展實務計算各項方法造成任何影響。
- 建立演化計算方法之數學模型: 提出數學模型並用以探討最佳化計算法方中所含有之廣域搜尋機制與區域搜尋機制的相對性質，以及其應如何調配方能取得平衡，以利設計出效能更佳之演算法。
- 分析粒子群演算法的收斂時間: 奠基於本實驗室之前所提出之粒子群演算法收斂的數學模型，進行收斂時間的推導，並將實際之粒子群演算法運用於處理具體的數學函數以獲得可驗證所推導之收斂時間的數值結果。
- 將研究成應用於 LT 編碼中之編碼密度機率模型最佳化: 我們首先尋找對 LT 編碼最佳化時所需的替代評估函數，再配合研究稀疏機率分佈的選擇以降低最佳化演算法的負擔，從而成功對編碼密度機率模型進行最佳化。
- 改進 LT 編碼機制使其得以客製化: 將演化計算領域經常使用的 tournament selection 概念，應用於改進 LT 編碼機制中，致使 LT 編碼採用者有機會能針對其不同的應用情境與狀況，將 LT 編碼客製化與最佳化。
- 撰寫報告並投稿相關期刊與重要會議論文以公開發表本計畫各項研究成果。本實驗室目前基於國科會之研究計畫補助，投稿與發表了以下的學術論文：
  - 期刊論文：
    - Chen, C.-M., Chen, Y.-p., Shen, T.-C., & Zao, J. K. A Practical Optimization Framework for the Degree Distribution in LT Codes. *IET Communications*. (Submitted)

- Lin, J.-Y., & Chen, Y.-p. (2013). Population sizing for inductive linkage identification. *International Journal of Systems Science*.

  doi: 10.1080/00207721.2011.577246. (SCI, EI). (Accepted)

- Chen, Y.-p., Chuang, C.-Y., & Huang, Y.-W. (2012). Inductive linkage identification on building blocks of different sizes and types. *International Journal of Systems Science, 43*(12), 2202–2213.

  doi: 10.1080/00207721.2011.566639. (SCI, EI).

- Lee, M.-C., Leu, F.-Y., & Chen, Y.-p. (2012). PFRF: An adaptive data replication algorithm based on star-topology data grids. *Future Generation Computer Systems, 28*(7), 1045–1057.

  doi: 10.1016/j.future.2011.08.015. (SCI, EI).

- Chen, C.-H., & Chen, Y.-p. (2011). Convergence time analysis of particle swarm optimization based on particle interaction. *Advances in Artificial Intelligence, 2011*(204750), 1–7.

  doi: 10.1155/2011/204750.

- Lin, J.-Y., & Chen, Y.-p. (2011). Analysis on the collaboration between global search and local search in memetic computation. *IEEE Transactions on Evolutionary Computation, 15*(5), 608–623.

  doi: 10.1109/TEVC.2011.2150754. (SCI, EI).

- Jiang, P., & Chen, Y.-p. (2011). Free lunches on the discrete Lipschitz class. *Theoretical Computer Science*, 412(17), 1614–1628.

  doi: 10.1016/j.tcs.2010.12.028. (SCI, EI).

■ 會議論文：

- Chen, C.-M., & Chen, Y.-p. Connection Choice Codes. *The 32nd IEEE International Conference on Computer Communications (IEEE INFOCOM 2013)*. (Submitted)

- Tsai, P.-C., Chen, C.-M., & Chen, Y.-p. (2012). Sparse degrees analysis for LT codes optimization. In *Proceedings of 2012 IEEE Congress on Evolutionary Computation (CEC 2012)* (pp. 2463–2468).

  doi: 10.1109/CEC.2012.6252861. (EI).

- Lin, J.-Y., & Chen, Y.-p. (2012). When and what kind of memetic algorithms perform well. In *Proceedings of 2012 IEEE Congress on Evolutionary Computation (CEC 2012)* (pp. 2716–2723).

  doi: 10.1109/CEC.2012.6252894. (EI).

参考文獻

[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," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006 (GECCO-2006),* pp. 1863-1864, 2006.

[9] D. E. Goldberg*, et al.*, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems,* vol. 3, pp. 493-530, 1989.

[10] Y.-p. Chen, *Extending the scalability of linkage learning genetic algorithms: Theory and practice*, 2005.

[11] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation,* vol. 1, pp. 67-82, 1997.

[12] H. Tarus*, et al.*, "Exploiting Redundancies to Improve Performance of LT Decoding," *Proceedings of the 6th Annual Conference on Communication Networks and Services Research (CNSR 2008),* pp. 198-202, 2008.

[13] F. Lu*, et al.*, "LT codes decoding: Design and analysis," *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009),* pp. 2492-2496, 2009.

[14] Q. Zhou*, et al.*, "Encoding and Decoding of LT Codes Based on Chaos," *Proceedings of the 3rd International Conference on Innovative Computing Information and Control (ICICIC '08),* pp. 451-451, 2008.

[15] E. Hyyti¨a*, et al.*, "Optimal Degree Distribution for LT Codes with Small Message Length," *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007),* pp. 2576-2580, 2007.

[16] E. A. Bodine and M. K. Cheng, "Characterization of Luby Transform Codes with Small Message Size for Low-Latency Decoding," *Proceedings of the IEEE International Conference on Communications,* pp. 1195-1199, 2008.

[17] V. Bioglio*, et al.*, "On the fly Gaussian elimination for LT codes," *IEEE Communications Letters,* pp. 953-955, 2009.

[18] M. Rossi*, et al.*, "SYNAPSE++: Code Dissemination in Wireless Sensor Networks Using Fountain Codes," *IEEE Transactions on Mobile Computing,* vol.

9, pp. 1749-1765, 2010.

[19] E. Hyyti¨a*, et al.*, "Optimizing the degree distribution of LT codes with an importance sampling approach," *Proceedings of the 6th InternationalWorkshop on Rare Event Simulation (RESIM 2006),* pp. 64-73, 2006.

[20] C.-M. Chen*, et al.*, "On the optimization of degree distributions in LT codes with covariance matrix adaptation evolution strategy," *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010),* pp. 3531-3538, 2010.

[21] C.-M. Chen*, et al.*, "Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition," *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010),* pp. 3635-3642, 2010.

[22] A. Talari and N. Rahnavard, "Rateless Codes with Optimum Intermediate Performance," *Proceedings of the Global Telecommunications Conference (GLOBECOM 2009),* pp. 1-6, 2009.

[23] C.-Y. Chuang and Y.-p. Chen, "Linkage identification by perturbation and decision tree induction," *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007),* pp. 357-363, 2007.

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

[25] Y.-w. Huang and Y.-p. Chen, "Detecting general problem structures with inductive linkage identification," *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2010),* pp. 508-515, 2010.

[26] Y.-W. Huang and Y.-p. Chen, "On the detection of general problem structures by using inductive linkage identification," *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO-2009),* pp. 1853-1854, 2009.

[27] Y.-p. Chen and P. Jiang, "Analysis on the facet of particle interaction in particle swarm optimization," *Theoretical Computer Science,* vol. 411, pp. 2101-2115, 2010.

[28] R. Karp*, et al.*, "Finite length analysis of LT codes," *Proceedings of 2004 IEEE International Symposium on Information Theory (ISIT 2004),* p. 39, 2004.

[29] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," *Proceedings of 2006 IEEE International Symposium on Information Theory (ISIT 2006),* pp. 2677-2679, 2006.

附錄：已發表之論文全文

期刊論文：

1. Chen, Y.-p., Chuang, C.-Y., & Huang, Y.-W. (2012). Inductive linkage identification on building blocks of different sizes and types. *International Journal of Systems Science, 43*(12), 2202–2213. doi: 10.1080/00207721.2011.566639. (SCI, EI).

2. Lee, M.-C., Leu, F.-Y., & Chen, Y.-p. (2012). PFRF: An adaptive data replication algorithm based on star-topology data grids. *Future Generation Computer Systems, 28*(7), 1045–1057. doi: 10.1016/j.future.2011.08.015. (SCI, EI).

3. Chen, C.-H., & Chen, Y.-p. (2011). Convergence time analysis of particle swarm optimization based on particle interaction. *Advances in Artificial Intelligence, 2011*(204750), 1–7. doi: 10.1155/2011/204750.

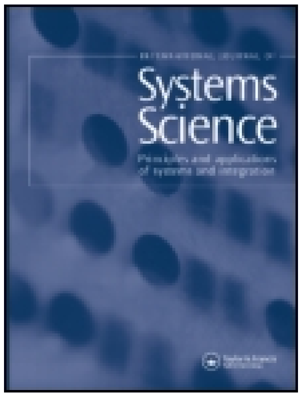
4. Lin, J.-Y., & Chen, Y.-p. (2011). Analysis on the collaboration between global search and local search in memetic computation. *IEEE Transactions on Evolutionary Computation, 15*(5), 608–623. doi: 10.1109/TEVC.2011.2150754. (SCI, EI).

5. Jiang, P., & Chen, Y.-p. (2011). Free lunches on the discrete Lipschitz class. *Theoretical Computer Science, 412*(17), 1614–1628. doi: 10.1016/j.tcs.2010.12.028. (SCI, EI).

會議論文：

1. Tsai, P.-C., Chen, C.-M., & Chen, Y.-p. (2012). Sparse degrees analysis for LT codes optimization. In *Proceedings of 2012 IEEE Congress on Evolutionary Computation (CEC 2012)* (pp. 2463–2468). doi: 10.1109/CEC.2012.6252861. (EI).

2. Lin, J.-Y., & Chen, Y.-p. (2012). When and what kind of memetic algorithms perform well. *In Proceedings of 2012 IEEE Congress on Evolutionary Computation (CEC 2012)* (pp. 2716–2723). doi: 10.1109/CEC.2012.6252894. (EI).



International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tsys20

Inductive linkage identification on building blocks of different sizes and types

Ying-ping Chen [a], Chung-Yao Chuang [a] & Yuan-Wei Huang [a]
[a] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Version of record first published: 11 Apr 2011.

To cite this article: Ying-ping Chen, Chung-Yao Chuang & Yuan-Wei Huang (2012): Inductive linkage identification on building blocks of different sizes and types, International Journal of Systems Science, 43:12, 2202-2213

To link to this article: http://dx.doi.org/10.1080/00207721.2011.566639

International Journal of Systems Science
Vol. 43, No. 12, December 2012, 2202–2213

Inductive linkage identification on building blocks of different sizes and types

Ying-ping Chen*, Chung-Yao Chuang and Yuan-Wei Huang

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

(*Received 24 March 2010; final version received 17 February 2011*)

The goal of linkage identification is to obtain the dependencies among decision variables. Such information or knowledge can be applied to design crossover operators and/or the encoding schemes in genetic and evolutionary methods. Thus, promising sub-solutions to the problem will be disrupted less likely, and successful convergence may be achieved more likely. To obtain linkage information, a linkage identification technique, called *Inductive Linkage Identification* (ILI), was proposed recently. ILI was established upon the mechanism of perturbation and the idea of decision tree learning. By constructing a decision tree according to decision variables and fitness difference values, the interdependent variables will be determined by the adopted decision tree learning algorithm. In this article, we aim to acquire a better understanding on the characteristics of ILI, especially its behaviour under problems composed of different-sized and different-type building blocks (BBs) which are not overlapped. Experiments showed that ILI can efficiently handle BBs of different sizes and is insensitive to BB types. Our experimental observations indicate the flexibility and the applicability of ILI on various elementary BB types that are commonly adopted in related experiments.

Keywords: inductive linkage identification; ILI; linkage learning; BBs; genetic algorithms; evolutionary computation

1. Introduction

Previous studies (Goldberg, Korb, and Deb 1989; Harik 1997) on genetic algorithms (GAs), which are widely utilised to handle control and engineering problems (Wang 2009; Li and Li 2010; Gladwin, Stewart, and Stewart 2011), have shown that the encoding scheme of solutions is one of the key factors to the success of GAs by demonstrating that simple GAs fail to handle problems of which the solutions are represented with loose encodings while genetic algorithms capable of learning linkage succeed. If strongly related variables, which are usually referred to as building blocks (BBs), are arranged loosely with the adopted representation, they are likely to be disrupted by crossover operations. Such a condition contributes to the divergence of population, instead of the convergence towards optimal solutions. Although encoding strongly related variables tightly or making crossover operators aware of such relationships could mitigate the problem and improve the GA performance (Stonedahl, Rand, and Wilensky 2008), both measures require the foreknowledge of the target problem, which is often not the case in which evolutionary algorithms are adopted.

In order to overcome the BB disruption problem, a variety of techniques have been proposed and developed in the past two decades and can be roughly classified into three categories (Munetomo and Goldberg 1998; Chen, Yu, Sastry, and Goldberg 2007):

(1) Evolving representations or operators;
(2) Probabilistic modelling for promising solutions;
(3) Perturbation methods.

The objective of the techniques in the first class is to make individual promising sub-solutions separated and less likely to be disrupted by crossover via manipulating the representation of solutions during optimisation. Various reordering and mapping operators have been proposed in the literature, such as self-crossover (Pal, Nandi, and Kundu 1998), which is proven able to generate any arbitrary permutation of the symbols, the messy GA (mGA) (Goldberg et al. 1989), and the fast mGA (fmGA) (Kargupta 1995), which is the more efficient descendant of mGA. The difficulty faced by these methods is that the reordering operator usually reacts too slow and loses the race against selection. Therefore, premature convergence at local optima occurs. Another technique, the linkage learning GA (LLGA) proposed by Harik (1997), uses circular structures as the representation with two-point crossover such that the tight linkage might be more likely preserved. LLGA works well while the shares of BBs are exponentially apportioned in the total fitness,

*Corresponding author. Email: ypchen@cs.nctu.edu.tw

ISSN 0020–7721 print/ISSN 1464–5319 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/00207721.2011.566639
http://www.tandfonline.com

which are usually referred to as exponentially scaled problems. However, it is inefficient when applied to uniformly scaled problems.

The methods in the second category are often referred to as the estimation of distribution algorithms (EDAs) (Mühlenbein and Paaß 1996; Larrañaga and Lozano 2001; Pelikan, Goldberg, and Lobo 2002). These approaches describe the dependencies among variables in a probabilistic manner by constructing a probabilistic model from selected solutions and then sample the built model to generate new solutions. Early EDAs began with assuming no interactions among variables, such as the population-based incremental learning (PBIL) (Baluja 1994) and the compact GA (cGA) (Harik, Lobo, and Goldberg 1999). Subsequent studies started to model pairwise interactions, e.g. the mutual-information input clustering (MIMIC) (de Bonet, Isbell, and Viola 1997), Baluja's dependency tree approach (Baluja and Davies 1997), and the bivariate marginal distribution algorithm (BMDA) (Pelikan and Mühlenbein 1999). Multivariate dependencies were then exploited, and more general interactions were modelled. Example methods include the extended compact GA (ECGA) (Harik 1999), the Bayesian optimisation algorithm (BOA) (Pelikan, Goldberg, and Cantú-Paz 1999), the factorised distribution algorithm (FDA) (Mühlenbein and Mahnig 1999) and the learning version of FDA (LFDA) (Mühlenbein and Höns 2005). Since model constructing in these methods requires no additional fitness evaluations, EDAs are usually considered efficient in the traditional viewpoints of evolutionary computation, especially when fitness evaluations involve time-consuming simulations. However, the model constructing mechanism itself is sometimes computationally expensive with a large population size, which usually occurs in evolutionary methods. The difficulty which EDAs often face is that the BBs contributing less to the total fitness are likely ignored rather than recognised.

Approaches in the third category observe the fitness differences caused by perturbing variables to detect dependencies. In the literature, the gene expression messy GA (GEMGA) (Kargupta 1996) models the sets of tightly linked variables as weights assigned to solutions and employs a perturbation method to detect them. GEMGA observes the fitness changes caused by perturbations on every variable for strings in the population and detects interactions among variables according to how likely the variables compose optimal solutions. Assuming that nonlinearity exists within a BB, the linkage identification by nonlinearity check (LINC) (Munetomo and Goldberg 1998) perturbs a pair of variables and observes the presence of nonlinearities to identify linkages. If the sum of fitness differences of perspective perturbations on two variables is equal to the fitness difference caused by simultaneously perturbing the two variables, linearity is confirmed, and thus, these two variables are considered independent. Instead of non-linearity, the descendant of LINC, linkage identification by non-monotonicity detection (LIMD) (Munetomo and Goldberg 1999), adopts non-monotonicity to detect interactions among variables. Compared to EDAs, the low salience BBs are unlikely ignored in these approaches. However, since obtaining fitness differences requires extra function evaluations, perturbation methods are usually considered demanding more computational efforts to detect linkages. In addition to empirical studies Heckendorn and Wright (2004) generalised these methods through Walsh analysis to obtain theoretical resource requirements. Zhou, Sun, and Heckendorn (2007) and Zhou, Heckendorn, and Sun (2008) later extended this study from the binary domain to high-cardinality domains.

An interesting approach combining the ideas of EDAs and perturbation methods, called *the dependency detection for distribution derived from fitness differences* (D^5), was developed by Tsuji, Munetomo, and Akama (2006). D^5 detects the dependencies of variables by estimating the distributions of strings clustered according to fitness differences. For each variable, D^5 calculates fitness differences by perturbations on that variable in the entire population and clusters the strings into sub-populations according to the obtained fitness differences. The sub-populations are examined to find k variables with the lowest entropies, where k is an algorithmic parameter for problem complexity, i.e. the number of variables in a linkage set. The determined k variables are considered forming a linkage set. D^5 can detect dependencies for a class of functions that are difficult for EDAs, e.g. functions containing low salience BBs, and requires less computational cost than other perturbation methods do. However, its major constraint is that it relies on parameter k, which may not be available due to the limited information of the problem structure. As a side-effect to parameter k, D^5 might be fragile in the situation where the problem is composed of subproblems of different sizes. Moreover, Ting, Zeng, and Lin (2010) recently utilised another data mining technique, Apriori Algorithm, to learn potential association rules between decision variables for linkage discovery. They reported that their proposal can improve D^5 in terms of solution quality and efficiency.

In our previous work, we proposed *inductive linkage identification* (ILI) based on perturbations and the integration with the Interative Dichotomiser (ID3) (Quinlan 1986) algorithm, which is widely used in machine learning. ILI is an unsupervised method without any parameter for the complexity of BBs.

Its scalability and efficiency against the increasing problem sizes have been demonstrated (Chuang and Chen 2007, 2008; Huang and Chen 2009, 2010). Compared to the conventional perturbation methods, such as LINC and LIMD, ILI utilises a data mining technique to analyse objective functions. Compared to D^5, which uses clustering, and the method proposed by Ting et al. (2010), which uses Apriori algorithm, ILI adopts the ID3 algorithm and behaves quite differently. In this article, we aim to address more detailed characteristics of ILI in order to gain deeper insights and better understandings of linkage learning. In particular, problems constructed by non-overlapped BBs of different sizes and sub-functions are studied and experimented upon. Our experimental results indicate that ILI holds the properties of robustness and efficiency when facing various configurations of BBs.

The remainder of this article is organised as follows. In Section 2, the background of linkage leaning in GA and decomposability of problems is briefly introduced. Section 3 gives an introduction to ILI, including a review of the ID3 decision tree learning algorithm, an example illustrating the proposed approach, and an algorithmic description of ILI. Section 4 presents the experiments conducted in this study and the results revealing the behaviour of ILI. Finally, Section 5 summaries and concludes this article.

2. Linkage and BBs

In this section, we briefly review the definitions and terminologies which will be used through out this article. As stated by de Jong, Watson, and Thierens (2005), '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', and such relationship between variables is often referred to as *linkage* in the GA literature. In order to obtain the full linkage information of a pair of variables, the fitness contribution or optimal setting of these two variables will be examined on all possible settings of the other variables.

Although obtaining the full linkage information is computationally expensive, linkage should be estimated using a reasonable amount of efforts if the target problem is decomposable. According to the Schema theorem (Holland 1992), short, low-order and highly fit substrings increase their share to be combined. Also stated in the BB hypothesis, GAs implicitly decompose a problem into sub-problems by processing BBs. It is considered that combining small parts is important for GAs and is consistent with human innovation (Goldberg 2002). These lead to a problem model called the *additively decomposable function* (ADF), which can be written as a sum of low-order sub-functions.

Let a string $\mathbf{s}$ of length ℓ be described as a series of variables, $\mathbf{s} = s_1 s_2 \cdots s_\ell$. We assume that $\mathbf{s} = s_1 s_2 \cdots s_\ell$ is a permutation of the decision variables $\mathbf{x} = x_1 x_2 \cdots x_\ell$ to represent the encoding scheme adopted by GA users. The fitness of string $\mathbf{s}$ is then defined as

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

where m is the number of sub-functions, f_i is the i-th sub-function and $\mathbf{s}_{v_i}$ is the substring to f_i. Each v_i is a vector specifying the substring $\mathbf{s}_{v_i}$. For example, if $v_i = (1, 2, 4, 8)$, $\mathbf{s}_{v_i} = s_1 s_2 s_4 s_8$. If f_i is also a sum of other sub-functions, it can be replaced by those sub-functions. Thus, each f_i can be considered as a nonlinear function.

By eliminating the ordering property of v_i, we can obtain a set V_i containing the elements of v_i. The variables belonging to the same set of V_i is regarded as interdependent because f_i is nonlinear. Thus, we refer to the set V_i as a linkage set. A related term, BBs, is referred to as the candidate solutions to sub-function f_i. In this article, only a subclass of the ADFs is considered. We concentrate on non-overlapping sub-functions. That is, $V_i \cap V_j = \emptyset$ if $i \neq j$. In addition, the strings are assumed to be composed of binary variables.

3. Inductive linkage learning

In this section, the ideas behind ILI will be presented. Then, the ID3 algorithm, which is proposed and widely utilised in the field of machine learning, will be briefly introduced. An example is given to illustratively explain the mechanism of ILI, followed by the pseudo code.

In ILI, linkage learning is regarded as the issue of decision tree learning. As an illustration, the fitness difference can be derived in the following equation within the ADF model:

$$\begin{aligned} f(s_1 s_2 \cdots s_8) &= f_1(s_1 s_2 s_3 s_4 s_5) + f_2(s_6 s_7 s_8) \\ \mathrm{d}f_1(\mathbf{s}) &= f(s_1 s_2 \cdots s_8) - f(\overline{s_1} s_2 \cdots s_8) \\ &= (f_1(s_1 s_2 s_3 s_4 s_5) + f_2(s_6 s_7 s_8)) \\ &\quad - (f_1(\overline{s_1} s_2 s_3 s_4 s_5) + f_2(s_6 s_7 s_8)) \\ &= f_1(s_1 s_2 s_3 s_4 s_5) - f_1(\overline{s_1} s_2 s_3 s_4 s_5). \end{aligned} \tag{2}$$

Equation (2) indicates that the fitness difference $\mathrm{d}f_1$ should be affected only by the bits belonging to the same sub-functions as the perturbed bits s_1, which are $s_1 s_2 \cdots s_5$. Since certain fitness difference values are respectively caused by particular bits arranged in some permutation of the sub-function where the perturbed variable belongs, we can consider the task as finding which values of variables will result in the corresponding fitness differences.

We found that this kind of tasks is similar to decision making in machine learning: given a condition composed of attributes, an agent (algorithm) should learn to make a decision with the given training instances. When the decision-making method is adopted for conducting linkage learning, decision variables are regarded as attributes and the fitness difference values stand for class labels. With this simple and direct mapping, linkage learning in GAs can potentially be handled with certain well-developed methods in machine learning.

3.1. *Decision tree learning: ID3*

The ID3 algorithm was proposed by Quinlan (1986) for the purpose of constructing a decision tree on a set of training instances. In its basic form, ID3 constructs a decision tree in a top–down manner without backtracking. When a decision tree is being constructed, each attribute is evaluated using a statistical property, called the *information gain*, to measure how well the attribute alone classifies the training instances. The best attribute, which leads to the highest information gain, is accordingly selected and used as the root node of the tree. A descendant node of the root is created for each possible value of the selected attribute, and the training instances are split into appropriate descendant branches. The entire process is repeated on the training instances associated with each descendant node.

The statistical property, information gain, of each attribute is simply the expected reduction in the impurity of instances after classifying the instances with the selected attribute. The impurity of an arbitrary collection of instances is called *entropy* in the information theory. Given a collection D, containing instances of c different target values, the entropy of D relative to this c-wise classification is defined as

$$Entropy(D) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i, \tag{3}$$

where p_i is the proportion of D belonging to class i. For simplicity, in all the calculations involving entropy, we define $0\log_2 0$ to be 0. In terms of entropy, the information gain, $Gain(D, A)$, of an attribute A relative to a collection of instances D, is defined as

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Val(A)} \frac{|D_v|}{|D|} Entropy(D_v), \tag{4}$$

Downloaded by [National Chiao Tung University] at 23:09 24 October 2012

where $Val(A)$ is the set of all possible values for attribute A and D_v is the subset of D of which attribute A has value v. In summary, ID3 can be described as the pseudo code given in Algorithm 1.

Algorithm 1: Pseudo code of ID3

```
procedure ID3($D$)
  Stop if no further classification is need
  for each attribute $A$ do
    Calculate $Gain(D, A)$
  end for
  Select the attribute with the highest information gain as a tree node
  for each possible value $v$ of the selected attribute do
    Create a branch for $D_v$, the subset of $D$ of which the selected attribute has value $v$
    Call ID3($D_v$) to construct this subtree
  end for
end procedure
```

In the proposal of ILI (Chuang and Chen 2007), the ID3 algorithm is adopted as a classification and relationship extraction mechanism. Linkage learning is then achieved by a sequence of decision tree constructions. In a classification problem, a training instance is composed of a list of attributes describing the instance and a target value which the decision tree is supposed to predict after training. For the purpose of linkage identification, the list of attributes is the solution string, and the target value is the fitness difference caused by perturbations.

3.2. *Exemplary illustration*

This section illustrates the idea that linkage learning is considered as decision learning with an example. We consider a trap function of size k defined as the following:

$$f_{trap_k}(s_1 s_2 \cdots s_k) = trap_k\left(u = \sum_{i=1}^{k} s_i\right) = \begin{cases} k, & \text{if } u = k; \\ k - 1 - u, & \text{otherwise,} \end{cases} \tag{5}$$

where u is the number of ones in the string $s_1 s_2 \cdots s_k$. Suppose that we are dealing with an eight-bit problem

$$f(s_1 s_2 \cdots s_8) = f_{trap_5}(s_1 s_2 s_3 s_4 s_5) + f_{trap_3}(s_6 s_7 s_8), \tag{6}$$

where $s_1 s_2 \cdots s_8$ is a solution string. In the black-box optimisation scenario, the structural decomposition of the objective function is unknown. Our goal here is to

Table 1. Population perturbed at s_1.

$s_1s_2\cdots s_8$	f	df_1	$s_1s_2\cdots s_8$	f	df_1
$\overline{0}$0001 011	3	1	$\overline{1}$0010 110	2	−1
$\overline{0}$0011 111	5	1	$\overline{1}$0011 000	1	−1
$\overline{0}$0100 001	3	1	$\overline{1}$0101 010	1	−1
$\overline{0}$0100 100	3	1	$\overline{1}$0101 100	1	−1
$\overline{0}$0101 111	5	1	$\overline{1}$0101 110	1	−1
$\overline{0}$0110 111	5	1	$\overline{1}$0110 101	1	−1
$\overline{0}$1000 111	6	1	$\overline{1}$0111 100	0	−1
$\overline{0}$1010 011	2	1	$\overline{1}$1001 011	1	−1
$\overline{0}$1101 010	1	1	$\overline{1}$1001 111	4	−1
$\overline{0}$1101 100	1	1	$\overline{1}$1010 011	1	−1
$\overline{0}$1101 101	1	1	$\overline{1}$1011 001	0	−1
$\overline{0}$1110 110	1	1	$\overline{1}$1011 010	0	−1
$\overline{0}$1111 001	0	−5	$\overline{1}$1100 000	1	−1
$\overline{0}$1111 110	0	−5	$\overline{1}$1100 010	1	−1
$\overline{0}$1111 111	3	−5	$\overline{1}$1100 011	1	−1
$\overline{1}$0000 010	3	−1	$\overline{1}$1110 101	0	−1
$\overline{1}$0001 010	2	−1	$\overline{1}$1111 010	5	5
$\overline{1}$0010 101	2	−1	$\overline{1}$1111 011	5	5

identify the two linkage sets $V_1=\{1,2,3,4,5\}$ and $V_2=\{6,7,8\}$, which correspond to the problem structural decomposition.

In the beginning, a population of strings is randomly generated as listed in Table 1. The first column lists the solution strings, and the second column lists the fitness values of the corresponding strings. After initializing the population, we perturb the first variable s_1 ($0\to 1$ or $1\to 0$) for all strings in the population in order to detect the variables interdependent on s_1. Note that the choice of first operating on s_1 in this example is not mandatory. Any un-grouped decision variable in the encoding may be chosen as the root node. The third column of Table 1 records the fitness differences, df_1, caused by perturbations at variable s_1.

Then, we construct an ID3 decision tree by using the perturbed population of strings as the training instances and the perturbed variable s_1 as the tree root. Variables in $s_1s_2\cdots s_8$ are regarded as attributes of the instances, and the fitness differences df_1 are the target values/class labels. Corresponding to Table 1, an ID3 decision tree shown in Figure 1 is constructed. By gathering all the decision variables on the non-leaf nodes, we can identify a group of s_1, s_2, s_3, s_4 and s_5. As a consequence, linkage set V_1 is correctly identified.

For the remainder of this example, since s_1, s_2, s_3, s_4 and s_5 are already identified as linkage set V_1, we proceed at s_6. The fitness differences after perturbing variable s_6 are shown in Table 2. Conducting the same procedure, an ID3 decision tree presented in Figure 2 is obtained. By gathering all the decision variables used in the decision tree, we obtain variables s_6, s_7 and s_8, which form linkage set V_2. Because all the decision

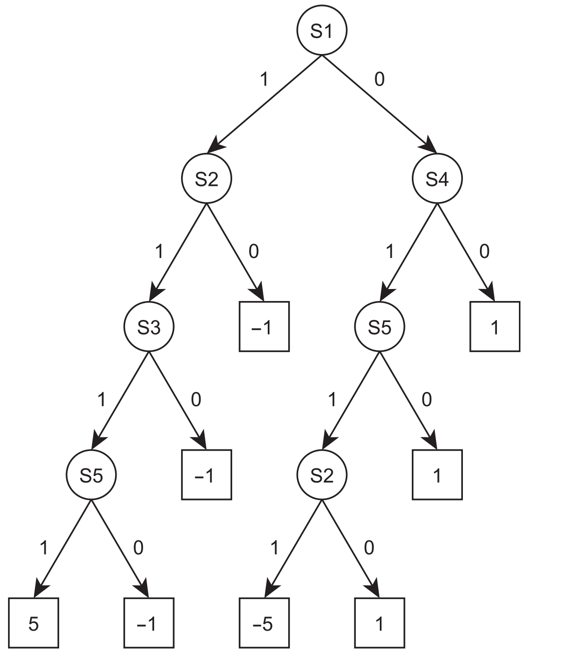


Figure 1. ID3 decision tree constructed according to Table 1.

Table 2. Population perturbed at s_6.

$s_1s_2\cdots s_8$	f	df_6	$s_1s_2\cdots s_8$	f	df_6
11100 $\overline{0}$00	1	0	10101 $\overline{1}$00	1	0
10011 $\overline{0}$00	1	0	01101 $\overline{1}$00	1	0
11011 $\overline{0}$01	0	0	00100 $\overline{1}$00	3	0
01111 $\overline{0}$01	0	0	10010 $\overline{1}$01	2	0
00100 $\overline{0}$01	3	0	10110 $\overline{1}$01	1	0
11111 $\overline{0}$10	5	0	11110 $\overline{1}$01	0	0
10101 $\overline{0}$10	1	0	01101 $\overline{1}$01	1	0
11100 $\overline{0}$10	1	0	01110 $\overline{1}$10	1	0
10001 $\overline{0}$10	2	0	01111 $\overline{1}$10	0	0
11011 $\overline{0}$10	0	0	01110 $\overline{1}$10	1	0
10000 $\overline{0}$10	3	0	10101 $\overline{1}$10	1	0
01101 $\overline{0}$10	1	0	01111 $\overline{1}$10	0	0
00001 $\overline{0}$11	3	−3	10010 $\overline{1}$10	2	0
00001 $\overline{0}$11	3	−3	00011 $\overline{1}$11	5	3
11010 $\overline{0}$11	1	−3	00011 $\overline{1}$11	5	3
11001 $\overline{0}$11	1	−3	01000 $\overline{1}$11	6	3
11111 $\overline{0}$11	5	−3	00101 $\overline{1}$11	5	3
11100 $\overline{0}$11	1	−3	11001 $\overline{1}$11	4	3
01010 $\overline{0}$11	2	−3	00110 $\overline{1}$11	5	3
10111 $\overline{1}$00	0	0	01111 $\overline{1}$11	3	3

variables are classified into their respective linkage sets, the linkage detecting task is accomplished. ILI finally reports two linkage sets, $V_1=\{s_1,s_2,s_3,s_4,s_5\}$ and $V_2=\{s_6,s_7,s_8\}$.

As illustrated in the example, the mechanism of ILI can detect size-varied BBs without assumptions. Such an ability implies that ILI should be capable of finding all relations among these variables as long as the

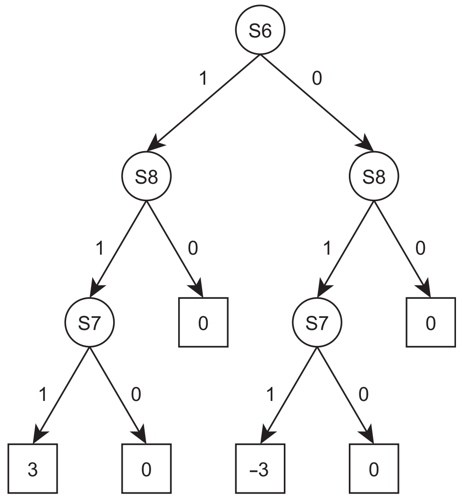


Figure 2. ID3 decision tree constructed according to Table 2.

population size is sufficiently large to provide significant statistics.

3.3. *Inductive linkage identification*

In this section, the idea demonstrated in the previous section is formalized as an algorithm, which is called ILI. The pseudo code of ILI is presented in Algorithm 2. Conceptually, ILI includes the following three main steps:

(1) Calculate fitness differences by perturbations;
(2) Construct an ID3 decision tree;
(3) Consider the tree nodes as a linkage set.

The three steps repeat until all the variables of the objective function are classified into their corresponding linkage sets.

ILI starts with initializing a population of strings. After initialization, ILI identifies one linkage set at a time using the following procedure: (1) a variable is randomly selected to be perturbed; (2) an ID3 decision tree is constructed according to the fitness differences caused by perturbations; (3) the variables used in the tree are gathered and considered as a linkage set.

Algorithm 2: Inductive linkage identification

```
procedure IDENTIFYLINKAGE($f$, $\ell$)
    Initialise a population $P$ with $n$ string of length $\ell$.
    Evaluate the fitness of strings in $P$ using $f$.
    $V \leftarrow \{1, \ldots, \ell\}$
    $m \leftarrow 0$
    while $V \neq \emptyset$ do
        $m \leftarrow m + 1$
        Select $v$ in $V$ at random.
        $V_m \leftarrow \{v\}$
        $V \leftarrow V - \{v\}$
        for each string $\mathbf{s^i} = s_1^i s_2^i \cdots s_\ell^i$ in $P$ do
            Perturb $s_v^i$.
            $\mathrm{d}f^i \leftarrow$ fitness difference caused by perturbation.
        end for
        Construct an ID3 decision tree using ($P$, d$f$).
        for each decision variable $s_j$ in tree do
            $V_m \leftarrow V_m \cup \{j\}$
            $V \leftarrow V - \{j\}$
        end for
    end while
    return linkage sets $V_1, V_2, \ldots, V_m$
end procedure
```

As clearly shown in Algorithm 2, there is no parameter needed for indicating the complexity of sub-functions. That is, ILI does not rely on any assumption on the size of BBs while other existing perturbation methods usually require the maximum size of BBs to be specified. This property distinguishes ILI from other existing methods. The only factor effecting the correctness of ILI is whether or not the solution strings in the population can provide sufficient information for the decision tree construction.

From our previous studies (Chuang and Chen 2007, 2008), we know that the required population size grows linearly with the problem size while the BBs size is constant. Such results indicate that ILI is more efficient than LINC, $\mathcal{O}(\ell^2) = \mathcal{O}(k^2m^2)$ (Munetomo and Goldberg 1998), and similar to D^5, $\mathcal{O}(\ell) = \mathcal{O}(km)$ (Tsuji et al. 2006), where ℓ is the problem size, k is the size (i.e. length or order) of BBs, and m is the number of BBs. Note that the comparison focuses on the amount of required computational resource instead of the identification quality. This is because given sufficient computational resource, all these methods can successfully identify every BB. In order to gain further understandings on the flexibility and applicability of ILI, in the next section, experiments on the BBs of different sub-functions as well as lengths are conducted and discussed.

4. Experiments and results

Experiments and results of ILI on binary and non-overlapped ADFs will be presented in this section. These experiments are designed to gain a better understanding of the behaviour of ILI on problems of different sub-functions compositions, including size-varied, size-mixed BBs and different sub-functions.

The required population size reflects the behaviour of ILI. Therefore, our experiments are designed to

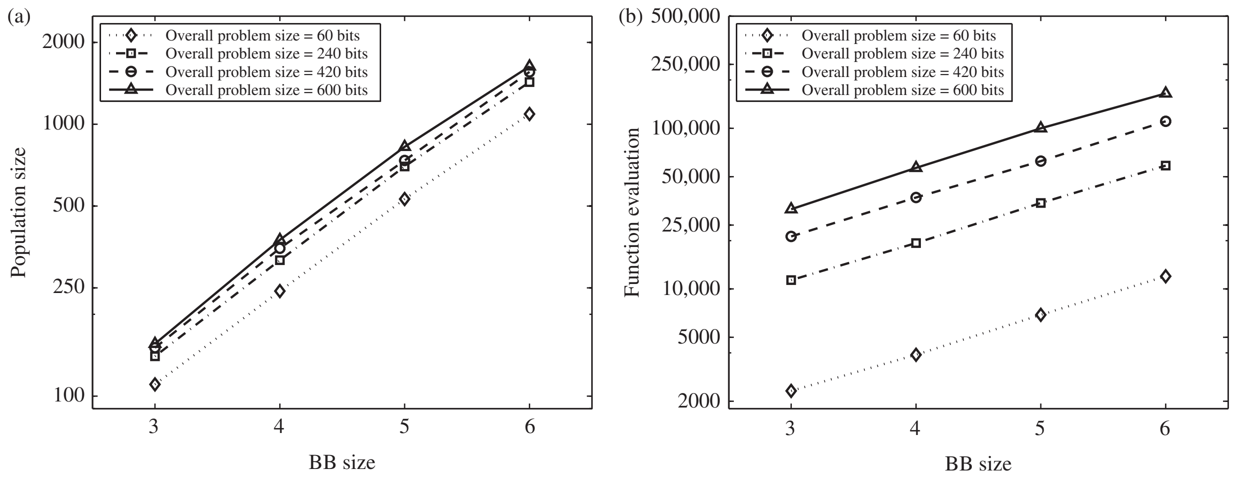


Figure 3. Requirements on different BB sizes; (a) Population size and (b) Function evaluation.

obtain the minimal population sizes required for different problem configurations. For a given problem, first a population size assuring successful trials of linkage identification, which means correctly identifying all the BBs within the problem for 30 consecutive and independent runs, is obtained by doubling the population size from 2500 until the first successful trial is archived. Once the upper bound of population sizes P_U is found, the required population size is determined in a bisection manner: the population size $P=(P_L+P_U)/2$ will be configured for ILI, where $P_L=1$ for the first iteration. If ILI can succeed with this population size P, then P will be regarded sufficiently large for the problem. The next iteration will perform on the range $[P_L, P]$. Otherwise, the range $[P, P_U]$ will be used. This procedure repeats until the range is smaller than a predefined distance, which is 2 in this study, and the last tested population size is considered as the minimal requirement for the current problem.

With the exponential regression of the experimental results, an estimation of $y=C\times 2^{a\times k}$ can be obtained, where a is a constant around 0.8 and C varies with different problem sizes. Earlier studies by Munetomo and Goldberg (1998) and Heckendorn and Wright (2004), respectively, suggested an empirical and a theoretical upper bounds of function evaluations, which are both in the form of $2^k\ell^j\log(\cdot)$ for problems of ℓ bits, composed of order-k BBs and each BB sharing j bits with others. Reviewing our empirical results with the upper bounds, ILI shows the same computational complexity of the exponential growth with k when overall sizes remain constant, such an observation is consistent with the upper bounds reported in the literature. However, the regression gives 0.8 as the base of exponent and thus indicates a practically better efficiency compared to the suggested upper bound when the complexity of sub-problem increases.

4.1. *Different BB sizes*

This section describes the experiment on problems of identical overall sizes but with different-sized sub-functions. From our experimental results with different configurations of the BB size k and the number of BBs m, we group those results with the overall problem sizes and arrange them with the BB size k. Thus, the results of the same problem size with different k can be examined.

Figure 3(a) and (b) shows the experimental results where the overall problem sizes are 60 bits, 240 bits, 420 bits and 600 bits with a log-scaled y-axis. The straight lines indicate that for identical overall problem sizes, the requirements of both the population size and the function evaluation grow exponentially.

4.2. *Mixed BB sizes*

One of the key features of ILI is unsupervised. In this section, we inspect this feature by conducting experiments on the problems consisting of non-overlapping BBs of order-k_1 and order-k_2 trap functions as

$$trap_{k_1+k_2}(\cdot)=\sum_{i=1}^{m}\left(trap_{k_1}(\cdot)+trap_{k_2}(\cdot)\right), \qquad (7)$$

where m is the number of $trap_{k_1}$ and $trap_{k_2}$. By designing the experiments in this way, the empirical results can be easily compared with those from problems consisting of identical sub-problem complexities in the following manner: for each problem size obtained from the experiment of $trap_{k_1+k_2}(\cdot)$, two

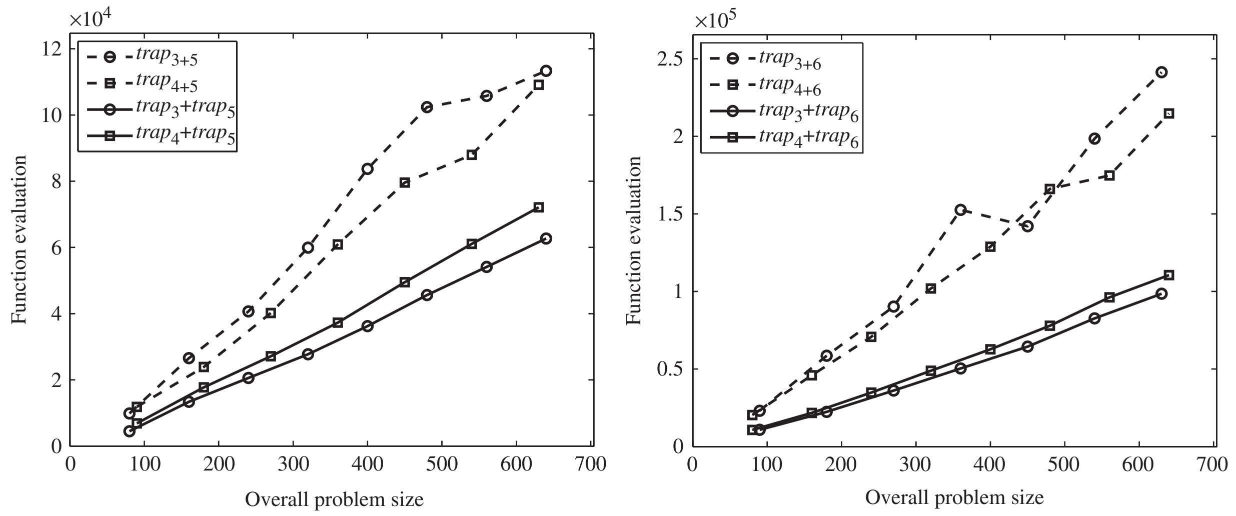


Figure 4. Problems with mixed BB sizes. The solid lines represent the actual experimental results while the dashed lines are the summed up calculations from Section 4.1.

results of the same amount of $trap_{k_1}$ and $trap_{k_2}$ from experiments in Section 4.1 are summed up to get the same problem size and total number of BBs, interpolation is utilised when there are no results of such configurations. These calculated numbers are denoted as $trap_{k_1} + trap_{k_2}$ in Figure 4 with the experimental results $trap_{k_1+k_2}$.

First, these results show that ILI is capable of detecting BBs of different sizes within one problem without any extra information regarding the complexity of sub-problems. Second, comparing with calculated data, it can be seen that although ILI requires more function evaluations for the problems composed of mixed BB sizes, the growth rate is still linear or very close to linear. The observation indicates that identifying size-varied BBs within a problem poses no particular difficulties for ILI. Such a property of robustness makes ILI more practical when being applied to real world problems where information regarding the sub-problem complexity is usually unavailable and no guideline exists to make appropriate assumptions.

4.3. *BBs of various elementary functions*

Despite of using $trap_k$ functions as the sub-function to construct BBs, the capability of ILI to handle BBs formed by other functions shown in Figure 5 is examined in this section. These elementary functions are used to compose the objective function according to the ADF model, and the complexity of order 4 is used in this section.

Figure 6 shows the experimental results. The required population sizes and function evaluations of $trap_4$, $nith_4$, $tmmp_4$ and $valley_4$ are plotted together, and the standard deviation of the results for $trap_4$ is also shown in the figures. Because the population and function evaluation requirements of these problems are similar, the behaviour of ILI should also be similar for problems constructed by mixing sub-problems of the same complexity. Moreover, the applicability of ILI on a wide range of problems is also confirmed. ILI is capable of detecting the interactions among variables as long as a sufficiently large population is employed to provide significant statistics.

5. Summary and conclusions

In this article, we examined ILI on several different configurations of BBs in order to gain better understandings. We focused on the mixed sizes of BBs and the elementary functions of different types. These series of experiments verified the efficiency of ILI on the population requirement growth, the robustness of ILI on mixed sizes of BBs, and the applicability of ILI on BBs formed with various elementary functions.

From the experiments of BB sizes, it is demonstrated that the required function evaluations grow exponentially with the size of BBs when the overall problem size remains constant. Such a result is consistent with the conclusions of previous studies from other researchers in the manner of Big-$\mathcal{O}$ while ILI demands less computational resource in practice. On the other hand, if computationally expensive real-world problems, such as parametric engineering design (Saridakis and Dentsoras 2009), are handled, and the optimisation framework has to be made much more efficient, techniques of the *surrogate-assisted*

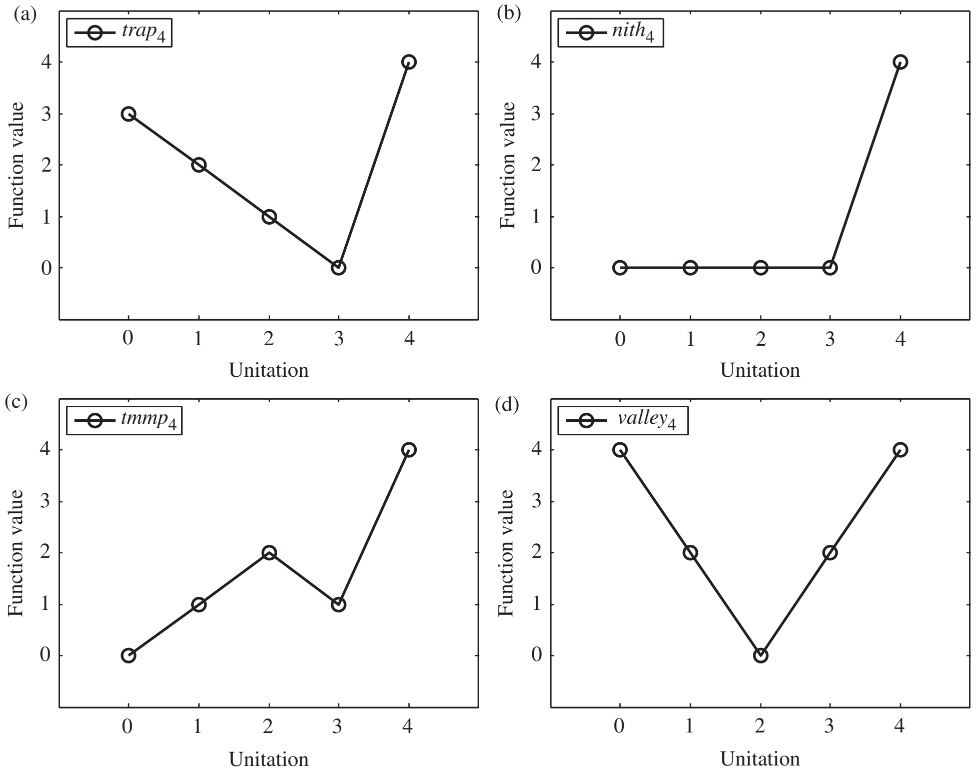


Figure 5. Elementary functions adopted in the series of experiments in Section 4.3; (a) *trap*$_4$, (b) *nith*$_4$, (c) *tmmp*$_4$ and (d) *valley*$_4$.

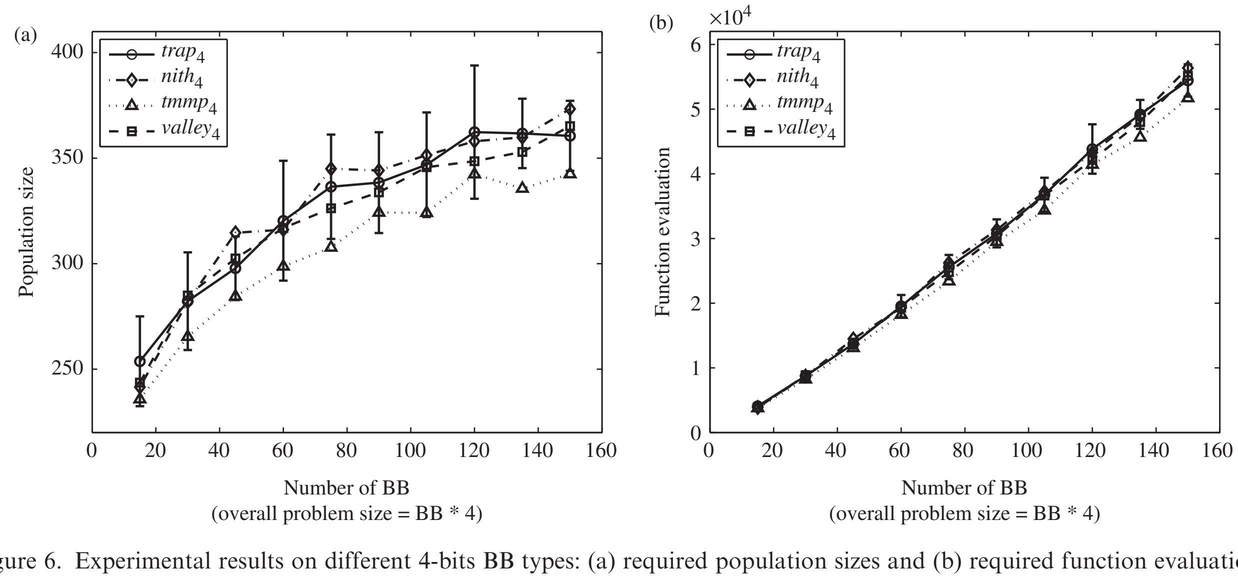


Figure 6. Experimental results on different 4-bits BB types: (a) required population sizes and (b) required function evaluations.

evolutionary algorithm (SAEA) (Sastry, Goldberg, and Pelikan 2001; Jin 2003; Lim, Jin, Ong, and Sendhoff 2010a; Lim, Ong, Setiawan, and Idris 2010b) may be adopted and utilised.

Another observation is that when ILI performs on problems composed of mixed-sized BBs, the computational complexity of ILI is still in the same order. This phenomenon indicates that detecting these more complicated problem structures poses no particular difficulty for ILI. Finally, the experimental results obtained by using four different elementary functions to construct BBs are quite similar. Thus, this series of experiments evidentially proves that ILI behaves similarly when handling sub-problem of different types.

As a consequence, we can now know that the most important factor that affects ILI's ability to identify linkage is the size of BBs. Compared with the BB size, ILI is relatively insensitive to other factors commonly studied by the related work, including the overall problem size, the number of BBs, and the type of BBs. Hence, ILI can be considered as a good linkage learning technique and can be adopted as a tool for analysing structures of target problems or a pre-processing procedure in frameworks of GAs.

Since its introduction, ILI as a linkage learning technique has been empirically proven efficient, robust and widely applicable. Research along this line includes integrating ILI into a GA framework, handling real-world applications with ILI, exploring ILI's capability of analysing problem structures and understanding the nature of linkage learning via getting deeper insights of ILI. As for the immediate future studies, the idea of 'linkage identification as decision learning' can be adapted to work with other advanced decision tree techniques. Characteristics of different decision tree algorithms might exhibit behaviour of different kinds and give us a better understanding of linkage identification. Such knowledge can be utilised to practically help the algorithmic development of GAs and theoretically reveal the working principle of evolutionary computation.

Acknowledgements

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

Notes on contributors

Ying-ping Chen received his BS and MS degrees in Computer Science and Information Engineering from National Taiwan University, Taiwan, in 1995 and 1997, respectively, and PhD in 2004 from the Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA. He has been an Assistant Professor from 2004 to 2009 and an Associate Professor since 2009 in the Department of Computer Science, National Chiao Tung University, Taiwan. His research interests in the field of genetic and evolutionary computation include theories, working principles, particle swarm optimisation, estimation of distribution algorithms, linkage learning techniques and dimensional/facet-wise models.

Chung-Yao Chuang received his BS and MS degrees in Computer Science from National Chiao Tung University, Taiwan, in 2006 and 2008, respectively. He is currently working at Academia Sinica, Taiwan for obligated citizen service and looking forward to being included in a PhD program starting at Fall 2012. His research interests include linkage problem in evolutionary algorithms, estimation of distribution algorithms, evolutionary computation and machine learning in general.

Yuan-Wei Huang received the BS and MS degrees in Computer Science from National Chiao Tung University, Taiwan, in 2008 and 2010, respectively. His research interests include linkage learning techniques, machine learning, and artificial intelligence.

References

Baluja, S. (1994), 'Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning', Technical Report, Pittsburgh, PA, USA.

Baluja, S., and Davies, S. (1997), 'Using Optimal Dependency-trees for Combinational Optimization', in *Proceedings of the Fourteenth International Conference on Machine Learning, ICML'97*, pp. 30–38.

Chen, Y.-p., Yu, T.-L., Sastry, K. and Goldberg, D.E. (2007), 'A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms', IlliGAL Report No. 2007014, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Chuang, C.Y., and Chen, Y.P. (2007), 'Linkage Identification by Perturbation and Decision Tree Induction', in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC, 2007)*, pp. 357–363.

Chuang, C.Y., and Chen, Y.P. (2008), '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, 2007)*, pp. 670–675.

de Bonet, J., Isbell, C., and Viola, P. (1997), 'MIMIC: Finding Optima by Estimating Probability Densities', *Advances in Neural Information Processing Systems*, 9, 424–430.

de Jong, E.D., Watson, R., and Thierens, D. (2005), 'On the Complexity of Hierarchical Problem Solving', in *Proceedings of Genetic and Evolutionary Computation Conference 2005 (GECCO, 2008)*, pp. 1201–1208.

Gladwin, D., Stewart, P., and Stewart, J. (2011), 'Internal Combustion Engine Control for Series Hybrid Electric Vehicles by Parallel and Distributed Genetic Programming/Multiobjective Genetic Algorithms', *International Journal of Systems Science*, 42, 249–261.

Goldberg, D.E. (2002), *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Norwell, MA, USA, Kluwer Academic Publishers.

Goldberg, D.E., Korb, B., and Deb, K. (1989), 'Messy Genetic Algorithms: Motivation, Analysis, and First Results', *Complex Systems*, 3, 493–530.

Harik, G.R. (1997), 'Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms', Ph.D. Dissertation, University of Michigan, Ann Arbor, MI, USA.

Harik, G. (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, 287–297.

Heckendorn, R.B., and Wright, A.H. (2004), 'Efficient Linkage Discovery by Limited Probing', *Evolutionary Computation*, 12, 517–545.

Holland, J.H. (1992), *Adaptation in Natural and Artificial Systems*, Cambridge, MA, USA: MIT Press.

Huang, Y.W., and Chen, Y.-p. (2009), 'On the Detection of General Problem Structures by Using Inductive Linkage Identification', in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2009 (GECCO, 2009)*, pp. 1853–1854.

Huang, Y.W., and 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, pp. 508–515.

Jin, Y. (2003), 'A Comprehensive Survey of Fitness Approximation in Evolutionary Computation', *Soft Computing*, 9, 3–12.

Kargupta, H. (1995), *SEARCH,* 'Polynomial Complexity, and the Fast Messy Genetic Algorithm', Technical Report, University of Illinois.

Kargupta, H. (1996), 'The Gene Expression Messy Genetic Algorithm', in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 814–819.

Larrañaga, P., and Lozano, J.A. (2001), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Boston, MA: Kluwer Academic Publishers.

Li, S., and Li, Z.Z. (2010), 'Spare Parts Allocation by Improved Genetic Algorithm and Monte Carlo Simulation', *International Journal of Systems Science*, First published on: 01 March 2010 (iFirst). DOI: 10.1080/00207720802556252.

Lim, D., Jin, Y., Ong, Y.S.O., and Sendhoff, B. (2010a), 'Generalizing Surrogate-assisted Evolutionary Computation', *IEEE Transactions on Evolutionary Computation*, 14, 329–355.

Lim, D., Ong, Y.S., Setiawan, R., and Idris, M. (2010b), 'Classifier-assisted Constrained Evolutionary Optimization for Automated Geometry Selection of Orthodontic Retraction Spring', in *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC, 2010)*, pp. 1449–1456.

Mühlenbein, H., and Höns, R. (2005), 'The Estimation of Distributions and the Minimum Relative Entropy Principle', *Evolutionary Computation*, 13, 1–27.

Mühlenbein, H., and Mahnig, T. (1999), 'FDA – A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions', *Evolutionary Computation*, 7, 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.

Munetomo, M., and Goldberg, D.E. (1998), 'Identifying Linkage by Nonlinearity Check', IlliGAL Report No. 98012, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Munetomo, M., and Goldberg, D.E. (1999), 'Identifying Linkage Groups by Nonlinearity/Non-monotonicity Detection', in *Proceedings of Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, pp. 433–440.

Pal, N.R., Nandi, S., and Kundu, M.K. (1998), 'Self-crossover – A New Genetic Operator and Its Application to Feature Selection', *International Journal of Systems Science*, 29, 207–212.

Pelikan, M., Goldberg, D.E., and Cantú-Paz, E. (1999), 'BOA: The Bayesian Optimization Algorithm', in *Proceedings of Genetic and Evolutionary Computation Conference 1999 (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, 5–20.

Pelikan, M., and Mühlenbein, H. (1999), 'The Bivariate Marginal Distribution Algorithm', *Advances in Soft Computing – Engineering Design and Manufacturing*, 521–535.

Quinlan, J.R. (1986), 'Induction of Decision Trees', *Machine Learning*, 1, 81–106.

Saridakis, K., and Dentsoras, A. (2009), 'Integration of Genetic Optimisation and Neuro-fuzzy Approximation in Parametric Engineering Design', *International Journal of Systems Science*, 40, 131–145.

Sastry, K., Goldberg, D.E., and Pelikan, M. (2001), 'Don't Evaluate, Inherit', in *Proceedings of Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, pp. 551–558.

Stonedahl, F., Rand, W., and Wilensky, U. (2008), 'CrossNet: A Framework for Crossover with Network-based Chromosomal Representations', in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2008 (GECCO, 2008)*, pp. 1057–1064.

Ting, C.K., Zeng, W.M., and Lin, T.C. (2010), 'Linkage Discovery Through Data Mining', *IEEE Computational Intelligence Magazine*, 5, 10–13.

Tsuji, M., Munetomo, M., and Akama, K. (2006), 'Linkage Identification by Fitness Difference Clustering', *Evolutionary Computation*, 14, 383–409.

Wang, K. (2009), 'Application of Genetic Algorithms to Robot Kinematics Calibration', *International Journal of Systems Science*, 40, 147–153.

Zhou, S., Heckendorn, R.B., and Sun, Z. (2008), 'Detecting the Epistatic Structure of Generalized Embedded Landscape', *Genetic Programming and Evolvable Machines*, 9, 125–155.

Zhou, S., Sun, Z., and Heckendorn, R.B. (2007), 'Extended Probe Method for Linkage Discovery Over High-cardinality Alphabets', in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007 (GECCO-2007)*, pp. 1484–1491.

Volume 28, issue 7, July 2012
ISSN 0167-739X
ELSEVIER
THE INTERNATIONAL JOURNAL OF
F FUTURE G GENERATION C COMPUTER S SYSTEMS
GRID COMPUTING AND eSCIENCE
Editor-in-Chief:
Peter Sloot
Associate Editors:
David Abramson
Ilkay Altintas
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Future Generation Computer Systems 28 (2012) 1045–1057

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

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

PFRF: An adaptive data replication algorithm based on star-topology data grids

Ming-Chang Lee [a], Fang-Yie Leu [b], Ying-ping Chen [a,*]

[a] *Department of Computer Science, National Chiao Tung University, Taiwan*

[b] *Department of Computer Science, Tunghai University, Taiwan*

ARTICLE INFO

Article history:
Received 5 November 2010
Received in revised form
5 July 2011
Accepted 5 August 2011
Available online 6 November 2011

Keywords:
Data grid
Data replication
Data access patterns
File popularity
PFRF

ABSTRACT

Recently, data replication algorithms have been widely employed in data grids to replicate frequently accessed data to appropriate sites. The purposes are shortening file transmission distance and delivering files from nearby sites to local sites so as to improve data access performance and reduce bandwidth consumption. Some of the algorithms were designed based on unlimited storage. However, they might not be practical in real-world data grids since currently no system has infinite storage. Others were implemented on limited storage environments, but none of them considers data access patterns which reflect the changes of users' interests, and these are important parameters affecting file retrieval efficiency and bandwidth consumption. In this paper, we propose an adaptive data replication algorithm, called the Popular File Replicate First algorithm (PFRF for short), which is developed on a star-topology data grid with limited storage space based on aggregated information on previous file accesses. The PFRF periodically calculates file access popularity to track the variation of users' access behaviors, and then replicates popular files to appropriate sites to adapt to the variation. We employ several types of file access behaviors, including Zipf-like, geometric, and uniform distributions, to evaluate PFRF. The simulation results show that PFRF can effectively improve average job turnaround time, bandwidth consumption for data delivery, and data availability as compared with those of the tested algorithms.

1. Introduction

Generally, a data grid, a specific grid system that provides users with a huge amount of storage space, often maintains a high volume of distributed data to serve users. Many recent large-scale scientific systems [1–4] and commercial applications [5], e.g., the Biomedical Informatics Research Network (BIRN) [6], the Large Hadron Collider (LHC) [7], the DataGrid Project (EDG) [8], and physics data grids [9,10], have collected a huge amount of data and performed complex experiments and analyses on the data [11–13].

According to the Pareto principle (also known as the 80/20 rule) [14], a part of data grid files is frequently accessed and transferred. If a file has no replicas distributed over the data grid, the efficiency of accessing the file is often poor since long distance data transfer always occupies a lot of bandwidth and causes long transmission delays [15]. Hence, how to decrease data access latency, lower bandwidth consumption for data transmission, and improve data availability have been the key issues in data grid research [16]. Data replication is a general and simple approach to achieve these goals. It has been widely used in many areas, such as the Internet, peer-to-peer systems, and distributed databases [17–21]. A well-defined data replication method should meet the requirements of being able to determine an appropriate time to replicate files, decide which files should be replicated, and store these replicas in appropriate locations [15,16,22–24].

On the other hand, the analyses of data access patterns have been the critical steps in designing efficient dynamic data replication schemes [25–27]. Several distributions have been used to model data access patterns, defined as the distribution of access counts on files of a system, and file popularity, defined as how often a file is accessed by users, i.e., how popular a file is [28,29]. Breslau et al. [28] claimed that using the Zipf-like distribution can more accurately model the distribution of webpage accesses. Cameron et al. [29] showed that the distribution of file accesses in data grids follows the Zipf-like distribution. Ranganathan and Foster [22, 30] claimed that the geometric distribution can properly model file access behaviors and the property of temporal/geographical locality.

Further, Ranganathan and Foster [31] derived file popularity by using both Zipf and geometric distributions on a multi-tier data grid with unlimited storage space. Tang et al. [23] also used Zipf-like and geometric distributions to simulate users' file access behaviors on a multi-tier data grid. Chang et al. [32,24] proposed two data replication strategies on a cluster-based data grid with limited storage. However, the strategy they proposed in [32] did

[*] Corresponding author.
E-mail addresses: mingchang1109@gmail.com (M.-C. Lee), leufy@thu.edu.tw (F.-Y. Leu), ypchen@nclab.tw, ypchen@cs.nctu.edu.tw (Y.-p. Chen).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.08.015

not consider the data access pattern. Hence, it might lead to inefficient data access as the users' access pattern changes; the strategy proposed in [24] only replicates the file most frequently accessed in the last time period, consequently resulting in long file transmission delays for those files with similar but low weights.

In this study, we propose an adaptive data replication algorithm, called the *P*opular *F*ile *R*eplicate *F*irst algorithm (PFRF for short), which is developed on a star-topology data grid with limited storage space. A star-topology data grid is a simplified tree-topology data grid with a central cluster that connects all other clusters. A link *l* between two arbitrary clusters will go through the central cluster, and *l* might comprise several routers, and physical links. Directly evaluating the components of *l* is difficult since too many analytical items might be involved. Hence, this study treats *l* as a logical link to simplify the original topology as a whole [33, 34]. The simplification process will be proposed. To adapt to the changes of users' interests in files, the PFRF aggregates file access information and replicates popular files to suitable clusters/sites. We simulate several cases in which file popularity follows a Zipf-like distribution, geometric distribution, and uniform distribution under the assumption that user behaviors vary with the changes of users' interests. The simulation results show that PFRF provides users with a system that has higher data availabilities, lower data transmission delays, and less bandwidth consumption for data access.

The rest of the paper is organized as follows. Section 2 introduces background and related work of this study. Section 3 describes the architecture of a star-topology data grid and the details of the PFRF. Simulation results are presented and discussed in Section 4. Section 5 concludes this article and addresses our future research.

2. Background and related work

In this section, we describe the architectures of data grids and several existing replication strategies and algorithms.

2.1. Data grid architecture

Data grids can be classified into multi-tier data grids, first proposed by the MONARC project [35], and cluster data grids, initially introduced by Chang et al. [32]. The multi-tier data grid architecture in which a leaf node represents a user or a computational node, and internal nodes are resource sites keeping sharable files. In this architecture, a file held by a site will also be held by all its ancestor sites. Therefore, the root site holds all files stored in the data grid. When an end user requires a file *F* which does not exist in his/her site, the user requests *F* from its immediate ancestor. If the ancestor does not have the file, it in turn requests *F* from its immediate ancestor. The process repeats until the user obtains the file from a node which holds the file. After that, the file will be replicated to all the nodes on this requesting path following the reverse direction of the requests. It is clear that file access latency can be reduced in a multi-tier data grid, but it leads to higher storage cost since files will be redundantly stored in multiple locations.

A cluster data grid consists of *n* clusters connected by the Internet [24]. Files are stored in these clusters. Each cluster has a header node (a header for short) responsible for managing site information and exchanging file access information with other cluster headers. A header periodically determines which file should be replicated by computing file weights. After that, the file with the highest weight will be replicated to clusters that need the file. Sites in these clusters can then locally and quickly retrieve the file. Compared with a multi-tier data grid, a cluster data grid consumes less storage to hold files.

2.2. Existing data replication algorithms/strategies

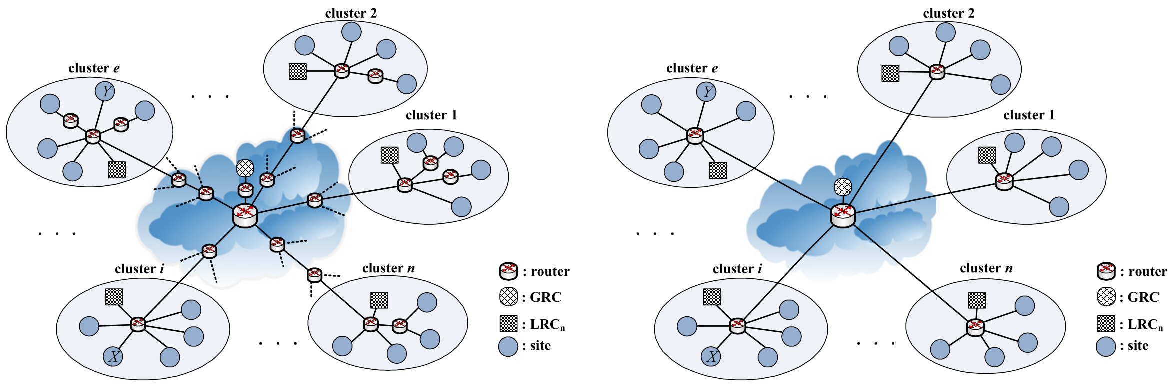
Least Frequently Used (LFU) [36] and Most Frequently Used (MFU) [36] are two simple dynamic replication strategies widely used in many areas, such as disk and cache memory duplication. If a storage device has insufficient space to hold a new file, LFU (MFU) will be invoked to choose the files that have been the least (most) frequently used as the victims to make room for the new one. In the experiments of this study, MFU and LFU are both involved, called the MFU/LFU strategy (M/LFU for short) in which MFU is used to choose the most frequently used files and LFU is employed to select victims once the destination cluster has insufficient storage space to save the replicated files.

Ranganathan and Foster [22] presented six replication/caching strategies for a multi-tier data grid: No Replication or Caching, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replication, and Fast Spread, and three types of localities: temporal locality, geographical locality, and spatial locality. The experimental results showed that the Fast Spread and Cascading Replication outperform the other four strategies and their file access latencies are shorter than those of the other four strategies. They also found that Fast Spread (Cascading) is better when the data access pattern is random (geographical locality). However, the six strategies cannot avoid the disadvantages of a multi-tier data grid, i.e., a file may be redundantly stored in a multi-tier. In fact, the storage space utilization and access latency are a trade-off [32]. Ranganathan and Foster [31] also proposed a suite of job scheduling and data replication algorithms for a multi-tier data grid and evaluated the performance of different combinations of the replication and scheduling strategies. One of the data replication algorithms, called DataRandom (DR for short), replicates a file when the corresponding access frequency exceeds a pre-defined threshold. Although DR is designed for an unlimited storage environment, it can also be run on a limited storage environment. DR is therefore involved in the experiments of this study.

Tang et al. [23] introduced Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU) algorithms to reduce the average data access response time for a multi-tier data grid. The basic idea of the two algorithms is to replicate a file to sites close to its requesting clients when the file's access rate is higher than a pre-defined threshold. SBU considers the file access history for individual site, but ABU aggregates the file access history for a system. With ABU, a node sends aggregated historical access records to its upper tiers, and the upper tiers do the same until these records reach the root. Due to the aggregation capability, ABU has a shorter job response time and less bandwidth consumption than those of SBU.

Khanli et al. [37] proposed an algorithm called Predictive Hierarchical Fast Spread (PHFS), which is an extended version of fast spread [22], in a multi-tier data grid. PHFS utilizes spatial locality [22,38] to predict data files required in the future, and pre-replicates these files to suitable sites to improve the performance of file accesses. Kunszt et al. [39] presented a replica management grid middleware to reduce file access/transfer time. Their experimental results showed that this middleware significantly reduces wide area transfer times. However, this model was developed for multi-tier data grids with unlimited storage space.

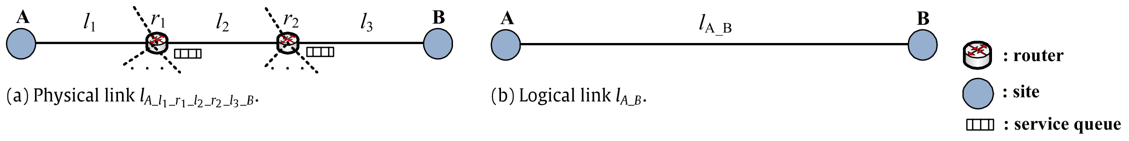
Chang et al. [24,32] presented two dynamic replication strategies, Latest Access Largest Weight (LALW) [24] and Hierarchical Replication Strategy (HRS) [32], on cluster-based data grids. LALW utilizes the half-life concept to evaluate file weights. A file with a higher access frequency has a larger weight. Their experimental results show that LALW outperforms LFU and no-replication data replication strategies [22] in network utilization and efficiency. However, LALW only replicates the most popular file in each time



(a) A tree-topology data grid.

(b) A star-topology data grid.

Fig. 1. The architectures of a tree-topology data grid and a star-topology data grid.



(a) Physical link $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$.

(b) Logical link l_{A_B}.

Fig. 2. (a) A physical link between site *A* and site *B*. (b) A logical link between site *A* and site *B*.

interval. Hence, the transmission delays of those files with similar but lower weights are still long. HRS, which is an extended version of BHR (Bandwidth Hierarchy based Replication) [40], aims to reduce expensive cluster-to-cluster replica transmission. Whenever the file is required, but cannot be retrieved from the local cluster, HRS replicates the file to a local site from a remote cluster. However, HRS does not consider data access patterns, and thus it might not be able to adapt to changes of user access behaviors and provide efficient data accesses.

3. System framework

The proposed data grid as shown in Fig. 1 consists of a *global replica controller* (GRC) and several clusters connected to the GRC through the Internet. As illustrated in Fig. 1(a), each connection comprises several routers and links, forming a tree-topology data grid rooted at the GRC. In other words, a file transmitted between two clusters will go through the routers and links on the two connections being considered. For example, when a site *X* in a cluster *i* issues a file access request to the GRC to request a file *F* stored in site *Y* which is a member of cluster *e*, the GRC then requests *Y* to deliver *F* to *X*. *F* then goes through routers and links between clusters *i* and *e*. Basically, it might not be easy to analyze the performance of the file transmission since in such a tree-topology too many network components and environmental parameters are involved. To simplify the evaluation of file transmission, in this study, we reduce a tree-topology data grid to a star-topology (see Fig. 1(b)).

3.1. Tree-to-star reduction

As shown in Fig. 2(a), site *A* connects to site *B* through link l_1, router r_1, link l_2, router r_2, and link l_3. The corresponding physical path is denoted by $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$. The bandwidth of the path is $\frac{|P|}{T_{l_1}+W_{r_1}+T_{l_2}+W_{r_2}+T_{l_3}}$ in which $|P|$ is the size of a packet *P* delivered through the path, $T_{l_i} = \frac{|P|}{B_{l_i}}$ is the transmission delays of link l_i, $W_{r_j} = \frac{1}{\mu_{r_j}-\lambda_{r_j}}$ is the service (queueing) delay of router r_j under the assumption that r_j is an M/M/1 queueing model where B_{l_i} is the bandwidth of l_i, $i = 1, 2$, and λ_{r_j} and μ_{r_j} are respectively the arrival rate and departure rate of r_j, $j = 1, 2, 3$. l_{A_B} shown in Fig. 2(b) is the logical link of $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$. Its bandwidth is $\frac{|P|}{T_{A\_B}}$ where T_{A_B} is the transmission delay of *P* from *A* to *B*. It is clear that $T_{A\_B} = T_{l_1} + W_{r_1} + T_{l_2} + W_{r_2} + T_{l_3}$. Therefore, we can conclude that $l_{A\_l_1\_r_1\_l_2\_r_2\_l_3\_B}$ can be reduced to a logical link l_{A_B}, and the tree-topology data grid shown in Fig. 1(a) could be reduced to a star-topology data grid shown in Fig. 1(b). Now a star-topology with the performance equivalent to that of the tree-topology can be obtained.

3.2. GRC and LRC

In the proposed architecture, each cluster comprises sites and a local replica controller (LRC). They are connected by a LAN or LANs. The LRC maintains a local replica table (*LRT*) includes filename, file location, access count, file weight, and master file fields, to record file access information. A master file is an original file that cannot be deleted from the data grid. File weight is the popularity of the file. Its calculation will be described later. File access count shows the frequency that the file is accessed by sites within a cluster. All master files are distributed to sites of different clusters. The GRC as a central server located somewhere in the Internet is responsible for aggregating file access records for all clusters and determining which files should be replicated to which clusters. The GRC maintains a global replica table (*GRT*) to collect the information recorded in *LRT*s. When the GRC decides to replicate a file to a cluster, it records the location of the new replica in *GRT* so that some time later when a LRC requests the location of the file, it can answer the LRC accordingly. Similarly, the cluster holding this new replica will record the related information in its *LRT*.

3.3. Zipf-like distribution and geometric distribution

To achieve a better file access performance, we need to keep track of users' file access behaviors to accordingly predict which files will be accessed frequently in the near future. The prediction is a main task of a data replication algorithm/strategy based on the assumption of temporal locality [22] in which a popular

```
PFRF algorithm:
Input: The total access count of $f_i$, $i = 1, 2, \ldots, N_k$, where $N_k$ is the number of files that cluster c
has in round $r$;
Output: Duplicating files that should be replicated to cluster c, denoted by $F_c = \{f_1, f_2, \ldots, f_{N_f}\}$, to
cluster c;
Procedure:
1:  for each cluster c, $c = 1, 2, \ldots, N_c${     /*$N_c$ is the number of clusters in the particular data grid*/
2:    Aggregate $A_c^r(f_i)$, $i = 1, 2, \ldots, N_k$;
3:    Sort all the files according to $A_c^r(f_i)$ in a descending order;
4:    Store the sorting result into set S;
5:    Calculate $TNF_c^r$ for $LRT_C$;
6:    Calculate $PW_c^r(f_i)$ and $PW_{avg}^r(f_i)$, $i = 1, 2, \ldots, N_k$;     /* Eqs. (3) and (4)*/
7:    Sort the set S according to the $PW_{avg}^r(f_i)$;
8:    Set the value of $x = 0.8$, $0 < x < 1$;     /*according to the 80/20 rule*/
9:    Calculate $N_f$;   /*$N_f$, derived with Eq. (5) for $x$=0.8, is the number of popular files*/
10:   Select the first $N_f$ files from S and put them into the set S';     /* S': the set of replication
11:                                                                            candidates*/
12:   for each file $f_j$ in S'{
13:       check $LRC_C$ to see whether cluster c has $f_j$;
14:       if cluster c does not have $f_j${
15:         if any site of cluster c has sufficient storage to save $f_j$
16:             replicate $f_j$ to the site from a nearest cluster which has $f_j$;
17:         else if{ for each site Y in cluster c { /* check storage space of each site Y in cluster c*/
18:                   for each file $f_k$ kept in site Y;
19:                   compare $PW_{avg}^r(f_k)$ with $PW_{avg}^r(f_j)$;
20:                   if there are $v$ files that are less popular than $f_j$ and the total size of these $v$
21:                      files plus the remaining storage space of site Y is sufficient to keep $f_j$
22:                    { PFRF deletes these $v$ files and replicates $f_j$ to Y;
23:                      break;}}}
24:         else /*The total size of all files that are less popular than $f_j$ plus the remaining storage
25:                 space is insufficient to hold $f_j$*/
26:         {     print ("no sites in cluster c can keep $f_j$"); /*PFRF will not replicate $f_j$*/
27: }}}}
```

Fig. 3. PFRF data replication algorithm.

file will be accessed more frequently than unpopular ones [23]. Breslau et al. [28] showed that webpage requests follow a Zipf-like distribution [29,41] derived from Zipf's law [42]. In the Zipf-like distribution, the access probability of the i-th most popular file, denoted by $P(f_i)$, is

$$P(f_i) = 1/i^{\alpha} \tag{1}$$

where $i = 1, 2, \ldots, n$ and α is a factor determining the file access distribution, $0 \le \alpha < 1$.

Ranganathan and Foster [30,31] adopted the geometric distribution to simulate file popularity in which the access probability of the i-th most popular file, denoted by $P(i)$, is

$$P(i) = (1-p)^{i-1} \cdot p \tag{2}$$

where $i = 1, 2, \ldots, n$ and $0 < p < 1$. A larger p represents that a smaller portion of files has been frequently accessed. As stated above, we assume that our users' access behaviors follow either Zipf-like or geometric distributions with different parameters.

3.4. Popular file replicate first (PFRF) algorithm

The PFRF algorithm, as illustrated in Fig. 3, is performed by the GRC at the end of a round, where a round is a fixed time period T_d in which y jobs, $y \ge 0$, are submitted by users from each cluster. A job might require several files as its input data. The algorithm comprises four phases: file access aggregate phase, file popularity calculation phase, file selection phase, and file replication phase.

1. File access aggregate phase: Between lines 2 and 5 of the algorithm, PFRF aggregates the access count for each file f_i stored in cluster c at round r, denoted by $A_c^r(f_i)$, sorts all the files on $A_c^r(f_i)s$ in a descending order, and stores the sorted result into a set S. After that, PFRF calculates the total number of files having been accessed by all sites in cluster c at round r, denoted by TNF_c^r, based on the information stored in LRT_C. Note that $1 \le i \le N_k$, and $1 \le c \le N_c$ where N_k is the number of files in cluster c in round r, and N_c is the number of clusters that the data grid has, and $r = 1, 2, 3, \ldots$.
2. File popularity calculation phase: In line 6, PFRF calculates a popularity weight for file f_i, denoted by $PW_c^r(f_i)$,

$$PW_c^r(f_i) = \begin{cases} PW_c^{r-1}(f_i) + A_c^r(f_i) \cdot a, & \text{if } A_c^r(f_i) > 0 \\ PW_c^{r-1}(f_i) - b, & \text{otherwise} \end{cases}, \quad r \ge 1,\ c \ge 1,\ i \ge 1. \tag{3}$$

where a and b are constants and $a < b$. The reason why $a < b$ is described later. If $A_c^r(f_i) > 0$, i.e., f_i has been accessed by users in round r, PFRF increases $PW_c^{r-1}(f_i)$ by $A_c^r(f_i) \cdot a$. Otherwise, it decreases $PW_c^{r-1}(f_i)$ by b. Basically, a higher $PW_c^r(f_i)$ implies that f_i is more popular. We assume that in round 0 all files follow the binomial distribution, i.e., $PW_c^0(f_i)$ =0.5, which means that the initial access probability of f_i is 0.5. Note that the minimum value of each $PW_c^{r-1}(f_i)$ is 0. From previous access records of f_i, PFRF derives the variation of the popularity of f_i and predicts the popularity of f_i for the next round, where $1 \le i \le N_k$. For instance, if f_3 has been accessed 5 times by cluster 2 in round 1, $PW_2^1(f_3) = 0.5 + 5 \cdot a$. After the derivation and prediction, PFRF calculates the average popularity of the files in all clusters, denoted by $PW_{avg}^r(f_i)$,

$$PW_{avg}^r(f_i) = \frac{\sum_{k=1}^{N_q} PW_c^r(f_i)}{N_q} \tag{4}$$

where N_q is the total number of clusters holding f_i in the data grid.
3. File selection phase: Between lines 7 and 10, PFRF sorts the set S on the average popular weights in a decreasing order, calculates N_f which is the number of files that might be replicated, and

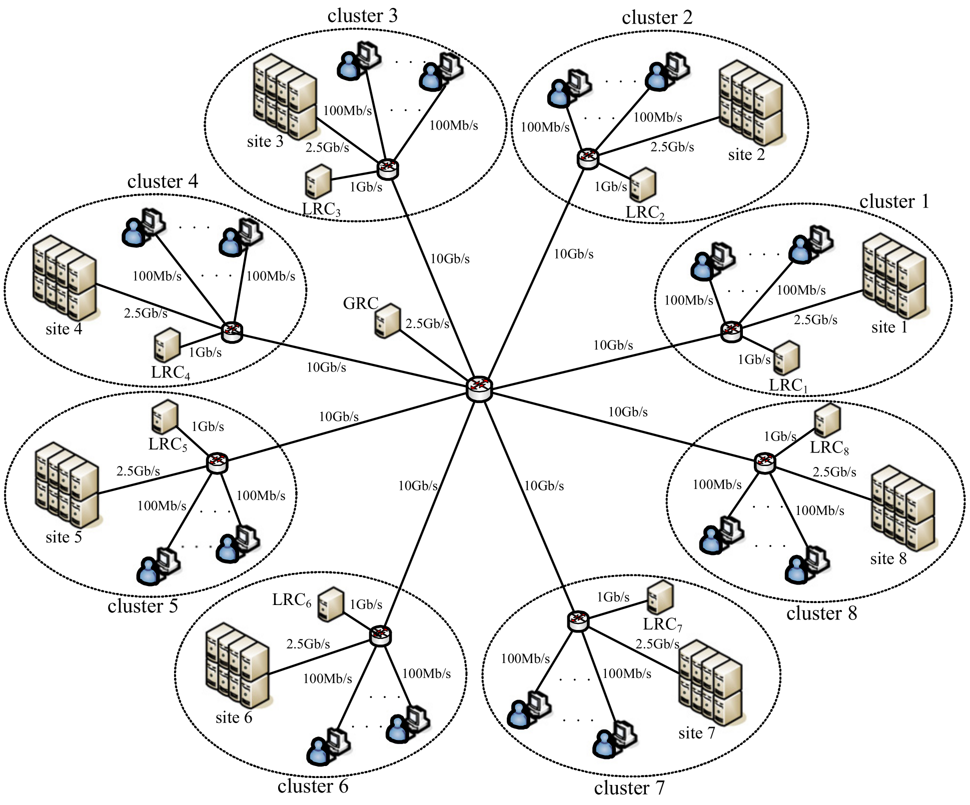


Fig. 4. The simulation topology.

selects the first N_f files as cluster c's replication candidates from S, where

$$N_f = \lfloor TNF_c^r \cdot (1 - x) \rfloor \quad (5)$$

in which x is a constant, $0 < x < 1$. In this study, we use the 80/20 rule as an example, i.e., PFRF will replicate the top 20% of frequently accessed files in this phase. Therefore, x is set to 0.8. If different rules or principles are employed, x may be changed.

4. File replication phase: Between lines 12 and 27, PFRF first checks to see whether each file, e.g., file f_j, in cluster c's replication candidates is stored in cluster c or not. If yes, PFRF does nothing. Otherwise, it further checks to see whether a site in cluster c has sufficient storage space to accommodate f_j or not. If yes, PFRF replicates f_j to the site from a nearest cluster holding f_j. Otherwise, PFRF deletes v files ($v \geq 1$) that are less popular than f_j from a site of cluster c so that it has enough storage space to keep f_j.

Since files and users may be located at different sites in the same cluster or different clusters in a star-topology data grid, a file transmission might go across clusters, consequently raising transmission delays. With the PFRF, popular files can be properly replicated and inter-cluster access can be dramatically reduced.

4. Simulation and performance comparison

To evaluate the proposed scheme, a real grid testbed or a grid simulator is required. However, to establish and maintain a real testbed is expensive. Instead, we choose a grid simulator as the simulation tool. Many grid simulators have been introduced, such as GridSim [43], MicroGrid [44], OptorSim [45], SimGrid [46], MONARC [47], and ChicSim [48], among which GridSim is chosen since it provides a flexible and extensible simulation environment and allows researchers to add new components/functions. In the following simulation, the existing data replication algorithms including M/LFU (recall MFU/LFU), DR (recall DataRandom), and No Replication (NR for short) are implemented and compared with the PFRF on a star-topology data grid.

4.1. Experimental environment and parameters

The test environment illustrated in Fig. 4 consists of a GRC and eight clusters. Each cluster has a LRC. The resource specifications and job parameters are listed in Tables 1 and 2, respectively. Each site comprises six computers, and each computer has four processors, i.e., a cluster has 24 processors. The processor rate of a processor is 1600 MIPS, i.e., the total processor rate of a cluster is 38,400 (=1600 × 24) MIPS. A cluster has 75 GB storage space to accommodate files. The inter-router, router-to-site, user-to-router, GRC-to-router, and LRC-to-router bandwidths are 10 Gb/s, 2.5 Gb/s, 100 Mb/s, 2.5 Gb/s, and 1 Gb/s, respectively. A total of 200 master files, each 1 GB in size, are randomly stored in this environment. A job randomly requests 5–10 files as its input files. On average, 10 jobs are submitted by each cluster in each round, i.e., on average a total of 80 jobs are submitted by all clusters in each round, and the time period of a round T_d is 1600 s.

In round r, M/LFU replicates cluster's most frequently used files to a cluster c in descending order. The replication is performed one by one until exhausting c's storage space. When c has insufficient storage space to replicate a remote file F, M/LFU selects k local files of c, denoted by $v = \{f_{c_1}, f_{c_2}, \ldots, f_{c_k}\}$, as the victims in the situation where the access count of f_{c_i} is smaller than that of F and k is the smallest integer that satisfies $\sum_{i=1}^{k} f_{c_i} \geq |F|$, where $|F|$ is the size of F. After that, M/LFU deletes the k files and replicates F to c.

We implemented two types of DR. One is DR-Local, in which a replication threshold of cluster c is set to the average access counts of all the files stored in c. For example, c has k files which have been accessed a total of m times in round r. The replication threshold is $\frac{m}{k}$. When a file, e.g., F, that is not stored in c has a

Table 1
Resources specifications of the following experiments.

Resources	Value
Total number of clusters	8
Total number of processors in a cluster	24
Single processor rate (MIPS)	1600
Total processor rate of a cluster	38,400 (=1600 × 24)
Storage available in a cluster	75 GB
Inter-router bandwidth	10 Gb/s
Router-to-site bandwidth	2.5 Gb/s
User-to-router bandwidth	100 Mb/s
GRC-to-router bandwidth	2.5 Gb/s
LRC-to-router bandwidth	1 Gb/s

Table 2
Job parameters of the following experiments.

Job parameters	Value
Total number of master files	200
Size of a master file	1 GB
Average number of jobs submitted by each cluster in a round	10
Average number of jobs submitted by all clusters in each round	80 (8 × 10 jobs)
Number of files accessed by a job	5–10
The duration of a round (T_d)	1600 s

higher access count than $\frac{m}{k}$ in round r, F will be replicated to c at the end of r, $1 \leq c \leq 8$. The other is DR-Global, in which the replication threshold is set to the average access counts of all files in all clusters, i.e., if the 200 files have been accessed h times in round r, the replication threshold will be $\frac{h}{200}$ for all clusters. When F has been accessed at least $\frac{h}{200}$ times by c in round r, it will be replicated to c. If c has insufficient space to hold the file, DR-Local and DR-Global will delete the files that have been least frequently accessed from c to make room for F.

Two types of NR were also implemented, denoted by NR-GRC and NR-LRC. In the NR-GRC, the 200 master files are all stored in the GRC. When a job requires a file, it remotely accesses the file from the GRC without storing the file locally. In the NR-LRC, the 200 master files are randomly distributed to the eight clusters. The GRC only maintains the *GRT*. A job locally accesses a file if the file is locally available. When it requires a remote file, it has to consult the GRC for the file location, and then remotely access the file. Due to duplicating no files, files are only stored in fixed sites and fixed clusters. Table 3 summarizes the master file settings for the tested algorithms.

4.2. Access patterns

Five access patterns listed in Table 4 were employed to simulate user access behaviors. File popularities follow Zipf-like (ZipfL for short), geometric (Geo for short), and uniform distributions (Uniform for short) where a uniform distribution represents that the probability of accessing a file by each user is the same. JRR, standing for job repeating rate, of round r is the probability of re-accessing those files that have been accessed in round $r - 1$, $0 \leq JRR \leq 1$, and parameters α and p are respectively used when ZipfL and Geo are employed.

Table 4
Different data access patterns employed.

No.	Data access pattern	α/p	JRR (%)	$p(f_i)/p(i)$
1	ZipfL-0.8	0.8	25	$1/i^{0.8}$
2	ZipfL-0.6	0.6	25	$1/i^{0.6}$
3	Geo-0.2	0.2	25	$(1-0.2)^{i-1} \cdot 0.2$
4	Geo-0.5	0.5	25	$(1-0.5)^{i-1} \cdot 0.5$
5	Uniform	None	25	None

To effectively analyze the algorithms, we evenly partitioned the popularities of the 200 master files into 10 levels and divided twenty consecutive rounds into three phases. As listed in Table 5, the first, the second, and the third phase respectively contain rounds 1–7, 8–14, and 15–20. In the first phase, we assume that File0–File19 are the most popular files, i.e., belonging to the first popularity level. File20–File39 are the second popular files, thus belonging to the second popularity level, and so on. In the second phase, we swap the files of the first two popularity levels, i.e., File20–File39 become the most popular, File0–File19 become the second, to simulate the change of file popularities, and other files' popularities remain unchanged. In the third phase, File40–File59 are the most popular files, File20–File39 the second, and File0–File19 the third. Other levels' file popularities remain unchanged.

To better understand the behaviors of data access patterns, i.e., file popularities, over the three phases, we first conduct the following experiments: 2000 jobs, instead of 80 jobs, were submitted for file accesses in each phase. The experimental results of ZipfL-0.8, ZipfL-0.6, Geo-0.2, Geo-0.5, and Uniform are illustrated in Figs. 5–9, respectively. Fig. 5(a) shows users' file access behaviors in the first phase; Fig. 5(b) and Fig. 5(c) respectively plot those in the second and the third phases. The access count (AC) of each most popular file in all the three phases/figures is about 215, and unpopular files, i.e., File60 to File199, are accessed less. In the case of ZipfL-0.6 (see Fig. 6), the AC of each most popular file in all the three phases is about 170. However, the ACs of the other popularity levels, i.e., between levels 4 and 10, are not evidently different, like a uniform distribution, in all three phases. When Geo-0.2 is invoked (see Fig. 7), the AC of each most popular file is about 175. When file IDs increase, the ACs decline more sharply than those in ZipfL-0.6 and ZipfL-0.8. In the Geo-0.5 case (see Fig. 8), the difference between/among the most popular files' ACs and those of the second and the third popular files in the three phases is significant. Generally, the ACs of the first three popularity levels on all access patterns are clearly different from those of the other popularity levels, implying that File0 to File59 are frequently accessed, while accesses of File140 to File199 are rare. The difference among the ACs of different popularity levels on the Uniform as shown in Fig. 9 is insignificant.

4.3. Simulation results

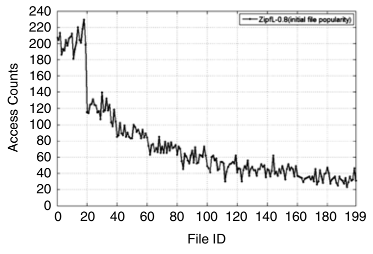
The tested algorithms were run on the same experimental environment so their performance can be fairly compared. Several

Table 3
Master files settings for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC algorithms.

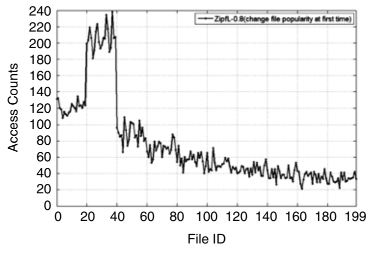
No.	Data replication algorithm	Setting
1	PFRF	200 master files are randomly distributed to the eight clusters
2	M/LFU	”
3	DR-Local	”
4	DR-Global	”
5	NR-GRC	200 master files are all stored in the GRC
6	NR-LRC	200 master files are randomly distributed to the eight clusters

Table 5
The file popularities in the three phases (Rounds 1–7, 8–14, and 15–20).

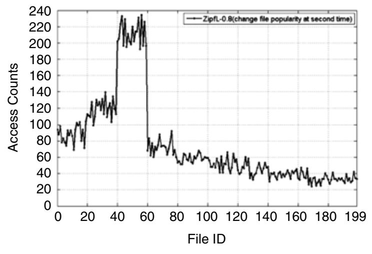
Popularity level (files)	Phases		
	Phase 1 (rounds 1–7)	Phase 2 (rounds 8–14)	Phase 3 (rounds 15–20)
1 (File0–File19)	1st popular	2nd popular	3rd popular
2 (File20–File39)	2nd popular	1st popular	2nd popular
3 (File40–File59)	3rd popular	3rd popular	1st popular
…	…	…	…
10 (File180–File199)	10th popular	10th popular	10th popular



(a) Phase 1.

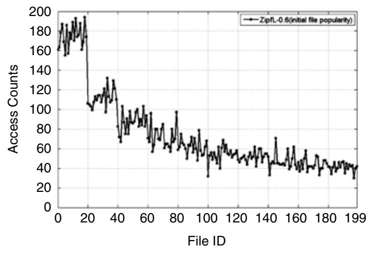


(b) Phase 2.

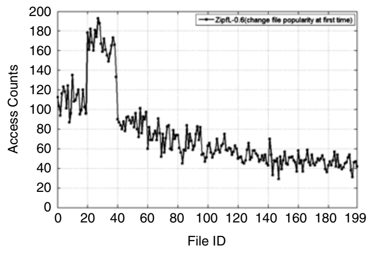


(c) Phase 3.

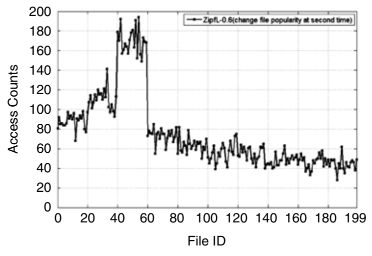
Fig. 5. Distributions of file requests against the 200 master files in the simulation process on ZipfL-0.8.



(a) Phase 1.

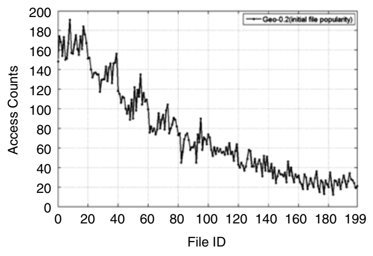


(b) Phase 2.

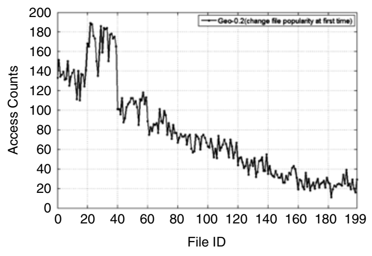


(c) Phase 3.

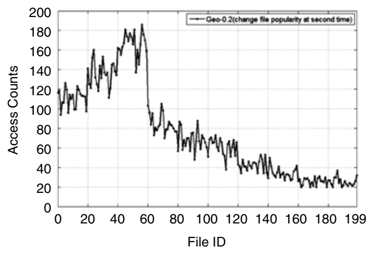
Fig. 6. Distributions of file requests against the 200 master files in the simulation process on ZipfL-0.6.



(a) Phase 1.

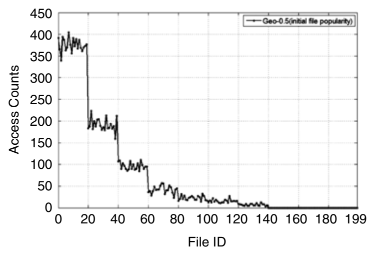


(b) Phase 2.

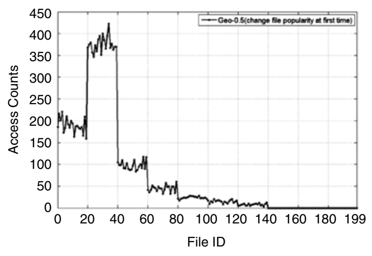


(c) Phase 3.

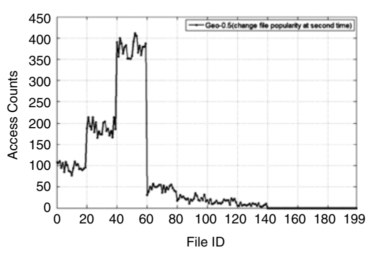
Fig. 7. Distributions of file requests against the 200 master files in the simulation process on Geo-0.2.



(a) Phase 1.



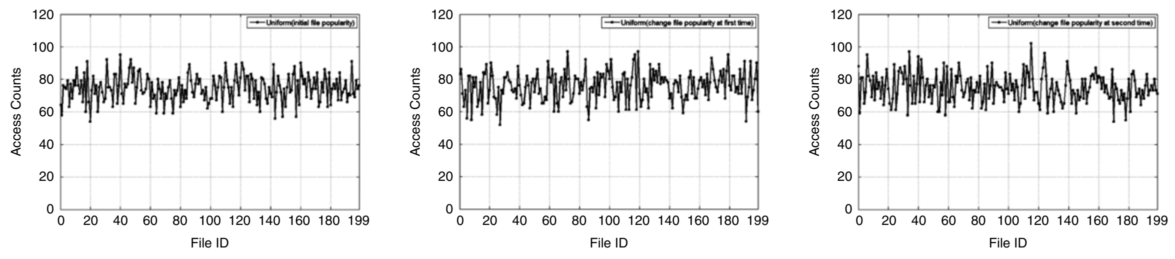
(b) Phase 2.



(c) Phase 3.

Fig. 8. Distributions of file requests against the 200 master files in the simulation process on Geo-0.5.

test metrics were used. The first is *average job turnaround time* (*ATT*), which is an average time interval from the time point when a job sends a file request to its LRC to the time point when the requested files are successfully received by the job. *ATT* is derived by dividing the total turnaround time of all jobs in all clusters in round *r* by the total number of jobs. The second is average



(a) Phase 1. (b) Phase 2. (c) Phase 3.

Fig. 9. Distributions of file requests against the 200 master files in the simulation process on Uniform.

data availability (*ADA*). Data availability, which was proposed by GridSim [43] for a job, e.g., job_x, in cluster c to access a file f_i of size d_{x_i}, denoted by $Avail_{x,c}$, is defined as

$$Avail_{x,c} = \frac{\sum_{i=1}^{k} t_{x_i,c}}{\sum_{i=1}^{k} d_{x_i}} \tag{6}$$

where k is the number of files accessed by job_x running on cluster c in round r, and $t_{x_i,c}$ is the time that job_x consumes to acquire f_i, locally or remotely. Let $avgAvail_c$ be the average data availability of cluster c which is defined as

$$avgAvail_c = \frac{\sum_{x \in jobs_c} Avail_{x,c}}{|jobs_c|} \tag{7}$$

where $jobs_c$ is the set of jobs submitted to cluster c by users to access files. The *ADA* of all clusters is defined as

$$ADA = \frac{\sum_{m=1}^{N_c} avgAvail_m}{N_c} \tag{8}$$

where N_c is the number of clusters in the data grid. The last is average bandwidth cost ratio (*ABCR*). *Bandwidth cost ratio* of cluster c in a round, denoted by BCR_c, is defined as

$$\begin{aligned} BCR_c &= \frac{LFA_c \cdot LC_c + RFA_c \cdot RC_c}{AFA_c \cdot C_{baseline}} \\ &= \frac{LFA_c \cdot LC_c + RFA_c \cdot RC_c}{(LFA_c + RFA_c) \cdot C_{baseline}} \end{aligned} \tag{9}$$

where $LFA_c (RFA_c)$ is the number of files that users in cluster c can locally (should remotely) access, $AFA_c = LFA_c + RFA_c$, LC_c is the cost for a user in cluster c to locally access a file. RC_c is the cost for the user to access a file from a remote cluster or the GRC, and $C_{baseline}$ is the average cost for a user to access a file from a remote cluster to local cluster c. If LFA_c is larger than RFA_c, that implies the particular data replication algorithm can more accurately predict user access behaviors. Otherwise, the algorithm due to inaccurate prediction would consume a lot of network resources to remotely access files. Eq. (9) only involves the number of files and neglects file sizes since in the simulation all files are of the same size, i.e., 1 GB. The *ABCR* used to determine whether a data replication algorithm could accurately predict popular files or not is defined as

$$ABCR = \frac{\sum_{m=1}^{N_c} BCR_m}{N_c} \tag{10}$$

where N_c is the total number of clusters in the data grid. A data replication algorithm with a smaller *ABCR* value will lead to better grid performance since most data can be locally retrieved. In the following, each simulation was performed ten times to obtain the values of the three performance metrics.

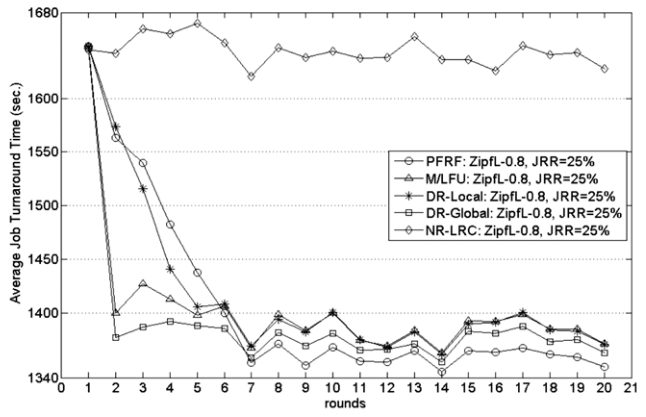


Fig. 10. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.8 with JRR = 25%.

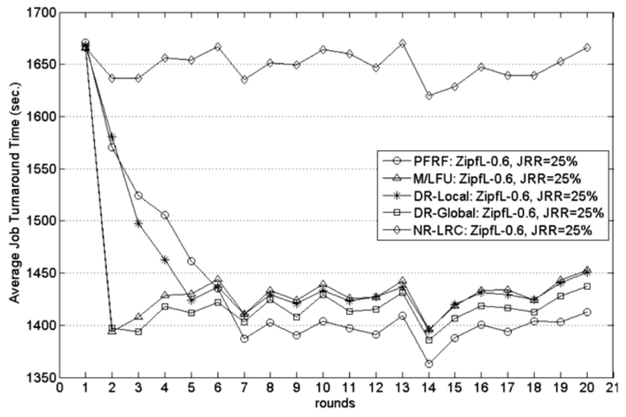


Fig. 11. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.6 with JRR = 25%.

4.3.1. *Average job turnaround time (ATT) and average data availability (ADA)*

Fig. 10 show the experimental results of *ATT*s for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on access pattern ZipfL-0.8 with JRR = 25%. The results of ZipfL-0.6, Geo-0.2, Geo-0.5, and Uniform are illustrated in Figs. 11–14, respectively. When ZipfL-0.8 with JRR = 25% is employed, as shown in Fig. 10, PFRF's *ATT*s are shorter than those of M/LFU, DR-Local, and DR-Global after the sixth round. This is also true on ZipfL-0.6 and Geo-0.2 (see Figs. 11 and 12, respectively). Fig. 13 plots the experimental results of Geo-0.5. On Uniform with JRR = 25% as shown in Fig. 14, *ATT*s of PFRF, M/LFU, DR-Local, and DR-Global are longer than those shown in Figs. 10–13 since the tested algorithms cannot effectively discriminate file popularity. It is clear that PFRF has shorter *ATT*s than those of M/LFU, DR-Local, and DR-Global on all of ZipfL-0.8, ZipfL-0.6, and Geo-0.2.

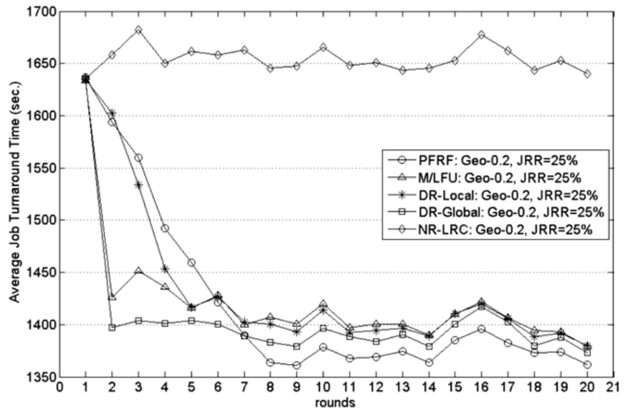


Fig. 12. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on Geo-0.2 with JRR = 25%.

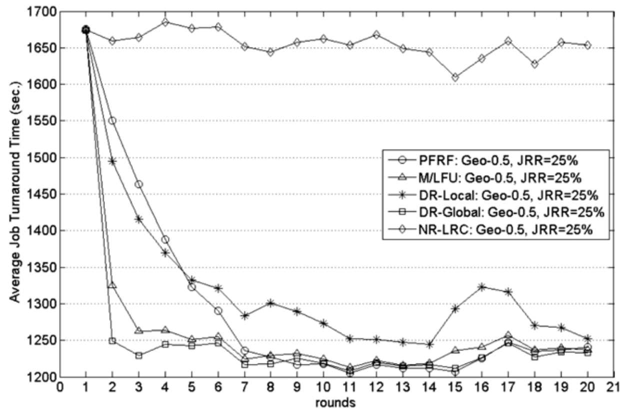


Fig. 13. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on Geo-0.5 with JRR = 25%.

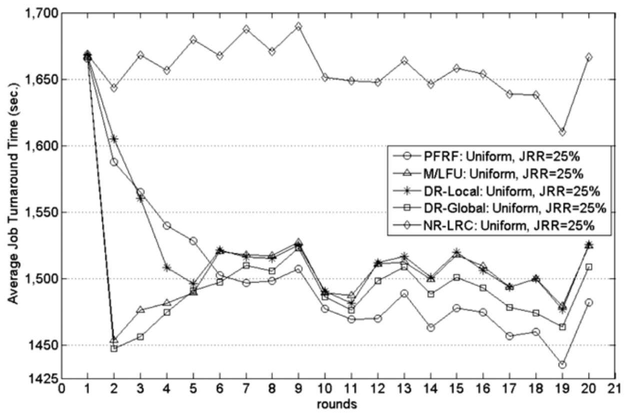


Fig. 14. Average job turnaround times for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on Uniform with JRR = 25%.

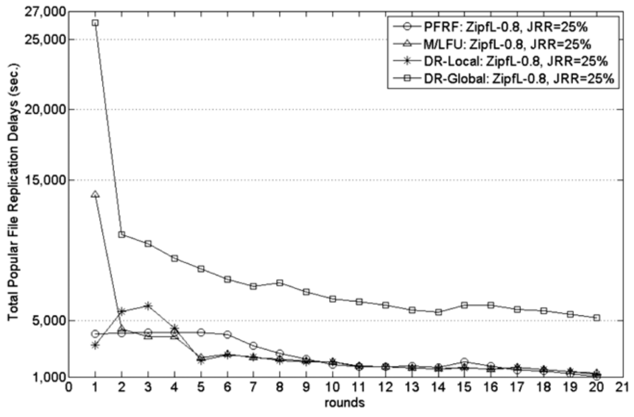


Fig. 15. Total popular file replication delays for PFRF, M/LFU, DR-Global, and DR-Local on ZipfL-0.8 with JRR = 25%.

When the data access pattern is Geo-0.5 with 25% (see Fig. 13), PFRF, M/LFU, and DR-Global have similar *ATT*s since the popular files shown in Fig. 8 can be easily identified. This is also why these algorithms on Geo-0.5 have the shortest *ATT*s compared with *ATT*s of these algorithms on other data access patterns (see scale of *Y*-axis of Figs. 10–14). Note that, on Geo-0.5, DR-Local does not effectively keep up with the change of user access behaviors especially at the end of phase 1 and phase 2. The reason is that it keeps accumulating the access count of certain popular files, resulting in a higher average access count as the replication threshold. It is impossible for a rising-popularity file, e.g., *F*, to have its access count suddenly larger than the threshold. Hence, *F* will not be replicated.

*ATT*s of PFRF on ZipfL-0.8, ZipfL-0.6, and Geo-0.2 in the twenty rounds are shown in Figs. 10–12, in which when file popularities change, i.e., between rounds 7 and 8 and between rounds 14 and 15 according to Table 5, *ATT*s of PFRF increase less sharply than those of other algorithms, implying that the PFRF can quickly adapt to the change of user access behaviors and provide better access performance as compared with all the other algorithms.

We have not mentioned NR-GRC and NR-LRC since NR-LRC's plots are high above those of the other four access patterns (see Figs. 10–14), and NR-GRC leads to a longer *ATT* (4714 s in average) which is about three times the highest scale of each figure. If we plot the results of NR-GRC in Figs. 10–14, the *ATT*s of the other algorithms will be close to each other and cannot be discriminated. The cause of longer *ATT*s for NR-GRC is that because there are no local files, all jobs have to remotely access all required files from the GRC. On NR-LRC, a job in each round spent about 1625–1675 s (see Figs. 10–14). Apparently, NR's access performance is not better than those of the other four algorithms on all access patterns.

Fig. 15 shows the total popular file replication delays of the PFRF, M/LFU, DR-Local, and DR-Global on ZipfL-0.8 with JRR = 25%. Note that NR does not replicate files. Hence, its experimental results are absent from this figure. Due to the page limit, we omit the total popular file replication delays of these algorithms on the other access patterns since they are similar to those illustrated in Fig. 15. According to the 80/20 rule, PFRF only replicates the top 20% of popular files, and thus PFRF always spends less than 5000 s to replicate popular files from remote clusters to local sites in each round. For each cluster, due to the number of remote files becoming less over time, *ATT*s of PFRF as shown in Figs. 10–14 reduce gradually. M/LFU replicates required files from remote clusters in each round, consequently, like that of PFRF, taking a longer time (about 14,000 s) to replicate files in the first round, and spending less in the later rounds.

After the fourth rounds, the total popular file replication delays of DR-Local are almost the same as those of M/LFU. Thus, *ATT*s of DR-Local are then similar to those of M/LFU on all data access patterns except on Geo-0.5. That is why in Figs. 10–12 and 14, the two curves almost overlap. DR-Global results in longer file replication delays than those of other algorithms and the delays are always higher than 5000 s in each round since it keeps replicating files in each round. That is why *ATT*s of DR-Global as shown in Figs. 10–14 can remarkably decline after the first round. However, *ATT*s of DR-Global are longer than those of PFRF after the sixth round in all access patterns except on Geo-0.5. The reason has been stated above.

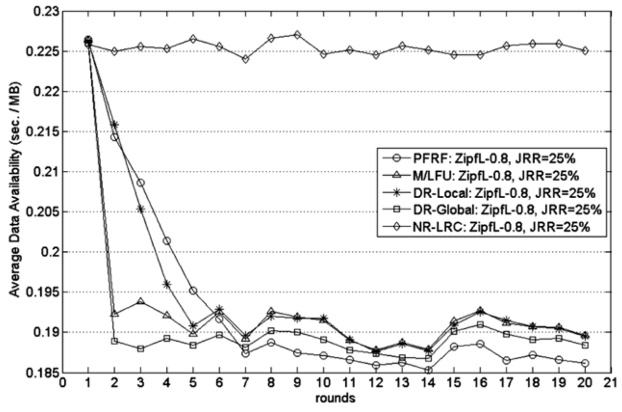


Fig. 16. Average data availabilities for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.8 with JRR = 25%.

Fig. 16 illustrates the *ADA*s for PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC on ZipfL-0.8 with JRR = 25%. According to the definition of $Avail_{x,c}$ presented in Eq. (6), the numerator $t_{x_i,c}$ is the turnaround time of job_x in cluster c, and hence the *ADA*s of all the tested algorithms have similar trends to that of their *ATT*s on all data access patterns. For example, all curves in Fig. 10 are similar to those in Fig. 16 on ZipfL-0.8 with JRR = 25%. Note that the *ADA*s of NR-GRC in each round are also omitted from Fig. 16 because they are the worst (about 0.637 which is far above the top scale, 0.23, of Fig. 16). The cause of these high *ADA*s was mentioned above.

4.3.2. Average bandwidth cost ratio (ABCR)

Fig. 17 illustrates the *ABCR*s of PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on ZipfL-0.8 with JRR = 25%. The bandwidths of the inter-router link, router-to-site link, user-to-router link, and GRC-to-router link as listed in Table 1 are respectively 10, 2.5, 0.1, and 2.5 Gb/s, and the corresponding unit costs are respectively $\frac{1}{10}$, $\frac{1}{2.5}$, $\frac{1}{0.1}$, and $\frac{1}{2.5}$. With the tested algorithms other than NR-GRC, files required by a job may be stored in remote clusters or a local cluster. To fairly compare all these algorithms, the router-to-site link and GRC-to-router link are given the same bandwidth, i.e., 2.5 Gb/s.

With NR-GRC, LC_c and LFA_c in Eq. (9) are zero since all master files are located at the GRC, i.e., $AFA_c = RFA_c$, $RC_c = \frac{1}{2.5} + \frac{1}{10} + \frac{1}{0.1} = 10.5$, and $C_{baseline} = \frac{1}{2.5} + \frac{1}{10} + \frac{1}{10} + \frac{1}{0.1} = 10.6$. Thus, $BCR_c = \frac{RC_c}{C_{baseline}} = 0.99$, $ABCR = BCR_c$, and $ABCR = 0.99$ in all rounds on all access patterns. For PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC, $RC_c = C_{baseline} = 10.6$, and $LC_c = \frac{1}{2.5} + \frac{1}{0.1} = 10.4$.

Comparing the plots shown in Figs. 17–21, *ABCR*s of NR-LRC in the five figures are all the worst, between 0.997 and 0.998. The reason is that NR-LRC does not replicate files among clusters; hence, each cluster has to access required files from remote clusters, even though it has frequently accessed these files. PFRF can effectively adjust file weights and lead to the best *ABCR*s on ZipfL-0.8, ZipfL-0.6, and Geo-0.2 after the sixth round, as compared with all the other algorithms. However, *ABCR*s of NR-GRC are better than those of other algorithms on ZipfL-0.6 (see Fig. 18) since all unpopular files have similar access count (see Fig. 6), like those of a uniform distribution. As shown in Fig. 20, PFRF, M/LFU, and DR-Global on Geo-0.5 have similar *ABCR*s, which are better than those of DR-Local, NR-GRC, and NR-LRC since the former three algorithms can effectively identify popular files and replicate them to the clusters requiring these files.

As shown in Fig. 21, NR-GRC on Uniform with JRR = 25% outperforms all the other algorithms since all required files can be accessed from the GRC rather than from remote clusters. On the other hand, *ABCR*s of PFRF, M/LFU, DR-Local, and DR-Global

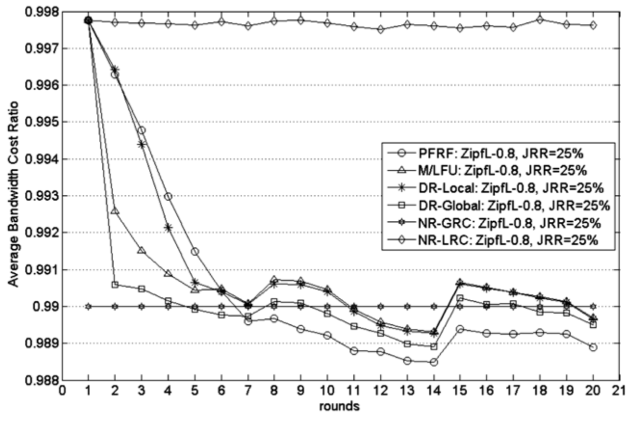


Fig. 17. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on ZipfL-0.8 with JRR = 25%.

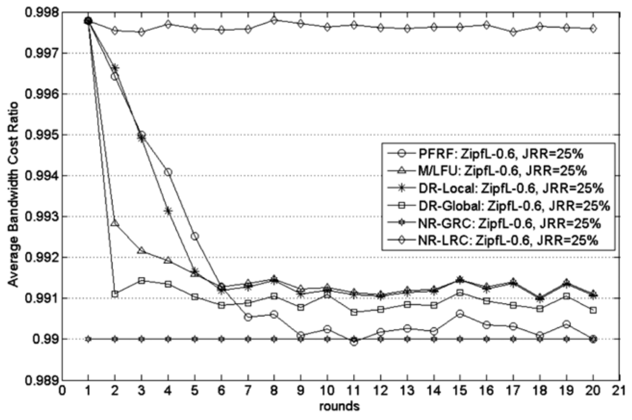


Fig. 18. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on ZipfL-0.6 with JRR = 25%.

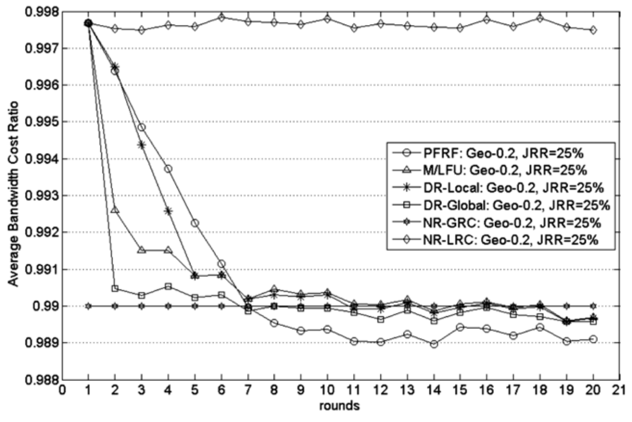


Fig. 19. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on Geo-0.2 with JRR = 25%.

are similar because popularities of all files as shown in Fig. 9 are similar and remain unchanged over time. According to Eqs. (9) and (10), the increase of RFA_cs will result in higher BCR_cs and *ABCR*s, indicating that the bandwidths consumed by all algorithms except NR-GRC on Uniform, of which *ABCR* is about 0.993 after the sixth (see Fig. 21), are higher than those on the other access patterns. Please compare the *ABCR* value 0.993 with those shown in Figs. 17–20, all between 0.983 and 0.992.

If we change the bandwidth of the user-to-router links shown Fig. 4 from 100 Mb/s to 1 Gb/s, when NR-GRC is employed, $RC_c = 1.5$, $C_{baseline} = 1.6$, and $BCR_c = ABCR = 0.937$ which is lower

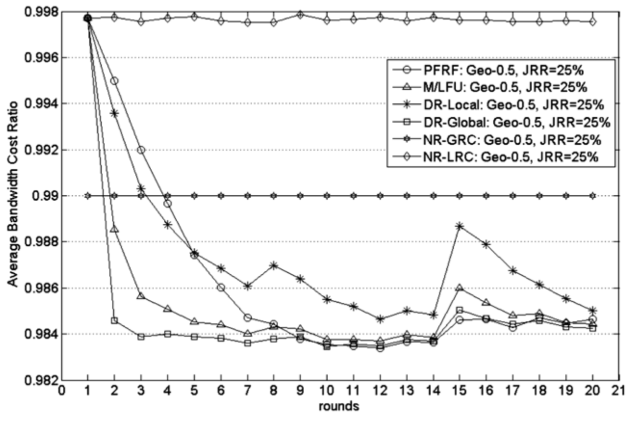


Fig. 20. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on Geo-0.5 with JRR = 25%.

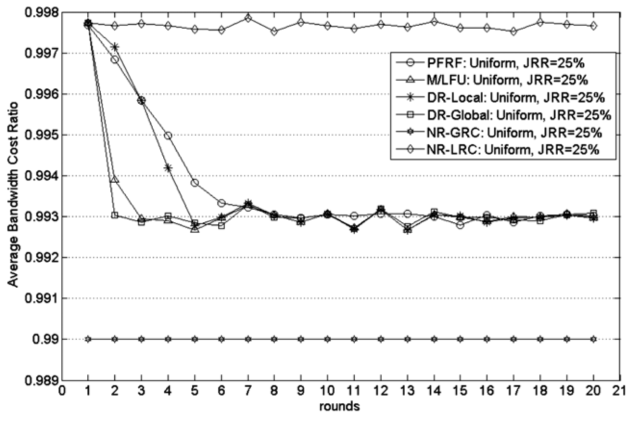


Fig. 21. Average bandwidth cost ratios for PFRF, M/LFU, DR-Local, DR-Global, NR-GRC, and NR-LRC on Uniform with JRR = 25%.

than that calculated above, i.e. 0.99. When NR-LRC is used, $RC_c = C_{baseline} = 1.6$, $LC_c = 1.4$, and $ABCR \approx 0.984$ which is also lower than the value calculated above for NR-LRC, i.e., between 0.997 and 0.998, implying that when there is a bottleneck along a path, the *ABCR* will be bigger. Other algorithms have similar phenomena.

4.3.3. Comparison of PFRF parameters a and b

The experimental environment used to evaluate the parameters a and b in Eq. (3) is the same as the one mentioned in Section 4.1. Fig. 22 shows experimental results for *ATT*s of PFRF when $a < b$(i.e., $a = 0.1, b = 0.15$), $a = b$ (i.e., $a = 0.1, b = 0.1$), and $a > b$ (i.e., $a = 0.15, b = 0.1$) on ZipfL-0.8 with JRR = 25%. Fig. 23, a zoomed-in portion of Fig. 22, illustrates the *ATT*s of PFRF from round 6 to round 20. Evidently, the case of ($a = 0.1, b = 0.15$) has the best *ATT*s, showing that it can accurately reflect which files are more popular. Therefore, in this study we select ($a = 0.1, b = 0.15$) to do the previous simulations.

4.3.4. Discussion

From an end user viewpoint, the goal of invoking a data replication algorithm is to shorten average turnaround time and enhance data availability. From the whole system viewpoint, the data replication algorithm should reduce bandwidth cost/consumption for grid systems. From the simulation results, we can see that PFRF, M/LFU, DR-Local, and DR-Global can improve turnaround time, bandwidth cost ratio, and data availability. Although NR-LRC can slightly reduce job turnaround time and improve data availability,

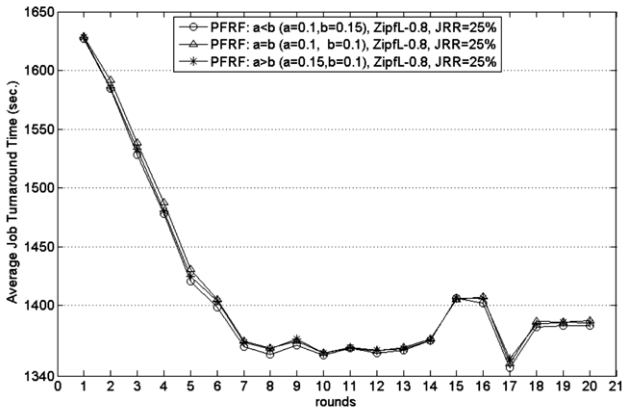


Fig. 22. The average job turnaround time against different rounds.

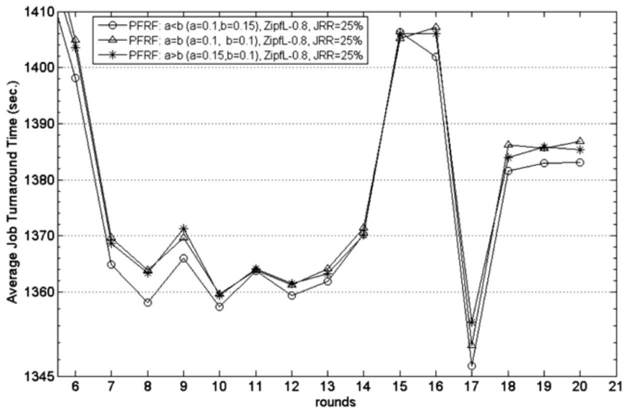


Fig. 23. The zoom-in figure between rounds 6 to 20 in Fig. 22.

its average bandwidth cost is still high. NR-GRC has the best average bandwidth cost ratios on the uniform distribution, but it has the worst turnaround time and data availability on all data access patterns.

In most situations, PFRF outperforms M/LFU, DR-Local, DR-Global, and NR-LRC on the three performance metrics. Although DR-Global has similar performance to that of PFRF, its high replication frequency will run out of storage space quickly, and consume considerable network bandwidth, resulting in high disk I/O costs for all clusters in a data grid. In our simulations, the average job turnaround time does not include the total popular file replication time. In other words, DR-Global is not really optimal.

Note that the GRC is only responsible for supplying the locations of files/replicas, instead of files/replicas themselves, to all clusters when PFRF, M/LFU, DR-Local, DR-Global, and NR-LRC algorithms are employed. Therefore, the bandwidth of the GRC will not be the bottleneck of these algorithms. However, with NR-GRC, all requested files are transmitted from the GRC. The bandwidth of the GRC might be a bottleneck when it receives a high volume of file requests from clusters.

5. Conclusions and future work

In this paper, we propose a novel data replication algorithm for a star-topology data grid with limited storage space to improve system performance. Although some previous studies have done that, such as providing shorter job response time, higher network usage, and less storage occupation on limited storage space, they did not consider data access patterns and the change of user access behaviors over time. Therefore, PFRF is designed to improve these weaknesses. It can effectively adapt to the changes of users' interests by dynamically adjusting file weights and replicating these files to appropriate clusters to improve performance of the whole system. We also analyze the average job turnaround time, average data availability, and average bandwidth cost ratio as the performance metrics of PFRF and compare them with those of five existing algorithms on five data access patterns. The simulation results show that PFRF outperforms all the tested algorithms when file popularity changes with time.

In the future, we plan to validate our simulation results on real data grids so that the proposed scheme can be evaluated on a real testbed. We would also like to enhance the reliability of the architecture by providing a hot-standby GRC like that presented in [49] to take over as the GRC when the GRC cannot work properly. We will also try to replicate popular files to users' local sites, rather than to users' local clusters. This can further reduce intra-cluster bandwidth consumption and unnecessary data transmission time. Finally, we plan to develop a reliability model to evaluate how many replicas are required for a file such that the file can stand against site failures. Those constitute our future studies.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grants NSC 99-2221-E-009-123-MY2 and NSC 100-2221-E-029-018.

The authors would like to thank the anonymous referees for their helpful comments that improved the quality of this paper.

References

[1] A. Folling, C. Grimme, J. Lepping, A. Papaspyrou, Robust load delegation in service grid environments, IEEE Transactions on Parallel and Distributed Systems 21 (9) (2010) 1304–1316.
[2] O. Sonmez, H. Mohamed, D. Epema, On the benefit of processor coallocation in multicluster grid systems, IEEE Transactions on Parallel and Distributed Systems 21 (6) (2010) 778–789.
[3] H. Li, Realistic workload modeling and its performance impacts in large-scale escience grids, IEEE Transactions on Parallel and Distributed Systems 21 (4) (2010) 480–493.
[4] H. Mohamed, D. Epema, Koala: a co-allocating grid scheduler, Concurrency and Computation: Practice and Experience 20 (16) (2008) 1851–1876.
[5] B. Tierney, W. Johnston, J. Lee, M. Thompson, A data intensive distributed computing architecture for grid applications, Future Generation Computer Systems 16 (5) (2000) 473–481.
[6] BIRN. http://www.nbrin.net/.
[7] LHC accelerator project. http://www-td.fnal.gov/LHC/USLHC.html.
[8] European DataGrid Project (EDG). http://www.eu-egee.org.
[9] GriPhyN: The Grid physics network project, 12 July 2010. http://www.griphyn.org.
[10] PPDG. http://www.ppdg.net.
[11] R.S. Chang, M.S. Hu, A resource discovery tree using bitmap for grids, Future Generation Computer Systems 26 (1) (2010) 29–37.
[12] J. Wu, X. Xu, P. Zhang, C. Liu, A novel multi-agent reinforcement learning approach for job scheduling in grid computing, Future Generation Computer Systems 27 (5) (2011) 430–439.
[13] S. Ebadi, L.M. Khanli, A new distributed and hierarchical mechanism for service discovery in a grid environment, Future Generation Computer Systems 27 (6) (2011) 836–842.
[14] M.E.J. Newman, Power laws, Pareto distributions and Zipf's law, Contemporary Physics 46 (2005) 323–351.
[15] K. Sashi, A.S. Thanamani, Dynamic replication in a data grid using a modified BHR region based algorithm, Future Generation Computer Systems 27 (2) (2011) 202–210.
[16] M. Lei, S.V. Vrbsky, X. Hong, An on-line replication strategy to increase availability in data grids, Future Generation Computer Systems 24 (2) (2008) 85–98.
[17] O. Wolfson, S. Jajodia, Y. Huang, An adaptive data replication algorithm, ACM Transactions on Database Systems 22 (2) (1997) 255–314.
[18] M. Rabinovich, I. Rabinovich, R. Rajaraman, Dynamic replication on the internet, Technical Report, HA6177000–980305-01-TM, AT&T Labs, March 1998.
[19] J.M. Perez, F. Garcia-Carballeira, J. Carretero, A. Calderon, J. Fernandez, Branch replication scheme: a new model for data replication in large scale data grids, Future Generation Computer Systems 26 (1) (2010) 12–20.
[20] M. Vrable, S. Savage, G.M. Voelker, Cumulus: filesystem backup to the cloud, ACM Transactions on Storage 5 (4) (2009).
[21] H. Shen, An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems, IEEE Transactions on Parallel and Distributed Systems 21 (6) (2010) 827–840.
[22] K. Ranganathan, I. Foster, Identifying dynamic replication strategies for a high performance data grid, in: Proceedings of the Second International Workshop on Grid Computing, Denver, CO, November 2001, pp. 75–86.
[23] M. Tang, B.S. Lee, C.K. Yeo, X. Tang, Dynamic replication algorithms for the multi-tier data grid, Future Generation Computer Systems 21 (2005) 775–790.
[24] R.S. Chang, H.P. Chang, A dynamic data replication strategy using access-weights in data grids, Journal of Supercomputing 45 (3) (2008) 277–295.
[25] S.Y. Ko, R. Morales, I. Gupta, New worker-centric scheduling strategies for data-intensive grid applications, in: Proc. ACM/IFIP/USENIX Int'l Conference on Middleware, 2007, pp. 121–142.
[26] L. Meyer, J. Annis, M. Wilde, M. Mattoso, I. Foster, Planning spatial workflows to optimize grid performance, in: Proc. ACM Symp. Applied Computing, 2006, pp. 786–790.
[27] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010).
[28] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: evidence and implications, in: Proceedings of IEEE INFOCOM'99, no.1, New York, USA, ppp. 126–134, 21–25 March 1999.
[29] D.G. Cameron, R. Carvajal-Schiaffino, A. Paul Millar, C. Nicholson, K. Stockinger, F. Zini, Evaluating scheduling and replica optimisation strategies in optorsim, in: The International Workshop on Grid Computing, Phoenix, Arizona, November 17, IEEE Computer Society Press, 2003.
[30] K. Ranganathan, I. Foster, Decoupling computation and data scheduling in distributed data intensive applications, in: International Symposium for High Performance Distributed Computing, HPDC-11, Edinburgh, 2002.
[31] K. Ranganathan, I. Foster, Simulation studies of computation and data scheduling algorithms for data grids, Journal of Grid Computing 1 (2003) 53–62.
[32] R.S. Chang, J.S. Chang, S.Y. Lin, Job scheduling and data replication on data grids, Future Generation Computer Systems 23 (7) (2007) 846–860.
[33] M. Coates, A. Hero, R. Nowak, B. Yu, Internet tomography, IEEE Signal Processing Magazine 19 (3) (2002).
[34] G. Levitin, Y.S. Dai, B.H. Hanoch, Reliability and performance of star topology grid service with precedence constraints on subtask execution, IEEE Transactions on Reliability 55 (3) (2006) 507–515.
[35] The MONARC project. http://monarc.web.cern.ch/MONARC/.
[36] A. Silberschatz, P.B. Galvin, G. Gagne, Operating System Concepts, 7th ed., Wiley, 2004.
[37] L.M. Khanli, A. Isazadeh, T.N. Shishavan, PHFS: a dynamic replication method, to decrease access latency in the multi-tier data grid, Future Generation Computer Systems 27 (3) (2011) 233–244.
[38] M.L. Yiu, H. Lu, N. Mamoulis, M. Vaitis, Ranking spatial data by quality preferences, IEEE Transactions on Knowledge and Data Engineering 23 (3) (2011).
[39] P. Kunszt, E. Laure, H. Stockinger, K. Stockinger, File-based replica management, Future Generation Computer Systems 21 (2005) 115–123.
[40] S.M. Park, J.H. Kim, Y.B. Ko, W.S. Yoon, Dynamic Data Grid Replication Strategy Based on Internet Hierarchy, in: Lecture Notes in Computer Science, vol. 3033, 2004, pp. 838–846.
[41] D.G. Cameron, R.C. Schiaffino, J. Ferguson, P. Millar, C. Nicholson, K. Stockinger, F. Zini, OptorSim v2.0 installation and user guide, November 2004. http://edgwp2.web.cern.ch/edg-wp2/optimization/optorsim.html.
[42] G. Kingsley Zipf, Relative frequency as a determinant of phonetic change, Reprinted from the Harvard Studies in Classical Philiology, Volume XL, 1929.
[43] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, R. Buyya, A toolkit for modelling and simulating data grids: an extension to gridsim, in: Concurrency & Computation: Practice and Experience, Wiley Press, New York, USA, 2008.
[44] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, A. Chien, The microgrid: a scientific tool for modeling computational grids, in: Proc. of IEEE Supercomputing Conference, Dallas, USA, November 4–10 2000.
[45] W. Bell, D. Cameron, L. Capozza, P. Millar, K. Stockinger, F. Zini, Simulation of dynamic grid replication strategies in optorsim, in: Proc. of the 3rd International Workshop on Grid Computing, GRID, Baltimore, USA, 18 November 2002.

[46] A. Legrand, L. Marchal, H. Casanova, Scheduling distributed applications: the simgrid simulation framework,in: Proc. of the 3rd International Symposium on Cluster Computing and the Grid, Tokyo, Japan, May 12–15 2003.
[47] C.M. Dobre, C. Stratan, Monarc simulation framework, in: Proc. of the RoEduNet International Conference, Timisoara, Romania, May 27–28 2004.
[48] ChicSim—the Chicago grid simulator, 5 October 2007. http://people.cs.uchicago.edu/_krangana/ChicSim.html.
[49] F.Y. Leu, C.T. Yang, F.C. Jiang, Improving reliability of a heterogeneous grid-based intrusion detection platform using levels of redundancies, Future Generation Computer Systems 26 (4) (2010) 554–568.

Ming-Chang Lee received the M.S. degree in Computer Science Department from TungHai University, Taiwan, in 2006. Currently, he is a Ph.D. student of National Chiao Tung University, Hsinchu, Taiwan. His research interests include grid applications, distributed computing, network security, and estimation of distribution algorithms.

Fang-Yie Leu received his B.S., Master and Ph.D. degrees from National Taiwan University of Science and Technology, Taiwan, in 1983, 1986 and 1991, respectively, and another Master's degree from Knowledge System Institute, USA, in 1990. His research interests include wireless communication, network security, Grid applications and Chinese natural language processing. He is currently a professor of Tunghai University, Taiwan, and the director of database and network security laboratory of the University. He is also a member of IEEE Computer Society.

Ying-ping Chen received the B.S. degree and the M.S. degree in Computer Science and Information Engineering from National Taiwan University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in 2004 from the Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA.

He is currently an associate professor in the Department of Computer Science, National Chiao Tung University, Taiwan. His research interests include data grid and MapReduce technologies in distributed computation as well as theories, working principles, and dimensional/facet-wise models in genetic and evolutionary computation.

Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2011, Article ID 204750, 7 pages
doi:10.1155/2011/204750

Research Article

Convergence Time Analysis of Particle Swarm Optimization Based on Particle Interaction

Chao-Hong Chen and Ying-ping Chen

Department of Computer Science, National Chiao Tung University, HsinChu 300, Taiwan

Correspondence should be addressed to Ying-ping Chen, ypchen@cs.nctu.edu.tw

Received 8 August 2011; Revised 7 December 2011; Accepted 21 December 2011

Academic Editor: Peter Tino

We analyze the convergence time of particle swarm optimization (PSO) on the facet of particle interaction. We firstly introduce a statistical interpretation of social-only PSO in order to capture the essence of particle interaction, which is one of the key mechanisms of PSO. We then use the statistical model to obtain theoretical results on the convergence time. Since the theoretical analysis is conducted on the social-only model of PSO, instead of on common models in practice, to verify the validity of our results, numerical experiments are executed on benchmark functions with a regular PSO program.

1. Introduction

Particle swarm optimizer (PSO), introduced by [1, 2], is a stochastic population-based algorithm for solving continuous optimization problems. As shown by [3] and by lots of real-world applications, PSO is an efficient and effective optimization framework. Although PSO has been widely applied in many fields [4–7], understanding of PSO from the theoretical point of view is still quite limited. Most of previous theoretical results [8–18] are derived under the system that assumes a fixed attractor or a swarm consisting of a single particle.

Due to the lack of theoretical analysis on PSO particle interaction, in this paper, we will make an attempt to analyze the convergence time for PSO on the facet of particle interaction. In particular, we will firstly introduce a statistical interpretation of PSO, proposed by [19], to capture the essence of particle interaction. We will then analyze the convergence time based on the statistical model. Finally, numerical experiments will be conducted to confirm the validity of our theoretical results obtained on simplified PSO, the social-only model, in a normal PSO configuration.

In the next section, we will briefly introduce the algorithm of PSO and the statistical interpretation of social-only PSO. In Section 3, we will analyze the convergence time of PSO based on the statistical model. The experimental results are presented in Section 4, followed by Section 5 which concludes this work.

2. Particle Swarm Optimization and the Statistical Interpretation

The social-only model of PSO can be described as pseudocode shown in Algorithm 1. In this paper, we will use boldface for vectors, for example, $\mathbf{X_i}$, $\mathbf{V_i}$. Without loss of generality, we assume that the goal is to minimize the objective function.

According to Algorithm 1, in the beginning, m particles are initialized, where m is the swarm size, an algorithmic parameter of PSO. Each particle contains three types of information: its location ($\mathbf{X_i}$), velocity ($\mathbf{V_i}$), and personal best position ($\mathbf{Pb_i}$). At each generation, each particle updates its personal best position ($\mathbf{Pb_i}$) and neighborhood best position ($\mathbf{Nb}$) according to its objective value. After updating the personal and neighborhood best positions, each particle updates the velocity according to $\mathbf{Pb_i}$ and $\mathbf{Nb}$. In the velocity update formula, w is the weight of inertia which is usually a constant. C_p and C_n are random values sampled from uniform distributions $U(0, c_p)$ and $U(0, c_n)$, where c_p and c_n are called acceleration coefficients. Finally, each particle updates its position according to the velocity and then goes to next generation.

```
procedure SOCIAL-ONLY PSO (Objective function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$)
    Initialize $m$ particles
    while the stopping criterion is not satisfied do
        for $i = 1, 2, \dots, m$ do
            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 $i = 1, 2, \dots, m$ do
            for $j = 1, 2, \dots, n$ do
                $\mathbf{V_i}_j \leftarrow w\mathbf{V_i}_j + C_n(\mathbf{Nb}_j - \mathbf{X_i}_j)$
                $\mathbf{X_i}_j \leftarrow \mathbf{X_i}_j + \mathbf{V_i}_j$
            end for
        end for
    end while
end procedure
```

ALGORITHM 1: Social-only model of PSO.

From the aforementioned brief description, we can already see that particle interaction is a crucial mechanism in the design of PSO. Although there have been previous studies on particle interaction and the PSO behavior, most of these studies were totally based on the assumption of fixed attractors, a false condition for PSO in action. In order to take particle interaction into consideration, we use an alternative view of PSO that regards the whole swarm as a unity. Instead of tracking the movement of each particle, we consider the overall swarm behavior by transforming the state of entire swarm into a statistical abstraction. Furthermore, in order to concentrate on particle interaction, we adopt the social-only model of PSO [20], which does not consider personal best positions.

The statistical interpretation of PSO we use in this paper is modified from [19], summarized in Algorithm 2. In the statistical model, the exact particle locations are not traced but modeled as a distribution θ over $\mathbb{R}^n$. Velocities are viewed as random vectors $\mathcal{V} \in \mathbb{R}^n$. The swarm size m is considered as the number of samples from distribution θ, since the geographic knowledge is embodied in the distribution, the neighborhood attractor can be viewed as the best of the m samples.

Each particle $\mathbf{P_i}$ is considered as a random vector sampled from θ, and the velocity $\mathbf{V_i}$ is sampled from $\mathcal{V}$. The neighborhood attractor can then be defined as $\mathbf{P_a} := \min_{\mathbf{P_i}}\{\mathcal{F}(P_1), \mathcal{F}(P_2), \dots, \mathcal{F}(P_m)\}$. At each generation, $\mathbf{P_i}_j$ is updated as $\mathbf{P_i}_j + w\mathbf{V_i}_j + C(\mathbf{P_a}_j - \mathbf{P_i}_j)$. The next distribution is thus the statistical characterization denoted by functions of the observed values:

$$\begin{aligned} \theta &\longleftarrow \mathcal{T}_p(\mathbf{P_1}, \dots, \mathbf{P_m}), \\ \mathcal{V} &\longleftarrow \mathcal{T}_v(\mathbf{P_1}, \dots, \mathbf{P_m}, \mathbf{V_1}, \dots, \mathbf{V_m}). \end{aligned} \tag{1}$$

Since w is a constant, distribution $\mathcal{V}$ can be removed because given two random vectors $\mathbf{X} \sim \theta$ and $\mathbf{V} \sim \mathcal{V}$, we can simply let θ' be the distribution of $\mathbf{X}' := \mathbf{X} + w\mathbf{V}$.

For simplicity, in this paper, we consider the positions of each dimension of a particle is independently sampled from distribution θ_i. Consider the random variable $X \sim \theta_i$ and let $E[X] = \mu$. If we divide the support of θ_i into s disjoint regions $R_1, \dots, R_s$, such that $\text{Prob}[X \in R_i] = 1/s$ for $i = 1, 2, \dots, s$, and each region is associated with a random variable of velocity $V_i \sim \mathcal{V}_i$. By picking $x_i \in R_i$ for each region, when s is sufficiently large, the swarm can be characterized as

$$\sum_{i=1}^{s} \frac{1}{s}(x_i + V_i) = \sum_{i=1}^{s} \frac{x_i}{s} + \sum_{i=1}^{s} \frac{V_i}{s} \approx \mu + \sum_{i=1}^{s} \frac{V_i}{s}. \tag{2}$$

Each component of $\sum_{i=1}^{s} V_i/s$ can be approximated with a normal distribution by the central limit theorem. As a consequence, normal distributions are a reasonable choice for describing the behavior of the entire swarm. We let the distribution of ith dimension, θ_i, be $N(\mu_i, \sigma_i^2)$, where $N(\mu_i, \sigma_i^2)$ is the normal distribution with mean μ_i and variance σ_i^2. The update of distribution becomes simply by calculating the mean and the variance.

The mean can be calculated by taking the average of updated positions, and the variance is calculated by using the maximum likelihood estimation (MLE). Let $\sigma_{\mathbf{t}}{}_i^2$ and $\mu_{\mathbf{t}}{}_i^2$ be the variance and mean of the ith dimension at the tth generation. Let $y_j = \mathbf{P_j}_i$ and $y'_j = \mathbf{P}'_{\mathbf{j}_i}$ for $j = 1, 2, \dots, m$ and let $y_a = \mathbf{P_a}_i$, $\overline{y} = (1/m)\sum_{j=1}^{m} y'_j$. To estimate the variance of the ith dimension at the $(t+1)$th generation, we use the maximum likelihood estimation (MLE). The likelihood function $L(\sigma_{\mathbf{t}}{}_i^2)$ is defined as the joint probability:

$$\begin{aligned} L(\sigma_{\mathbf{t}}{}_i^2) &:= \prod_{j=1}^{m} \left(\frac{1}{\sqrt{2\pi}\sigma_{\mathbf{t+1}}{}_i^2} \right) \exp\left(\frac{-\left(y'_j - \overline{y}\right)^2}{2\sigma_{\mathbf{t+1}}{}_i^2} \right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma_{\mathbf{t+1}}{}_i^2} \right)^m \exp\left(\frac{-\sum_{j=1}^{m}\left(y'_j - \overline{y}\right)^2}{2\sigma_{\mathbf{t+1}}{}_i^2} \right). \end{aligned} \tag{3}$$

```
procedure STATISTICAL INTERPRETATION OF SOCIAL-ONLY PSO (Objective
Function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$)
    Initialize: $\sigma \leftarrow \sigma_0$, $\mu \leftarrow \mu_0$
    while the stopping criterion is not satisfied do
        for $i = 1, 2, \ldots, m$ do
            for $j = 1, 2, \ldots, n$ do
                $\mathbf{P_i}_j \sim N(\mu_j, \sigma_j^2)$
            end for
        end for
        $\mathbf{P_a} = \min_{\mathbf{P_i}}\{\mathcal{F}(\mathbf{P_i})\}$
        for $i = 1, 2, \ldots, m$ do
            for $j = 1, 2, \ldots, m$ do
                $\mathbf{P}'_{\mathbf{i}j} \leftarrow \mathbf{P_i}_j + C(\mathbf{P_a}_j - \mathbf{P_i}_j)$
            end for
        end for
        $\mu \leftarrow (\sum_{i=1}^m \mathbf{P}'_\mathbf{i})/m$
        $\sigma^2 \leftarrow \mathrm{MLE}(\mathbf{P}'_1, \mathbf{P}'_2, \ldots, \mathbf{P}'_\mathbf{m})$
    end while
end procedure
```

ALGORITHM 2: Statistical model of social-only PSO. Distribution θ is represented by $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$. Acceleration coefficient $C \sim U(0, c)$.

To find ${\sigma_{\mathbf{t+1}}}_i^2$ that maximizes $L({\sigma_\mathbf{t}}_i^2)$, we differentiate $L({\sigma_\mathbf{t}}_i^2)$ with respect to ${\sigma_{\mathbf{t+1}}}_i^2$:

$$\begin{aligned} L'\left({\sigma_\mathbf{t}}_i^2\right) = &-\left(\frac{m}{2}\right)\left(\frac{1}{\sqrt{2\pi}}\right)^m {\sigma_{\mathbf{t+1}}}_i^{-m-2} \\ &\times \exp\left(\frac{-\sum_{j=1}^m \left(y'_j - \overline{y}\right)^2}{2{\sigma_{\mathbf{t+1}}}_i^2}\right) \\ &+ \left(\frac{1}{\sqrt{2\pi}}\right)^m \frac{\sum_{j=1}^m \left(y'_j - \overline{y}\right)^2}{2} {\sigma_{\mathbf{t+1}}}_i^{-m-4} \\ &\times \exp\left(\frac{-\sum_{j=1}^m \left(y'_j - \overline{y}\right)^2}{2{\sigma_{\mathbf{t+1}}}_i^2}\right), \end{aligned} \tag{4}$$

the value of ${\sigma_{\mathbf{t+1}}}_i^2$ that maximizes $L({\sigma_\mathbf{t}}_i^2)$ is $\sum_{j=1}^m (y'_j - \overline{y})^2/m$. As a result, in our model of PSO, the results of MLE is ${\sigma_{\mathbf{t+1}}}_i^2 = \sum_{j=1}^m (y'_j - \overline{y})^2/m$ for $i = 1, 2, \ldots, n$.

3. Convergence Time Analysis

In this section, we will analyze the PSO convergence time based on the aforementioned statistical interpretation of the social-only model. As the first step, we must define the state of convergence. Since, in this work, we regard the entire swarm as a distribution, the state of convergence is then referred to as the variance of the distribution. We define the state of convergence as the variance for every dimension is less than a given value $\epsilon > 0$. By using this definition, we can now start our analysis of PSO convergence time. To estimate the variance after distribution update, we need the following lemma from [21].

Lemma 1. *Let $X_1, X_2, \ldots, X_m \sim N(\mu, \sigma^2)$. Define $S = \sum_{i=1}^m (X_i - \overline{X})^2/(m-1)$, where $\overline{X} = \sum_{i=1}^m X_i/m$. One has $(m-1)S \sim \sigma^2\chi^2_{m-1}$, where χ^2_{m-1} is the chi-square distribution with $m-1$ degrees of freedom.*

With this lemma, we can obtain the following.

Lemma 2. *Given the swarm size m, acceleration coefficient c, and variance of the ith dimension at the tth generation ${\sigma_\mathbf{t}}_i^2$, one has $E[{\sigma_{\mathbf{t+1}}}_i^2] = [(1/3)c^2 - c + 1][(m-1)/m]{\sigma_\mathbf{t}}_i^2$.*

Proof. We know ${\sigma_{\mathbf{t+1}}}_i^2 = \sum_{j=1}^m (y'_j - \overline{y})^2/m$. The expected value is

$$\begin{aligned} &E\left[\frac{1}{m}\sum_{j=1}^m \left(y'_j - \frac{\sum_{k=1}^m y'_k}{m}\right)^2\right] \\ &= \frac{1}{m}E\left[\sum_{j=1}^m \left(y_j + C\left(y_a - y_j\right) - \frac{\sum_{k=1}^m y_k + C\left(y_a - y_k\right)}{m}\right)^2\right] \\ &= \frac{1}{m}E\left[\sum_{j=1}^m \left(\frac{m(1-C)y_j - (1-C)\sum_{k=1}^m y_k}{m}\right)^2\right] \\ &= \frac{1}{m^3}E\left[(1-C)^2\sum_{j=1}^m \left(my_j - \sum_{k=1}^m y_k\right)^2\right] \\ &= \frac{1}{m}E\left[(1-C)^2\right]E\left[\sum_{j=1}^m \left(y_j - \frac{1}{m}\sum_{k=1}^m y_k\right)^2\right]. \end{aligned} \tag{5}$$

Let $S = \sum_{j=1}^m (y_j - (1/m)\sum_{k=1}^m y_k)^2$. Since $y_j \sim N({\mu_{\mathbf{t+1}}}_i, {\sigma_{\mathbf{t+1}}}_i^2)$ for $j = 1, 2, \ldots, m$ and $y_1, y_2, \ldots, y_m$ are i.i.d., by Lemma 1, $S \sim {\sigma_{\mathbf{t}}}_i^2 \chi_{m-1}^2$, and $E[S] = (m-1){\sigma_{\mathbf{t}}}_i^2$. Then, we can obtain

$$\begin{aligned} E[{\sigma_{\mathbf{t+1}}}_i^2] &= \frac{1}{m} E\left[(1-C)^2\right] E\left[\sum_{j=1}^{m}\left(y_j - \frac{1}{m}\sum_{k=1}^{m} y_k\right)^2\right] \\ &= \frac{1}{m} E[1-2C+C^2] E[S] \\ &= \frac{1}{m}\left(\frac{1}{3}c^2 - c + 1\right)(m-1){\sigma_{\mathbf{t}}}_i^2 \\ &= \left(\frac{1}{3}c^2 - c + 1\right)\frac{m-1}{m}{\sigma_{\mathbf{t}}}_i^2. \end{aligned} \tag{6}$$

□

Lemma 2 is derived under the condition that ${\sigma_{\mathbf{t}}}_i^2$ is given. The following lemma will derive the relationship between $E[{\sigma_{\mathbf{t}}}_i^2]$ and $E[{\sigma_{\mathbf{t+1}}}_i^2]$.

Lemma 3. $E[{\sigma_{\mathbf{t+1}}}_i^2] = ((1/3)c^2 - c + 1)((m-1)/m)E[{\sigma_{\mathbf{t}}}_i^2]$.

Proof.

$$\begin{aligned} E[{\sigma_{\mathbf{t}}}_i^2] &= \int_{\sigma^2\in\mathbb{R}^+} E[{\sigma_{\mathbf{t+1}}}_i^2 \mid {\sigma_{\mathbf{t}}}_i^2 = \sigma^2]\operatorname{Prob}\{{\sigma_{\mathbf{t}}}_i^2 = \sigma^2\}d\sigma^2 \\ &= \int_{\sigma^2\in\mathbb{R}^+}\left(\frac{1}{3}c^2 - c + 1\right)\frac{m-1}{m}\sigma^2\operatorname{Prob}\{{\sigma_{\mathbf{t}}}_i^2 = \sigma^2\}d\sigma^2 \\ &\qquad\qquad\text{(by Lemma 2)} \\ &= \left(\frac{1}{3}c^2 - c + 1\right)\frac{m-1}{m}\int_{\sigma^2\in\mathbb{R}^+}\sigma^2\operatorname{Prob}\{{\sigma_{\mathbf{t}}}_i^2 = \sigma^2\}d\sigma^2 \\ &= \left(\frac{1}{3}c^2 - c + 1\right)\frac{m-1}{m}E[{\sigma_{\mathbf{t}}}_i^2]. \end{aligned} \tag{7}$$

□

Now, we can obtain the relationship of convergence time and algorithmic parameters of PSO.

Theorem 4. *Given swarm size m, acceleration coefficient c, ϵ, and $\sigma_{\mathbf{0}}$. Let $h = \max_i\{{\sigma_{\mathbf{0}}}_i^2\}$. One has $E[{\sigma_{\mathbf{t}}}_i^2] < \epsilon$ for $i = 1,2,\dots,n$ when $[(1/3)c^2 - c + 1][(m-1)/m] < 1$ and $t > \log(\epsilon/{\sigma_0}_h^2)/\log([(1/3)c^2 - c + 1][(m-1)/m])$.*

Proof. From Lemma 3, we know

$$\begin{aligned} E[{\sigma_{\mathbf{t}}}_i^2] &= \left(\frac{1}{3}c^2 - c + 1\right)^t\left(\frac{m-1}{m}\right)^t E[{\sigma_{\mathbf{0}}}_i^2] \\ &= \left(\frac{1}{3}c^2 - c + 1\right)^t\left(\frac{m-1}{m}\right)^t {\sigma_{\mathbf{0}}}_i^2. \end{aligned} \tag{8}$$

Since $[(1/3)c^2 - c + 1][(m-1)/m] < 1$, we have

$$\begin{aligned} E[{\sigma_{\mathbf{t}}}_i^2] &= \left(\frac{1}{3}c^2 - c + 1\right)^t\left(\frac{m-1}{m}\right)^t {\sigma_{\mathbf{0}}}_i^2 \\ &< \left(\frac{1}{3}c^2 - c + 1\right)^t\left(\frac{m-1}{m}\right)^t {\sigma_{\mathbf{0}}}_h^2 \\ &< \epsilon. \end{aligned} \tag{9}$$

The last inequality holds because

$$\begin{aligned} &\log\left(\left(\frac{1}{3}c^2 - c + 1\right)^t\left(\frac{m-1}{m}\right)^t {\sigma_{\mathbf{0}}}_h^2\right) \\ &\quad = t\log\left(\left(\frac{1}{3}c^2 - c + 1\right)\left(\frac{m-1}{m}\right)\right) + \log {\sigma_{\mathbf{0}}}_h^2 \\ &\quad < \log\left(\frac{\epsilon}{{\sigma_{\mathbf{0}}}_h^2}\right) + \log {\sigma_{\mathbf{0}}}_h^2 \\ &\quad = \log\epsilon. \end{aligned} \tag{10}$$

□

We have two corollaries immediately from Theorem 4.

Corollary 5. *Given swarm size m, acceleration coefficient c, and level of convergence ϵ such that $[(1/3)c^2 - c + 1][(m-1)/m] < 1$ and $\epsilon < 1$, one has $E[{\sigma_{\mathbf{t}}}_i^2] < \epsilon$ for $i = 1,2,\dots,n$ for $t = O(-\log\epsilon)$.*

Corollary 6. *Given swarm size, m, c, and ϵ such that $[(1/3)c^2 - c + 1][(m-1)/m] < 1$ and $\epsilon < 1$, there exists a constant $c' < 1$ such that for $t = O(-1/\log c'(1 - 1/m))$, one has $E[{\sigma_{\mathbf{t}}}_i^2] < \epsilon$ for $i = 1,2,\dots,n$.*

Corollary 5 reveals the linear relationship between the level of convergence and the convergence time, and the interpretation of Corollary 6 is that when the swarm size is sufficiently large, the effect of enlarging swarm size on the convergence time is not important. In the next section, we will empirically examine the two corollaries with a common practical PSO configuration.

4. Experiments

In this section, we verify the validity of Corollaries 5 and 6 by running standard PSO 2006 downloaded from Particle Swarm Central. We use two objective functions in our experiments:

(i) sphere function [22]:

$$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2, \qquad \mathbf{x} \in [-100, 100]^D, \tag{11}$$

(ii) schwefel's problem 1.2 [22]:

$$f_2(\mathbf{x}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j^2\right), \qquad \mathbf{x} \in [-100, 100]^D. \tag{12}$$

We have $D = 10$ for both $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ in the following experiments.

We firstly examine Corollary 5. The PSO algorithmic parameters are given as $c_p = 1$, $c_n = 1$, $w = 1/(2\ln 2)$, and swarm size = 50. The value of ϵ is varied from 10^{-1} to 10^{-10}. For each value of ϵ, we perform 100 independent runs. For each run, we count the number of generations

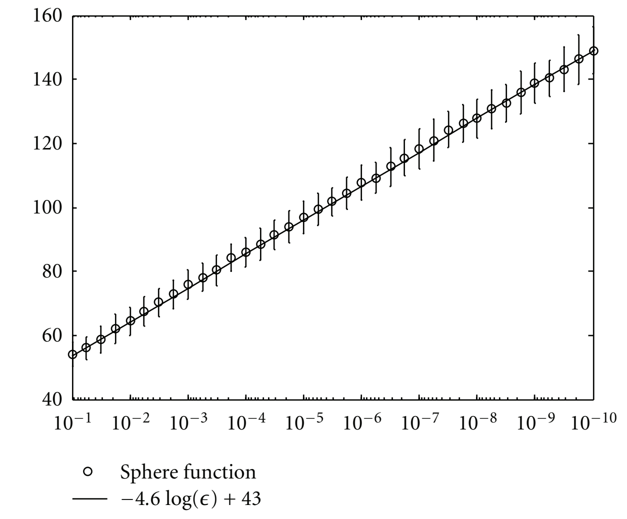


FIGURE 1: Comparison of experimental results and theoretical results from Corollary 5 of $f_1(\mathbf{x})$. The x-axis represents the value of ϵ, and y-axis represents the mean number of generation. The experimental results are very close to $\mathcal{O}(-\log \epsilon)$.

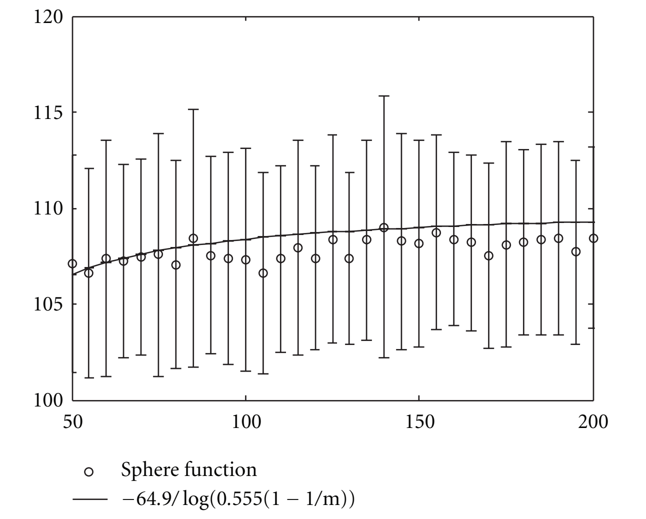


FIGURE 3: Comparison of experimental results and theoretical results from Corollary 6 of $f_1(\mathbf{x})$. x-axis represents the swarm size ranging from 50 to 200, and y-axis represents the mean number of generation. The experimental results are very close to $\mathcal{O}(-1/\log c'(1-1/m))$ with $c' < 1$.

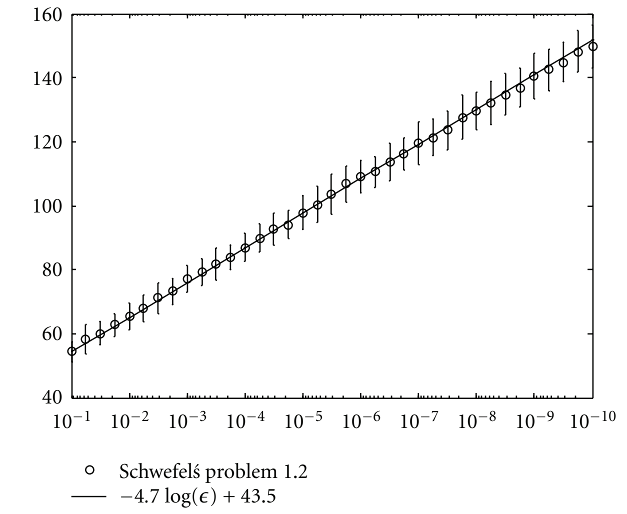


FIGURE 2: Comparison of experimental results and theoretical results from Corollary 5 of $f_2(\mathbf{x})$. The x-axis represents the value of ϵ, and y-axis represents the mean number of generation. The experimental results are very close to $\mathcal{O}(-\log \epsilon)$.

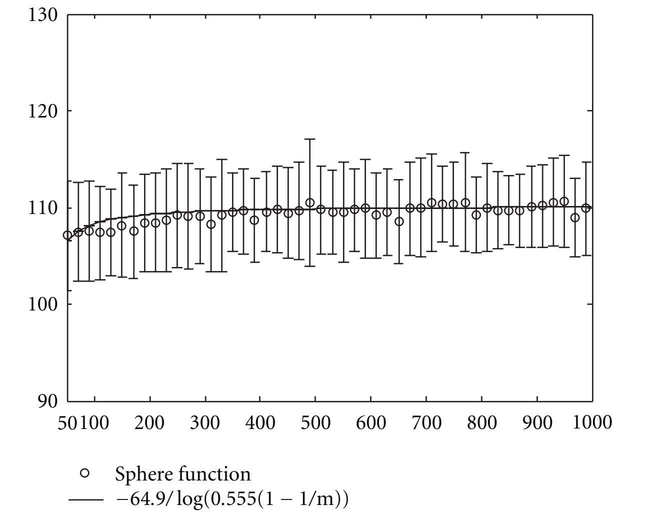


FIGURE 4: Comparison of experimental results and theoretical results from Corollary 6 of $f_1(\mathbf{x})$. x-axis represents the swarm size ranging from 50 to 1000, and y-axis represents the mean number of generation. The experimental results are very close to $\mathcal{O}(-1/\log c'(1-1/m))$ with $c' < 1$.

from initialization to the state in which variances for all dimensions are smaller than ϵ, and we calculate the mean number of generations for the 100 runs.

The comparison of these experimental results and our theoretical results is shown in Figures 1 and 2. From Figure 1, we can see that the experimental results of $f_1(\mathbf{x})$ are very close to $-4.6 \log \epsilon + 43 = O(-\log \epsilon)$, and from Figure 2, the experimental results of $f_2(\mathbf{x})$ are very close to $-4.7 \log \epsilon + 43.5 = O(-\log \epsilon)$. The experimental results agree with our estimation in Corollary 5, in which the value of $-\ln \epsilon$ and the PSO convergence time are linearly related.

After Corollary 5 is empirically verified with the standard PSO, we now examine Corollary 6. The parameters we used in PSO are given as $c_p = 1$, $c_n = 1$, $w = 1/(2 \ln 2)$, and $\epsilon = 10^{-6}$. The swarm size ranges from 50 to 1000 with step 5. For each swarm size, we perform 100 independent runs and record the mean as we did in last experiment.

The comparison of experimental and theoretical results is shown in Figures 3, 4, 5, and 6. From Figures 3 and 4, we can see that the convergence time is close to $-64.9/\log 0.555(1 - 1/m) = \mathcal{O}(-1/\log c'(1 - 1/m))$, where $c' = 0.555$, and in Figures 5 and 6, the convergence time is close to

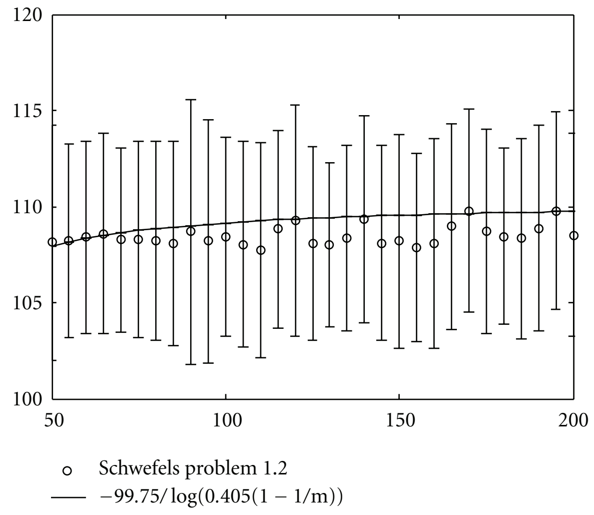


FIGURE 5: Comparison of experimental results and theoretical results from Corollary 6 of $f_2(\mathbf{x})$. x-axis represents the swarm size ranging from 50 to 200, and y-axis represents the mean number of generation. The experimental results are very close to $\mathcal{O}(-1/\log c'(1-1/m))$ with $c' < 1$.

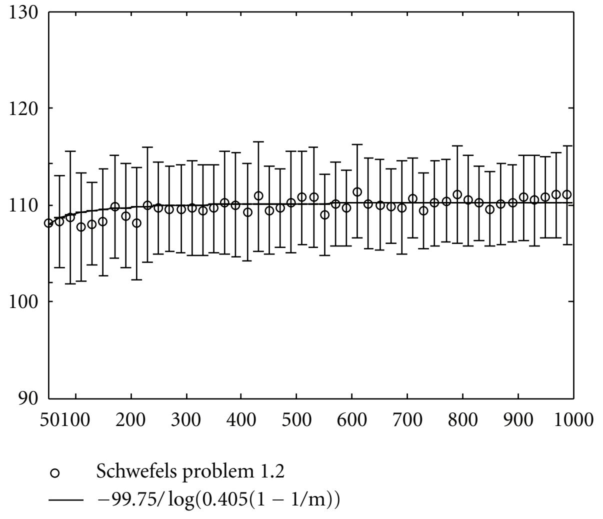


FIGURE 6: Comparison of experimental results and theoretical results from Corollary 6 of $f_2(\mathbf{x})$. x-axis represents the swarm size ranging from 50 to 1000, and y-axis represents the mean number of generation. The experimental results are very close to $\mathcal{O}(-1/\log c'(1-1/m))$ with $c' < 1$.

$-99.75/\log 0.405(1-1/m) = \mathcal{O}(-1/\log c'(1-1/m))$, where $c' = 0.405$. As we can observe from these figures, when the swarm size becomes large, the increase of convergence time is insignificant, confirming our estimation in Corollary 6.

5. Conclusions

In this paper, a statistical interpretation of a simplified model of PSO was adopted to analyze the PSO convergence time. In order to capture the essence of particle interaction, the statistical model adopted in this paper assumed no fixed attractors. The effect of particle interaction was included in our analysis. Our theoretical results revealed the relationship between the convergence time and the level of convergence as well as the relationship between the convergence time and the swarm size. Numerical results, in the standard settings of PSO, were obtained to empirically verify our theoretical results derived with a simplified PSO configuration. The agreement between the experimental and theoretical results indicated the importance of particle interaction in PSO. Consequently, more research effort should be invested into analyzing the working of particle interaction in order to better understand particle swarm optimization.

Some future extensions of this study are now ready to be explored. First of all, the relationship between PSO and the number of dimensions, that is, in the adopted model, the relationship between t and $E[\sigma_{\mathbf{t}(n)}{}^2]$, where $\sigma_{\mathbf{t}(n)}{}^2 = \max\{\sigma_{\mathbf{t}1}^2, \sigma_{\mathbf{t}2}^2, \ldots, \sigma_{\mathbf{t}n}^2\}$. Second, the theoretical analysis conducted in this study is independent of objective functions. In Section 4, we verify the analysis with only two objective functions. More functions of various features and properties, which do not violate the settings of the adopted statistical model, should be used to examine the estimation. Third, the distribution which we used in this paper is the normal distribution. However, there should exist some objective functions that enforce the swarm to distribute according to different distributions. The analysis presented in this paper will fail on those objective functions. As a result, more sophisticated models should be adopted to provide good descriptions of the PSO macrobehavior and to enable researchers to derive more accurate PSO estimations. Finally, the social-only PSO model we adopted in this paper does not take the personal experience into consideration. We also need more sophisticated models to analyze the PSO macrobehavior influenced by the personal experience.

Acknowledgment

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2.

References

[1] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1942–1948, December 1995.

[2] J. Kennedy and R. C. Eberhart, “A new optimizer using particle swarm theory,” in *Proceedings of the 6th International Symposium on Micromachine and Human Science*, pp. 39–43, Nagoya, Japan, 1995.

[3] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’99)*, vol. 3, pp. 1945–1950, 1999.

[4] F. Pan, G. Wang, and Y. Liu, “A multi-objective-based non-stationary UAV assignment model for constraints handling using PSO,” in *In Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC ’09)*, pp. 459–466, Shanghai, China, June 2009.

[5] P. Yan and L. Tang, “PSO algorithm for a scheduling parallel unit batch process with batching,” in *In Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC ’09)*, pp. 703–708, Shanghai, China, June 2009.

[6] T. M. Alkhamis and M. A. Ahmed, “Simulation-based optimization for repairable systems using particle swarm algorithm,” in *In Proceedings of the 37th Conference on Winter Simulation*, pp. 857–861, Orlando, Fla, USA, December 2005.

[7] R. Wrobel and P. H. Mellor, “Particle swarm optimisation for the design of brushless permanent magnet machines,” in *In Proceedings of the IEEE Industry Applications Conference: 41st IAS Annual Meeting*, vol. 4, pp. 1891–1897, Tampa, Fla, USA, October 2006.

[8] J. Kennedy, “The behavior of particles,” in *Proceedings of the 7th International Conference on Evolutionary Programming*, pp. 581–589, 1998.

[9] E. Ozcan and C. K. Mohan, “Analysis of a simple particle swarm optimization system,” *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 8, pp. 253–258, 1998.

[10] E. Ozcan and C.K. Mohan, “Particle swarm optimization: surfing the waves,” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’99)*, pp. 1939–1944, 1999.

[11] M. Clerc and J. Kennedy, “The particle swarm-explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[12] F. van den Bergh, *An analysis of particle swarm optimizers*, Ph.D. thesis, University of Pretoria, 2002.

[13] K. Yasuda, A. Ide, and N. Iwasaki, “Adaptive particle swarm optimization,” in *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1554–1559, October 2003.

[14] Y. L. Zheng, L. H. Ma, L. Y. Zhang, and J. X. Qian, “On the convergence analysis and parameter selection in particle swarm optimization,” in *In Proceedings of the 2nd International Conference on Machine Learning and Cybernetics*, pp. 1802–1807, November 2003.

[15] I. C. Trelea, “The particle swarm optimization algorithm: convergence analysis and parameter selection,” *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[16] F. van den Bergh and A. P. Engelbrecht, “A study of particle swarm optimization particle trajectories,” *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[17] V. Kadirkamanathan, K. Selvarajah, and P. J. Fleming, “Stability analysis of the particle dynamics in particle swarm optimizer,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 245–255, 2006.

[18] M. Jiang, Y. P. Luo, and S. Y. Yang, “Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm,” *Information Processing Letters*, vol. 102, no. 1, pp. 8–16, 2007.

[19] Y. P. Chen and P. Jiang, “Analysis of particle interaction in particle swarm optimization,” *Theoretical Computer Science*, vol. 411, no. 21, pp. 2101–2115, 2010.

[20] J. Kennedy, “Particle swarm: social adaptation of knowledge,” in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC ’97)*, pp. 303–308, April 1997.

[21] M.R. Spiegel, *Schaum’s Outline of Theory and Problems of Probability and Statistics*, Mcgraw-Hill, 1975.

[22] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

Analysis on the Collaboration Between Global Search and Local Search in Memetic Computation

Jih-Yiing Lin and Ying-Ping Chen, *Member, IEEE*

***Abstract*—The synergy between exploration and exploitation has been a prominent issue in optimization. The rise of memetic algorithms, a category of optimization techniques which feature the explicit exploration-exploitation coordination, much accentuates this issue. While memetic algorithms have achieved remarkable success in a wide range of real-world applications, the key to successful exploration-exploitation synergies still remains obscure as conclusions drawn from empirical results or theoretical derivations are usually quite algorithm specific and/or problem dependent. This paper aims to provide a theoretical model that can depict the collaboration between global search and local search in memetic computation on a broad class of objective functions. In the proposed model, the interaction between global search and local search creates a set of local search zones, in which the global optimal points reside, within the search space. Based on such a concept, the quasi-basin class (QBC) which categorizes problems according to the distribution of their local search zones is adopted. The subthreshold seeker, taken as a representative archetype of memetic algorithms, is analyzed on various QBCs to develop a general model for memetic algorithms. As the proposed model not only well describes the expected time for a simple memetic algorithm to find the optimal point on different QBCs but also consists with the observations made in previous studies in the literature, the proposed model may reveal important insights to the design of memetic algorithms in general.**

***Index Terms*—Global search, local search, memetic algorithms, quasi-basin class, subthreshold seeker.**

I. Introduction

OPTIMIZATION, finding the optimal element among a set of feasible ones, is a type of problem commonly encountered in many fields. Many real-world and theoretical problems can be formulated as optimization problems and solved by applying or developing various optimization techniques. Early optimization techniques, such as Newton's method, simplex method, conjugate gradient algorithm, and the like, have been well developed on problems with certain mathematical characteristics. However, as many real-world optimization problems are black-box problems of which *a priori* problem knowledge is not available, the use of meta-heuristics started to prevail. Meta-heuristics are generally population-based algorithms which explore the search space stochastically according to some heuristics. As they are not problem-specific, they have a good chance to perform well on black-box optimization problems. Evolutionary algorithms, particle swarm optimizations, ant colony algorithms, and the like, are some of the renowned meta-heuristics which have been widely adopted.

The generality of meta-heuristics which provides the wide applicability also limits the efficiency of meta-heuristics. When complicated problems are encountered, without taking advantages of problem-specific information given *a priori* or retrieved during optimization, meta-heuristics can merely deliver mediocre performance. As problem-specific heuristics can generally take advantages of problem-specific information, techniques that hybrid general meta-heuristics and problem-specific heuristics have been developed to provide more efficient optimization techniques for more complicated problems. These techniques which employ general meta-heuristics as global search and problem-specific heuristic as local search are commonly referred to as memetic algorithms (MAs). With an appropriate coordination, memetic algorithms cannot only exhibit a good explorative ability as a population-based global search algorithm does but also deliver a good exploitive performance as a local search algorithm does. As a result, memetic algorithms perform better than pure population-based global search algorithms or stand-alone local search algorithms. As the research interests and activities of memetic algorithms thrive, memetic computing has been evolved from hybridization of global search and local search to hybridization with adaptation and has the potential to be applied to computational intelligence [1]. Manifold of successful memetic algorithms in various application domains, ranging from NP-hard combinatorial problems to non-linear programming problems, have been reported [2]. Besides the various application domains mentioned in [2], recent memetic algorithm applications in Cartesian robot control [3], e-learning systems [4], image segmentation [5], feature selection [6], mission management [7], and portfolio selection [8] also demonstrate the efficacy of memetic algorithms in different application domains.

Among these memetic algorithms, in addition to the selection of the global search component and the local search operator, the synergy between global search and local search has always been one of the key design issues. The design of most memetic algorithms follows the seminal studies on memetic algorithms proposed in [9] and [10]. In these studies, the authors observed that memetic algorithms favor infrequent

Manuscript received April 30, 2010; revised September 29, 2010 and March 11, 2011; April 19, 2011. Date of current version September 30, 2011. This work was supported in part by the National Science Council of Taiwan, under Grant NSC 99-2221-E-009-123-MY2.

The authors are with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: jylin@nclab.tw; ypchen@nclab.tw).

Digital Object Identifier 10.1109/TEVC.2011.2150754

starts and long running time of local search. They also proposed several renowned strategies for selecting solution candidates on which the local search operator is applied: the fitness based selection and the diversity based selection. However, with the aids of these guidelines, designing a memetic algorithm for a specific problem still requires considerable time as the optimal design is not only algorithm specific but also problem dependent. To cope with this issue, the concept of systematically adjusting the parameters of local search is proposed [11]. Although this technique is robust, it does not guarantee the best performance. Another line of research is regarding the concept of memes [12]–[14]. In these studies, the local search algorithms, encoded as memes, can adapt to the underlying problem and thus improve the efficiency as the memetic algorithm progresses. This framework is robust as well as efficient with the expense of the learning cost of memes.

In spite of the light shed on the design issue of memetic algorithms by the aforementioned studies, the question of how one can achieve the optimal design of memetic algorithms on a specific problem remains. The key to achieve this ultimate goal apparently include a full awareness of the physics behind the algorithm and the problem. As theoretical studies can help to understand the internal mechanism of algorithms, they can provide important insights to the design issue. Compared to the progress of theoretical studies on evolutionary computation, which is still in its infancy [15]–[23], theoretical studies of memetic algorithms are even scarce. Recent studies [24], [25] investigated the behavior of simple memetic algorithms on several classes of functions. The proposed theoretical models on the demonstrative classes of functions reaffirmed that parameterizing memetic evolutionary algorithms can be extremely difficult. As these theoretical models are developed according to different classes of functions, they are capable of depicting the algorithmic behavior from their respective perspectives on the adopted classes of functions instead of providing a unified principle for the design of memetic algorithms.

The concept of basins of attraction [26] provides another perspective and gives an opportunity to conduct general analysis on memetic algorithms. In [27] and [28], the search space is viewed as a union of basins of attraction, and the optimal allowable local search length of simple memetic algorithms is theoretically estimated. A similar concept, quasi-basins defined by the subthreshold seeker, was adopted to prove the searchability of general functions [29] and to investigate the subthreshold seeking behavior [30].

In this paper, we aim to establish a theoretical model that can depict the collaboration between global search and local search in memetic computation on a wide range of problems. To achieve this, we propose the concept of local search zones which are the regions that local search exploits. In this perspective, these local search zones are defined by the landscape of the problem as well as the collaboration between global search and local search. As local search zones are generally not easy to assess, we adopt quasi-basins to estimate local search zones and define the quasi-basin class (QBC) which categorizes problems by their quasi-basin distributions as the basis on which memetic algorithms are investigated. Then, we analyze the performance of the subthreshold seeker, which is regarded as a representative archetype of memetic algorithms, to develop a theoretical model for the global-local search collaboration in memetic computation. The derived theoretical model can describe how the distribution of local search zones and the efficiency of the global search algorithm and the local search algorithm are related to the expected time for a memetic algorithm to find the optimal solution. Because this model, empirically verified, is consistent with the observations made in many previous studies in the literature, it may be considered valid for representing various memetic algorithms on a wide range of problems and may give important insights to the future design of advanced memetic algorithms.

The rest of this paper is arranged in the following manner. Section II gives a survey on the current progress of analysis on memetic algorithms and elaborates the need of a general theoretical model which can describe the collaboration between global search and local search in memetic computation on a broad range of problems. Section III expounds the fundamental concepts on the analysis of memetic algorithms and provides the definitions of our framework to form the basis for further derivation. As a memetic algorithm comprises global search and local search, we first analyze the global search component of the subthreshold seeker and discuss how this analysis is related to the behavior of common global search algorithms in Section IV. Based on the analysis of global search and the concept of QBC, we derive and empirically verify the formula that describes the behavior of the subthreshold seeker working with local search operators of different efficiency on various QBCs in Section V. After the empirically verifying the proposed model, we expound how our model can represent the general behavior of memetic algorithms and discuss possible extensions and future work of the proposed model in Section VI. Finally, we recap the significance of our model and conclude this paper in Section VII.

II. Background

Designing a memetic algorithm requires not only selecting a global search mechanism as well as a local search operators but also establishing a subtle coordination to exhibit the vantage of both ends. Hart [9] in his seminal study for designing efficient memetic algorithms investigated the following four questions on continuous optimization problems.

1) How often should local search be applied?
2) On which solutions should local search be used?
3) How long should local search be run?
4) How efficient does local search need to be?

In his framework, he noted that the memetic algorithms that employ elitism will be most efficient with large population sizes and infrequent local search. He also proposed two strategies, fitness based selection and diversity based selection, for selecting solution candidates to apply local search. He concluded that these two strategies help much. Land [10] extended Hart's study to combinatorial domains. In his study, he adopted steady state genetic algorithms as global search and proposed a local search potential based strategy in selecting local search candidates. The local search potential strategy turned out to be not very useful. Yet, he observed that his steady state memetic algorithm favored smaller rates and longer runtime for local search, consistent with Hart's study.

Although limited to specific problems, the studies of Hart and Land gave some insights to the first three questions and have inspired the successive memetic algorithms in a wide variety of applications. The concepts of selecting the best or some qualified individuals for local search which resemble the fitness based selection have been adopted in [31]–[33]. The steady state memetic algorithm with adaptive local search has been applied in [34]–[36], while other studies exhibit the vantage of utilizing the diversity information in their design of memetic algorithms [37], [38]. Investigations into the balance between global search and local search for some applications available in the literature also accord with Hart's and Land's observations [39], [40]. Despite that the accordance of these results reveals some essential design principles of efficient memetic algorithms, designing a memetic algorithm still requires a considerable amount of effort due to the lack of detailed knowledge on how the key mechanism of memetic algorithms, the synergy between global search and local search, working on the underlying problem. An interesting technique of adapting local search intensity in a simulated annealing way was proposed to cope with the MA parameterizing issue [11]. More robust than the fixed local search intensity setting, this method still requires a range setting and does not guarantee the best performance.

In addition to the parameterizing issue caused by using memetic algorithms to handle different problems, the efficiency of a local search operator is particularly problem dependent. [41] provided a landscape analysis for memetic algorithms. Following this, the concept of memes [12]–[14] was proposed. In these frameworks, the local search component is designed to adapt to the underlying problem as the optimization progresses. These meme evolving or learning memetic algorithms are robust regardless of the underlying problem and efficient. Recent studies [42], [43] have also proposed several metrics to assess the improvement of applying a local search algorithm on a problem.

Furthermore, theoretical analysis has always been a prevalent way to provide clues to the design of algorithms. For continuous problems, convergence analysis is widely adopted in performance assessment for evolutionary computation [16], [19], [22]. For discrete problems, the (1+1) evolutionary algorithm (EA) has been widely adopted in theoretical analysis on evolutionary algorithms [15], [17], [18], [20], [21], [23]. The (1+1)-EA is a rather simple algorithm with one individual and an evolutionary operator flipping each bit of the individual with a uniform probability. Following these studies, the theoretical analysis of memetic algorithms starts from the (1+1)-MA and goes to the (μ+λ)-MA [24], [25]. On three discrete functions, Sudholt investigated the behavior of the (1+1)-MA and the (μ+λ)-MA, and these studies reaffirmed the parameterizing of memetic algorithms is extremely hard.

Theoretical models developed in this way are capable of providing different perspectives, according to the adopted classes of functions, to analyze a memetic algorithm. Reference [44] illustrated that different problems favor different population sizes, while [45] and [46], which investigated the effect of recombination operators, provided counter perspectives. The issue of such an analysis technique is that the derived theoretical behavior is naturally confined and largely determined by the adopted objective functions. As an undesirable result, the different conclusions obtained from various theoretical models cannot form a unified guideline to the design of algorithms.

Another line of analysis involves the concept of basins of attraction. The basin of attraction of a local optimum is the set of points in the search space such that a local search process starts from any member within a basin will eventually find the local optimum in that basin [26]. In this line of research, the search space is a union of basins of attraction. References [27] and [28] adopted this concept to estimate the optimal local search length. In these papers, basins of attraction in the search space are categorized into two types, in which target solutions can or cannot be reached. The optimal local search length is estimated via acquiring the probability of hitting the former basins.

A closely related concept, quasi-basin defined by subthreshold seeker, was introduced by [29] in investigating searchable functions in which the No Free Lunch theorem does not hold. The submedian seeker which starts local search when hitting a point with a submedian value and turns to do random search when hitting a point with a supermedian value was considered. By applying the submedian seeker to functions with a certain degree of self-similarity, that the functions exhibiting self-similarity are searchable was proved. Whiteley and Rowe further proposed the subthreshold seeker, a generalized 1-D submedian seeker, and investigated its seeking behavior [30]. In their work, the subthreshold seeking behavior, the ratio of the sampled subthreshold points to superthreshold points, was used as a performance index. Their theoretical analysis detailed the conditions under which the subthreshold seeker could outperform random search and showed that a higher bit-precision could improve the performance.

Finally, in this paper, we aim to provide a general model for the collaboration between global search and local search in memetic algorithms on a broad class of problems. The proposed model will describe how the expected performance of a memetic algorithm is related to the efficiency of the local search operator, the landscape of the problem, and the collaboration between global search and local search. In order to achieve this goal, we propose the concept of local search zones. Local search zones are the regions which local search prefers and the global optimal point resides in. As generally local search zones are not easy to assess, we adopt the idea of quasi-basins to estimate local search zones and define the QBC to categorize problems according to their quasi-basin distributions. The subthreshold seeker, taken as a representative archetype of memetic algorithms, is analyzed over different QBCs as a general theoretical model for memetic algorithms. Thus, the proposed model can depict the essence of the collaboration between global search and local search in memetic algorithms on various problems and may shed light on the design of memetic algorithms.

III. Quasi-Basin Classes and Subthreshold Seeker

In this section, we introduce the concept of local search zones and give definitions to the fundamental terminologies of our framework. The concept of the local search zones is described based on the formal definitions of the search process of an algorithm on a problem and the search space viewed by a search process. Then, based on the concept of local search zones, we introduce the QBC and the generalized subthreshold seeker on which the theoretical analysis is based.

A. Local Search Zones

The task to handle an optimization problem is to optimize a given objective function $f : \mathcal{X} \rightarrow \mathcal{Y}$. For convenience, we specify our optimization goal as to find a point $x^* \in \mathcal{X}$ with the minimum value $y^* \in \mathcal{Y}$. We assume that both $\mathcal{X}$ and $\mathcal{Y}$ are finite sets. Such an assumption makes a practical sense because optimization problems are generally numerically solved on digital computers. In this paper, for simplifying the derivation, we also assume that every function maps different $x \in \mathcal{X}$ to different $y \in \mathcal{Y}$. In order to formally describe a search process of an algorithm on a function, we adopt part of the terminologies defined in [47] as the following definitions.

Definition 1 (Search Process): Given two finite sets $\mathcal{X}$ and $\mathcal{Y}$:

1) A trace of length m is a sequence $T_m := ((x_i, y_i))_1^m = ((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ with distinct x_i. "$x \in T_m$" denotes that $x = x_i$ for some $i \in \{1, 2, \ldots, m\}$. Let T_0 be the empty sequence and $\mathcal{T}^\ell$ be the set containing all the traces of a length smaller than or equal to ℓ.
2) Let A_T, where $T \in \mathcal{T}^{|\mathcal{X}|-1}$, be a random variable over $\mathcal{X}$ satisfying that $Prob\{A_T = x\} = 0$ for all $x \in T$. An algorithm A is a collection of such random variables, i.e., $A = \{A_T \mid T \in \mathcal{T}^{|\mathcal{X}|-1}\}$.
3) The search process of A on f, $S(A, f)$, is a stochastic process $(X_i, Y_i := f(X_i))$ over $\mathcal{X} \times \mathcal{Y}$ defined by $X_1 \sim A_{T_0}$ and $X_{k+1} \sim A_{(X_i,Y_i)_1^k}$.

For generality, we interpret the search space viewed by a memetic algorithm as a graph. Since a local search algorithm usually starts from a candidate solution and iteratively moves to a neighbor solution, the local search algorithm defines the neighborhood of a candidate solution in the search space viewed by the memetic algorithm which utilizes it. Thus, we define the search space viewed by a memetic algorithm as a graph of which the vertices are the set of points of $\mathcal{X}$ and the edges are the set of pairs of points connected by the local search algorithm of the memetic algorithm as follows.

Definition 2 (Search Space): Given a memetic algorithm MA, a function f, and LS, the local search algorithm adopted by MA. Let $N_{LS}(v)$ denote the neighborhood of a vertex v defined by LS. The search space viewed by MA on f can be represented by a graph $G = (V, E)$, where $V(G) := \mathcal{X}$ and $E(G) := \{\langle v_i, v_j\rangle \mid v_j \in N_{LS}(v_i), \forall i, j\}$.

In the rest of this paper, the terms $\mathcal{X}$ and $V(G)$ are used exchangeably. Now, with all these fundamental terminologies, we can formally define the local search zone as follows.

Definition 3 (Local Search Zone): Given a search process of a memetic algorithm MA on function f, $S(MA, f)$, and LS, the local search algorithm adopted by MA:

1) The local search points of a search space G viewed by $S(MA, f)$ are defined as the set of points $S_{LSZ} = \{v \mid E[Pr(X_{k+1} = u, u \in N_{LS}(v) | v \in T_k, u \notin T_k)] > 0.5, \forall v \in V(G)\}$.[1]
2) A local search zone LSZ is defined as a maximal subset in S_{LSZ} such that there exists a path[2] between all the pairs of vertices.
3) The size of local search zones is denoted by $|S_{LSZ}|$.

By this definition, a local search point is a vertex which if is visited by MA, MA would tend to visit one of its unvisited adjacent vertices in the future, and the local search zones are where the local search points reside. In other words, the local search zones are where local search prefers when a memetic algorithm is applied. In our perspective, the distribution of the local search zones has a great influence on the performance of a memetic algorithm. As the local search of practical memetic algorithms favors the points that have high potential to lead to the optimal point, fitness-relevant and diversity-relevant criteria are adopted. These criteria are somehow dynamic, complicated, and difficult to analyze. Abstracting the exploration behavior of global search and the exploitation behavior of local search, we consider fitness as the prime index of the potential to find the optimal point regardless of the diversity-relevant metrics which are often auxiliary for diversity maintenance. Thus, the local search zones can be estimated by zones consisting of qualified high-fitness points. Based on this way of thinking, we define the QBC to represent different problem classes which possess different local search zone distribution. Then, we take the subthreshold seeker as a representative archetype of memetic algorithms and analyze its behavior on various QBCs to develop a general theoretical model for the core mechanism of memetic algorithms.

B. Quasi-Basin Classes

The QBC conceptually defines problem classes according to the number of local search zones and the size of local search zones. To define QBC, we first define the quasi-basin (QB) as follows.

Definition 4 (QB):

1) For any function f, function value $\beta_m(f)$, defined as $\beta_m(f) := \min\left\{\arg_y \{|\{x \in \mathcal{X} \mid f(x) \leq y\}| = m\}\right\}$, denotes a threshold, and there are $m - 1$ points with an objective value less than $\beta_m(f)$.
2) For any function f, the set that contains all the points with an objective value less than $\beta_m(f)$ is defined as $S_m(f) := \{x \in \mathcal{X} | f(x) \leq \beta_m(f)\}$.
3) Given a graph G, for any function $f : V(G) \rightarrow \mathcal{Y}$, a quasi-basin QB is defined as a maximal subset in $S_m(f)$ such that there is a path between all the pairs of vertices.

[1]The $E[]$ notation indicates the expected value of $Pr(X_{k+1} = u, u \in N_{LS}(v) | v \in T_k, u \notin T_k)$ over all $k \in |\mathcal{X}|$ and all possible T_k which containing v and not containing u.

[2]A path in a graph is a sequence of vertices such that from each vertex there exists an edge to the next vertex in the sequence except for the last one.

As generally the points residing in quasi-basins are better than the other points in the search space and are favored by fitness-relevant local search criterion, $S_m(f)$ conceptually estimates the set of points residing in the local search zones with a size m while a quasi-basin QB can be regarded as a local search zone in the search space. Based on the fundamental definitions, we define the QBC and the uniform quasi-basin class (uQBC) in Definitions 5 and 6.

Definition 5 (QBC): Given a graph G and a co-domain $\mathcal{Y}$, the corresponding discrete QBC with b distinct quasi-basins and m subthreshold vertices is defined as

$$\mathcal{Q}(G, \mathcal{Y}, m, b) := \\ \{f : V(G) \to \mathcal{Y} \mid S_m(f) = \bigcup_{i=1}^{b} QB_i, \bigcap_{i=1}^{b} QB_i = \emptyset, \\ |QB_i| \geq 1, 1 \leq i \leq b\} .$$

Definition 6 (uQBC): Given a graph G and a co-domain $\mathcal{Y}$, the corresponding uniform discrete QBC with b distinct quasi-basins and m subthreshold vertices is defined as

$$\mathcal{Q}_u(G, \mathcal{Y}, m, b) := \\ \{f : V(G) \to \mathcal{Y} \mid S_m(f) = \bigcup_{i=1}^{b} QB_i, \bigcap_{i=1}^{b} QB_i = \emptyset, \\ \left\lfloor \frac{m}{b} \right\rfloor \leq |QB_i| \leq \left\lceil \frac{m}{b} \right\rceil, 1 \leq i \leq b\} .$$

The QBC defines a class of problems with the points of m smallest function values distributed among b distinct quasi-basins. Thus, we can categorize problems according to their distribution of quasi-basins conceptually mapping to the distribution of local search zones. The uniform QBC further restricts the sizes of quasi-basins to be uniform. Note that m is naturally restricted to be less than or equal to $|\mathcal{X}|$ and greater than or equal to b. b is a positive integer which is less than the minimum of m and $|\mathcal{X}| - m$.

C. Subthreshold Seeker

Global search and local search of the subthreshold seeker are coordinated by the threshold θ. A subthreshold seeker globally searches by sampling the space uniformly at random (u.a.r.) until it encounters a subthreshold point, a point with a function value lower than the threshold. When this event occurs, the subthreshold seeker starts local search to exploit the quasi-basin, a maximal set of connected subthreshold points, in which the subthreshold point resides. The local search part keeps visiting the neighbors of the current vertex until it could no longer walk on a subthreshold vertex. After local search in the quasi-basin is done, the subthreshold seeker continues global search until another subthreshold pointer is encountered. The subthreshold seeker will continue switching between global search and local search until the stopping criterion is met. Fig. 1 illustrates how the subthreshold seeker proceeds. In Whitley and Rowe's work, their subthreshold seeker was applicable only to 1-D functions. In our present work, we generalize their subthreshold seeker to more dimensions by utilizing the graph representation in the definition of search space for more general applications.

```
procedure SUBTHRESHOLD-SEEKER($\mathcal{X}$, $\mathcal{Y}$, $N : \mathcal{X} \to 2^{\mathcal{X}}$, $f : \mathcal{X} \to \mathcal{Y}$, $\theta$, $s$)
    while the stopping criterion is not satisfied do
        if Queue is not empty then
            $x \leftarrow$ Queue.pop();
        else
            Select $x$ from $\mathcal{X}$ u.a.r.
        end if
        if $f(x) \leq \theta$ then
            Queue.push($N_s(x)$)
        end if
    end while
end procedure
```

Fig. 1. Generalized subthreshold seeker.

The notation $N_s(x)$ denotes the set of virtual neighbors of vertex x defined by the local search parameter s. When $s = 1$, all the neighbors of vertex x in G defined by the local search operator will be visited. In other words, for $N_1(x)$, all the points in the quasi-basin under local search will be eventually visited. In the rest of this paper, we refer to the exhaustive local search as the local search with the local search parameter $s = 1$. When $s > 1$, only $1/s$ of the points in the quasi-basin will be visited in one local search run. To simulate the effect of this parameter, we manipulate the local search to have a step size s. Thus, the virtual neighbors of vertex x are those vertices who are s distance away from x. Here, s distance refers to the length of a path consisting s edges on the graph. Note that this subthreshold seeker does not sample visited points to avoid the performance declination caused by repeated sampling.

IV. Stochastic Global Search Time

Before we start to analyze the collaboration between global search and local search in the subthreshold seeker, we first investigate the behavior of the global search part employed by the subthreshold seeker. In this section, we theoretically and empirically analyze the behavior of the global search part with respect to the number of subthreshold points m and the number of quasi-basins b. We will derive the expected number of visited points required by the random search, the global search part, to find the first subthreshold point. The expected number of visited points is referred to as the expected first global search time $E(T_\theta)$, where θ is $\beta_m(f)$. Then, we will further approximate the expected kth global search time. The theoretical results will be empirically verified, and the discussion on the implication of the model will be presented.

A. First Global Search Time

In this section, we estimate the expected number of visited points for global search to find the first point x with $f(x) \leq \theta$, referred to as the first global search time T_θ. The first global search time can be interpreted in the following manner. Since $\theta = \beta_m(f)$, the number of points with their function values less than or equal to θ is m. Let N be the size of $\mathcal{X}$. As global search is uniform random sampling without repetition, the search space is of size N and contains m desired points, the probability for q visited points to contain exactly one subthreshold point follows the hypergeometric distribution with parameters N, m, and q

$$P(X = 1; N, m, q) = \frac{\binom{m}{1}\binom{N-m}{q-1}}{\binom{N}{q}} .$$

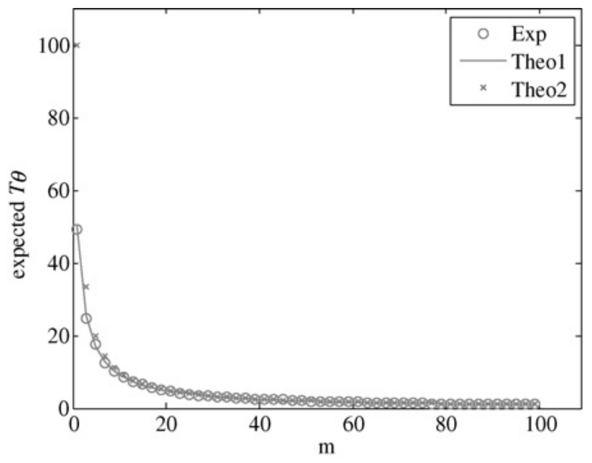


Fig. 2. Expected T_θ with respect to m when $N = 100$. Exp represents the actual average T_θ over 1000 independent simulation runs. Theo1 represents the theoretical expected T_θ of non-repeated uniform random sampling, and Theo2 represents the theoretical expected T_θ of uniform random sampling that allows us to sample visited points.

The probability to hit a subthreshold point at the qth visited points is therefore

$$\frac{1}{q}P(X = 1; N, m, q)\ .$$

Let $E(T_\theta)$ be the expected first global search time. We have

$$\begin{aligned} E(T_\theta) &= \sum_{i=1}^{N-m+1} i\ \frac{1}{i}P(X = 1; N, m, i) \\ &= \sum_{i=1}^{N-m+1} \frac{\binom{m}{1}\binom{N-m}{i-1}}{\binom{N}{i}} \\ &= m \sum_{i=1}^{N-m+1} \frac{i}{N} \prod_{j=0}^{i-2} \frac{N-m-j}{N-1-j}\ . \end{aligned} \quad (1)$$

Fig. 2 illustrates the expected value of T_θ with respect to m when $N = 100$. In this figure, we compare (1) (the solid line, Theo1) with N/m (the crosses, Theo2) and the average first global search time in 1000 independent simulation runs (the circles, Exp). The N/m is the expected T_θ for allowing sampling visited points which is obviously an upper bound of (1). In the figure, we can find that (1) consists of the empirical result perfectly, while the trend of N/m gradually converges toward the other two. For non-repeated random sampling, half of points in the search space are expected to be visited before finding the minimum point. As m increases, indicating that $S_m(f)$ contains more points, time to meet a point in $S_m(f)$ decreases rapidly regardless of whether or not sampling visited points is allowed. It indicates that although finding several specific points in a search space via random search takes a considerable amount of time, finding a point in a small but large enough set can be attained within a relatively shorter time.

Fig. 3 illustrates the differences among the actual T_θ averaged over 1000 runs, and the two theoretical expected T_θ in ratio. The circle represents the difference between the empirical result and N/m in ratio with respect to the empirical result, and the cross represents that between (1) and N/m. From this figure we can find that (1) can be approximated by N/m as it only deviates significantly from (1) when m is

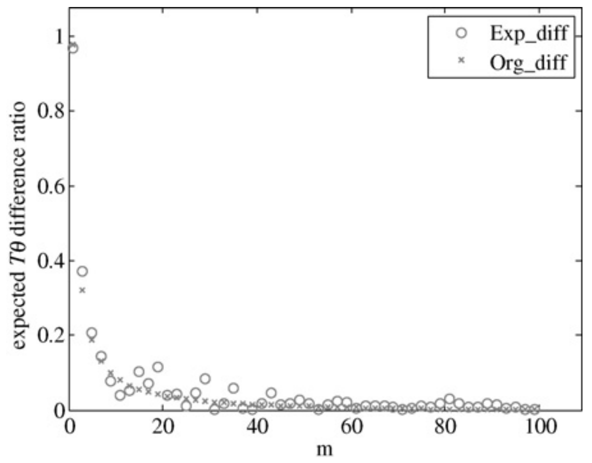


Fig. 3. Difference ratio of the expected T_θ with respect to m when $N = 100$. The Exp_diff represents the difference ratio between the actual average T_θ and N/m. The Org_diff represents the difference ratio between the theoretical expected T_θ and N/m.

rather small. As (1) is a complicated formula and difficult to analyze, we approximate the expected T_θ with N/m.

B. kth Global Search Time

In this section, we further measure the expected time for global search to find a subthreshold point after $k-1$ runs of local search have been executed. In other words, we estimate the time for the kth global search. As the local search frequency and the global search frequency are related to the landscape of the problem, for simplicity, we derive the model on the uniform quasi-basin class $\mathcal{Q}_u(G, \mathcal{Y}, m, b)$. All the problems in this class have their quasi-basin sizes fixed to $\lfloor m/b \rfloor$ or $\lceil m/b \rceil$. Because each local search run in a quasi-basin will eventually visit about $1/s$ of the points in the quasi-basin, each local search run will visit $\lfloor m/bs \rfloor$ or $\lceil m/bs \rceil$ points. For convenience of derivation, we adopt m/bs instead of $\lfloor m/bs \rfloor$ or $\lceil m/bs \rceil$ for the number of points visited by a local search run. For non-revisit search, since the first global search time is approximated as N/m, when in the second global search run, there will be $N - N/m - m/bs$ unvisited points and $m - m/bs$ unvisited subthreshold points, the time required for the second global search run is

$$\frac{N - \dfrac{N}{m} - \dfrac{m}{bs}}{m - \dfrac{m}{bs}}\ .$$

The ith global search time is denoted as F_i, where i is referred to as the number of global search runs. We have

$$\begin{aligned} F_1 &= \frac{N}{m} \\ F_2 &= \frac{N - F_1 - \dfrac{m}{bs}}{m - \dfrac{m}{bs}} \\ F_3 &= \frac{N - (F_1 + F_2) - \dfrac{2m}{bs}}{m - \dfrac{2m}{bs}} \end{aligned}$$

$$F_k = \frac{N - \sum_{i=1}^{k-1} F_i - \dfrac{(k-1)m}{bs}}{m - \dfrac{(k-1)m}{bs}} . \tag{2}$$

Fig. 4 illustrates the global search time, estimated by (2), with respect to the number of global search runs for different distributions of quasi-basins. Fig. 4, with $N = 1000$, $m = 10$, $b = 10$, and $s = 1$, represents a case of a scarce small quasi-basin distribution. In this case, the global search time is large and does not change much as the number of global search runs increases. The second case illustrated in Fig. 4 represents a case of a scarce large quasi-basin distribution with $N = 1000$, $m = 900$, $b = 10$, and $s = 1$. The global search time is small and slightly increases as the number of runs increases. The last case illustrated in Fig. 4 represents a case of fully uniform distributed quasi-basins with $N = 1000$, $m = 500$, $b = 500$, and $s = 1$. In this case, the global search time is also small and slightly decreases as the number of runs increases.

As the QBC only defines the number of subthreshold points and the number of quasi-basins, local search of the subthreshold seeker can be considered as stochastic non-repeated sampling in the set of subthreshold points. Since the minimum resides in $S_m(f)$ with m points, for the stochastic non-repeated sampling, it is expected to sample $(m+1)/2$ points before the minimum point can be found. In the uniform QBC, each quasi-basin is about the same size, $\lfloor m/b \rfloor$ or $\lceil m/b \rceil$, and a local search run visits $\lfloor m/bs \rfloor$ or $\lceil m/bs \rceil$ of the points in a quasi-basin. It is expected to require $k = \left\lceil \dfrac{(m+1)/2}{m/bs} \right\rceil \approx \lceil bs/2 \rceil$[3] local search runs to find the minimum, which implies $k = \lceil bs/2 \rceil$ global search runs are required. Thus, the final global search time is

$$F_f = \frac{N - \displaystyle\sum_{i=1}^{\lceil bs/2 \rceil - 1} F_i - \dfrac{(\lceil bs/2 \rceil - 1)m}{bs}}{m - \dfrac{(\lceil bs/2 \rceil - 1)m}{bs}} . \tag{3}$$

We are now ready to calculate the upper bound for the last global search run. When bs is even, the last global search run requires

$$\begin{aligned} F_f &< \frac{bsN - (\lceil bs/2 \rceil - 1)m}{bsm - (\lceil bs/2 \rceil - 1)m} \\ &= \left(\frac{2bs}{bs+2}\right)\frac{N}{m} - \left(\frac{bs-2}{bs+2}\right) . \end{aligned} \tag{4}$$

When bs is odd, the last global search run requires

$$\begin{aligned} F_f &< \frac{bsN - (\lceil bs/2 \rceil - 1)m}{bsm - (\lceil bs/2 \rceil - 1)m} \\ &= \left(\frac{2bs}{bs+1}\right)\frac{N}{m} - \left(\frac{bs-1}{bs+1}\right) . \end{aligned} \tag{5}$$

The lower bound for the last global search run can be derived as follows:

$$\begin{aligned} F_f &> \frac{N - \sum_{i=1}^{k-1} F_i}{m} \\ &> \frac{N - N/2}{m} = \frac{1}{2}\frac{N}{m} . \end{aligned} \tag{6}$$

Both (4) and (5) indicate that the final global search time would be no more than twice of the amount of the first global search. On the other hand, (6) shows that the final global

[3]For simplicity, we approximate $(m+1)/2$ with $m/2$.

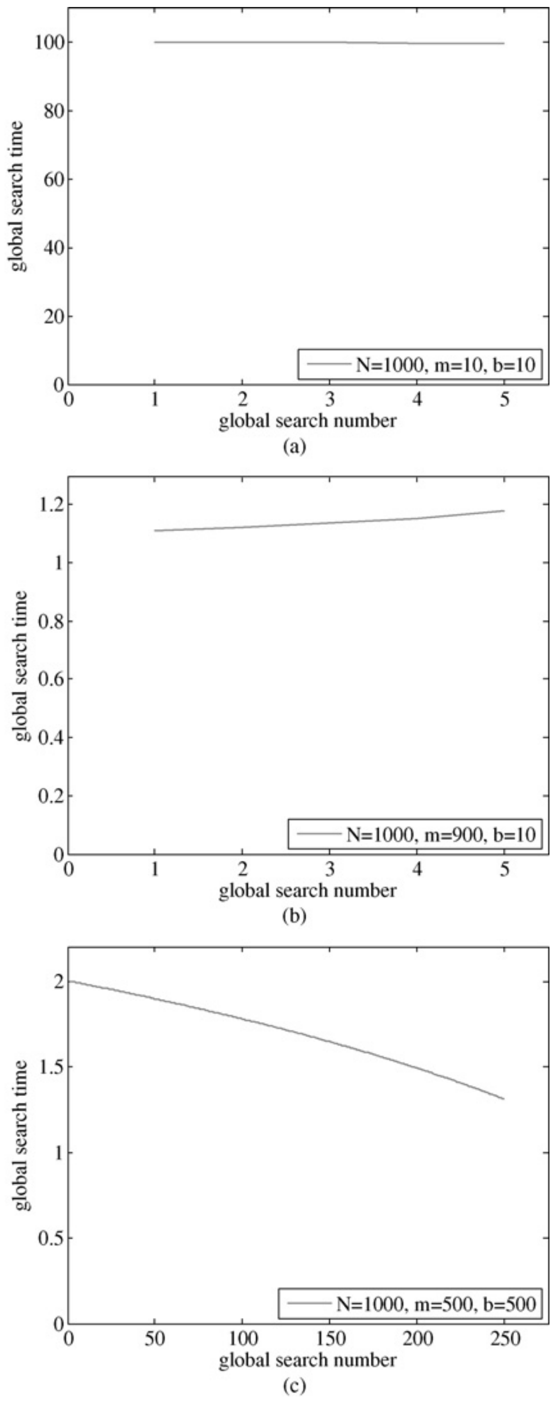


Fig. 4. Global search time with respect to the number of global search runs when the exhaustive local search is applied. (a) Case of a scarce small quasi-basin distribution. (b) Case of a scarce large quasi-basin distribution. (c) Case of fully uniform distributed quasi-basins.

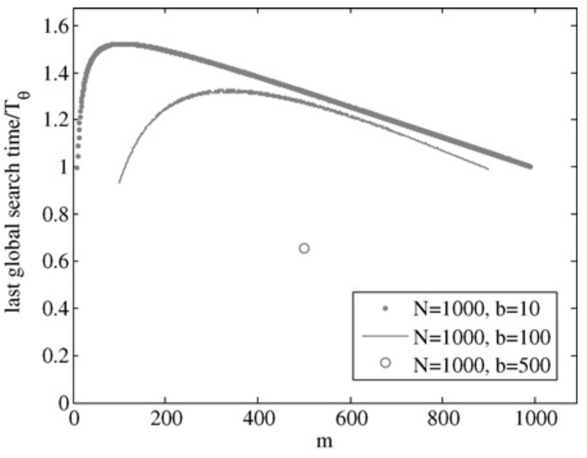


Fig. 5. Last global search time divided by the first global search time, T_θ, with respect to m when the exhaustive local search is applied.

search time would be greater than half of the amount of the first global search. Fig. 5 illustrates the last global search time, F_f, divided by the first global search time, T_θ, with respect to m when the exhaustive local search is applied. The last global search times of $b = 10$ and $b = 100$, initially increase as m increases and reach a peak, followed by gradually degradation. The smaller b is, the smaller m the peak appears at with a greater peak value. Generally, when the number of quasi-basins is considerably large, the smaller the last global search time is. Overall, for $N = 1000$ the last global search time is within 0.6 to 1.6 times of T_θ. As indicated in Fig. 4, the variation of the global search time with respect to the number of global search runs is approximately linear. Thus, we can approximate the average global search time as the average of the first global search time and the last global search time. The resultant upper bound and lower bound of the approximated average global search time are then 0.75 and 1.5 times of T_θ.

C. Discussion

Overall, in this section, we can see that the expected global search time to hit a subthreshold point in local search zones is inversely proportional to the size of local search zones in the search space. Because the uniform random search is employed as the global search component, such results illustrate a baseline behavior of global search in common definitions. It can be observed that when the size ratio between local search zones and the whole search space is very small, the expected global search time will be immensely long because finding a local search zone is very difficult. If the ratio is not very small and permits an acceptable probability to be hit by global search, the expected global search time will drop dramatically. In this case, since the size of local search zones is still small, the local search operator requires a relatively short time to find the optimum solution.

V. Subthreshold Seeker on QBC

In this section, we formulate the expected evaluation time for a subthreshold seeker on a $\mathcal{Q}(G, \mathcal{Y}, m, b)$ as the sum of expected total global search time and the expected total local search time. The expected total global search time is the product of the expected time for global search to enter a local search zone and the expected number of global search runs. The expected total local search time is merely the expected time for local search to find the global optimal points among the local search zones which is proportional to the size of the local search zones in the search space. In this manner, the derived formula can depict how the collaboration between global search and local search influences the performance of memetic algorithms. Then, we propose a sampling test scheme to empirically verify the behavior of the subthreshold seeker on various QBCs. Finally, the empirical results are illustrated to validate the proposed theoretical model.

A. Evaluation Time of Subthreshold Seeker

With the global search time ready, we can now estimate the time to find the minimum point, i.e., the evaluation time T of subthreshold seeker, with the equation

$$T = \frac{cN}{m}\left\lceil \frac{bs}{2} \right\rceil + \frac{m+1}{2} \; . \tag{7}$$

The expected total time over a QBC is considered as the sum of the expected total global search time, the first term, and the expected total local search time, the second term. As discussed in the previous section, it is expected to apply $\lceil bs/2 \rceil$ local search runs in order to find the global optima, and thus, $\lceil bs/2 \rceil$ global search runs. $\frac{cN}{m}$ represents the average global time with c varies between 0.75 and 1.5, and $\frac{m+1}{2}$ corresponds to the expected time for the local search to find the minimum among subthreshold points. To derive the m that achieves the minimum evaluation time, we solve the following equation with the first derivative of (7)[4] to be zero:

$$T' = -\frac{bscN}{2m^2} + \frac{1}{2} = 0 \; . \tag{8}$$

The solution of this equation is $m = \sqrt{bscN}$. Setting m to about $\sqrt{bscN}$ in (7), the subthreshold seeker can achieve the minimum evaluation time T around $\sqrt{bscN}$.[5] Note that the total global search time and the total local search time are near identical when the overall evaluation time is minimum. The following sections verify (7) with the results obtained by our experiments.

B. Sampling Test Scheme

For empirical convenience, we implement the simplest case of QBC, pathwise quasi-basin class (PQBC). PQBC is the class of functions with a simple path spatial structure and a distinct integer value in $\mathcal{Y} = \{1, 2, \cdots, n\}$, where $n = |\mathcal{X}|$, on each vertex. PQBC is formally defined as

Definition 7 (PQBC): Given a finite set $\mathcal{Y} = \{1, 2, \cdots, n\} \subset \mathbb{N}$ and a simple path $G = \overline{v_1 v_2 \ldots v_n}$, the pathwise QBC with b distinct quasi-basins and m subthreshold vertices is defined as $\mathcal{Q}^{+}(G, \mathcal{Y}, m, b)$.

To investigate the expected subthreshold seeker behavior over a specific PQBC, we sample functions from a specific

[4]For convenience, we omit the ceiling.

[5]The actual value is $\sqrt{bscN}$+0.5, we omit 0.5 as it is a rather small quantity.

```
1: procedure PATHWISE QBC SAMPLER($\overline{v_1 v_2 \ldots v_n}$, $\mathcal{Y} = \{1,2,\ldots,n\}, m, b, Usize$)
2:     $ST \leftarrow \{1,2,\ldots,m\}$
3:     $GT \leftarrow \{m+1, m+2, \ldots, n\}$
4:     $i \leftarrow 1$
5:     while $i \leq b$ do
6:         $gt \leftarrow UniformPick(GT)$
7:         $st \leftarrow UniformPick(ST)$
8:         $qb_i \leftarrow (gt, st)$
9:         $i \leftarrow i + 1$
10:    end while
11:    while $ST \neq \emptyset$ do
12:        $st \leftarrow UniformPick(ST)$
13:        if $Usize$ then
14:            $i \leftarrow i + 1 \mod b$
15:        else
16:            $i \leftarrow Uniform([1, b])$
17:        end if
18:        $qb_i \leftarrow (qb_i, st)$
19:    end while
20:    $QB \leftarrow \cup qb_i$
21:    $S \leftarrow GT \cup QB$
22:    $i \leftarrow 1$
23:    while $i \leq n$ do
24:        $v \leftarrow UniformPick(S)$
25:        while $length(v) > 1$ do
26:            $f(v_i) \leftarrow Pop(v)$
27:            $i \leftarrow i + 1$
28:        end while
29:        $f(v_i) \leftarrow v$
30:        $i \leftarrow i + 1$
31:    end while
32:    return $f$
33: end procedure
```

Fig. 6. Pathwise QBC sampler.

PQBC via the PQBC sampler of which the pseudo code is shown in Fig. 6 to generate functions in the pathwise QBC with uniform basins and non-uniform basins. Function *UniformPick* in Fig. 6 samples the input set uniformly at random, returns the sampled value, and removes that value from the input set. Function *Pop* outputs the first value of a sequence and removes the first value from the sequence. The pathwise QBC sampler separates the input values $1, 2, \ldots, n$ into two sets: the set with superthreshold points (GT) and the set with subthreshold points (ST) and then uniformly randomly picks one point from GT and one from ST to construct the basic sequence of a quasi-basin. After every quasi-basin has its basic sequence, the next step is to assign all the subthreshold points to each quasi-basin. When the input boolean parameter *Usize* is set to *True*, the sampler uniformly assigns subthreshold points to every quasi-basin. Otherwise, each subthreshold point is assigned to an arbitrary quasi-basin. We then uniformly randomly pick members in the quasi-basin sequences and GT. When a quasi-basin sequence is picked, this sequence is assigned as the values of next vertices. Fig. 7 illustrates an example of functions in the pathwise QBC which consists 20 points with three quasi-basins containing eight subthreshold points.

In order to empirically verify the time for a subthreshold seeker to find the minimum point of a given PQBC $\mathcal{Q}^{+}(G, \{1, 2, \cdots, n\}, m, b)$, we set the subthreshold seeker's threshold θ to $\beta_m(f)$. In the following sections, we verify (7) with the average time for a subthreshold seeker to find the minimum on various PQBCs. For each PQBC, the performance of the subthreshold seeker is measured by averaging 50 function instances with 20 independent runs on each function instance.

C. Experimental Results

Fig. 8 compares the average evaluation time for a subthreshold seeker with the exhaustive local search ($s = 1$) and the

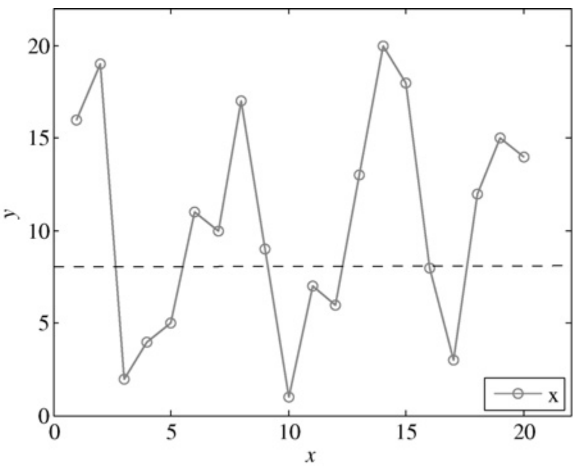


Fig. 7. Example of functions belonging to $\mathcal{Q}^{+}(G, \{1, 2, \cdots, 20\}, 8, 3)$.

theoretical evaluation time derived by (7) with respect to m on different pathwise uQBCs with n = 1000 and b = 1, 10, and 250. The solid lines, Ttheo and Ttheo1, indicate the theoretical evaluation time derived from (7) with c = 1, while the dashed line Ttheo2 indicates that with c = 1.5. The circle (ls) and the cross (gs) represent the average total number of sampling used by local search and global search respectively.

Because there is only one quasi-basin in Fig. 8(a), one global search is required. The proposed model matches the empirical result in this case. In Fig. 8(b), the empirical result matches the proposed model with c = 1.5. Such a situation may be caused by the significant global search time growth we observed in Fig. 5. The global search time grows as high as 1.5 when both b and m are quite small. Fig. 8(c) illustrates with large b, the subthreshold seeker performs worse than random search with its evaluation time exceed half of the search space size. Such a result, consisting with the proposed model, indicates that when the number of basins are greater than a quarter of the search space, the problem is unsearchable. Note that in these three cases, the average total local search time and the average total global search time also consist with our theoretical model. The average local search time is about $m/2$ while the average global search time matches (1).

Fig. 9 illustrates the evaluation time of a subthreshold seeker with the exhaustive local search on a non-uniform pathwise QBCs with n = 1000 and b = 10. Compare the results to that shown in Fig. 8(b), we can observe that although the deviations of the empirical results on non-uniform QBCs is slightly greater than that on uniform QBCs, the two sets of results basically resemble each other. Because the non-uniform pathwise QBCs have every basin's expected size identical, it can be expected that a subthreshold seeker behave statistically similarly on non-uniform and uniform QBCs.

Figs. 10 and 11 compare the theoretical optimal evaluation time with the empirical results with respect to b and n, respectively. In both cases, the exhaustive local search is applied. The solid lines in both figures indicate the optimal theoretical evaluation time predicted by (7) with c = 1, and the dashed line indicates that with c = 1.5. Both figures demonstrate that the theoretical prediction and the empirical results are in good

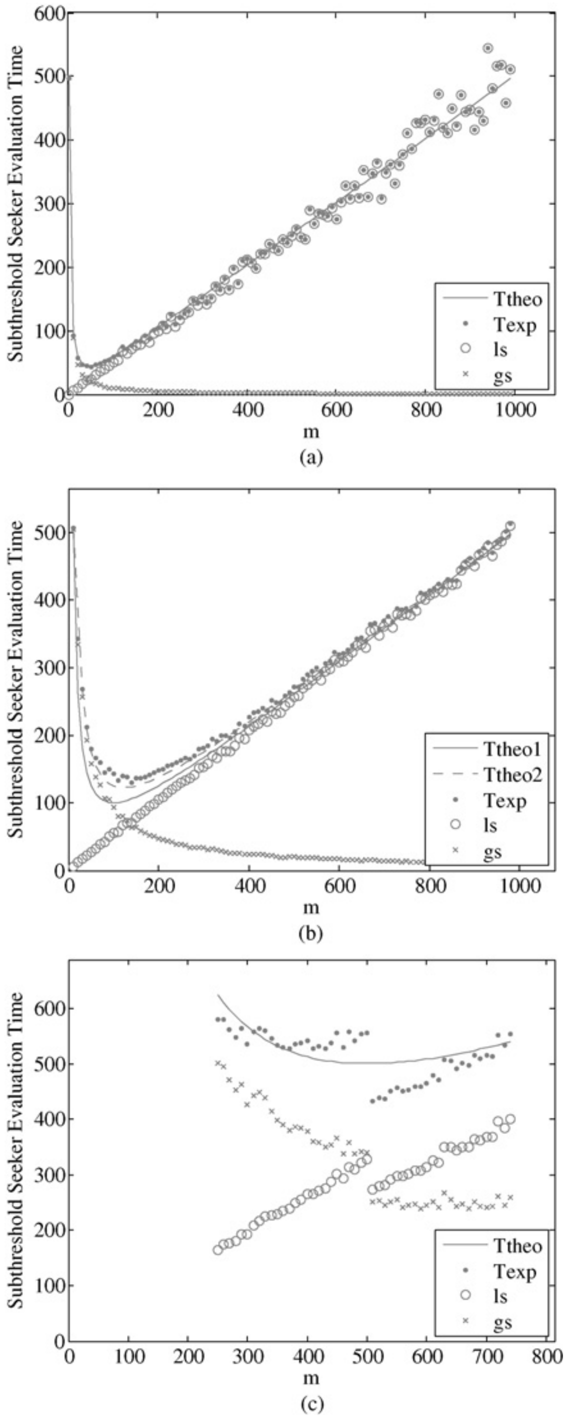


Fig. 8. Time for a subthreshold seeker to find the minimum with respect to m when (a) n = 1000 and b = 1, (b) n = 1000 and b = 10, and (c) n = 1000 and b = 250. The lines, Ttheo, Ttheo1, and Ttheo2, represent the theoretical values derived from (7), and the dot, Texp, represents the average time for a subthreshold seeker to find the minimum on different PQBCs. The average total sampling counts used by local search and global search are also recorded as ls and gs, respectively.

agreement, and therefore, (7) is dimensionally validated for different factors.

Fig. 12 illustrates the evaluation time with respect to m when non-exhaustive local search components, i.e., $s > 1$, are used. In both cases, the solid lines represent the theoretical evaluation time predicted by (7) with $c = 1.5$. These

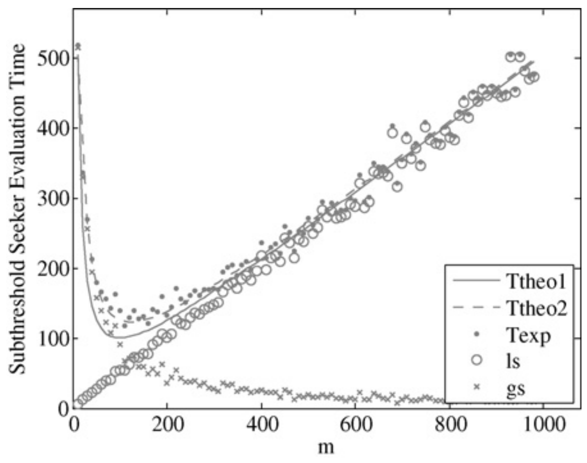


Fig. 9. n = 1000, b = 10, non-uniform quasi-basin.

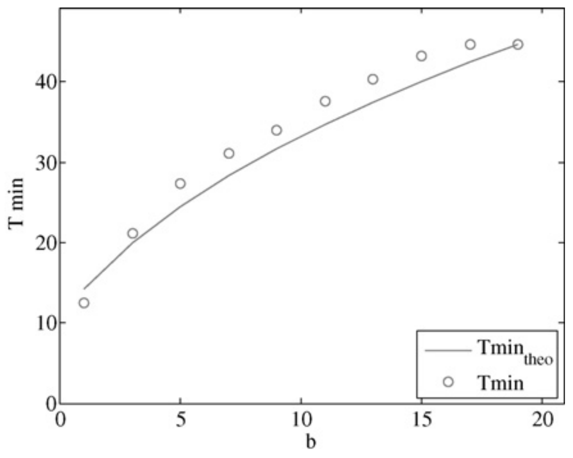


Fig. 10. Optimal evaluation time versus b when n = 100. The solid line indicates the theoretical value predicted by (7) with c = 1, and the dots indicate the empirical results.

empirical results also well match the proposed theoretical model (7).

VI. Discussion

In this section, we first explain how the subthreshold seeker can be regarded as a representative archetype of MAs and how the theoretical model can depict the general behavior of MAs. Then, we connect the proposed model to previous related studies in the literature. Finally, we discuss the extensions and future work of the proposed model.

A. Subthreshold Seeker as a Representative Archetype of MA

Since the proposed model of the subthreshold seeker on different QBCs has been validated by the empirical results in the previous section, in this section, we revisit our framework and discuss how our theoretical model is representative of MAs on a broad range of problems. The origin of our framework is the concept of local search zones. Based on this concept, the search space viewed by a search process can be partitioned into local search zones, which are areas preferred by exploitation, and parts of no interests. Global

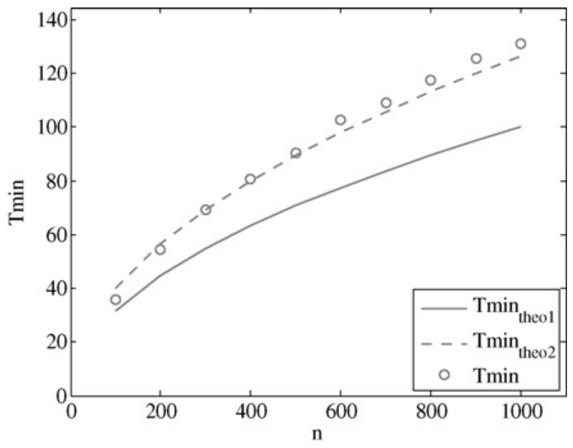


Fig. 11. Optimal evaluation time versus n when $b = 10$. The solid line indicates the theoretical value predicted by (7) with $c = 1$, the dashed line indicates that with $c = 1.5$, and the dots indicate the empirical results.

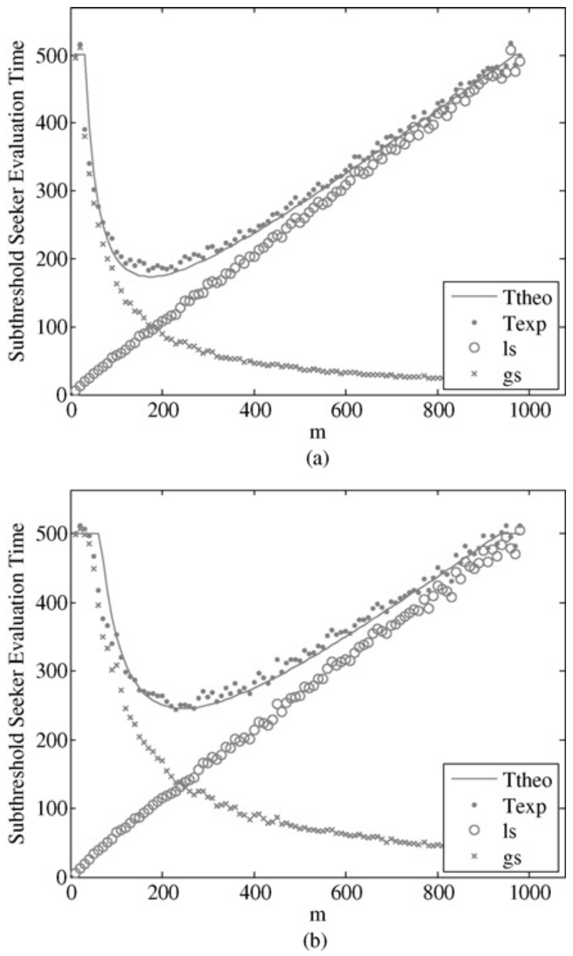


Fig. 12. Evaluation time for subthreshold seekers with a non-exhaustive local search component ($s > 1$). (a) $n = 1000$, $b = 10$, $s = 2$, uniform quasi-basin. (b) $n = 1000$, $b = 10$, $s = 4$, uniform quasi-basin.

search explores the whole space to find a local search zone for local search to exploit. The size of the local search zones in the search space affects the time for global search to find a point in local search zones and the time for local search to find the optimal solution in local search zones. The number of local search zones and the efficiency of local search further influence the required local search runs and global search runs. The performance of a memetic algorithm is thus determined by the efficiency of global search, the efficiency of local search, and the distribution of the local search zones. As the distribution of local search zones is dictated by the landscape of the search space and the local search criterion, by assessing the impact of the distribution of local search zones on the evaluation time, we can analyze the physics behind the collaboration between global search and local search on various problem classes.

Fitness-relevant and diversity-relevant metrics are common local search criteria in practical memetic algorithms. They form complicated local search zones which are difficult to measure. As we aim to model the pure collaboration between global search and local search, we consider the global search exhibits fair exploration and the local search exhibits fair exploitation. Hence, the diversity-relevant metrics which are commonly used to balance the exploration and the exploitation can be ignored. To build a more comprehensible model, we adopt fitness values as a representative fitness-relevant local search criterion. This criterion forms local search zones that can be referred to as quasi-basins. The QBC, which is accordingly defined, then categorizes all problems according to their quasi-basin distributions. In this way, our model is capable of describing the general behavior of memetic algorithms on a broad range of problems.

Generally, the QBC categorizes all the problems according to their search space landscape and the local search threshold. Besides the number of subthreshold points and the number of the quasi-basins, the QBC does not put any other constraints on the problems belonging to the same class. In other words, except that the subthreshold points may tend to gather according to the number of quasi-basins, both the subthreshold points and the superthreshold points of an instance that belongs to one QBC can be arbitrarily distributed in local search zones and the rest of the search space, respectively. As the expected performance of a local search algorithm on a QBC is calculated over all the possible instances belonging to the QBC, the local search processes on an instance of a QBC can be considered virtually as random sequences of subthreshold points. Thus, the expected performance of a local search generally would resembles that of a random sampling algorithm. On the other hand, as the subthreshold points tend to gather as quasi-basins, a greedy global search may perform worse than random sampling because it is not likely to discover a quasi-basin around a discovered quasi-basin. In fact, a random sampling algorithm may be the perfect explorer on a QBC due to its full diversity. From this point of view, the subthreshold seeker which virtually employs random search as its global search and local search can be considered a representative archetype of memetic algorithms on QBCs.

The proposed theoretical model manifests and gives explanations to the following facts.

1) Memetic algorithms which perform local search to few qualified points perform better.
2) The efficiency of local search greatly influences the evaluation time of memetic algorithms.
3) The physical landscape of a problem greatly influence the evaluation time of memetic algorithm.

We can assume that the average global search time to enter a local search zone is inversely proportional to the size ratio between local search zones and the search space, while the average total local search time is proportional to the size of the local search zones. Putting these two terms together, we can obtain the "V-shaped" curve which resembles those derived from (7). This V-shaped curve implies that a good collaboration between global search and local search should guarantee a short average global search time to hit local zones and sufficiently small sizes of local zones for the local search to exploit. Regarding the influence of the size of local search zones on the average global search time to find local search zones and the average time for local search to find the optimal point, memetic algorithms which have small sized local search zones will perform better. As mentioned in Section III-A, local search zones are generally zones consisting of qualified high-fitness points. A small size of local search zones implies that local search only be applied to few qualified individuals. This observation is consistent with the use of elitism in local search candidate selection and the infrequent local search principle in quite a number of research works [9], [10], [31]–[33]. It is also notable that several studies adopt a local search/global search ratio which is consistent with our theoretical model [36].

In our model, the local search component adopted by the subthreshold seeker exploits a quasi-basin via visiting the neighbors of current search point. The local search parameter of the employed local search operator is connected to how well a quasi-basin is exploited. Recall that when $s = 1$, the exhaustive local search will eventually visit all the points in a quasi-basin. In this case, the local minimum of a quasi-basin, which may be the global minimum, will be visited, and thus, only one local search run for each basin is required. For other local search parameter greater than one, there are chances for one local search run to miss the global minimum in a quasi-basin, and thus, more local search runs on this quasi-basin and more global search runs to hit this quasi-basin are required. The cost will be the extra global search time to enter the quasi-basin again when the algorithm guarantees non-repeated sampling. Fig. 12 and the factor s in (7) demonstrate the effect of the degree of exploitation of a basin. Such an effect implies that a good local search operator ought to fully exploit the given quasi-basin, at least the local minimum resides in the quasi-basin should be found, to guarantee a good local search and global search coordination. This inference is consistent with the empirical results of those studies [10], [39], [40] concluding that longer but not excessive local search lengths are favored in memetic algorithms.

Another notable factor is the number of local search zones in the search space. Our model illustrates that the number of global search runs is proportional to this factor. Recall that we represent the search space viewed by a search process as a graph composed of the neighborhood defined by the employed local search algorithm. The size of local search zones is determined by the local search criterion and the connectivity formed by the local search operator. Given the same local search criterion, the local search operator which forms fewer local search zones will perform better. This suggests that a good local search operator should be able to find local search points regardless of the physical landscape of a problem. This is somehow difficult for naive greedy local search algorithms to achieve and may require landscape knowledge given by the user or learned from the search process. However, operators with this kind of ability to cross the physical landscape of a problem somehow deviate from the traditional definition of local searchers. Thus, for typical local search, the number of local search zones are primarily defined by the physical landscape of a problem and the local search criterion. The physical landscape of a problem is usually connected to the number of niches of a problem. Our model also takes into account this crucial factor and delineates the relationship between this factor and the evaluation time. The proposed theoretical model indicates that for a fair memetic algorithm, the expected evaluation time should scale at most as the square root of the number of niches of a problem.

Despite the aforementioned consistency between the proposed model and the elitism based strategy in local search candidate selection, infrequent local search, long local search length, and local search/global search ratio, some previous studies also show a strong connection to our model. In an investigation on the balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling [39], the authors examined 132 combinations of 11 values of k, which is the maximum number of examined neighbors of the current solution, and 12 values of p_{LS}, which is the local search probability applied to the tournament selected individuals. The former factor k connects to the degree of how well a feasible sub-region can be exploited, and the second factor p_{LS} connects to the threshold that triggers local search. The authors found that the combination of the maximum k value and the minimum nonzero p_{LS} value achieved the best performance, the lowest cost of flowshop scheduling, in their experiments. The V-shaped curve of cost along the axis of the maximum k with respect to p_{LS} in their Fig. 13 resembles our V-shaped curve of the evaluation time in Fig. 8. Because the stop criterion of their experiments is the evaluation of a fixed number of points, the factor combinations that require less evaluation time to find the global minimum will have better solution quality, i.e., lower cost. This agreement implies that the proposed model may be adopted to give a theoretical explanation to the internal working of their multiobjective memetic algorithms.

Another set of intriguing empirical results is presented in the study of parameterizing local search [11]. In that study, the authors applied a hybrid approach to the memory cost minimization problem with various local search parameter settings. The local search parameter refers to the intensity of the local search method, a tractable algorithm called code size dynamic programming post optimization, applied to every individual in the population. The authors depicted in Fig. 13 in their paper that when a fixed runtime is used, the number of generations completed decreases rapidly as the local parameter increases. That the global search time is proportional to the number of generations implies the curve, indicating the global search time, resembles the expected T_θ in our Fig. 2. As the expected global search time is illustrated and the expected local search time will be proportional to the intensity of local search, summing up the expected global search time and the expected local search time, a V-shaped curve of the expected evaluation time with respect to the intensity of local search will be obtained. Fig. 12 in their study illustrates the attained solution quality, lower cost preferred, versus the setting of local search intensity. As previously discussed, lower attained cost in a given fixed time leads to shorter expected time to find the optimal solution. This figure also resembles the V-shaped curve of our theoretical model which confirms that the proposed model is quite applicable to their conclusions.

Although in these two studies, practical memetic algorithms, instead of the subthreshold seeker, are employed and investigated, the trend of their evaluation time resembles the proposed model developed based on the subthreshold seeker. It indicates that our theoretical model is indeed representative of memetic algorithms as we previously inferred. Another interesting study is the optimal bounds on finding fixed points of contraction mappings proved by [48]. In this investigation, the authors presumed that the expected lower bound of a randomized algorithm to find the fixed point of a contraction mapping $f : M \to M$ on a finite metric space (M, d) is $\Omega(\sqrt{|M|})$ and proved this bound is valid. In this fixed point problem, given any point $x \in M$ with the $d(x, f(x))$ the kth largest, one can find the fixed point with k steps via a valid deterministic algorithm. Consider the set exploited by the deterministic algorithm as the subthreshold sub-space which consists only one quasi-basin and the random sampling process to find a starting point for the deterministic algorithm as global search, according to (7), the best size of the subthreshold sub-space should be $\Omega(\sqrt{|M|})$ resulting in an expected optimal evaluation time of $\Omega(\sqrt{|M|})$. Thus, our theoretical model can also provide a reasonable, theoretical interpretation to this presumed value $\Omega(\sqrt{|M|})$.

Although our model is developed based on the subthreshold seeker, the subthreshold seeker can be taken as a representative archetype of memetic algorithms as it employs a fair explorer as its global search and a fair exploiter as its local search. The proposed model delineates the general behavior of memetic algorithms: how the global search behaves with respect to local search criteria, how the local search behaves with respect to local search criteria, how the local search criteria coordinate global search and local search, and how the local search efficiency and the problem landscape influence the evaluation time. The preceding paragraphs illustrated the consistence between our model and various memetic algorithm-problem complexes studied in the literature, either discrete problems or continuous problems, validate that our theoretical model is capable of providing a unified explanation to the physics behind memetic algorithms.

B. Extensions and Future Work

In our model, we propose the concept of local search zones and link it to the QBCs. Then, the subthreshold seeker, which employs random search as global search and local search, clearly illustrates the collaboration of global search and local search on various QBCs. In this presentation, estimating local search zones as quasi-basins, the global search can be considered as a baseline explorer and the local search can be considered as a standard exploiter. Thus, the model can reveal the essential relationship among the performance of memetic algorithms, the problem class categorized by QBCs, and the collaboration of global search and local search. In practical memetic algorithms, not only global searchers are population-based greedy approaches but also local searchers are greedy approaches on continuous problems. The local search criteria constantly depend on the status of the search process and the corresponding local search zones are difficult to measure. Although quasi-basins can roughly estimate local search zones, further studies on local search zones are required if more accurate models are to be developed. Extending our model to an instance of algorithm-problem complexes requires much further investigations into the following scopes.

1) The relationship between local search criteria and local search zones on discrete problems.
2) The behavior of the population-based greedy global search on discrete problems.
3) The behavior of the greedy local search on discrete problems.
4) All the three items in infinite and/or continuous domains.

Here, we discuss the first three scopes via adopting a memetic algorithm in the present framework. The memetic algorithm illustrated in Fig. 13 is a modified version of (μ+λ)-MA adopted in [25]. The algorithm first samples an initial population of size μ from $\mathcal{X}$, and then in each generation, generates λ children via the parent selection, mutation, and local search operations. The mutation operation flips each bit in x independently with probability $1/\ell$ where ℓ is $\lceil \log_2(|\mathcal{X}|) \rceil$. If the mutated offspring x' satisfies the local search criterion, it undergoes the local search operation. The best μ of the μ+λ individuals are selected as the survivors of that generation. The algorithm continues till its stopping criterion is satisfied. The first problem we confront is that for a local search criterion that other than a fitness value threshold, we must find a way to transfer the local search criterion to a fitness value threshold or the size of local search zones in a way that we can link the algorithm-problem complex to a QBC. Though most local search criteria are fitness-relevant and favor elitists, they are dependent on the current population and dynamic along generations. Further investigations are required to be devoted to this issue to provide some proper measurement of the size of local search zones. However, as the dynamics change slightly between generations, approximating the resultant size of local search zones with some statistical techniques may provide good solutions to this issue.

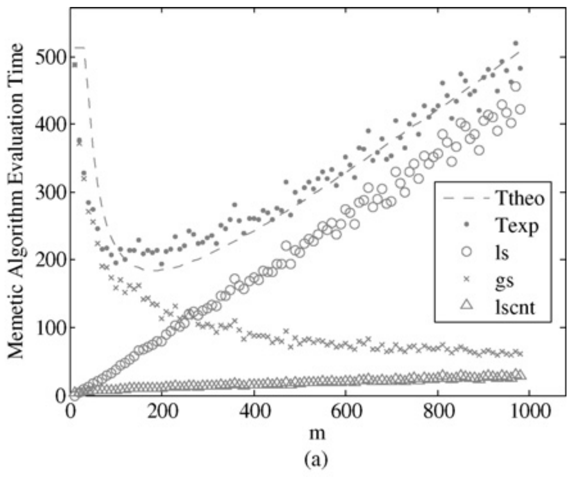
To manifest how population based greedy searchers perform on QBCs, the local search criterion of the (μ+λ)-MA is set the same as the subthreshold seeker. In other words, when the fitness value of the mutated individual is better than the threshold value, local search is applied to the mutated individual. Fig. 14 shows how a (20+20)-MA behaves with different local search operators on QBCs. The greedy local search keeps on moving to a better neighbor until no further move can be made. The exhaustive local search acts identically as the aforementioned subthreshold seeker does. In Fig. 14(a), the evaluation time of this (20+20)-MA can be approximated by the dashed line *Ttheo* which is (7) with $c \cdot s = 3.3$. In

```
1:  procedure ($\mu$+$\lambda$)-MEMETIC ALGORITHM($\mathcal{X}$, $\mathcal{Y}$, $f : \mathcal{X} \rightarrow \mathcal{Y}$)
2:      $t \leftarrow 1$
3:      Initialize population $P_1$ with $\mu$ individuals
4:      while the stopping criterion is not satisfied do
5:          $P'_t \leftarrow \emptyset$
6:          $i \leftarrow 1$
7:          while $i \leq \lambda$ do
8:              Choose $x \in P_t$ uniformly at random
9:              $x' \leftarrow mutation(x)$
10:             if $x'$ satisfies local search criterion then
11:                 $x'' \leftarrow localsearch(x')$
12:             else
13:                 $x'' \leftarrow x'$
14:             end if
15:             $P'_t \leftarrow P'_t \cup x''$
16:             $i \leftarrow i + 1$
17:         end while
18:         $P_{t+1} \leftarrow$ best $\mu$ individuals in $P'_t \cup P_t$
19:         $t \leftarrow t + 1$
20:     end while
21: end procedure
```

Fig. 13. (μ+λ)-memetic algorithm.

Fig. 14(b), although the evaluation time could not be properly approximated by (7), the V-shape remains. The triangles, *lscnt*, in these two figures represent the number of local search runs. From these figures, we can find that the applied greedy local search is much less efficient than the exhaustive search with its limited total local search time denoted by the circles, *ls*. The higher *lscnt* than the total local search time in the case of MA with greedy local search suggests revisiting of local searched points. Note also that in both figures, the memetic algorithm resembles the global search behavior of random search with an offset and the greedy local search resembles the exhaustive search with a degraded gradient. In these two memetic algorithms, our theoretical model is still capable of capturing the essence of the collaboration between global search and local search.

To accurately estimate the evaluation time of an instance of MA search process, one needs to take into account the influence of the population size and the exploration ability limited by its greediness to estimate its expected global search time and assess the efficiency of local search which corresponds to the parameter *s*. Another notable characteristic is that both memetic algorithms in the two cases perform worse than the subthreshold seeker on QBCs. This is consistent with our earlier statement that the random sampling is a better global explorer than any greedy algorithms on QBCs. This may seem contrary to the practical memetic algorithms. However, as the QBC categorizes arbitrary problems according to the quasi-basin distribution, it does not guarantee that all the problems within a QBC exhibit a regularity which a greedy algorithm can take advantages of. To manifest the optimization characteristics of greedy algorithms, fast convergence versus degrading diversity, the QBC framework must be extended to define classes of continuous-like discrete problems. In our opinion, adopting the concept of discrete Lipschitz class (DLC) [47] may be a good choice. In [47], the Lipschitz functions, functions with bounded slope, are transferred to DLC to describe continuous problems in discrete domains. Combining



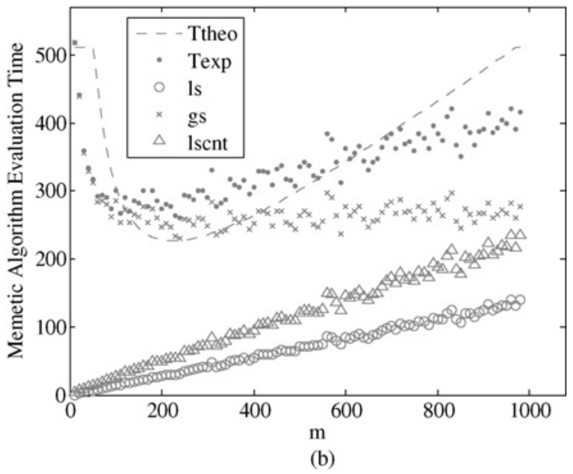


Fig. 14. Evaluation time for (20+20) memetic algorithms. (a) n = 1024, b = 10, MA with exhaustive local search. (b) n = 1024, b = 10, MA with greedy local search.

the DLC and QBC may provide a desired model in discrete domains that exhibits the characteristics of optimization in continuous domains.

For continuous problems, further efforts are required to extend all the analysis from discrete problems to continuous problems. In continuous domains, both $\mathcal{X}$ and $\mathcal{Y}$ are infinite sets. The first question may be how to extend the search space represented by a graph to fit the continuous scenario. If we can define the local search zones in a similar manner in continuous domains, we may start to investigate the behavior of memetic algorithms based on the modified framework. It may be much harder to estimate in continuous domains the expected global search time to enter local search zones, the expected total local search time to find the optimal solution, and the efficiency of local search of a given memetic algorithm. Once all these issues are resolved, an instance of algorithm-problem complex in continuous domains can be successfully delineated as well as the performance of a global search algorithm and a local search algorithm can be measured and compared in continuous domains. Memetic algorithm designers can thus select their global search algorithms and local search algorithms and design the local search criterion by following the guideline provided by the modified framework. As practical problems are generally black-box optimization, a designer with the aforementioned knowledge must dynamically estimate the number of local search zones and the size ratio of the local search zones and the search space to accordingly adjust the local search criterion in order to achieve the best collaboration between global search and local search.

Overall, although our model in this paper depicts the core, general behavior of memetic algorithms, it might potentially be extended to specific instances of memetic algorithm-problem complexes. Based on the concept of local search zones, the expected performance of a memetic algorithm can be assessed by analyzing the following components individually: the expected time for a global search algorithm to find a local search point, the expected time for a local search algorithm to find the optimal point in the local search zones, and the efficiency of a local search algorithm. With all the information available, algorithm designers can compare and select proper global search algorithms and local search algorithms and adopt the optimal local search criterion on the target problem accordingly. As designing an optimal memetic algorithm on a given problem has been the primary goal of memetic algorithms, extending our model to more practical memetic algorithms may provide a feasible way to achieve the goal. This may be an interesting and challenging task in the field of memetic algorithms.

VII. Summary and Conclusion

In this paper, we proposed the concept of local search zones. Based on this concept, we introduced the QBC to estimate the local search zones and adopted the subthreshold seeker as a representative archetype of memetic algorithms in order to analyze the collaboration between global search and local search on various quasi-basin classes. The derived theoretical model was capable of depicting the essence of the collaboration between a baseline global searcher and a standard local searcher. The efficiency of local search algorithms and the niches of problems were also taken into account in the proposed model.

The proposed theoretical model indicates that the global search time to find a point to start local search is inversely proportional to the size ratio between local search zones and the search space. The total local search time is proportional to the size of local search zones. Appropriate settings of local search criteria should guarantee sufficiently small sizes of local search zones. As the theoretical model cannot only well describe the behavior of the subthreshold seeker for the empirical results but can also capture the general behavior of various memetic algorithms proposed and observed in the literature, it can provide a unified explanation to the physics behind memetic algorithms and may reveal important insights to the design of memetic algorithms.

Furthermore, the proposed model is also capable of being extended to describe some specific memetic algorithms. The concept of local search zones provides an alternative way to assess the performance of a memetic algorithm by analyzing individually the performance of global search algorithms and the performance of local search algorithms. In this way, memetic algorithm designers may compare and select their global search algorithms and local search algorithms and adopt appropriate local search criteria for their problem. As the research direction of this paper may be a feasible way to achieve a better memetic algorithm design, along this line, much effort may be worth putting into further investigations.

Acknowledgment

The authors are grateful to the National Center for High-Performance Computing for computer time and facilities.

References

[1] R. Meuth, M.-H. Lim, Y.-S. Ong, and D. Wunsch, "A proposition on memes and meta-memes in computing for higher-order learning," *Memetic Comput.*, vol. 1, no. 2, pp. 85–100, 2009.

[2] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation: Past, present and future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.

[3] F. Neri and E. Mininno, "Memetic compact differential evolution for Cartesian robot control," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 54–65, May 2010.

[4] G. Acampora, M. Gaeta, and V. Loia, "Exploring e-learning knowledge through ontological memetic agents," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 66–77, May 2010.

[5] L. Jiao, M. Gong, S. Wang, B. Hou, Z. Zheng, and Q. Wu, "Natural and remote sensing image segmentation using memetic computing," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 78–91, May 2010.

[6] Z. Zhu, S. Jia, and Z. Ji, "Toward a memetic feature selection paradigm," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 41–53, May 2010.

[7] R. Meuth, E. Saad, D. Wunsch, and J. Vian, "Memetic mission management," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 32–40, May 2010.

[8] R. Ruiz-Torrubiano and A. Suarez, "Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 92–107, May 2010.

[9] W. E. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. California, San Diego, 1994.

[10] M. W. S. Land, "Evolutionary algorithms with local search for combinatorial optimization," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. California, San Diego, 1998.

[11] N. Bambha, S. Bhattacharyya, J. Teich, and E. Zitzler, "Systematic integration of parameterized local search into evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 137–155, Apr. 2004.

[12] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, Faculty Comput. Eng. Math. Sci., Univ. West England, Bristol, U.K., 2002.

[13] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[14] J. Smith, "Coevolving memetic algorithms: A review and progress report," *IEEE Trans. Syst. Man Cybern. B Cybern.*, vol. 37, no. 1, pp. 6–17, Feb. 2007.

[15] H. Mühlenbein, "How genetic algorithms really work: I. Mutation and hillclimbing," in *Proc. 2nd Conf. PPSN*, 1992, pp. 15–25.

[16] G. Rudolph, *Convergence Properties of Evolutionary Algorithms*. Hamburg, Germany: Verlag Dr. Kovač, 1997.

[17] S. Droste, T. Jansen, and I. Wegener, "A rigorous complexity analysis of the (1 + 1) evolutionary algorithm for separable functions with Boolean inputs," *Evol. Comput.*, vol. 6, no. 2, pp. 185–196, 1998.

[18] J. Garnier, L. Kallel, and M. Schoenauer, "Rigorous hitting times for binary mutations," *Evol. Comput.*, vol. 7, no. 2, pp. 173–203, 1999.

[19] H.-G. Beyer, *The Theory of Evolution Strategies* (Natural Computing Series). Berlin, Germany: Springer, 2001.

[20] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, nos. 1–2, pp. 51–81, 2002.

[21] J. He and X. Yao, "Toward an analytic framework for analyzing the computation time of evolutionary algorithms," *Artif. Intell.*, vol. 145, nos. 1–2, pp. 59–97, 2003.

[22] M. Jiang, Y. Luo, and S. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Inform. Process. Lett.*, vol. 102, no. 1, pp. 8–16, 2007.

[23] T. Jansen and I. Wegener, "The analysis of evolutionary algorithms: A proof that crossover really can help," *Algorithmica*, vol. 34, no. 1, pp. 47–66, 2008.

[24] D. Sudholt, "On the analysis of the (1 + 1) memetic algorithm," in *Proc. ACM SIGEVO GECCO*, 2006, pp. 493–500.

[25] D. Sudholt, "The impact of parametrization in memetic evolutionary algorithms," *Theor. Comput. Sci.*, vol. 410, no. 26, pp. 2511–2528, 2009.

[26] L. Kallel, B. Naudts, and C. R. Reeves, "Properties of fitness functions and search landscapes," in *Theoretical Aspects of Evolutionary Computing*. Berlin, Germany: Springer-Verlag, 2001, pp. 175–206.

[27] A. Sinha, Y.-P. Chen, and D. E. Goldberg, "Designing efficient genetic and evolutionary algorithm hybrids," in *Recent Advances in Memetic Algorithms* (Studies in Fuzziness and Soft Computing, vol. 166). Heidelberg, Germany: Physica-Verlag, 2004, pp. 259–288.

[28] Q. H. Nguyen, Y.-S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, Jun. 2009.

[29] S. Christensen and F. Oppacher, "What can we learn from no free lunch? A first attempt to characterize the concept of a searchable function," in *Proc. GECCO*, 2001, pp. 1219–1226.

[30] D. Whitley and J. Rowe, "Subthreshold-seeking local search," *Theor. Comput. Sci.*, vol. 361, no. 1, pp. 2–17, 2006.

[31] K.-H. Liang, X. Yao, and C. Newton, "Evolutionary search of approximated n-dimensional landscapes," *Int. J. Knowl. Based Intell. Eng. Syst.*, vol. 4, no. 3, pp. 172–183, 2000.

[32] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning," *IEEE Trans. Syst. Man Cybern. B Cybern.*, vol. 37, no. 1, pp. 62–69, Feb. 2007.

[33] Y.-H. Liu, "A hybrid scatter search for the probabilistic traveling salesman problem," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 2949–2963, 2007.

[34] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evol. Comput.*, vol. 12, no. 3, pp. 273–302, 2004.

[35] S. Garcia, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognit.*, vol. 41, no. 8, pp. 2693–2709, 2008.

[36] D. Molina, M. Lozano, C. Garcia-Martinez, and F. Herrera, "Memetic algorithms for continuous optimization based on local search chains," *Evol. Comput.*, vol. 18, no. 1, pp. 27–63, 2010.

[37] F. Neri, J. Toivanen, G. L. Cascella, and Y.-S. Ong, "An adaptive multimeme algorithm for designing HIV multidrug therapies," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 4, no. 2, pp. 264–278, Apr.–Jun. 2007.

[38] J. Tang, M. H. Lim, and Y. S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems," *Soft Comput.*, vol. 11, no. 9, pp. 873–888, 2007.

[39] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.

[40] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Trans. Syst. Man Cybern. B Cybern.*, vol. 37, no. 1, pp. 70–76, Feb. 2007.

[41] P. Merz and B. Freisleben, "Fitness landscapes and memetic algorithm design," in *New Ideas in Optimization*. New York: McGraw-Hill, 1999, pp. 245–260.

[42] I. Paenke, T. J. Kawecki, and B. Sendhoff, "The influence of learning on evolution: A mathematical framework," *Artif. Life*, vol. 15, no. 2, pp. 227–245, 2009.

[43] M. Le, Y.-S. Ong, Y. Jin, and B. Sendhoff, "Lamarckian memetic algorithms: Local optimum and connectivity structure analysis," *Memetic Comput.*, vol. 1, no. 3, pp. 175–190, 2009.

[44] C. Witt, "Runtime analysis of the (μ + 1) EA on simple pseudo-Boolean functions," *Evol. Comput.*, vol. 14, no. 1, pp. 65–86, 2006.

[45] B. Doerr, N. Hebbinghaus, and F. Neumann, "Speeding up evolutionary algorithms through asymmetric mutation operators," *Evol. Comput.*, vol. 15, no. 4, pp. 401–410, 2007.

[46] A. H. Wright and J. N. Richter, "Strong recombination, weak selection, and mutation," in *Proc. ACM SIGEVO GECCO*, 2006, pp. 1369–1376.

[47] P. Jiang and Y.-P. Chen, "Free lunches on the discrete Lipschitz class," *Theor. Comput. Sci.*, vol. 412, no. 17, pp. 1614–1628, 2011.

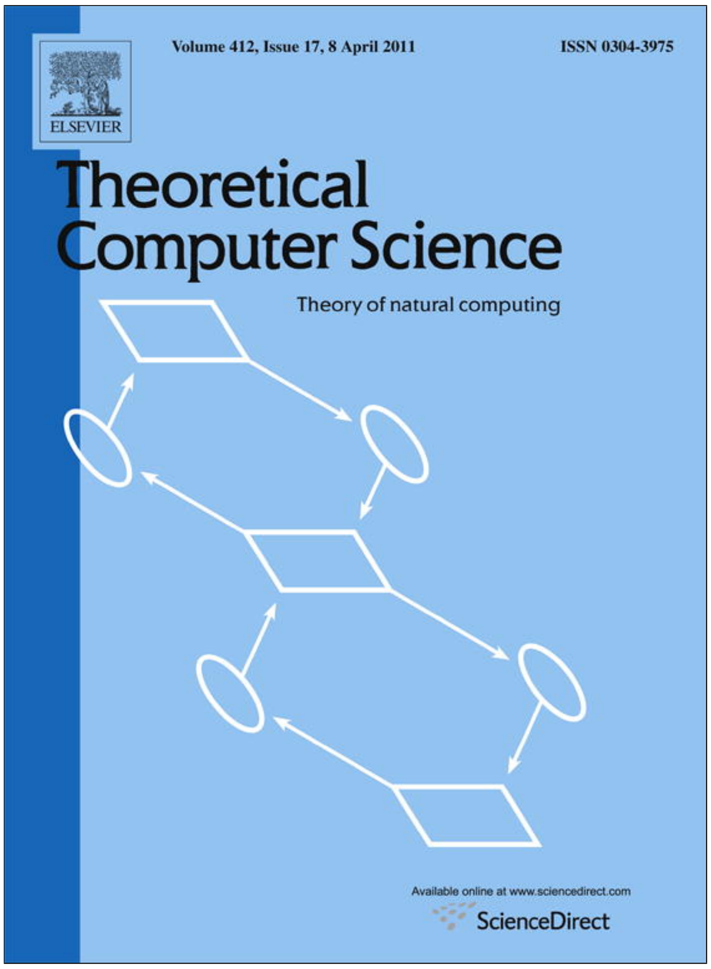
[48] C.-L. Chang and Y.-D. Lyuu, "Optimal bounds on finding fixed points of contraction mappings," *Theor. Comput. Sci.*, vol. 411, nos. 16–18, pp. 1742–1749, 2010.

Jih-Yiing Lin received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000 and 2002, respectively. She is currently working toward the Ph.D. degree in computer science from the Department of Computer Science, National Chiao Tung University.

She was with Sunplus Technology, Hsinchu, from 2002 to 2007. Her current research interests in the field of genetic and evolutionary computation include theories, working principles, particle swarm optimization, and linkage learning techniques.

Ying-Ping Chen (M'04) received the B.S. and M.S. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree from the Department of Computer Science, University of Illinois, Urbana, in 2004.

He was an Assistant Professor from 2004 to 2009 and has been an Associate Professor since 2009 with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan. His current research interests in the field of genetic and evolutionary computation include theories, working principles, particle swarm optimization, estimation of distribution algorithms, linkage learning techniques, and dimensional/facet-wise models.


Volume 412, Issue 17, 8 April 2011
ISSN 0304-3975
ELSEVIER
Theoretical Computer Science
Theory of natural computing
Available online at www.sciencedirect.com
ScienceDirect

Theoretical Computer Science 412 (2011) 1614–1628

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Free lunches on the discrete Lipschitz class

Pei Jiang, Ying-ping Chen [*]

Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, HsinChu City 300, Taiwan

ARTICLE INFO

Article history:
Received 14 January 2009
Received in revised form 30 November 2010
Accepted 8 December 2010
Communicated by G. Rozenberg

Keywords:
No-Free-Lunch theorem
Lipschitz continuity
Discrete Lipschitz class
Subthreshold-seeker
Sampling-test scheme

ABSTRACT

The No-Free-Lunch theorem states that there does not exist a genuine general-purpose optimizer because all algorithms have the identical performance on average over all functions. However, such a result does not imply that search heuristics or optimization algorithms are futile if we are more cautious with the applicability of these methods and the search space. In this paper, within the No-Free-Lunch framework, we firstly introduce the discrete Lipschitz class by transferring the Lipschitz functions, i.e., functions with bounded slope, as a measure to fulfill the notion of continuity in discrete functions. We then investigate the properties of the discrete Lipschitz class, generalize an algorithm called subthreshold-seeker for optimization, and show that the generalized subthreshold-seeker outperforms random search on this class. Finally, we propose a tractable sampling-test scheme to empirically demonstrate the superiority of the generalized subthreshold-seeker under practical configurations. This study concludes that there exist algorithms outperforming random search on the discrete Lipschitz class in both theoretical and practical aspects and indicates that the effectiveness of search heuristics may not be universal but still general in some broad sense.

1. Introduction

In the 1980s there was a belief in the field of evolutionary computation that evolutionary algorithms are more widely applicable and have superior overall performance while they may not perform as well as the specialized algorithm for a specific optimization problem. However, in 1995, Wolpert and Macready proposed the No-Free-Lunch (NFL) theorem [1,2] which formally states that every algorithm performs equally well on average over all functions. A direct implication of NFL is that, given a performance measure, the better performance of an algorithm on some problems is balanced by the worse performance on others. In other words, there is no such thing as robustness under the NFL framework, or all algorithms are considered robust. Therefore, it is no surprise that the proposition of the NFL theorem causes a great deal of controversy in the optimization and heuristic search community [3], as the NFL theorem sets a limitation on the pursuit of general-purpose optimizers.

Indeed, the implications of the NFL theorem seem to disagree with empirical observations of the effectiveness of optimization algorithms and search heuristics, since general-purpose optimizers such as gradient-based methods, simulated-annealing, and biologically inspired algorithms do have their share of significance in real-world applications. On the other hand, the NFL theorem is a mathematical theorem, which means that it is absolutely true when all the assumptions are given. As a consequence, previous studies intending to address the incoherence between theoretical results and empirical observations are mostly aiming at the assumptions of the NFL theorem, especially the notion of “all functions”. Droste et al. [4,5] systematically described a few scenarios of functions and claimed that the scope of the NFL theorem is too enormous to be realistic. Streeter [6] proved that the NFL theorem does not hold over the problems with sufficiently bounded

[*] Corresponding author. Tel.: +886 3 5712121; fax: +886 3 6126520.
E-mail addresses: pjiang@nclab.tw (P. Jiang), ypchen@nclab.tw, ypchen@cs.nctu.edu.tw (Y.-p. Chen).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.028

description length. Igel and Toussaint [7] extended the NFL theorem and investigated the necessary and sufficient conditions for the NFL theorem based on distributions of target functions. Beyond identifying a subset of problems on which the NFL result cannot be applied, Christensen and Oppacher [8] started with a more direct standpoint by proposing the submedian-seeker and demonstrated such an algorithm can outperform random search on certain types of functions. Thereafter, Whitley and Rowe [9] simplified and extended Christensen and Oppacher's work and showed that a more generic subthreshold-seeker can outperform random search on uniformly sampled polynomials in the sense of the number of subthreshold points visited in a given time span.

In addition to the studies following the NFL theorem directly, there is a notable amount of efforts investigating the expected runtime of search heuristics on various fitness functions. As early as 1992, when the NFL theorem had not been introduced yet, Mülenbein [10] demonstrated the expected runtime of $(1+1)$-EA on the ONEMAX function and the (k, ℓ)-deceptive function. Thereafter, because of the simplicity of $(1+1)$-EA, it has been widely explored in numerous theoretical studies in the past decade, e.g., the analysis of $(1+1)$-EA on linear functions [11,12], unimodal functions [4,11], the needle-in-a-haystack problem [13], the maximum matching problem [14], and the minimum spanning tree problem [15]. While in these studies, the NFL theorem might not be the central topic, the rigorous analysis thereof provides a firm theoretical basis to compare optimization algorithms and indicates superior performance of some algorithm over another on certain functions [16], thus relating the applicability of optimization algorithms to the NFL theorem in a subtle manner.

In the aforementioned studies, the topics may be different, but a common goal is shared – addressing the generality of optimization algorithms and search heuristics. This study serves the same purpose. Borrowing the notion of Lipschitz functions in real analysis, we introduce the discrete Lipschitz class as an attempt to capture the continuity of a discrete search space. The property of similarities in objective values within a neighborhood is possessed by many real-world problems, and such a problem structure does facilitate the search process. In particular, we prove that a generalized subthreshold-seeker indeed outperforms random search on the discrete Lipschitz class in theory as well as demonstrate the theoretical result can be carried over into practice by proposing a sampling-test scheme and conducting numerical experiments with comparisons.

The remainder of this paper is organized as follows: In Section 2, we briefly review the NFL framework to establish and unify the terminology and definitions as preliminaries. Section 3 introduces the discrete Lipschitz class and describes the relationship between the class and the theorem with a focus on the condition under which the NFL theorem holds over the discrete Lipschitz class. In Section 4, we generalize the subthreshold-seeker and discuss its performance on the discrete Lipschitz class in comparison with random search. In Section 5, we propose a sampling-test scheme as an alternative way to examine the effectiveness of optimizers in practice, followed by the conclusions in Section 6.

2. A brief review of NFL

The No-Free-Lunch (NFL) theorem, in short, states that all algorithms have the same overall performance. As plain as this statement may seem, there are several aspects to be clarified. Firstly, "algorithms" in the realm of NFL are restricted to the scope of "non-repeating black-box algorithms". The term "black-box algorithm", referred to as "blind search" in some papers, is used to describe the class of algorithms only employing the result of function evaluations as information. The requirement of non-repeating ensures that the search process can be viewed as a permutation of the elements in search space, and revisiting points merely increases the running time without rendering any assistance for identifying the optimum. In fact, when the performance is averaged over all functions, based on NFL, the best an algorithm can do is try not to re-sample.

The concept of "all problems" is another intriguing point for its inherent vagueness. If the set of all problems is defined in the most general sense that there are no restrictions, e.g., requiring all problems to share a fixed and finite domain, imposed on the scope of problems, one of the fundamental results in computability is that the set of problems is uncountably infinite. If we consider the collection of all feasible regions of optimization problems as a formal language, we can easily use the diagonalization method to show that such a language is not recursively enumerable or Turing-recognizable. In other words, in this case, there exists a problem whose feasible region is not recognizable by any Turing machine. Therefore, to bypass this difficulty, the original NFL framework takes a more practical stand here by considering the functions defined on a finite domain with a finite codomain.[1]

Within the NFL framework, the concepts of optimization problems and search algorithms can be formalized in the following definition:

Definition 1 (*NFL Framework*)**.** Given two finite sets $\mathcal{X}$ and $\mathcal{Y}$,

(1) The set of all functions $\mathcal{F}_{\mathcal{X},\mathcal{Y}}$, with respect to $\mathcal{X}$ and $\mathcal{Y}$, is defined as $\mathcal{F}_{\mathcal{X},\mathcal{Y}} := \{f \mid f : \mathcal{X} \to \mathcal{Y}\}$.

(2) A *trace* of length m is a sequence $T_m := ((x_i, y_i))_1^m = ((x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ with distinct x_i's. "$x \in T_m$" denotes that x appears in T_m, i.e., $x = x_i$ for some $i \in \{1, 2, \dots, m\}$. Let T_0 be the empty sequence and $\mathcal{T}^\ell$ be the set containing all the traces of a length smaller than or equal to ℓ.

(3) Let A_T, where $T \in \mathcal{T}^{|\mathcal{X}|-1}$, be a random variable over $\mathcal{X}$ satisfying that $Prob\{A_T = x\} = 0$ for all $x \in T$. An *algorithm* A is a collection of such random variables, i.e., $A = \{A_T \mid T \in \mathcal{T}^{|\mathcal{X}|-1}\}$.

1 While the original NFL theorem assumes finiteness of the search space, studies extending the applicability of the NFL theorem on continuous domains have arisen recently [17,18].

(4) The *search process* of A on f, $S(A, f)$, is the stochastic process $(X_i, Y_i := f(X_i))$ over $\mathcal{X} \times \mathcal{Y}$ defined by $X_1 \sim A_{T_0}$ and $X_{k+1} \sim A_{((X_i,Y_i))_1^k}$. Let $S(A, f, k) := ((X_i, Y_i))_1^k$, and $S_y(A, f, k) := (Y_i)_1^k$ is called the *performance vector.*

(5) Let $\mathcal{V} := \bigcup_{i=1}^{|\mathcal{X}|} \mathcal{Y}^i$ be the set containing all possible performance vectors. A *performance measure* is any function mapping $\mathcal{V}$ to $\mathbb{R}$.

The terminology in Definition 1 mostly follows those adopted in [2] and [19] with a few slight modifications applied to avoid the situation that an algorithm is undefinable on a complete trace and to make search processes able to be expressed in a naturally stochastic way.

Example 2 (*Random Search in NFL*). Let R_{T_m} be a random variable that $Prob\{R_{T_m} = x\} = 1/(|\mathcal{X}| - m)$ for all $x \notin T_m$. In the NFL framework, random search can be accordingly defined as $RS := \bigcup_{m=0}^{|\mathcal{X}|-1} \{R_{T_m} \mid T_m \in \mathcal{T}^{|\mathcal{X}|-1}\}$.

Now, the NFL theorem can be given as Theorem 3. The complete proof can be found in the original NFL papers [1,2].

Theorem 3 (*NFL Theorem*). *If $v \in \mathcal{V}$ is a performance vector with length ℓ, $\sum_f Prob\{S_y(A, f, \ell) = v\} = c$, where c is a constant independent of A.*

3. Discrete Lipschitz class

3.1. Definition of the discrete Lipschitz class

In real analysis, Lipschitz functions refer to the functions with bounded slope. Given a set $\mathcal{C} \subseteq \mathbb{R}$, $f : \mathcal{C} \to \mathbb{R}$ is a *Lipschitz function* if there exists a constant $K > 0$ such that $|f(a) - f(b)| \le K|a - b|$ for all $a, b \in \mathcal{C}$. The Lipschitz condition is a stronger condition than normal continuity, because any Lipschitz function is uniformly continuous. On the other hand, the functions that are not everywhere differentiable may still be Lipschitz, e.g., $f(x) = |x|$. On a closed interval, the Lipschitz class lies between continuous functions and the functions having continuous derivatives [20].

For the discrete space, there is no such thing as continuity. However, if there is some sort of distance defined in some discrete space, the Lipschitz condition can still be applied, and therefore a natural way to simulate continuity in the discrete space can be obtained. In combinatorics, the spatial structures are typically formed via graph theory. If we view the vertex set as the search space and the edge set as the specification of the geometry, the Lipschitz condition can be transferred here by restricting the difference of objective values between any two adjacent vertexes. The merit of such definition is that we do not put any constraints on the global structure directly such as to demand the functions to be polynomial or the description length to be bounded. Instead, we only expect some similarities of the objective values within a neighborhood in the search space.

Before commencing to detail our work, we explain the notations adopted in this paper for clarity. Given a graph G, the vertex set and the edge set of G are denoted as $V(G)$ and $E(G)$. Also, $deg(v)$ and $N(v)$ indicate the degree and the neighborhood of a vertex $v \in V(G)$, respectively. Since we will focus on the discrete Lipschitz class in the remainder of this paper, the domain $\mathcal{X}$ is always the vertex set $V(G)$, representing the spatial structure. Hence, the two notations $\mathcal{X}$ and $V(G)$ are used exchangeably. Furthermore, the property of $\mathcal{Y}$ of interest is the ordering, so without loss of generality, $\mathcal{Y}$ is assumed to be a subset of $\mathbb{N}$ of the form $\{0, 1, \dots, m\}$ unless specified otherwise. The discrete Lipschitz class (DLC) can now be introduced.

Definition 4 (*Discrete Lipschitz Class, DLC*). Given a connected graph G and a finite set $\mathcal{Y} \subset \mathbb{R}$, the corresponding discrete Lipschitz class with Lipschitz constant K is defined as

$$\mathcal{L}(G, \mathcal{Y}, K) := \{f : V(G) \to \mathcal{Y} \mid \forall \overline{v_1 v_2} \in E(G), |f(v_1) - f(v_2)| \le K\}.$$

Such a definition provides a means to represent the intrinsically real-parameter optimization problems through discretization (for practical computing devices). For instance, if a cube $\mathcal{C} \subset \mathbb{R}^n$ is discretized uniformly into a set of grid points, $V(G) = \{x_1, x_2, \dots, x_M\}^n \subset \mathbb{R}^n$ with $x_{i+1} - x_i = u > 0$, we can let $E(G) = \{\overline{v_i v_j} \mid v_i, v_j \in V(G) \text{ and } \|v_i - v_j\|_1 = u\}$. $\mathcal{L}(G, \mathcal{Y}, K)$ then forms a class containing all functions, defined on $\mathcal{C}$ differentiable with the absolute values of partial derivatives upper-bounded by K/u, discretized over $V(G)$. Furthermore, since $\mathcal{Y}$ is bounded, this class contains all functions mapping $V(G)$ to $\mathcal{Y}$ with sufficient large K (e.g., $K = \max \mathcal{Y} - \min \mathcal{Y}$).

The simplest case of DLC is the class of functions defined on $\mathbb{R}$, in which the graph representing the spatial structure is a simple path. Fig. 1 gives an illustrative example of such functions.

Definition 5 (*Pathwise Discrete Lipschitz Class, PDLC*). Given a finite set $\mathcal{Y} \subset \mathbb{R}$ and a simple path $G = \overline{v_1 v_2 \dots v_n}$, the pathwise discrete Lipschitz class with Lipschitz constant K is defined as $\mathcal{L}(G, \mathcal{Y}, K)$.

3.2. Combinatorial optimization problems and DLC

Because DLC is discrete by definition, if a combinatorial problem can be represented in the form of its solutions, we can then consider it an instance in DLC by specifying the neighborhood structure and the Lipschitz constant. For example, the search space of the Traveling Salesman Problem (TSP, see, e.g., [21]) consists of all possible tours, i.e., Hamiltonian cycles. For each tour, there is a corresponding edge set containing all the edges through which the tour passes. Therefore, we can

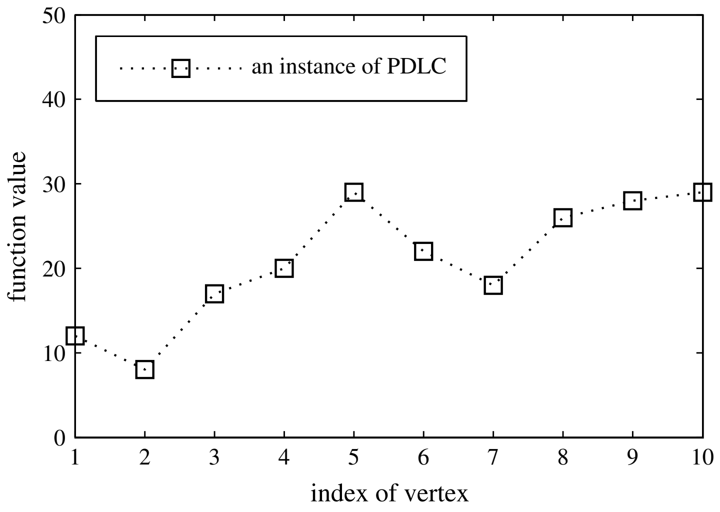


Fig. 1. A function instance of PDLC with 10 vertexes and $K = 10$.

define the neighborhood of a tour t as the set of tours which agree on all but two edges with t, and the Lipschitz constant can be written in terms of the maximum weight, as shown in the following example:

Example 6 (*TSP and DLC*)**.** If the weights of edges are non-negative and upper-bounded by a constant K, then every traveling salesman problem is an instance of $\mathcal{L}(G, \mathcal{Y}, 2K)$, where each vertex of G corresponds to a tour in the original graph and if two tours differ in only two edges, they are considered neighbors.

Note that G is not the original graph on which the TSP defined.

The Minimum Spanning Tree problem (MST, [21]) can also be similarly associated to DLC.

Example 7 (*MST and DLC*)**.** If the weights of edges are non-negative and upper-bounded by a constant K, then every MST problem is an instance of $\mathcal{L}(G, \mathcal{Y}, 2K)$, where each vertex of G corresponds to a spanning tree on the original graph, and if two spanning trees differ in only two edges, they are considered neighbors.

In addition to problems directly defined on graphs, the optimization version of the set-partition problem [21], which aims to partition a set into two equally weighted subsets, can also be connected to DLC. For a set $S \subset \mathbb{N}$, we denote the sum of elements in S as M_S, i.e., $M_S = \sum_{x \in S} x$.

Example 8 (*Set-partition Problem and DLC*)**.** Given a finite set $S \subset \mathbb{N}$ with $\max\{S\} = K$, the solution space of the set-partition problem is the power set of S. For a subset U of S, the corresponding objective value is $f(U) = |M_U - M_S/2|$. Two subsets A and B of S, where $A \subset B$, are considered neighbors if $|B - A| = 1$, and thus $f(A) \le f(B) = |M_B - M_A + M_A - M_S/2| \le f(A) + |M_A - M_B| \le f(A) + K$. Therefore, the set-partition problem on S is an instance of DLC with Lipschitz constant K.

Generally speaking, if a combinatorial optimization problem involves a weight set, it can be considered as an instance of DLC by interpreting the solutions as nodes and the relationship between solutions as the neighborhood structure. In the following section, we will show that the NFL theorem does not hold over DLC under most circumstances, and consequently, it is possible to contrive algorithms outperforming random search on these combinatorial problems.

3.3. DLC and NFL

In this section, we will investigate DLC within the NFL framework and derive a condition under which the NFL theorem holds. In order to determine whether the NFL theorem holds over a problem class, Schumacher et al. [19] provided a criterion for the NFL theorem based on permutation closure.

Definition 9 (*Permutation Closure*)**.** If π is a permutation on $\mathcal{X}$, define f_π as $f_\pi(x) := f(\pi(x))$ for all $x \in \mathcal{X}$. $F \subseteq \mathcal{F}_{\mathcal{X},\mathcal{Y}}$ is closed under permutation if for all $f \in F$ and for every permutation π on $\mathcal{X}$, $f_\pi \in F$.

Lemma 10. *The NFL theorem holds over F if and only if F is closed under permutation.*

Although in [19], this criterion is proposed for deterministic algorithms, since a randomized algorithm is simply a mixed strategy, i.e., a distribution over all possible deterministic strategies [22,5], this criterion still holds for randomized algorithms in the sense of expectation. Utilizing Lemma 10, a simple criterion for whether or not the NFL result can be applied to a DLC can be obtained.

Theorem 11 (*Criterion for NFL on DLC*)**.** *Let $\mathcal{L}(G, \mathcal{Y} = \{0, 1, \ldots, m\}, K)$ with $m > K \ge 1$ be a DLC. NFL holds over $\mathcal{L}(G, \mathcal{Y}, K)$ if and only if G is complete.*

Proof. By Lemma 10, it is sufficient to show that $\mathcal{L}(G, \mathcal{Y}, K)$ is closed under permutation if and only if G is complete.

- If G is complete, for every $f \in \mathcal{L}(G, \mathcal{Y}, K)$, we have $|f(v_i) - f(v_j)| \leq K$ for all v_i and $v_j \in V(G)$. For any permutation π on $\mathcal{X}$ and for all v_i and $v_j \in V(G)$, $|f_\pi(v_i) - f_\pi(v_j)| = |f(\pi(v_i)) - f(\pi(v_j))| \leq K$. Therefore, $f_\pi \in \mathcal{L}(G, \mathcal{Y}, K)$.
- If $\mathcal{L}(G, \mathcal{Y}, K)$ is closed under permutation, suppose for contradiction that G is not complete. The incompleteness and connectivity of G imply that there exist v_i and $v_j \in V(G)$ with $\overline{v_i v_j} \notin E(G)$. Select $v_k \in N(v_i)$, where $N(v_i)$ is the neighborhood of v_i. Obviously, $v_k \neq v_j$. Consider the function $f \in \mathcal{L}(G, \mathcal{Y}, K)$:

$$f(v) = \begin{cases} 0 & \text{if } v = v_i; \\ K+1 & \text{if } v = v_j; \\ K & \text{otherwise.} \end{cases}$$

and the permutation π:

$$\pi(v) = \begin{cases} v_k & \text{if } v = v_j; \\ tv_j & \text{if } v = v_k; \\ v & \text{otherwise.} \end{cases}$$

$|f_\pi(v_k) - f_\pi(v_i)| = |f(\pi(v_k)) - f(\pi(v_i))| = |f(v_j) - f(v_i)| = K + 1$, so $f_\pi \notin \mathcal{L}(G, \mathcal{Y}, K)$, a contradiction. □

The condition of completeness implies that the NFL theorem holds over DLC only in the extreme case that the entire search space is in the same neighborhood. While such a situation is theoretically possible, yet somewhat trivial. Taking PDLC as an example, when $m > K$, the NFL theorem sustains over a PDLC only if there are merely two vertexes in the search space.

4. DLC and subthreshold-seeker

The subthreshold-seeker (STS), introduced by Whitley and Rowe [9] and proved to outperform random search on uniformly sampled polynomials of one variable, is a metaheuristic that employs the threshold as a switch of local search. In essence, it is a selective local search method as it conducts local search if a given condition is satisfied. In this section, a generalization of subthreshold-seeker is firstly presented, and we will demonstrate that the generalized subthreshold-seeker can outperform random search on DLC.

4.1. Generalized subthreshold-seeker

In Whitley and Rowe's work, the subthreshold-seeker is an optimization algorithm aiming at functions with a one-dimensional domain, i.e., functions defined on a subset $\mathcal{C} \subseteq \mathbb{R}$. The subthreshold-seeker will successively select a point from the search space uniformly at random (u.a.r.) until a subthreshold point is encountered. Once encountering a subthreshold point, the subthreshold-seeker will search through the quasi-basin where that subthreshold point resides. In Whitley and Rowe's definition, a quasi-basin is a set of contiguous points with objective values below the threshold. In other words, the threshold is used to determine whether the subthreshold-seeker enters the local search phase, and the subthreshold-seeker can be viewed as an optimizer with an exhaustively local search operator.

According to this point of view, we generalize the subthreshold-seeker to the extent that it is applicable to any function of which the domain possesses a neighborhood structure as in Algorithm 1.

Algorithm 1 (*Generalized Subthreshold-seeker*).

```
procedure SUBTHRESHOLD-SEEKER($\mathcal{X}$, $\mathcal{Y}$, $N : \mathcal{X} \to 2^{\mathcal{X}}$, $f : \mathcal{X} \to \mathcal{Y}$)
    while the stopping criterion is not satisfied do
        if Queue is not empty then
            $x \leftarrow$ Queue.pop();
        else
            Select $x$ from $\mathcal{X}$ u.a.r.
        end if
        if $f(x) \leq \theta$ then
            Queue.push($N(x)$)
        end if
    end while
end procedure
```

Following the NFL framework, the parts of selecting and pushing are both restricted to unvisited points. Such a task can be achieved by a bookkeeping manner. Since the performance of an algorithm is judged by the performance vector, all overheads other than function evaluations will not count under the NFL framework.

The only control parameter of the subthreshold-seeker is the threshold. The elegance of the subthreshold-seeker is that it comprises the two fundamental operations of search heuristics, local search and global restart, and yet still stays in a simple form.

4.2. Subthreshold-seeker on DLC

Christensen and Oppacher [8] defined the performance measure as the number of submedian points visited by an algorithm, and Whitley and Rowe [9] generalized this notion to any threshold less than or equal to the median. That is, given a predefined stopping time L and $\alpha \in (0, 1/2]$, the performance measure is the number of points visited in the first L function evaluations with the top $\alpha|\mathcal{X}|$ values in the objective space.

This performance measure may seem odd at the first glance, for typically the performance of an optimizer is measured in terms of the time in which the optimum is located. However, even focusing on functions as simple as unimodal functions that are monotone with respect to the distance from the optimum, the time complexity analysis is still a difficult task. For instance, to the best of our limited knowledge, the time complexity of $(1 + 1)$-ES [23] on such functions has not been analyzed until recently [24]. Hence, it seems unlikely to analyze the runtime of an algorithm that is more sophisticated than random search over a broad class of problems. Furthermore, as mentioned in Section 2, within the NFL framework, the performance measure can be any function defined on the set containing all performance vectors, and roughly speaking, with more subthreshold points visited, it is more likely to identify a point with a satisfiable objective value. Therefore, Whitley and Rowe's notion appears in between theoretically analyzable and practically meaningful.

For any function f, we define $\beta_\alpha(f)$ be the maximum objective value below the performance threshold, i.e.,

$$\beta_\alpha(f) := \max\left\{ y \in \mathcal{Y} \;\middle|\; \sum_{i=0}^{y} |\{x \in \mathcal{X} \mid f(x) = i\}| \le \alpha|\mathcal{X}| \right\}.$$

If the set following the "max" notation is empty, then $\beta_\alpha(f)$ is defined to be $-\infty$. Let $\Psi_{\alpha,f}(v)$ be a performance measure that maps a performance vector v to the number of components of v below performance threshold, i.e., $\Psi_{\alpha,f}((v_1, v_2, \dots, v_L)) = |\{v_i \mid v_i \le \beta_\alpha(f)\}|$. It is noteworthy that the performance threshold should be distinguished from the algorithmic threshold. The latter should be regarded as a control parameter of the algorithm and hence is not related to the performance measure.

Whitley and Rowe showed that if f is a uniformly sampled polynomials of one variable, and $\beta_\alpha(f)$ is known in advance, setting $\theta = \beta_\alpha(f)$, under certain conditions the subthreshold-seeker outperforms random search on f. In this study, we will show that the subthreshold-seeker with θ within some range of codomain, rather than a specific value, will outperform random search in the sense that for all functions in the DLC, the expected number of points below the performance threshold visited by the subthreshold-seeker is greater than or equal to that by random search, and there does exist a function such that the inequality is strict.

Theorem 12 (*Equal or Better Performance of STS on DLC*)**.** *Let $\mathcal{L}(G, \mathcal{Y} = \{0, 1, \dots, m\}, K)$ with $m > K$ be a DLC. For all $f \in \mathcal{L}(G, \mathcal{Y}, K)$ if the algorithmic threshold θ of a subthreshold-seeker satisfies $\theta \le \beta_\alpha(f) - K$, then $E[\Psi_{\alpha,f}(S_y(STS, f, L))] \ge E[\Psi_{\alpha,f}(S_y(RS, f, L))]$ for all L with $1 \le L \le |\mathcal{X}|$.*

Proof. Let f be any function belonging to $\mathcal{L}(G, \mathcal{Y}, K)$. Suppose $S(STS, f, L) = ((X_{si}, Y_{si}))_{i=1}^{L}$ and $S(RS, f, L) = ((X_{ri}, Y_{ri}))_{i=1}^{L}$. Define the indicator variable I_{si} as $I_{si} = 1$ when $Y_{si} \le \beta_\alpha(f)$ and $I_{si} = 0$ otherwise, and I_{ri} is defined in a similar way for random search. We can obtain that $\Psi_{\alpha,f}(S_y(STS, f, L)) = \sum_{i=1}^{L} I_{si}$ and $\Psi_{\alpha,f}(S_y(RS, f, L)) = \sum_{i=1}^{L} I_{ri}$.

We prove the theorem by induction on L. Let $U := |\{x \in V(G) \mid f(x) \le \beta_\alpha(f)\}|$ be the total number of points below the performance threshold. When $L = 1$, since both strategies select a point u.a.r. from $\mathcal{X}$ in the first move, clearly $E[I_{s1}] = U/|\mathcal{X}| = E[I_{r1}]$. Suppose $E[\sum_{i=1}^{L} I_{si}] \ge E[\sum_{i=1}^{L} I_{ri}]$ for $1 \le L < |\mathcal{X}|$. Then,

$$\begin{aligned} E\left[\sum_{i=1}^{L+1} I_{si}\right] &= E\left[\sum_{i=1}^{L} I_{si}\right] + E[I_{sL+1}] \\ &= E\left[\sum_{i=1}^{L} I_{si}\right] + \sum_{(x_i)_{i=1}^{L} \in \mathcal{X}^L} E\left[I_{sL+1} \mid (X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}\right] Prob\{(X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}\}. \end{aligned} \quad (1)$$

If X_{si} is popped out from the queue, $f(X_{si}) \le \theta + K \le \beta_\alpha(f) - K + K = \beta_\alpha(f)$, and hence, $I_{si} = 1$. Otherwise, if X_{si} is selected from $\mathcal{X}$ u.a.r., then $Prob\{I_{si} = 1\} = (U - k)/(|\mathcal{X}| - i + 1)$, where k is the number of points visited in the first $i - 1$ steps with objective values smaller than or equal to $\beta_\alpha(f)$. Let C_L be the set collecting all $(x_i)_{i=1}^{L} \in \mathcal{X}^L$ such that if $(X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}$, the queue will be nonempty in the $(L + 1)$th move. Therefore,

$$\begin{aligned} &\sum_{(x_i)_{i=1}^{L} \in \mathcal{X}^L} E[I_{sL+1} \mid (X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}] Prob\{(X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}\} \\ &\quad = \sum_{(x_i)_{i=1}^{L} \in C_L} E[I_{sL+1} \mid (X_{si})_{i=1}^{L} \in C_L] Prob\{(X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}\} \\ &\quad + \sum_{(x_i)_{i=1}^{L} \notin C_L} E[I_{sL+1} \mid (X_{si})_{i=1}^{L} \notin C_L] Prob\{(X_{si})_{i=1}^{L} = (x_i)_{i=1}^{L}\} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{(x_i)_{i=1}^L \in C_L} 1 \cdot Prob\{(X_{si})_{i=1}^L = (x_i)_{i=1}^L\} \\
&+ \sum_{(x_i)_{i=1}^L \notin C_L} \frac{U - \left|\{x_i \in (x_i)_{i=1}^L \mid f(x_i) \le \beta_\alpha(f)\}\right|}{|\mathcal{X}| - L} Prob\{(X_{si})_{i=1}^L = (x_i)_{i=1}^L\} \\
&\ge \sum_{(x_i)_{i=1}^L \in \mathcal{X}^L} \frac{U - \left|\{x_i \in (x_i)_{i=1}^L \mid f(x_i) \le \beta_\alpha(f)\}\right|}{|\mathcal{X}| - L} Prob\{(X_{si})_{i=1}^L = (x_i)_{i=1}^L\} \\
&= \sum_{k=0}^{L} \frac{U-k}{|\mathcal{X}| - L} Prob\left\{\sum_{i=1}^{L} I_{si} = k\right\}. \qquad (2)
\end{aligned}
$$

Substituting into (1),

$$
\begin{aligned}
E\left[\sum_{i=1}^{L+1} I_{si}\right] &\ge \sum_{k=0}^{L} kProb\left\{\sum_{i=1}^{L} I_{si} = k\right\} + \sum_{k=0}^{L} \frac{U-k}{|\mathcal{X}|-L} Prob\left\{\sum_{i=1}^{L} I_{si} = k\right\} \\
&= \frac{U}{|\mathcal{X}|-L} + \frac{|\mathcal{X}|-L-1}{|\mathcal{X}|-L} \sum_{k=0}^{L} kProb\left\{\sum_{i=1}^{L} I_{si} = k\right\} \\
&= \frac{U}{|\mathcal{X}|-L} + \frac{|\mathcal{X}|-L-1}{|\mathcal{X}|-L} E\left[\sum_{i=1}^{L} I_{si}\right] \\
&\ge \frac{U}{|\mathcal{X}|-L} + \frac{|\mathcal{X}|-L-1}{|\mathcal{X}|-L} E\left[\sum_{i=1}^{L} I_{ri}\right] \\
&= \sum_{k=0}^{L} kProb\left\{\sum_{i=1}^{L} I_{ri} = k\right\} + \sum_{k=0}^{L} \frac{U-k}{|\mathcal{X}|-L} Prob\left\{\sum_{i=1}^{L} I_{ri} = k\right\} \\
&= E\left[\sum_{i=1}^{L} I_{ri}\right] + \sum_{k=0}^{L} E\left[I_{rL+1} \middle| \sum_{i=1}^{L} I_{ri} = k\right] Prob\left\{\sum_{i=1}^{L} I_{ri} = k\right\} \\
&= E\left[\sum_{i=1}^{L+1} I_{ri}\right]. \qquad (3)
\end{aligned}
$$

Inequality (3) follows from the induction hypothesis. □

Furthermore, next theorem guarantees that for any $f \in \mathcal{L}(G, \mathcal{Y}, K)$, if there exists a point above performance threshold and the subthreshold-seeker ever enters the local search phase, the subthreshold-seeker will outperform random search strictly in expectation according to the performance measure $\Psi_{\alpha,f}$.

Theorem 13 (*Strictly Better Performance of STS on DLC*). *Let $\mathcal{L}(G, \mathcal{Y} = \{0, 1, \dots, m\}, K)$ with $m > K$ be a DLC. For all $f \in \mathcal{L}(G, \mathcal{Y}, K)$ and for every subthreshold-seeker STS with $\theta \le \beta_\alpha(f) - K$ satisfy:*

(1) $\exists v \in V(G)$ with $f(v) > \beta_\alpha(f)$, and
(2) $\exists v \in V(G)$ with $f(v) \le \theta$,

$E[\Psi_{\alpha,f}(S_y(STS, f, L))] > E[\Psi_{\alpha,f}(S_y(RS, f, L))]$ for all $L \in [2, |\mathcal{X}| - 1]$.

Proof. If there are no such functions in $\mathcal{L}(G, \mathcal{Y}, K)$, the theorem holds vacuously. Otherwise, let f be any function satisfying the two conditions and define $((X_{si}, Y_{si}))_{i=1}^L$, $((X_{ri}, Y_{ri}))_{i=1}^L$, I_{si}, I_{ri}, U, and C_L in the same way as in Theorem 12. We prove by induction on L. When $L = 2$, since in the second step, the queue is nonempty if and only if $f(X_{si}) \le \theta$, $C_1 = \{v \in V(G) \mid f(v) \le \theta\} \ne \emptyset$ by Condition (2). Therefore,

$$
\begin{aligned}
E[I_{s1} + I_{s2}] &= E[I_{s1}] + \sum_{x \in \mathcal{X}} E[I_{s2} \mid X_{s1} = x] Prob\{X_{s1} = x\} \\
&= E[I_{s1}] + \sum_{x: f(x) \le \theta} 1 \cdot Prob\{X_{s1} = x\} + \sum_{x: \theta < f(x) \le \beta_\alpha(f)} \frac{U-1}{|\mathcal{X}|-1} Prob\{X_{s1} = x\} \\
&\quad + \sum_{x: f(x) > \beta_\alpha(f)} \frac{U}{|\mathcal{X}|-1} Prob\{X_{s1} = x\}
\end{aligned}
$$

$$
\begin{aligned}
&> E[I_{s1}] + \sum_{x:f(x)\le\beta_\alpha(f)} \frac{U-1}{|\mathcal{X}|-1} Prob\{X_{s1}=x\} + \sum_{x:f(x)>\beta_\alpha(f)} \frac{U}{|\mathcal{X}|-1} Prob\{X_{s1}=x\}\\
&= E[I_{s1}] + \sum_{k\in\{0,1\}} \frac{U-k}{|\mathcal{X}|-1} Prob\{I_{s1}=k\}\\
&= E[I_{r1}] + \sum_{k\in\{0,1\}} \frac{U-k}{|\mathcal{X}|-1} Prob\{I_{r1}=k\}\\
&= E[I_{r1}+I_{r2}].
\end{aligned}
$$

The inequality follows from $C_1 \ne \emptyset$ and $(U-1)/(|\mathcal{X}|-1) < 1$, for Condition (1) implies $U < |\mathcal{X}|$. For the induction hypothesis, suppose $E[\sum_{i=1}^{L} I_{si}] > E[\sum_{i=1}^{L} I_{ri}]$ for L with $2 \le L < |\mathcal{X}|-1$. In the $(L+1)$th step, from the proof of Theorem 12, we always have

$$
\begin{aligned}
E\left[\sum_{i=1}^{L+1} I_{si}\right] &\ge \frac{U}{|\mathcal{X}|-L} + \frac{|\mathcal{X}|-L-1}{|\mathcal{X}|-L} E\left[\sum_{i=1}^{L} I_{si}\right]\\
&> \frac{U}{|\mathcal{X}|-L} + \frac{|\mathcal{X}|-L-1}{|\mathcal{X}|-L} E\left[\sum_{i=1}^{L} I_{ri}\right]\\
&= E\left[\sum_{i=1}^{L+1} I_{ri}\right]. \qquad (4)
\end{aligned}
$$

Since $(|\mathcal{X}|-L-1)/(|\mathcal{X}|-L) > 0$ when $L < |\mathcal{X}|-1$, and $E[\sum_{i=1}^{L} I_{si}] > E[\sum_{i=1}^{L} I_{ri}]$ from the induction hypothesis, Inequality (4) is strict. $\square$

The following example illustrates a set of functions on which subthreshold-seekers strictly outperform random search.

Example 14. Given $V(G) = \{v_1, v_2, \ldots, v_{2n}\}$ and a positive integer K, the function f defined as $f(v_i) = iK$ is an instance of the PDLC $\mathcal{L}(G, \mathcal{Y} = \{0, 1, \ldots, 2nK\}, K)$. If $\alpha = 1/2$, then all subthreshold-seekers with $(n-1)K \ge \theta \ge K$ strictly outperform random search on f.

Let $d := \max\{deg(v) \mid v \in V(G)\}$ be the maximum degree of the graph and $dis(u, v)$ be the length of the shortest path from u to v. For any subthreshold-seeker, if we are able to set its θ within some interval, the following corollary gives a sufficient condition of the existence of functions on which the subthreshold-seeker strictly outperforms random search.

Corollary 15. *Let $\mathcal{L}(G, \mathcal{Y} = \{0, 1, \ldots, m\}, K)$ be a DLC with a maximum degree $d > 2$. Given $\alpha \in (0, 1/2]$ and an integer $C > 1$ with $CK + 1 \le m$, if*

$$\alpha|V(G)| > \frac{d(d-1)^C - 2}{d-2},$$

then there exists a function $f \in \mathcal{L}(G, \mathcal{Y}, K)$ such that $E[\Psi_{\alpha,f}(S_y(STS, f, L))] > E[\Psi_{\alpha,f}(S_y(RS, f, L))]$ for all L with $2 \le L \le |\mathcal{X}|-1$, where STS is a subthreshold-seeker with $\theta \in \beta_\alpha(f) - [K, CK]$.

Proof. We prove this corollary constructively. Select a vertex v_0 from $V(G)$ arbitrarily. Consider the function f defined as

$$f(v) = \begin{cases} 0 & \text{if } v = v_0;\\ dis(v, v_0)K & \text{if } 1 \le dis(v, v_0) \le C;\\ CK+1 & \text{otherwise.} \end{cases}$$

Since

$$
\begin{aligned}
&|v_o| + |\{v \in V(G) \mid 1 \le dis(v, v_0) \le C\}|\\
&\quad \le 1 + \left(d + d(d-1) + d(d-1)^2 + \cdots + d(d-1)^{C-1}\right)\\
&\quad = 1 + \frac{d\left[(d-1)^C - 1\right]}{(d-1)-1}\\
&\quad = \frac{d(d-1)^C - 2}{d-2} < \alpha|V(G)|,
\end{aligned}
$$

from the definition of $\beta_\alpha(f)$, $\beta_\alpha(f) = CK$. Furthermore, there must exist $v_1 \in V(G)$ that $f(v_1) = CK + 1$, for $|v_o| + |\{v \in V(G) \mid 1 \le dis(v, v_0) \le C\}| < |V(G)|$. Therefore, we have $f(v_0) \le \theta$ and $f(v_1) > \beta_\alpha(f)$. Thereby Theorem 13 can be applied. $\square$

For PDLC, since the maximum degree is upper-bounded by 2, the sufficient condition can be reduced to a simpler form.

Corollary 16. *Let $\mathcal{L}(G, \mathcal{Y} = \{0, 1, \ldots, m\}, K)$ be a PDLC. Given $\alpha \in (0, 1/2]$ and an integer $C > 1$ with $CK + 1 \le m$, if*

$$|V(G)| > (1 + 2C)\alpha^{-1},$$

then there exists a function $f \in \mathcal{L}(G, \mathcal{Y}, K)$ such that $E[\Psi_{\alpha,f}(S_y(STS, f, L))] > E[\Psi_{\alpha,f}(S_y(RS, f, L))]$ for all L with $2 \le L \le |\mathcal{X}| - 1$, where STS is a subthreshold-seeker with $\theta \in \beta_\alpha(f) - [K, CK]$.

Proof. For any $v_0 \in V(G)$, $|v_o| + |\{v \in V(G) \mid 1 \le dis(v, v_0) \le C\}| \le 1 + 2C$. □

Combining Theorems 12 and 13, if we manage to set $\theta \le \beta_\alpha(f) - K$, the subthreshold-seeker will perform at least as good as random search on a DLC. If the subthreshold-seeker has a chance to conduct local search, it will strictly outperform random search. Corollaries 15 and 16 show that if d and C remain unchanged, we can obtain a DLC satisfying the conditions by increasing $|V(G)|$, for α as a predefined performance threshold. In other words, with the same neighborhood structure, if the capability of a subthreshold-seeker to sample a decent threshold is unaffected by the increasing domain size, which is generally true and will be discussed later, then the conditions in Corollaries 15 and 16 hold with a sufficiently large domain. Estimating θ within some range should be more practical than gauging a specific value such as $\beta_\alpha(f)$. In the next section, we will explore this possibility and empirically confirm the theoretical results obtained in this section by proposing and adopting a sampling-test scheme.

5. Sampling-test scheme

Conventionally, the effectiveness of an optimizer is examined via experiments on a suite of test functions that serves as a benchmark. These test functions are selected according to some prior knowledge of the importance thereof. Here we propose and adopt a different approach in order to confirm the theoretical results obtained in the previous section from an empirical aspect. We draw a sample of functions randomly from PDLC in a manner similar to select respondents in a campaign survey and conduct experiments on these sampled functions. There is no bias in favor of which functions should be selected. We expect the arbitrariness delivers information about the composition of the problem class.

A uniform sampler for PDLC is firstly given in Section 5.1. Experiments are then presented to summarize this section and demonstrate how the Lipschitz condition facilitates the search process in a practical standpoint.

5.1. A uniform sampler for PDLC

In order to conduct the sampling test, we need a uniform sampler in the first place. The following algorithm generates problem instances of PDLC with Lipschitz constant K u.a.r.

Algorithm 2 (*Uniform PDLC Sampler*).

```
procedure UNIFORM PDLC SAMPLER($\overline{v_1 v_2 \ldots v_n}$, $\mathcal{Y} = \{0, 1, \ldots, m\}$, $K$)
   $f(v_1) \leftarrow Uniform([0, m])$
   $i \leftarrow 2$
   while $i \le n$ do
      $f(v_i) \leftarrow f(v_{i-1}) + Uniform([-K, K])$
      if $f(v_i) > m$ or $f(v_i) < 0$ then
         $f(v_1) \leftarrow Uniform([0, m])$
         $i \leftarrow 1$;
      end if
      $i \leftarrow i + 1$
   end while
   return $f$
end procedure
```

Here $Uniform([a, b])$ denotes the function that selects an integer u.a.r. from the closed interval $[a, b]$. Such a sampler belongs to the category of accept–reject algorithms [25]. It generates a problem instance with bounded difference between any two successive vertexes u.a.r., and if the instance at hand exceeds the range of the codomain, the sampler rejects the instance. The accept–reject mechanism guarantees the uniformity. Once the sampler halts, the output is always an instance of the PDLC.

Since this sampler is a Las Vegas algorithm, in which the answer is guaranteed to be correct but the resources used are not bounded [26], we need to address its time complexity for the practicality. For each candidate instance, the sampler will go through at most $|\mathcal{X}|$ steps to assign all the vertex objective values, so it remains to show how many candidate instances it takes to generate a legit instance successfully. The accept–reject process is geometrically distributed, and therefore the expected number of instances consumed is the inverse of the acceptance probability. The following theorem provides an upper bound for the rejection probability.

Lemma 17. *Suppose $|\mathcal{Y}| = 2m+1$, where m is an integer, and $|\mathcal{X}| = n$. If*

$$m > \sqrt{\frac{(n-1)(K^2+K)}{3}} \geq 2,$$

then the rejection probability is less than

$$\frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}|\mathcal{Y}|} - \frac{4(n-1)(K^2+K)}{3|\mathcal{Y}|^2} + \frac{5}{|\mathcal{Y}|}.$$

Proof. Without loss of generality, suppose $\mathcal{Y} = \{-m, -m+1, \ldots, m\}$. Let (K_i) be a sequence of i.i.d. random variables that $K_i = j$ with probability $1/(2K+1)$ for $j \in \{-K, -K+1, \ldots, K\}$ and $S_j := \sum_{i=1}^{j} K_i$. When $f(v_1) = i$, the instance is rejected if and only if $i + S_j \geq m+1$ or $i + S_j \leq -m-1$ for some $1 \leq j \leq n-1$, so the occurrence of rejection always implies $\max_{1\leq j\leq n-1} |S_j| \geq \min\{|m+1-i|, |-m-1-i|\}$. Moreover, the symmetry indicates that $Prob\{\text{rejection} \mid f(v_1) = i\} = Prob\{\text{rejection} \mid f(v_1) = -i\}$ for $|i| \leq m$. Therefore,

$$\begin{aligned}
Prob\{\text{rejection}\} &= \sum_{i=-m}^{m} Prob\{\text{rejection} \mid f(v_1) = i\} Prob\{f(v_1) = i\} \\
&= \frac{\sum_{i=-m}^{m} Prob\{\text{rejection} \mid f(v_1) = i\}}{2m+1} \\
&\leq \frac{Prob\{\max_{1\leq j\leq n-1} |S_j| \geq m+1\} + 2\sum_{i=1}^{m} Prob\{\max\limits_{1\leq j\leq n-1} |S_j| \geq m+1-i\}}{2m+1} \\
&= \frac{Prob\{\max_{1\leq j\leq n-1} |S_j| \geq m+1\} + 2\sum_{i=1}^{m} Prob\{\max\limits_{1\leq j\leq n-1} |S_j| \geq i\}}{2m+1}.
\end{aligned}$$

Using Kolmogorov's inequality [27], we can get

$$Prob\{\max_{1\leq j\leq n-1} |S_j| \geq i\} \leq \min\left\{\frac{Var[S_{n-1}]}{i^2}, 1\right\}.$$

Since $Var[K_i] = 2(1^2+2^2+\cdots+K^2)/(2K+1) = (K^2+K)/3$, $Var[S_{n-1}] = (n-1)Var[K_i] = (n-1)(K^2+K)/3$. Moreover, $Var[S_{n-1}]/i^2 \leq 1$ if and only if $i \geq \sqrt{Var[S_{n-1}]}$, we have

$$\begin{aligned}
Prob\{\text{rejection}\} &\leq \frac{\frac{Var[S_{n-1}]}{(m+1)^2} + 2\left(\sum\limits_{i=1}^{\lceil\sqrt{Var[S_{n-1}]}\rceil-1} 1 + \sum\limits_{i=\lceil\sqrt{Var[S_{n-1}]}\rceil}^{m} \frac{Var[S_{n-1}]}{i^2}\right)}{2m+1} \\
&\leq \frac{\frac{Var[S_{n-1}]}{(m+1)^2} + 2\left(\lceil\sqrt{Var[S_{n-1}]}\rceil - 1 + Var[S_{n-1}]\int_{x=\lceil\sqrt{Var[S_{n-1}]}\rceil-1}^{m} x^{-2}dx\right)}{2m+1} \\
&\leq \frac{\frac{Var[S_{n-1}]}{(m+1)^2} + 2\left(\sqrt{Var[S_{n-1}]} - \frac{Var[S_{n-1}]}{m} + \frac{Var[S_{n-1}]}{\lceil\sqrt{Var[S_{n-1}]}\rceil-1}\right)}{2m+1}.
\end{aligned}$$

Since $x/(x-1)$ decreases when $x > 1$, we obtain

$$\begin{aligned}
\frac{Var[S_{n-1}]}{\lceil\sqrt{Var[S_{n-1}]}\rceil - 1} &\leq \frac{Var[S_{n-1}]}{\sqrt{Var[S_{n-1}]} - 1} \\
&= \sqrt{Var[S_{n-1}]} + \frac{\sqrt{Var[S_{n-1}]}}{\sqrt{Var[S_{n-1}]} - 1} \\
&\leq \sqrt{Var[S_{n-1}]} + 2.
\end{aligned}$$

According to the assumption that $Var[S_{n-1}]/(m+1)^2 < 1$,

$$\begin{aligned}
Prob\{\text{rejection}\} &< \frac{4\sqrt{Var[S_{n-1}]} - \frac{2Var[S_{n-1}]}{m} + 5}{2m+1} \\
&= \frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}(2m+1)} - \frac{2(n-1)(K^2+K)}{3m(2m+1)} + \frac{5}{2m+1} \\
&< \frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}|\mathcal{Y}|} - \frac{4(n-1)(K^2+K)}{3|\mathcal{Y}|^2} + \frac{5}{|\mathcal{Y}|}. \quad \square
\end{aligned}$$

Theorem 18 (*Upper Bound for the Rejection Probability*). *Define $m := \lfloor (|\mathcal{Y}|-1)/2 \rfloor$. If $m > \sqrt{(n-1)(K^2+K)/3} \geq 2$, then the rejection probability is less than*

$$\frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}|\mathcal{Y}|} - \frac{4(n-1)(K^2+K)}{3|\mathcal{Y}|^2} + O\left(|\mathcal{Y}|^{-1}\right).$$

Proof. If $|\mathcal{Y}| = 2m+1$, then we are done by the previous lemma. Otherwise, if $|\mathcal{Y}| = 2m+2$, without loss of generality, suppose that $\mathcal{Y} = \{-m, -m+1, \ldots, m+1\}$ and let $\mathcal{Y}' = \{-m, -m+1, \ldots, m\}$. Therefore,

$$\begin{aligned}
Prob\{\text{rejection}\} &= Prob\{f(v_1) \in \mathcal{Y}'\} Prob\{\text{rejection} \mid f(v_1) \in \mathcal{Y}'\} \\
&\quad + Prob\{f(v_1) \notin \mathcal{Y}'\} Prob\{\text{rejection} \mid f(v_1) \notin \mathcal{Y}'\} \\
&= \left(\frac{2m+1}{2m+2}\right) Prob\{\text{rejection} \mid f(v_1) \in \mathcal{Y}'\} \\
&\quad + \left(\frac{1}{2m+2}\right) Prob\{\text{rejection} \mid f(v_1) = m+1\}.
\end{aligned}$$

When $f(v_1) \in \mathcal{Y}'$, if f exceeds the range of $\mathcal{Y}$, then f also exceeds the range of $\mathcal{Y}'$, so from the previous lemma we have

$$Prob\{\text{rejection} \mid f(v_1) \in \mathcal{Y}'\} < \frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}(2m+1)} - \frac{4(n-1)(K^2+K)}{3(2m+1)^2} + \frac{5}{2m+1}.$$

As a result,

$$\begin{aligned}
Prob\{\text{rejection}\} &\leq \left(\frac{2m+1}{2m+2}\right) Prob\{\text{rejection} \mid f(v_1) \in \mathcal{Y}'\} + \left(\frac{1}{2m+2}\right) \\
&< \frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}(2m+2)} - \frac{4(n-1)(K^2+K)}{3(2m+1)(2m+2)} + \frac{6}{2m+2} \\
&< \frac{4\sqrt{(n-1)(K^2+K)}}{\sqrt{3}|\mathcal{Y}|} - \frac{4(n-1)(K^2+K)}{3|\mathcal{Y}|^2} + O\left(|\mathcal{Y}|^{-1}\right). \quad \square
\end{aligned}$$

Corollary 19. *If $|\mathcal{Y}| = C\sqrt{(n-1)(K^2+K)} > C \cdot 2\sqrt{3}$ for some constant $C \geq \sqrt{3}$, then the rejection probability is less than*

$$\frac{4\sqrt{3}C - 4}{3C^2} + O\left(|\mathcal{Y}|^{-1}\right).$$

Proof. If $C \geq \sqrt{3}$,

$$\begin{aligned}
m = \left\lfloor \frac{|\mathcal{Y}|-1}{2} \right\rfloor &\geq \frac{|\mathcal{Y}|}{2} - 1 \\
&\geq \frac{\sqrt{3(n-1)(K^2+K)}}{2} - 1 \\
&= \sqrt{\frac{(n-1)(K^2+K)}{3}} + \frac{\sqrt{(n-1)(K^2+K)}}{2\sqrt{3}} - 1 \\
&> \sqrt{\frac{(n-1)(K^2+K)}{3}}.
\end{aligned}$$

Substituting $\sqrt{(n-1)(K^2+K)}/|\mathcal{Y}|$ by $1/C$ and applying Theorem 18, the corollary is proved. $\square$

For instance, if $C = \sqrt{3}$ and $|\mathcal{Y}|$ is so large that $O(|\mathcal{Y}|^{-1})$ is negligible, the expected number of instances consumed is no more than 9. Multiplying the time to assign all vertexes' values, the expected runtime, in terms of the number of assignments, is no more than $9|\mathcal{X}|$. In other words, asymptotically speaking, if $|\mathcal{X}|$ and $|\mathcal{Y}|$ are about equal and $|\mathcal{Y}|$ is larger than K^2 to some extent, then the expected runtime is approximately linear.

5.2. *Experimental settings and results*

As demonstrated in Section 4, the virtues of the subthreshold-seeker rely on a proper algorithmic threshold. Although the main results in Section 4 hold when $\theta \le \beta_\alpha(f) - K$, because we do not set a performance threshold literally to scrutinize how many subthreshold points are visited in real-world applications, in an experimental setting, we can examine the subthreshold-seeker more practically in terms of the time to identify the optimum. Therefore, the algorithmic threshold should be utilized for optimization, or more specifically, to minimize the objective function in this case.

We will compare the subthreshold-seeker with random search. Here we present three different subthreshold-seekers. For the theoretical purpose, the first one uses the actual median of all objective values, in the form of exterior knowledge, as θ. The second one firstly selects a 100 points u.a.r. and then employs the calculated median as θ. The third one also starts with obtaining 100 points u.a.r., but it computes the mean and the standard deviation of these points and sets θ to the mean minus the standard deviation. Moreover, the three subthreshold-seekers and random search obey the NFL framework and hence are non-repeating.

In advance of experiments, we need to determine the size of the set PDLC problems to be sampled. Suppose we want to estimate a population proportion $q \in [0, 1]$. We draw a sequence of samples uniformly and independently from the population with replacement. For each sample, we observe if it belongs to the variety of interest. With a large sample size, we expect the proportion in the sample approximates the real proportion. The following theorem depicts the relationship between the sample size and the error bound.

Theorem 20 (*Sample Size and Error Bound*)**.** *Let (Z_i) be a sequence of i.i.d. indicator variables with $E[Z_i] = q$. For all $\delta, \epsilon \in (0, 1)$, if*

$$n \ge -\frac{\ln(\delta/2)}{2\epsilon^2},$$

then

$$Prob\left\{\left|\frac{\sum_{i=1}^{n} Z_i}{n} - q\right| > \epsilon\right\} \le \delta.$$

Proof. Let $\overline{Z} = (\sum_{i=1}^{n} Z_i)/n$. Applying Hoeffding's inequality [28], for $0 < \epsilon < 1 - q$, we have

$$Prob\{\overline{Z} - q > \epsilon\} \le e^{-2n\epsilon^2},$$

and for $0 < \epsilon < q$,

$$Prob\{\overline{Z} - q < -\epsilon\} \le e^{-2n\epsilon^2}.$$

Moreover, if $\epsilon \ge 1 - q$,

$$Prob\{\overline{Z} - q > \epsilon\} \le Prob\{\overline{Z} > 1\} = 0 \le e^{-2n\epsilon^2}.$$

Similarly, if $\epsilon \ge q$,

$$Prob\{\overline{Z} - q < -\epsilon\} \le Prob\{\overline{Z} < 0\} = 0 \le e^{-2n\epsilon^2}.$$

Hence, we conclude that for all $\epsilon \in (0, 1)$,

$$Prob\{|\overline{Z} - q| > \epsilon\} \le 2e^{-2n\epsilon^2}.$$

Finally,

$$n \ge -\frac{\ln(\delta/2)}{2\epsilon^2}$$

implies $2e^{-2n\epsilon^2} \le \delta$, and we complete the proof. □

In particular, with the conventional setting of $(\epsilon, \delta) = (0.03, 0.05)$, a sample of size 2, 050 suffices. In other words, if we draw a sample of size 2, 050, $[\overline{Z} - 0.03, \overline{Z} + 0.03]$ forms a confidence interval for q with confidence level at least 95%.

Table 1
Successful rate.

θ	Category	$(\lvert\mathcal{X}\rvert, \lvert\mathcal{Y}\rvert)$ $(10^4, 10^4)$	$(10^5, 10^5)$	$(10^6, 10^6)$
γ	$>$	0.9995 (2,049)	0.9985 (2,047)	0.9976 (2,045)
	$> .2$	0.9951 (2,040)	0.9620 (1,972)	0.9624 (1,973)
$\hat{\gamma}$	$>$	0.9995 (2,049)	0.9990 (2,048)	0.9985 (2,047)
	$> .2$	0.9937 (2,037)	0.9620 (1,972)	0.9732 (1,995)
$\hat{\mu} - \hat{\sigma}$	$>$	1.0000 (2,050)	1.0000 (2,050)	1.0000 (2,050)
	$> .2$	1.0000 (2,050)	1.0000 (2,050)	1.0000 (2,050)

γ: median. $\hat{\gamma}$: estimated median. $\hat{\mu}$: estimated mean. $\hat{\sigma}$: estimated standard deviation. “$>$": proportion of instances where the subthreshold-seeker outperforms random search. “$>$ .2": proportion of instances where the subthreshold-seeker outperforms random search by a 20% margin.

Table 2
Mean time steps to locate the minimum.

Algorithm	$(\lvert\mathcal{X}\rvert, \lvert\mathcal{Y}\rvert)$ $(10^4, 10^4)$	$(10^5, 10^5)$	$(10^6, 10^6)$
STS, $\theta = \gamma$	2037.58	22913.23	229232.26
STS, $\theta = \hat{\gamma}$	2221.44	23170.58	229532.04
STS, $\theta = \hat{\mu} - \hat{\sigma}$	918.29	8095.78	80322.92
random search	4972.50	49724.74	496912.49

γ: median. $\hat{\gamma}$: estimated median. $\hat{\mu}$: estimated mean. $\hat{\sigma}$: estimated standard deviation.

The sampler generates 2,050 instances of PDLC with $(|\mathcal{X}|, |\mathcal{Y}|) = (10^4, 10^4)$, $(10^5, 10^5)$, and $(10^6, 10^6)$, respectively. The Lipschitz constant K is set to 100 for the concern of execution time, as previously discussed. For each problem instance, we test each algorithm for 50 independent runs. If the average time of a subthreshold-seeker to find the optimum is less than that of random search, the instance is counted as a success. We also count the number of instances that a subthreshold-seeker outperforms random search by a 20% margin, i.e., the instance where the average optimization time of a subthreshold-seeker is less than 80% of that of random search. Table 1 displays the empirical results.

All three subthreshold-seekers outperform random search in most of the sampled problem instances. Furthermore, the subthreshold-seeker with $\theta = \hat{\mu} - \hat{\sigma}$ outperforms random search in all 2,050 instances sampled, even with the requirement of a 20% margin. The statistical significance of such results is obvious to see: suppose the population proportion that the subthreshold-seeker with $\theta = \hat{\mu} - \hat{\sigma}$ outperforms random search is q. To obtain the result that random search is outperformed in all instances, the probability is q^{2050}. Even if q is as high as 0.995, the above probability is just 0.000034. To more formally rephrase, if the null hypothesis is “$q \le 0.995$”, the p-value is merely 0.000034.

Table 2 displays the averaged optimization time over the 2,050 sampled problem instances. The subthreshold-seeker with $\theta = \hat{\mu} - \hat{\sigma}$ outperforms others by a significant margin. Random search averages approximately $|\mathcal{X}|/2$ to find the minimum, which is expected. The subthreshold-seeker using the actual median and the one using the sample median both take about half time steps of that needed by random search to optimize the function.

The subthreshold-seekers with $\theta = \hat{\mu} - \hat{\sigma}$ and $\theta = \hat{\gamma}$ are indeed black-box algorithms, for there is no exterior knowledge exerted and the only information they can use are function evaluations, but they outperform random search by a remarkable difference.

The performance difference between $\theta = \hat{\gamma}$ and $\theta = \gamma$ is insignificant. Such a result suggests that in this case, an estimation of median may be adequate. Suppose that P with $|P| = N$ is a subset of real numbers, and for all $i \in P$, $R(i)$ is defined to be the rank (i.e., ordering) of i in P. For instance, $R(\min P) = 1$ and $R(\max P) = N$. For simplicity, we assume that N is odd and hence the median of P is the element i with $R(i) = \lceil N/2 \rceil$. Now we want to estimate the median of P. If a point sample S of size n, where n is assumed odd, is drawn by successively selecting an element u.a.r. from P with replacement, the estimated median, γ, is presumed to be the sampled median, and we want the error is bounded by $\epsilon > 0$, i.e., $|R(\gamma) - \lceil N/2 \rceil| \le \epsilon N$.

If $R(\gamma) < \lceil N/2 \rceil - \epsilon N$, there are at least $\lceil n/2 \rceil$ selections with ranks less than $\lceil N/2 \rceil - \epsilon N$. Let X_i be the indicator variable that indicates if the ith selection is less than $\lceil N/2 \rceil - \epsilon N$, $X_i = 1$ with probability $p := (\lceil N/2 \rceil - \lfloor \epsilon N \rfloor - 1)/N$. $R(\gamma) < \lceil N/2 \rceil - \epsilon N$ if and only if $\sum_{i=1}^{n} X_i \ge \lceil n/2 \rceil$. Since $E[\sum_{i=1}^{n} X_i] = np$, applying another form of Hoeffding's inequality [28], we have

$$Prob\left\{R(\gamma) < \left\lceil \frac{N}{2} \right\rceil - \epsilon N\right\} = Prob\left\{\sum_{i=1}^{n} X_i \ge \left\lceil \frac{n}{2} \right\rceil\right\}$$

$$
\begin{aligned}
&\leq Prob\left\{\sum_{i=1}^{n} X_i \geq \frac{n}{2}\right\}\\
&= Prob\left\{\frac{1}{n}\sum_{i=1}^{n} X_i \geq p + \left(\frac{1}{2} - p\right)\right\}\\
&\leq \left[\left(\frac{p}{p+\frac{1}{2}-p}\right)^{p+\frac{1}{2}-p}\left(\frac{1-p}{1-p-(\frac{1}{2}-p)}\right)^{1-p-(\frac{1}{2}-p)}\right]^{n}\\
&= [4p(1-p)]^{\frac{n}{2}}\,.
\end{aligned}
$$

Moreover, the symmetry implies that

$$
Prob\left\{R(\gamma) > \left\lceil\frac{N}{2}\right\rceil + \epsilon N\right\} \leq [4p(1-p)]^{\frac{n}{2}}\,.
$$

Therefore,

$$
Prob\left\{\left|R(\gamma) - \left\lceil\frac{N}{2}\right\rceil\right| > \epsilon N\right\} \leq 2\,[4p(1-p)]^{\frac{n}{2}}\,.
$$

Now the only quantity left is p. By definition,

$$
p = \frac{\lceil\frac{N}{2}\rceil - \lfloor\epsilon N\rfloor - 1}{N} \approx \frac{1}{2} - \epsilon.
$$

For instance, if we set $\epsilon = 0.1$ and $n = 100$, the probability of exceeding the error bound is less than 0.26. If the sample size n increases to 2, 000, even with a small $\epsilon = 0.03$, the probability reduces to just 0.054. It is noteworthy that the effect of the population size N is negligible. Therefore, the required number of samples remains the same, even if the search space is immense. Although in real-world applications P is usually a multiset, if the multiplicities of P are not too large, such a gauge should not diverge significantly.

6. Conclusions

In this study, we introduced and investigated the properties of the discrete Lipschitz class. A generalized subthreshold-seeker was then proposed and shown to outperform random search on this broad function class. Finally, we proposed a tractable sampling-test scheme to empirically demonstrate the performance of the generalized subthreshold-seeker under practical configurations. We showed that optimization algorithms outperforming random search on the discrete Lipschitz class do exist from both theoretical and practical aspects.

As controversial as it may be, the NFL theorem provides an alternative standpoint to review the position of optimization algorithms and search heuristics. The NFL theorem expels the false hope to conquer all possible functions with only limited information available, as it points out the expectation to find a universally black-box optimizer is definitely over-optimistic. However, the NFL theorem does not imply the utter infertility in the land of search heuristics by any means if our goals are appropriately placed. In this paper, the discrete Lipschitz class, as a simulation of continuous functions in a discrete space, is shown to be a class of problems on which black-box optimizers have performance advantages in both theory and practice. The only constraint imposed on the search space is bounded differences within a neighborhood. Under such a minor condition, black-box optimizers can still be effective over a broad, meaningful, and practical problem class as suggested by this study.

Acknowledgements

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] D.H. Wolpert, W.G. Macready, No free lunch theorems for search, Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute, 1995.
[2] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 4 (1997) 67–82.
[3] J.C. Culberson, On the futility of blind search: An algorithmic view of no free lunch, Evolutionary Computation 6 (1998) 109–127.
[4] S. Droste, T. Jansen, I. Wegener, Perhaps not a free lunch but at least a free appetizer, Tech. Rep. ISSN 1433-3325, Department of Computer Science, University of Dortmund, 1998.
[5] S. Droste, T. Jansen, I. Wegener, Optimization with randomized search heuristics – the (a)nfl theorem, realistic scenarios, and difficult functions, Theoretical Computer Science 287 (2002) 131–144.
[6] M.J. Streeter, Two broad classes of functions for which a no free lunch result does not hold, in: Proceedings of the Genetic and Evolutionary Computation Conference 2003, 2003, pp. 1418–1430.

[7] C. Igel, M. Toussaint, A no-free-lunch theorem for non-uniform distributions of target functions, Journal of Mathematical Modelling and Algorithms 3 (4) (2004) 313–322.
[8] S. Christensen, F. Oppacher, What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function, in: Proceedings of the Genetic and Evolutionary Computation Conference 2001, 2001, pp. 1219–1226.
[9] D. Whitley, J. Rowe, Subthreshold-seeking local search, Theoretical Computer Science 361 (2006) 2–17.
[10] H. Mülenbein, How genetic algorithms really work i. mutation and hillclimbing, in: Proceedings of the Parallel Problem Solving from Nature, 2, 1992, pp. 15–26.
[11] S. Droste, T. Jansen, I. Wegener, On the analysis of the $(1+1)$ evolutionary algorithm, Theoretical Computer Science 276 (2002) 51–81.
[12] J. He, X. Yao, Towards an analytic framework for analysing the computation time of evolutionary algorithms, Artificial Intelligence 145 (1–2) (2003) 59–97.
[13] P.S. Oliveto, C. Witt, Simplified drift analysis for proving lower bounds in evolutionary computation, in: Proceedings of Parallel Problem Solving from Nature, 2008, pp. 82–91.
[14] O. Giel, I. Wegener, Evolutionary algorithms and the maximum matching problem, in: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, 2003, pp. 415–426.
[15] F. Neumann, I. Wegener, Randomized local search, evolutionary algorithms, and the minimum spanning tree problem, Theoretical Computer Science 378 (1) (2007) 32–40.
[16] T. Jansen, I. Wegener, A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-boolean functions of unitation, Theoretical Computer Science 386 (1–2) (2007) 73–93.
[17] A. Auger, O. Teytaud, Continuous lunches are free!, in: Proceedings of the Genetic and Evolutionary Computation Conference 2007, 2007, pp. 916–922.
[18] J. Rowe, M. Vose, A. Wright, Reinterpreting no free lunch, Evolutionary Computation 17 (1) (2009) 117–129.
[19] C. Schumacher, M.D. Vose, L.D. Whitley, The no free lunch and problem description length, in: Proceedings of the Genetic and Evolutionary Computation Conference 2001, 2001, pp. 565–570.
[20] R. Courant, F. John, Introduction to Calculus and Analysis, Vol. 1, Springer-Verlag, 1989.
[21] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd edition, The MIT Press, 2001.
[22] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
[23] I. Rechenberg, Evolutionsstrategie '94, Frommann Holzboog, 1994.
[24] J. Jägersküpper, Algorithmic analysis of a basic evolutionary algorithm for continuous optimization, Theoretical Computer Science 379 (3) (2007) 329–347.
[25] C. Robert, G. Casella, Monte Carlo Statistical Methods, Springer-Verlag, 1999.
[26] Mikhail J. Atallah (Ed.), Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, ISBN-10: 0849326494, ISBN-13: 978-0849326493.
[27] Y.S. Chow, H. Teicher, Probability Theory: Independence, Interchangeability, Martingales, 3rd edition, Springer, 1997.
[28] W. Hoeffding, Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association 58 (1963) 13–30.

WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

Sparse Degrees Analysis for LT Codes Optimization

Pei-Chuan Tsai
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: pctsai@nclab.tw

Chih-Ming Chen
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: ccming@nclab.tw

Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: ypchen@nclab.tw

***Abstract*—Luby Transform (LT) codes are a new member in the family of forward error correction codes without a fixed code rate. The property called *rateless* is attractive to researchers in last decade, and lots of studies have been proposed and attempted to improve the performance of LT codes. One variation is the use of a sparse degree distribution instead of a full one referred to in the encoding process of LT codes to reduce the search space. Observing a fact that the ability of a sparse degree distribution is limited by the nonempty degrees, we introduce a tag selection scheme to choose reasonable sparse degrees for LT codes in this paper. We firstly investigate the influence of different degrees on the error rate of LT codes and then propose a general selection algorithm based on our observations. After that, the covariance matrix adaptation evolution strategy (CMA-ES) is applied to find the optimal sparse degree distributions of which the degrees are defined by our selection algorithm. Finally, the experimental results are presented as evidence to show the proposed scheme is effective and practical.**

I. Introduction

Digital fountain [1], [2] is a new class of forward error correction codes, which are used in erasure channel for recovering erased data. The most important property of digital fountain codes is *rateless*. The property allows encoder to generate encoded packets unlimitedly without a fixed code rate. Encoded packets are continuously delivered like a fountain, and any receiver can reconstruct the source data once a sufficient amount of packets are received without considering the order. It means that the performance of digital fountain codes is independent of channel parameters, such as the erasure rate. It is a great advantage to deliver data through an unclear channel or broadcast information to many users whose connection qualities are unequal. The first practical implementation of digital fountain is Luby Transform (LT) codes [3] proposed by Michael Luby in 2002. LT codes are not only rateless but also have low computation complexity due to its simple coding procedure. As other linear block codes, the coding structure of LT codes can be expressed as a tanner graph. Tanner graph is a bipartite graph consists of bit nodes, check nodes and connection edges for indicating the coding relation between the nodes. To generate a check node, LT codes decide the degree of a node according to a particular probability distribution called *degree distribution*. The performance of LT codes totally depends on the size of bit nodes, k, and a given degree distribution. Luby hence suggested two general expressions to define practical degree distributions for LT codes. However, the proposed distribution named as *soliton distribution* is near optimal only when the size of source data gets close to infinite. In other words, there is still room to improve LT codes with moderate source data size.

The approach to improve the LT codes can be modeled as an optimization problem in which the probability on each degree is considered as a decision variable and the objective is to search for degree distributions with better performance. Many kinds of methods have been proposed to solve such an optimization problem. The researchers firstly focused on optimizing the performance of LT codes for a short data length [4], [5]. [5] even presented an numerical approach which can compute the optimal degree distributions. Unfortunately the optimization scale is limited to $k \leq 30$ due to huge computational cost. For a large k size, [6] made the first attempt to apply heuristic search algorithms to optimize degree distributions. [7] and [8] are our previous studies in which evolutionary algorithms were introduced to address the issue and several distributions better than soliton distributions were obtained for LT codes, while [9] presented the similar idea and focused on maximizing the intermediate recovery rate of LT codes by using multi-objective optimization algorithms.

Studies show the feasibility to optimize degree distributions by evolutionary algorithms. In the optimization framework, the challenge of huge search space is a key issue. The problem dimensionality is equal to the source data length k since an individual should represent the probability on each degree from 1 to k. It is hard to deal with so many variables while k rises to hundreds. Adopting a *sparse degree distribution* is an alternative solution which has been frequently used in LT codes optimization. In a sparse degree distribution, nonzero probabilities distribute on only partial degrees which are predefined by a set of *tags*. The set of tags is a subset of all degrees and used to limit the search in a sub-space whose size is much less than full degrees.

[6] chose the power of 2 to compose sparse degree distributions for optimization, and Fibonacci numbers were used in [7], [8]. All the previous studies proposed practical approaches which can find better distributions in a sparse form. However, the best distribution in sub-space defined by given tags is merely near-optimal for LT codes. Previously, the tags were usually decided according to experimental experience, and there exists no complete investigation on how to choose tags. As a result, this paper employs evolutionary algorithms as a research tool and observes the influence of different

tags on decoding error probability of LT codes. Based on the observations, a sparse degree selection algorithm is proposed to define appropriate tags, and the optimal distribution formed by the selected tags may get close to the global optimal.

The remainder of the paper is organized as follows. Section II gives the details of coding mechanism of LT codes and introduces two evaluation approaches for degree distributions. Section III investigates the probability reallocation of degree distributions and presents the influences of different tags. In section IV, our selection algorithm is proposed and described in detail. Several optimization results with different parameter settings are presented in section V to examine the selection algorithm. Finally, the conclusion and our contribution are given in section VI.

II. LT CODES

Before describing main work of the paper, the operation of LT codes is introduced in this section as a background. The encoding and decoding procedures are given in section II-A. Source data in general are divided into k fragments with an identical length. These fragments are bit nodes or called *input symbols* if the length of each fragment is only a bit. Similarly *encoding symbols* denote the codewords generated by the encoding procedure in LT codes. For a clear presentation, the terms, input symbols, and encoding symbols, are consistently used in this paper. Section II-B introduces *soliton degree distributions*, which were proposed according to a theoretical analysis. Furthermore, because evaluating the quality of degree distributions is necessary in optimization, section II-C introduces two different approaches for evaluating the decoding error rate of LT codes for a given degree distribution.

A. Encoding and Decoding

Given the source data, we suppose that the source data are cut into k input symbols. Before an encoding symbol is generated, a degree d is chosen at random according to the adopted degree distribution $\Omega(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \Omega(d) = 1$. The degree d decides how many distinct input symbols are involved to compose an encoding symbol. Then d input symbols, also named *neighbors*, are chosen uniformly at random and accumulated by XOR. In the design of LT codes, random number plays an essential role in the encoding process. The approach employed by LT codes for a sender to inform receivers of all information is to synchronize a random number generator with a specified seed.

At the receiver side, when n, which is usually slightly larger than k, encoding symbols arrive, *belief propagation* is used to reconstruct the source data step by step. All encoding symbols are initially covered in the beginning. In the first step, all encoding symbols with only one neighbor can be directly released to recover their unique neighbor. For an input symbol that has been recovered but not processed yet, it is called a *ripple* and will be pushed into a queue. At each subsequent step, ripples are popped from the queue as a processing target one by one. A ripple is removed from all encoding symbols that have it as a neighbor. If an encoding symbol has only one remaining neighbor after 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 becomes empty with uncovered input symbols. In other words, more encoding symbols are required to continue the decoding process. The decoding process succeeds if all input symbols are recovered at the end.

B. Soliton distribution

The coding behavior of LT codes is determined by the degree distribution, $\Omega(d)$, and the number of received encoding symbols, n. Reception overhead, $\varepsilon = n/k$, denotes the delivery efficiency of LT codes, and ε depends on the given degree distribution. Based on the theoretical analysis, Luby proposed the *ideal soliton distribution* which can achieve the best performance, $\varepsilon = 1$, 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 decoding step. Hence, there is exactly one expected ripple generated at each step when the encoding symbol size is k. After k processing step, ideally, the source data can be fully recovered.

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 that 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 size that must be maintained to complete the decoding process. Therefore, in the same paper by Luby, a modified version called *robust soliton distribution* was also proposed.
Robust soliton distribution $\mu(d)$:

$$S = c \ln(k/\delta)\sqrt{k} \, ,$$

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

$$\beta = \sum_{d=1}^{k} (\rho(d) + \tau(d)) \, , \tag{3}$$

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k \, , \tag{4}$$

where c and δ are two parameters for controlling the character of a robust soliton distribution. c controls the average degree of the distribution, and δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripples as aforementioned. Robust soliton distribution can ensure that only $n = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoding symbols are required to recover the source data successfully with a probability at least 1-δ.

C. Evaluation of Degree Distributions

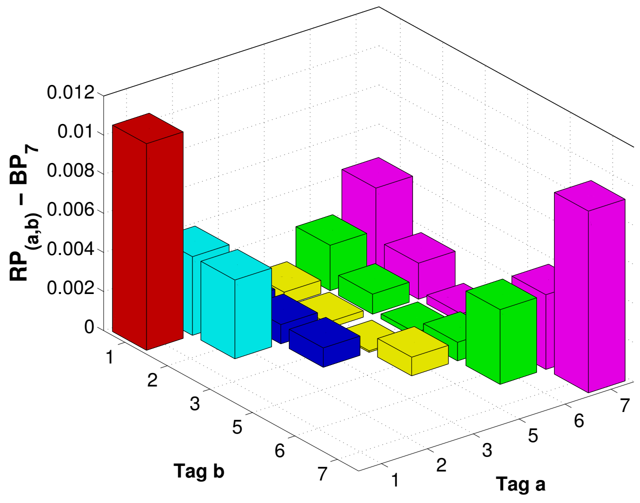
In this section, we outline two approaches to evaluate degree distributions. One intuitive indicator for evaluating degree distributions is the reception overhead, ε. For the reasons that the coding process of LT codes is stochastic and there is uncertainty in the transmission channel, a successful decoding cannot be guaranteed in the condition of a constant overhead. Moreover, in order to obtain the average overhead, a large amount of simulations are required to estimate ε. Therefore, an alternative approach is to evaluate the error probability of LT codes with some particular reception overhead. In 2004, an effective evaluation method was proposed in [10] for LT codes with a finite size of k. Dynamic programming was utilized to construct the distribution of ripple size during the decoding process. In short, the error probability of LT codes can be deterministically evaluated by the method when the input symbol size k and a fixed overhead are given. Unfortunately, the computation considers a 3-dimensional matrix of which the edge size is k and the complexity is too high. Even when several reduction techniques are applied, the computational complexity is still $\mathcal{O}(k^3 \log^2(k) \log\log(k))$. Subsequently, a new model for rigorous analysis on LT codes has been proposed in 2006 [11]. The difference in this approach is to make an assumption that the number of received symbols is a random variable with mean n. Based on the assumption, a fast evaluation with time complexity $\mathcal{O}(k^2 \log(k))$ was presented. Both approaches are feasible to serve as an evaluation function and are employed in this paper for different requirements.

III. Probability Reallocation

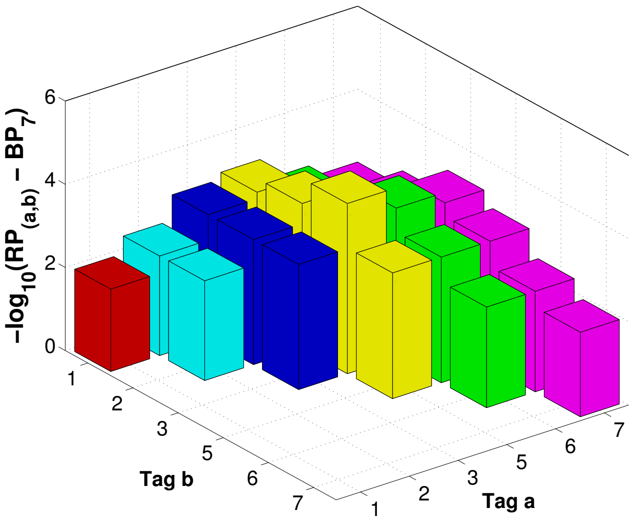
For seeking a reasonable strategy to select sparse tags, we conduct experiments to observe the effects of the probability reallocation on the degree distribution. We first define the probability reallocation process. For a given optimal degree distribution $\Omega(d)$, we choose degrees i, a, and b, where $a, b \neq i$ to apply reallocation and get the new degree distribution $\Omega'(d)$:

$$\Omega'(d) = \begin{cases} \Omega(d) & \text{for} \quad 1 \leq d \leq k, d \neq i, a, b \\ 0 & \text{for} \quad d = i \\ \Omega(d) + \Delta(d) & \text{for} \quad d = a, b \end{cases},$$

where $\Delta(a) + \Delta(b) = \Omega(i)$, i.e. reallocate the probability of degree i to degrees a and b. After determing degrees i, a, and b, an evolutionary algorithm is applied to find out the best distribution of the probability reallocation on these degrees. The objective of the best distribution denotes the least increment of error probability after reallocation. In [5], a method that can find the optimal degree distributions for LT codes with small k size was presented. The obtained optimal degree distributions guarantee the minimal error probability for overhead $\varepsilon = 1$. Calculation of the error probability for LT codes at $\varepsilon = 1$ can be done by Karp's evaluation approach. Hence, the optimal degree distribution for input symbol size $k = 7$ was chosen as our experimental subject and the covariance matrix adaptation evolution strategy [12], [13], [14](CMA-ES) is employed to search for the best reallocation. In the experiment, we denote the minimal error probability for input symbol size $k = 7$ as BP_7 and for reallocated distribution with pair (a, b) as $RP_{(a,b)}$. Figure 1(a) shows the differences of error probabilities between the optimal and the reallocated degree distributions.



(a) Error probability variances



(b) Error probability variances in log scale

Fig. 1. Error probability variances after reallocation.

In the experiment, we choose the optimal degree distribution for $k = 7$ as the observation target and redistribute the probability of degree 4 to any combinations of degree (a, b) with $a, b \neq 4$. The results illustrate two factors of error probability variances that: 1) the distance between the reallocated and the removed degrees; 2) the complementary property of the reallocated degrees. For the first factor, the figure shows that the differences of error probabilities would grow as the distances of a, b to 4 become larger. For instance, if we fix $a = 1$ and change b from 1 to 7 excluding 4, it can be found that the differences would be decreased when the value of b close to 4. For the latter, the complementary property means one of the pair (a, b) is bigger than the removed degree and the other is smaller would be better. It can be

Algorithm 1 Tags Selection Function

```
Input: The source symbol size $k$, the density parameter $d$;
Output: The set of sparse tags;
 1: procedure TSF($k$, $d$)
 2:     $D \leftarrow$ Ideal soliton distribution for size $k$;
 3:     $S \leftarrow 0, E \leftarrow 1, Tags \leftarrow []$;
 4:     while $i < k$ do
 5:         $S \leftarrow S + E \times D(i)$
 6:         if $S > (1/d)$ then
 7:             $Tags \leftarrow [Tags, i]$;
 8:             $S \leftarrow 0$;
 9:             $E \leftarrow 1$;
10:         else
11:             $E \leftarrow E + 1$;
12:         end if
13:     end while
14:     return vector $[1, Tags, k]$;
15: end procedure
```

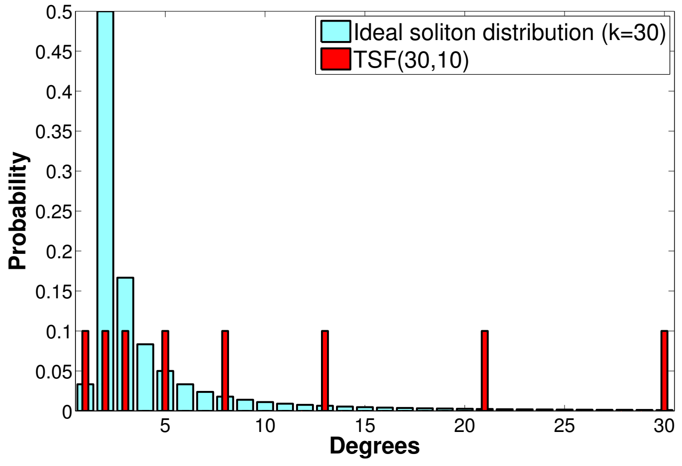


Fig. 2. Illustration of the work done by TSF(k, d).

observed by comparing the selections of $(a, b) = (a_1, 4 + i)$ to $(a, b) = (a_1, 4 - i)$, e.g. comparisons between $(1, 2)$ and $(1, 6)$. The distance influence of such two pairs are the same but the pair $(1, 6)$ with complementary property observably has a smaller difference.

Given the above considerations, it turns out that the reallocation of the probability to degrees 3 and 5, the nearest two degrees to 4, would be the most close to the optimum. For a better view of showing how close the adjusted distributions could approach to the optimal one, we illustrate the differences of error probabilities in logarithm scale in Figure 1(b). It also gives the proof that the error probability of the original distribution could be well approximated by the reallocated one. These observations inspire us regarding how to choose the tags for a sparse degree distribution.

IV. Selection function for Sparse Tags

In addition to the factors observed from experiments, we also take some intuitive properties into account. For example, the higher probability to be reallocated would result in higher difference of the error probability to the optimal one. Summing up all of above, the main considerations of our degree selection strategy for sparse tags would be as follows:

1) The number and value of probabilities around each degree.
2) The distance between the probability reallocated degrees and the removed one.

The first criterion comes from the positive correlation of the reallocated probability and the error probability. The second one accounts for the results of the experiments that replacing the tag to be removed with two adjacent tags would have the best approximation for error probability. According to these criterion, we propose the sparse tags selection function in Algorithm 1.

We consider the *ideal soliton distribution* since it would be the optimal degree distribution in the ideal case. The density parameter d acts as the bound that base on the first main consideration, degree i would not be removed if its probability was larger than $1/d$. On the other hand, we group the degrees with probabilities below $1/d$ and concentrate those probabilities to a nearby degree. The second main consideration would be applied to the selection of the representative degree of each group. We accumulate the probabilities multiplied the distance factors and take the degree while the sum exceeds the bound $1/d$. In addition, considering the complementary property of the selected degrees to distribute the probability, we reserve the degrees 1 and k to ensure there always exists degrees satisfied this property to be chosen. Figure 2 illustrates the work done by Tags Selection Function (TSF) and shows the tags selected for $(k, d) = (30, 10)$. Tags 1 and 30 were selected to meet the complementary property. The tags $2, 3$ were selected since the probabilities of these tags in ideal soliton distribution were above the density criteria. The remaining tags were the representations of the grouped tags of which probabilities were below the density bound.

Following we provide some examples of sparse tags selected by Algorithm 1 for $k = 100$:

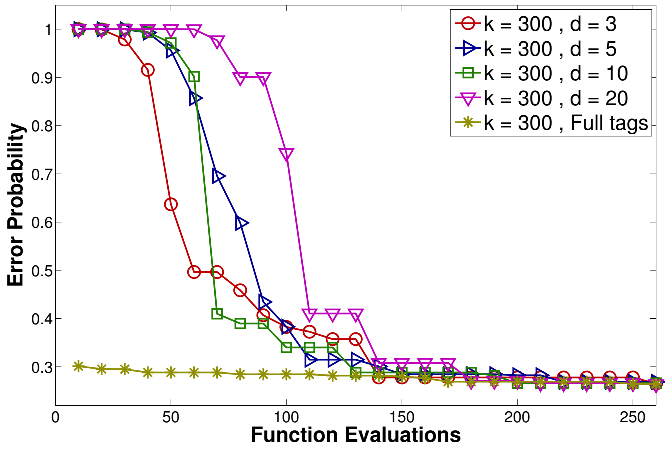
1) TSF$(100, 1)$ = [1, 4, 23, 100].
2) TSF$(100, 3)$ = [1, 2, 5, 12, 30, 76, 100].
3) TSF$(100, 5)$ = [1, 2, 4, 8, 16, 32, 64, 100].
4) TSF$(100, 10)$ = [1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 100].
5) TSF$(100, 20)$ = [1, 2, 3, 4, 6, 8, 11, 15, 21, 29, 40, 56, 78, 100].

We can see that the sparse tags selected in TSF$(100, 5)$ were close to the series of power of 2 and those selected in TSF$(100, 10)$ were close to Fibonacci series. Such a result gives an explanation for the good performance of choosing these series, power of 2 and the Fibonacci series, as approximation to the full tags in a certain extent.

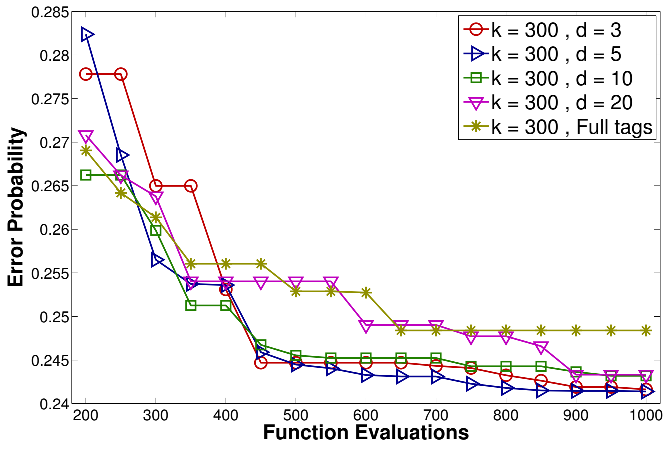
TABLE I

THE ERROR PROBABILITIES OF THE BEST DEGREE DISTRIBUTIONS DISCOVERED IN EACH 30-RUN EXPERIMENT.

Tag Type	k = 100	k = 150	k = 200	k = 250	k = 300
d = 3	0.56332720	0.46689819	0.37998428	0.30450876	0.24139928
d = 5	0.56296776	0.46627322	0.37872813	0.30325441	0.24007702
d = 10	0.56293832	0.46622717	0.37871451	0.30324585	0.24008108
d = 20	0.56291529	**0.46619257**	**0.37866506**	**0.30319090**	**0.24001890**
Full Tags	**0.56291403**	0.46619313	0.37869051	0.30323273	0.24023868
Min.	0.56291403	0.46619257	0.37866506	0.30319090	0.24001890



(a) Results in early stages for observing different initializations



(b) Results for sparse degree distributions converge faster

Fig. 3. The evolutionary trends of fitness values in optimization.

V. Experimental Results

In this section, we make a complete examination on the effects of the proposed sparse tags selection function. We set up the input parameters $k = \{100, 150, 200, 250, 300\}$ and $d = \{3, 5, 10, 20\}$. For each (k, d) pair, we firstly use the TSF function to determine the corresponding tags and then apply the CMA-ES algorithm to search for the optimal sparse degree distribution with these tags. Finally, we compare the minimal error probabilities of the sparse degree distribution and the full degree distribution. For each optimization setting, 30 independent runs were tested due to the natural randomness of evolutionary algorithms. The minimal error probability in 30 runs was recorded for each generation. The maximum number of function evaluation is limited to 3×10^4 and the optimization result is presented in Figure 3. Figure 3(a) shows the early phase of optimization process. The results of all sparse degree distributions are with high error rates at beginning because the initial probabilities on sparse degrees were set by random values. In contrast, the full degree distribution was initialized as ideal soliton distribution to avoid the failure of optimizing a large number of decision variables. As the number of evaluations increases, the curves of sparse distributions drop down to the same level of error probability as the full degree distribution. Since the density parameter, d, of our selection function effects the size of tags, it also reflects on the convergence of the curve of each sparse distribution. It can be observed that the curve with $d = 3$ firstly converges and $d = 20$ is the last. Figure 3(b) shows the same experimental result but in a different interval of x-axis. We can clearly observe that under the same optimization approach, the optimized results of sparse distributions are better than that of the full degree after hundreds of function evaluations.

Fast convergence is an expected advantage for adopting sparse degree distributions. Furthermore, it can be expected that the tags defined by our selection strategy could approximate full degrees on performance as nicely as possible. To examine our argument, the minimal error probabilities of different tag types are listed in Table I for comparison. The values were the minimal results found by CMA-ES in 30 runs and 3×10^4 function evaluations. For each column, the minimal error probability is marked as bold and copied to the last row. The sparse degree distribution with $d = 20$ leads four of totally five different k size experiments. For giving a more convenient view to compare the results, we survey the differences between each entry and the minimal value in same column. Accounting for the minimal error probability of the column will be zero after all entries minus the minimal one, we add a base value (10^{-7}) to calculate differences for letting all data be plotted in a logarithmic coordinate system. Figure 4 visualizes the values in Table I and shows the distance between each tag types and the near optimal distribution that we have found. Although a larger value of d which means a larger subset of degrees will cause slow convergence in optimization, more tags can form a sparse distribution with a lower error probability. Our experimental result definitely confirms the argument and also illustrates that the selection strategy is practical. On the other hand, the full degree distribution is

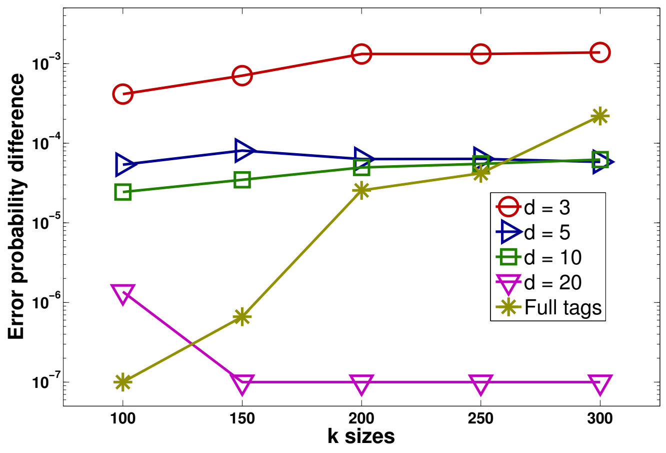


Fig. 4. Differences between the error probabilities and the minimal one.

considered to have global minimal error probability because it is the universal set of all distributions and forms the complete search space. However, the optimization result of full degree distributions gets worse and worse as input symbol size k increases. For the same optimization algorithm and evaluation function are implemented for each degree set, the worse results of full degree distributions could be explained by that the number of decision variables is too many for CMA-ES to handled in limited function evaluations.

VI. Conclusion

Using evolutionary algorithms to optimize the degree distribution for LT codes is a promising research topic. Sparse degree distributions are frequently used to replace full degrees for reducing the search space. How good the performance could be achieved by a sparse degree distribution depends on the set of its non-zero entries, i.e., tags. However, no investigation has been done regarding how to decide appropriate tags to construct sparse degree distributions with good performance. In this paper, the authors analyzed the influence of different degrees on decoding rate and proposed a tag selection algorithm to choose tags for LT codes optimization. Finally, the presented experimental results were evidentially illustrate the practicality of the proposed tag selection algorithm.

In previous studies, researchers manually chose tags for sparse degree distributions according to their own experimental experience. Even though the chosen subset of degrees worked well, the detailed mechanism was still unknown. This work made an effort to find out guidelines for choosing appropriate tags. The proposed selection algorithm can be applied for any input size and control the level of sparseness conveniently by adjusting the density parameter. This solution can help researchers to put more attention in the optimization algorithm rather than the individual encoding. The paper presented the qualitative analysis of probability reallocation in a distribution. The variances of error probability were compared for changing the reallocated degree and then quantitative analysis is needed in advance. If the quantity of variance can be measured precisely, developing a local search based on the measure approach to enhance certain optimization framework for LT codes will be possible. Research of this line is definitely worth pursuing, and the authors are currently taking the challenge.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2.

References

[1] 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*. ACM, 1998, pp. 56–67.

[2] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, 2002.

[3] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, 2002, pp. 271–282.

[4] E. A. Bodine and M. K. Cheng, "Characterization of Luby Transform codes with small message size for low-latency decoding," in *Proceedings of the IEEE International Conference on Communications*, 2008, pp. 1195–1199.

[5] E. Hyytiä, 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.

[6] ——, "Optimizing the degree distribution of LT codes with an importance sampling approach," in *Proceedings of the 6th InternationalWorkshop on Rare Event Simulation (RESIM 2006)*, 2006, pp. 64–73.

[7] C.-M. Chen, Y.-p. Chen, T.-C. Shen, and J. K. Zao, "On the optimization of degree distributions in LT code with covariance matrix adaptation evolution strategy," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2010, pp. 3531–3538.

[8] ——, "Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2010, pp. 3635–3642.

[9] A. Talari and N. Rahnavard, "Rateless codes with optimum intermediate performance," in *Proceedings of the Global Telecommunications Conference (GLOBECOM 2009)*, 2009, pp. 1–6.

[10] 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.

[11] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2006)*, 2006, pp. 2677–2679.

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

[13] 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.

[14] ——, "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.

WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

When and What Kind of Memetic Algorithms Perform Well

Jih-Yiing Lin
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: jylin@nclab.tw

Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City, TAIWAN
Email: ypchen@nclab.tw

***Abstract*—The synergy between exploration and exploitation has been a prominent issue in optimization. The rise of memetic algorithms, a category of optimization techniques which feature the explicit exploration-exploitation coordination, much accentuates this issue. While memetic algorithms have achieved remarkable success in a wide range of real-world applications, the key to a successful exploration-exploitation synergy still remains obscure. Manifold empirical results and theoretical derivations have been proposed and provided various perspectives from different algorithm-problem complexes to this issue. In our previous work, the concept of *local search zones* was proposed to provide an alternative perspective depicting the general behavior of memetic algorithms on a broad range of problems. In this work, based on the local search zone concept, we further investigate how the problem landscape and the way the algorithm explores and exploits the search space affect the performance of a memetic algorithm. The collaborative behavior of several representative archetypes of memetic algorithms, which exhibit different degrees of explorability and exploitability, are illustrated empirically and analytically on problems with different landscapes. As the empirical results consist with the local search zone concept and describe the behavior of various memetic algorithms on different problems, this work may reveal some essential design principals for memetic algorithms.**

I. Introduction

Optimization, finding the optimal element among a set of feasible ones, is a type of problems commonly encountered in many fields. Numerous real-world and theoretical problems can be formulated as optimization problems and solved by applying or developing various optimization techniques. Among them, general meta-heuristics, population-based algorithms which explore the search space stochastically according to some common heuristics, exhibit good explorative ability and have a good chance to perform well on many real-world optimization problems which are generally black-box problems with little *a priori* problem knowledge available. Some of the renowned meta-heuristics are evolutionary algorithms, particle swarm optimization, ant colony algorithms, and the like. However, the generality of meta-heuristics which provides the wide applicability also limits the efficiency of meta-heuristics. When complicated problems are encountered, without taking advantages of problem specific information given *a priori* or retrieved during optimization, meta-heuristics can merely deliver mediocre performance. In an attempt to incorporate the good explorative ability of general meta-heuristics and the good exploitive performance of problem specific algorithms to provide more efficient techniques for more complicated problems, techniques which employ general meta-heuristics as global search and problem specific algorithms as local search, referred to as memetic algorithms (MAs), have thrived. A variety of successful memetic algorithms in various domains, ranging from NP-hard combinatorial problems to non-linear programming problems, also have been reported [1].

Among these memetic algorithms, the synergy between global search and local search has always been one of the key design issues. The seminal study on memetic algorithms [2] and its succeeding work [3] both suggest that memetic algorithms favor infrequent starts and long running time of local search. They also proposed several renowned strategies for selecting solution candidates on which the local search operator is applied: the fitness based selection and the diversity based selection. However, with the aids of these guidelines, designing a memetic algorithm for a specific problem still requires considerable time as the optimal design is not only algorithm specific but also problem dependent. Parameterizing difficulties in the design of memetic algorithms have been practically encountered in applications and also theoretically proved on several problem classes [4], [5]. To cope with this issue, memetic algorithms have been evolved from hybridization of global search and local search to hybridization with adaptation [6]–[8]. These algorithms adopting adaptive local search, referred as to memes, are robust and efficient with the expense of the learning cost of memes.

These studies on different algorithm-problem complexes have provided manifold aspects to the behavior of different memetic algorithms on different problems. In our previous work [9], we proposed the concept of *local search zone* to provide an alternative perspective which aims to depict the general behavior of memetic algorithms. In the previous work, a theoretical model depicting the exploration-exploitation synergy of the subthreshold seeker, a representative archetype of memetic algorithms, on different Quasi-Basin Classes (QBCs) was formulated to represent the general behavior of collaboration between global search and local search in memetic computation on a broad class of objective functions. As the theoretically and empirically verified model not only well depicts the collaborative behavior of the representative archetype of memetic algorithms but also consists with the empirical results in quite a few studies in the literature, further investigation on the effect of problem landscapes and how the explorability of global search and the exploitability of local search affect the optimization performance is possible.

In this work, as practical problems generally exhibit some degree of continuity rather than distributions of different sets of stochastic points described in QBCs, we further extend the QBC to Discrete Lipschitz Quasi-Basin Class(DLQBC) which categorizes Lipschitz continuous problems according to the number of basins and the roughness of landscape. Then we apply several representative archetypes of memetic algorithms on different DLQBCs to investigate the collaborative behavior of memetic algorithms exhibiting different explorability and exploitability on different problem landscapes. As the empirical results of this framework not only consist with the theoretical model proposed in our previous work but also well delineate how different types of memetic algorithms behave on different problems, this work may shed light into the design of memetic algorithms.

The rest of this paper is arranged in the following manner. We firstly introduce the framework of subthreshold seeker on QBCs as the basis of this study in Section II, then provide the definition of Discrete Lipschitz Quasi-Basin Class (DLQBC) and its sampling test scheme in Section III, and propose several representative archetypes of memetic algorithms for investigation in Section IV. The empirical results of the proposed representative archetypes memetic algorithms on different DLQBCs are presented and discussed in Section V. Finally, we conclude this paper in Section VI.

II. Subthreshold Seeker on Quasi-Basin Classes

In this section, the concept of local search zones and the theoretical model proposed in our previous work [9] are briefly reviewed to form the basis of this study.

A. Quasi-Basin Classes

In [9], an optimization problem is to optimize a given objective function $f : \mathcal{X} \to \mathcal{Y}$, and the optimization goal is to find $x^* \in \mathcal{X}$ with the minimum value $y^* \in \mathcal{Y}$. The $\mathcal{X}$ is assumed a finite set as optimization problems are generally numerically solved on digital computers. To simplify the derivation, every function maps different $x \in \mathcal{X}$ to different $y \in \mathcal{Y}$ is also assumed. For generality, the search space is interpreted as a graph viewed by an optimization algorithm. In this interpretation, the vertices are the set of points of $\mathcal{X}$ and the edges are the set of pairs of points which are neighbors viewed by an optimization algorithm. The terms $\mathcal{X}$ and $V(G)$ are used exchangeably in the following text.

Based on the fundamental definition of the search space, the local search zones are defined as regions where local search prefers and are virtually determined by the adopted local search criterion and the landscape of the search space. Thus, frameworks based on the concept of local search zones are especially suitable for investigating the collaboration between global search and local search. As local search zones are generally hard to measure, quasi-basins are used to approximate local search zones. Given a threshold $\beta_m(f)$, which forms a set $S_m(f)$ consisting m vertices with their objective values smaller than or equal to the threshold, a quasi-basin (QB) is defined as a maximal connected subset in $S_m(f)$. Accordingly, the Quasi-Basin Class(QBC) is proposed to define a class of problems with m *subthreshold points*, points with objective values that are smaller than or equal to $\beta_m(f)$, distributed among b distinct quasi-basins. As the vertices residing in quasi-basins are better than the other vertices in the search space and are favored by fitness-relevant local search criteria, QBCs categorize the problems according to their distribution of quasi-basins which is conceptually mapping to the distribution of local search zones.

B. Subthreshold Seeker

To illustrate how the distribution of local search zones affects the performance of a memetic algorithm, [9] adopts Subthreshold Seeker (SS) as a representative archetype of memetic algorithms. SS is a simplistic, minimal optimization algorithm that coordinates random global search and exhaustive local search according to a threshold. It explores the search space by uniformly randomly sampling the search space. When a subthreshold point is encountered by global search, SS exploits the quasi-basin defined by the local search threshold where the encountered subthreshold point resides in. In other words, the exhaustive local search will eventually visit all the points in the encountered quasi-basin. After local search in the quasi-basin is done, SS continues to global search until another subthreshold point is encountered. This switching between global search and local search proceeds until the stopping criterion is satisfied.

C. Theoretical Model of SS on QBCs

Now we are ready for the theoretical model of SS on QBCs. Given a QBC $\mathcal{Q}(G, \mathcal{Y}, m, b)$, the theoretical expected samplings, T, for SS, with its local search threshold set to $\beta_m(f)$, to find the minimum of a function in the QBC is derived in [9] as

$$T = \frac{cN}{m}\left\lceil \frac{b}{2} \right\rceil + \frac{m+1}{2}\ , \tag{1}$$

where N denotes $|G|$, the size of G, and c denotes a parameter between 0.75 and 1.5. The first term and the second term of this formula are corresponding to the global search time and the local search time of SS respectively. The global search time of SS depends on the expected time for the random search to find one of the m subthreshold points in N points. It is inversely proportional to m and dramatically drops as m increases. On the other hand, the local search time of SS linearly scales with m as SS employs exhaustive local search. Summing up the global search time and the local search time, a v-shape of the expected evaluation time along the m-axis with its optimal setting of m around $\sqrt{bcN}$ can be obtained. This theoretical model suggests that applying local search on small amount of qualified individuals generally delivers better performance than applying local search on several superior individuals or conducting massive local search on indistinguishable individuals.

III. Discrete Lipschitz Quasi-Basin Class

In order to demonstrate the effect of problem landscapes, we adopt the concept of Lipschitz continuity to define the Discrete Lipschitz Quasi-Basin Class (DLQBC) as a class that categorizes problems exhibiting continuity according to the number of modals and the roughness of landscape. The Discrete Lipschitz Quasi-Basin Class (DLQBC) is defined in Definition 1.

Definition 1: (Discrete Lipschitz Quasi-Basin Class, DLQBC). Given a graph G and a co-domain $\mathcal{Y}$, the corresponding discrete Lipschitz quasi-basin class with b distinct quasi-basins, Lipschitz constant K and landscape monotonicity m_o is defined as

$$\begin{aligned}
&\mathcal{L}(G,\mathcal{Y},b,K,m_o) := \\
&\{f : V(G) \to \mathcal{Y} \mid \forall \overline{v_1v_2} \in E(G), |f(v_1) - f(v_2)| \leq K, \\
&\forall \overline{v_1v_2}, \overline{v_2v_3} \in E(G), \\
&E[P_{rob}[(f(v_1) - f(v_2)) \times (f(v_2) - f(v_3)) > 0]] \approx m_o, \\
&S_{|V(G)/2|}(f) = \bigcup_{i=1}^{b} QB_i, \bigcap_{i=1}^{b} QB_i = \emptyset, \\
&\lfloor |V(G)|/2b \rfloor \leq |QB_i| \leq \lceil |V(G)|/2b \rceil, 1 \leq i \leq b\} \ .
\end{aligned}$$

In this definition, a function that belongs to a DLQBC has its value difference of any two neighbor points in the search space bounded to less than or equal to the Lipschitz constant K. The m_o parameter, defined as the approximated expected value of the probability of the monotonicity among three connected points, decides the roughness of landscape. An m_o value of 1 indicates a smooth landscape while a m_o value of 0.5 indicates a rough landscape. We also set the number of subthreshold points to $V(G)/2$ and adopt uniform basin size, thus the b specified in a DLQBC can match the general sense of the number of modals of a problem.

A. Sampling Test Scheme

For empirical convenience, we implement the simplest case of DLQBC, pathwise Discrete Lipschitz Quasi-Basin Class (PDLQBC), which is the class of functions with a simple path spatial structure, $G = \overline{v_1v_2\ldots v_n}$, with $\mathcal{Y} = \mathbb{R}$.

To investigate the expected behavior of an optimization algorithm over a specific PDLQBC, we sample functions from a specific PDLQBC via the PDLQBC sampler of which the pseudo code is shown in Fig. 1. The PDLQBC sampler firstly determines the size of each basin and hill, then constructs and concatenates each basin and hill. In DLQBC, the subthreshold points and the rest points have equal amount, thus θ is set to $n/2$[*]. The $UniformPick(\theta-1,b)$ function uniformly randomly picks b integers from the set $\{1,\ldots,\theta-1\}$. Thus after sorting GTL and concatenating θ to the gtl_i sequence,

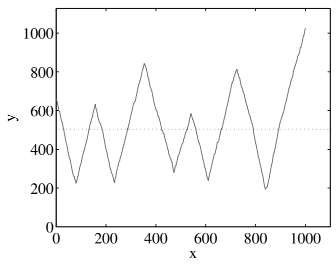
```
1: procedure Pathwise DLQBC Sampler($\overline{v_1v_2\ldots v_n}$, $\mathcal{Y}=\mathbb{R}, b, K, m_o$)
2:     $\theta \leftarrow n/2$
3:     $GTL \leftarrow UniformPick(\theta-1, b)$
4:     $\{gtl_1, gtl_2, \ldots, gtl_{b+1}\} \leftarrow Sort(GTL), \theta$
5:     $Size_b \leftarrow \lfloor \theta/b \rfloor$
6:     $f(v)_1^{gtl_1} \leftarrow StartSampler(gtl_1, \theta, K, m_o)$
7:     $idx \leftarrow gtl_1$
8:     for $i$ = 1 to $b$ do
9:         if $i \leq mod(\Theta, b)$ then
10:            $l \leftarrow Size_b + 1$
11:        else
12:            $l \leftarrow Size_b$
13:        end if
14:        $f(v)_{idx+1}^{idx+l} \leftarrow SubSampler(l, \theta, -K, m_o)$
15:        $idx \leftarrow idx + l$
16:        $l \leftarrow gtl_{i+1} - gtl_i$
17:        if $i \leq b$ then
18:            $f(v)_{idx+1}^{idx+l} \leftarrow SubSampler(l, \theta, K, m_o)$
19:        else
20:            $f(v)_{idx+1}^{idx+l} \leftarrow EndSampler(l, \theta, K, m_o)$
21:        end if
22:        $idx \leftarrow idx + l$
23:    end for
24:    return $f$
25: end procedure
```

Fig. 1. Pathwise DLQBC Sampler.

$gtl_{i+1} - gtl_i$ can be utilized as the size of the i'th hill. As the basin sizes are uniform, either $\lfloor \theta/b \rfloor$ or $\lfloor \theta/b \rfloor + 1$ are assigned as the basin sizes. Each basin or hill $\{f(v_a),\ldots,f(v_b)\}$, indicated by $f(v)_a^b$, is constructed by $StartSampler$ or $SubSampler$ or $EndSampler$. The $SubSampler$ constructs a basin (in a $\cup$ shape) or a hill (in a $\cap$ shape) according to the specified basin/hill size l, θ, the Lipschitz constant K, and the landscape monotonicity m_o. When K is negative, the $SubSampler$ samples a path of length l consisting half of down-hilling subthreshold points and half of up-hilling subthreshold points which obey the continuity specified by the Lipschitz constant K and the landscape monotonicity m_o. Furthermore, the starting point and end point of this $\cup$-shape path have their value within $\theta - K$ to guarantee the Lipschitz continuity of all the concatenated basins and hills. Similarly, when K is positive, the $SubSampler$ samples a $\cap$-shape path which fulfils the Lipschitz continuity and landscape monotonicity constraints. The $StartSampler$ and $EndSampler$ samples a downhill path and a uphill path that fit the aforementioned constraints respectively. Via concatenating the paths constructed by $StartSampler$, $SubSampler$, and $EndSampler$, we can sample an instance that belongs to a PDLQBC.

In our framework, all the PDLQBC instances have their G set to $\overline{v_1v_2\ldots v_{1000}}$ and K set to 10. The notation $b\#$ denotes that the b of the PDLQBC is set to $\#$. The notations $easy$

[*]For simplicity, we illustrate the sampler only when n is even.


(a) b5-easy

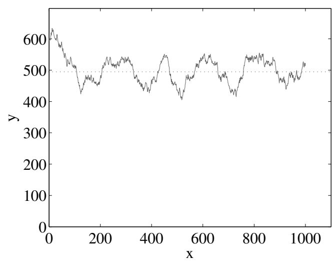

(b) b5-hard

Fig. 2. Instances of PDLQBCs

and $hard$ denote that m_o of the PDLQBC is set to 1 and 0.5 respectively. Fig. 2 illustrates two instances of PDLQBCs.

IV. Representative Archetypes of Memetic Algorithms

In this framework, we investigate how the explorability and exploitability of a memetic algorithm affect its performance. In our perspective, according to the degree of explorability and exploitability, basic optimization algorithms are mainly of three types: random search, heuristic search, and exhaustive search. Here we refer to pure random sampling as random search, refer to sampling according to some rules as heuristic search, and refer to sampling all the vertices as exhaustive search. Accordingly, we investigate representative archetypes of memetic algorithms consisting of different combinations of random search, heuristic search, and exhaustive search.

A. Nelder-Mead Method

As heuristic search algorithms are various in forms, we adopt the Nelder-Mead method (NM) as a representative heuristic search algorithm. The Nelder-Mead method utilizes the concept of a simplex, which is a special polytope of $N+1$ vertices in N dimensions, and features its capability of approximating a local optimum of a problem with N variables when the objective function varies smoothly and is unimodal. As our sampling test scheme samples instances of PDLQBC which are fundamentally one dimension problems, we adopt the 1-D Nelder Mead Algorithm shown in Fig. 3. The $Uniform(\mathcal{X}, 2)$ indicates a uniformly random sampling of two vertices from the search space $\mathcal{X}$. To avoid stagnation during a global search, when x_1 and x_2 appear to be the same vertex, we re-sample x_1 and x_2. Note that when this Nelder-Mead method is utilized as a local search algorithm in the following text, the stop criterion is set to the convergence of the simplex. In other words, it will stop when x_1 and x_2 appear to be the same vertex.

B. Memetic Algorithm Types

Fig. 4 illustrates the pseudo code of a generalized memetic algorithm. The S is the resultant set of vertices of the current global search and the reference for the next global search. If one of the vertex s_i in S satisfies the local search criterion, local search will be performed on the point. This process

```
1:  procedure 1-D NELDER MEAD ALGORITHM($\mathcal{X}$, $\mathcal{Y}$, $f : \mathcal{X} \to \mathcal{Y}$)
2:      $\{x_1, x_2\} \leftarrow Uniform(\mathcal{X}, 2)$
3:      while the stopping criterion is not satisfied do
4:          $\{x_1, x_2\} \leftarrow Sort(\{x_1, x_2\})$
5:          $x_r \leftarrow x_1 + (x_1 - x_2)$
6:          if $f(x_r) < f(x_1)$ then
7:              $x_e \leftarrow x_1 + 2(x_1 - x_2)$
8:              if $f(x_e) < f(x_1)$ then
9:                  $x_2 \leftarrow x_e$
10:             else
11:                 $x_2 \leftarrow x_r$
12:             end if
13:         else
14:             $x_c \leftarrow x_2 + (x_1 - x_2)/2$
15:             $x_2 \leftarrow x_c$
16:         end if
17:         if $x_1 = x_2$ then
18:             $\{x_1, x_2\} \leftarrow Uniform(\mathcal{X}, 2)$
19:         end if
20:     end while
21: end procedure
```

Fig. 3. 1-D Nelder-Mead Algorithm

```
1:  procedure GENERALIZED MEMETIC ALGORITHM($\mathcal{X}$, $\mathcal{Y}$, $f : \mathcal{X} \to \mathcal{Y}$)
2:      $S \leftarrow Initialization()$
3:      while the stopping criterion is not satisfied do
4:          $S \leftarrow GlobalSearch(S, \mathcal{X}, \mathcal{Y}, f : \mathcal{X} \to \mathcal{Y})$
5:          for $s_i \in S$ do
6:              if local search criterion is satisfied then
7:                  $s_i \leftarrow LocalSearch(s_i, \mathcal{X}, \mathcal{Y}, f : \mathcal{X} \to \mathcal{Y})$
8:              end if
9:          end for
10:     end while
11: end procedure
```

Fig. 4. A generalized memetic algorithm.

will continue until the stopping criterion is satisfied. In this framework, we set the stopping criterion to the sampling of the global optima, i.e., the minimum. To illustrate how local search criteria affect the collaboration between global search and local search based on the concept of local search zones, we set the local search criterion as the subthreshold seeker does. Thus, local search will be performed on any vertex s_i in S with an objective value $f(s_i)$ less than or equal to $\beta_m(f)$, the local search threshold, and we can then investigate the behavior of the subject memetic algorithms based on the model in [9].

To identify representative archetypes of memetic algorithms for investigation, we firstly categorize all optimization algorithms into three types: random search, heuristic search, and exhaustive search. Among these three types, heuristic search exhibits both fair explorability and exploitability while random search and exhaustive search deliver full explorability and full

TABLE I
LIST OF REPRESENTATIVE ARCHETYPES OF MEMETIC ALGORITHMS

Memetic Algorithm	Global Search	Local Search
SS	random search	exhaustive search
MA1	Nelder-Mead method	exhaustive search
MA2	random search	Nelder-Mead method

exploitability, respectively. Based on the paradigm of memetic algorithms described in Fig. 4, the three algorithms listed in Table. I are proposed as three representative archetypes of memetic algorithms. They are Subthreshold Seeker (SS), Memetic Algorithm Type 1 (MA1), and Memetic Algorithm Type 2 (MA2). SS, as introduced in Section II, employs random global search and exhaustive local search. MA1 employs NM as its global search and exhaustive search as its local search. MA2 employs random search as its global search and NM as its local search. In this manner, we consider SS, MA1, and MA2 are three representative archetypes for memetic algorithms. Note that following our previous work [9], the evaluation time of all the optimization algorithms, SS, MA1, MA2, and NM, in this framework is considered as the count of non-repeated sampling to find the minimum of a PDLQBC instance. In other words, repeated samplings are not counted.

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

In this section, we investigate the behavior of all the proposed representative archetypes of memetic algorithms on different DLQBCs. The collaborative behavior of different archetypes of memetic algorithms on different problems categorized by the number of modals and the landscape roughness will be illustrated by the evaluation times of the proposed algorithms with different local search thresholds on four representative PDLQBCs. The notations MA2-1 and MA2-10 are used for MA2 with one and ten step distances for the initial simplex of the NM local search respectively. That is, one vertex of the initial simplex is set to the encountered subthreshold point, and the other is set to a point that is one or ten steps far from the encountered subthreshold point.

All the proposed algorithms are tested on four PDLQBCs, b1-easy, b1-hard, b10-easy, and b10-hard. These four PDLQBCs are used as representative problem instances of smooth unimodal problems, rough unimodal problems, smooth multi-modal problems, and rough multi-modal problems, respectively. The local search threshold of each algorithm is set to $\beta_m(f)$ and m is set from 1 to 991 with a step of 10. For each algorithm with a specified local search threshold, its evaluation time on a PDLQBC is measured by averaging 50 function instances with 20 independent runs on each instance.

As the theoretical model derived in [9], modeling the evaluation time as the summation of global search time and local search time, describes the behavior of SS on QBCs well, we extend this modeling approach to explain the behavior of different MAs on DLQBCS. In the remainder of this section, we will firstly analyze the behavior of NM as a global search algorithm and as a local search algorithm. Then, we will investigate and discuss the behavior of the proposed archetypes of MAs on different problems.

A. Nelder-Mead Method as Global Search and Local Search

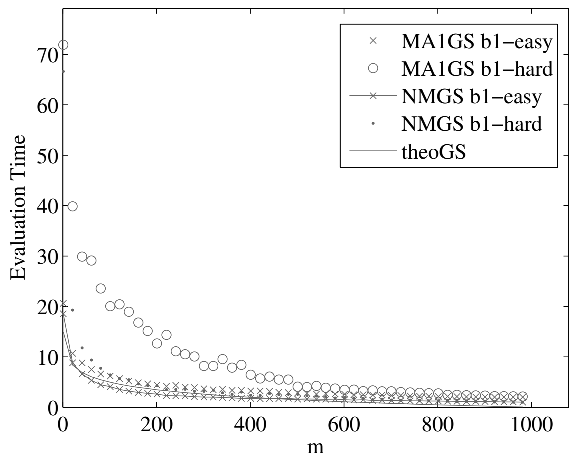
In order to explain the behavior of the proposed archetypes of memetic algorithms in a similar way to the theoretical model of SS on QBCs, we demonstrate the global search behavior and local search behavior of NM, which is employed as global search as well as local search in the proposed archetypes. The explorability and exploitability of NM are assessed by measuring the average time for NM to find one of the m subthreshold points in N points and the minimum among m points, respectively. In both assessments, we test them on b1-easy and b1-hard problems. The G in the explorability assessment is set to $\overline{v_1 v_2 \dots v_{1000}}$ and in the exploitability assessment is set to $\overline{v_1 v_2 \dots v_m}$. Figs. 5(a) and 5(b) illustrate the global search behavior and the local search behavior of NM with respect to m. As NM is employed as global search in MA1 and employed as local search in MA2, we also compare the global search time of MA1, MA1GS, and the local search time of MA2-10, MA2LS, with the explorability of NM, NMGS, and the exploitability of NM, NMLS.

In Fig. 5(a), theoGS is calculated as $1.5\log_2(N/m)$. On b1-easy problems, both NMGS and MA1GS could be approximated by theoGS and are logarithmic inversely proportional to m. On b1-hard problems, the roughness has more significant impact to the NM's global search behavior, either NMGS or MA1GS, when m is rather small. In Fig. 5(b), theoLS is calculated as $2\log_2((m+1)/2)$. On b1-easy problems, both NMLS and MA2LS could be approximated by theoLS and exhibit a logarithmic growth with respect to m. On b1-hard problems, the roughness of the landscape has more impact on both NMLS and MA2LS as m increases. As the goal of MA1 and MA2 is to find the minimum of a problem instance, several local search rounds and thus several global search rounds are required on rough landscapes and this could results in the larger values of MA1GS and MA2LS than the values of NMGS and NMLS.

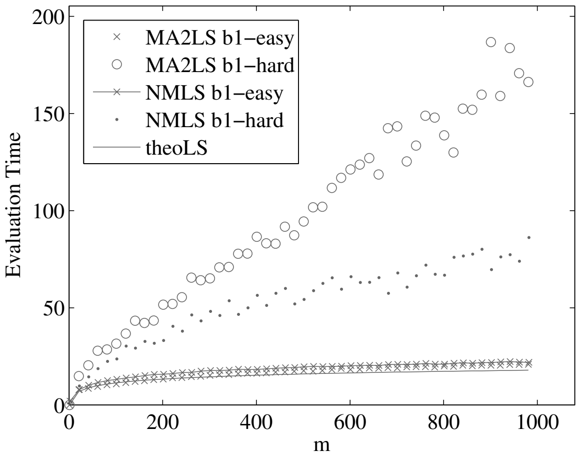
In short, when NM is employed as global search on unimodal problems, it performs significantly better than random search when m is small. When NM is employed as local search, it is more robust to the size of local search space than exhaustive search when the landscape is smooth. However landscape roughness could alter this robustness.

B. Fundamental Observations

Now we investigate the behavior of the proposed archetypes of memetic algorithms on different problems. Firstly, Fig. 6 and 7 illustrate the evaluation times of the proposed archetypes of memetic algorithms on different PDLQBCs, b1-easy, b1-hard, b10-easy, and b10-hard. On these four types of problems, the evaluation times of NM and the $theo$ lines are included to provide baselines for performance comparison of the subject algorithms. The $theo$ lines are depicted according to Equation(1), the theoretical evaluation time of SS on QBCs, with c set to 1 when $b1$ and set to 1.5 when $b10$.



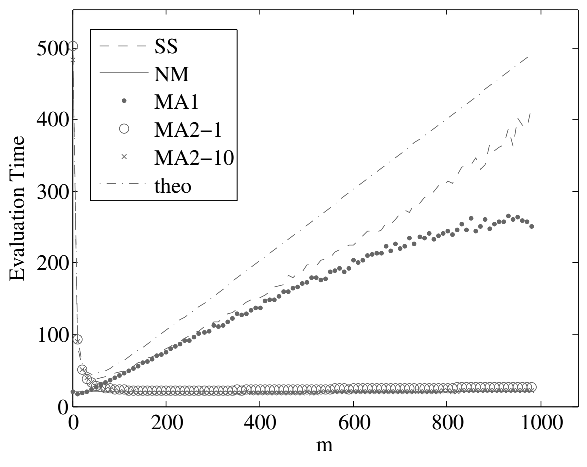
(a)



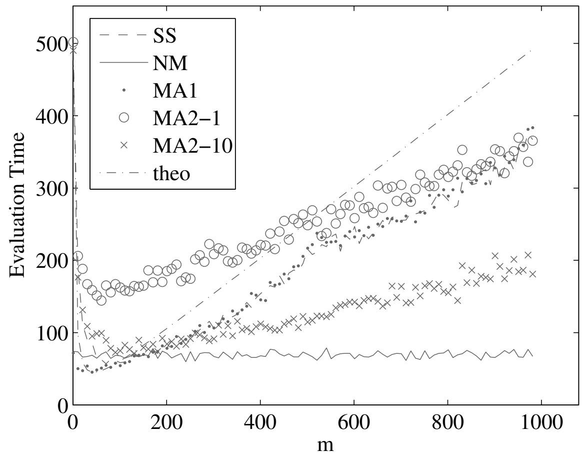
(b)

Fig. 5. The global search behavior and local search behavior of NM on DLQBCs. Fig. 5(a) illustrates the average evaluation time for the Nelder-Mead method to find one subthreshold point among N points when there are m subthreshold points. Fig. 5(b) illustrates the average evaluation time for the Nelder-Mead method to find the minimum point among m points.

Note that in Fig. 6 and 7, the evaluation times of all the algorithms exhibit either v-shapes or L-shapes along the m axis. As the evaluation time is the summation of global search time and local search time, and global search time generally dramatically decreases as m increases while local search time basically increases as m increases, the global search time and the local search time dominate the evaluation time when m is rather small and when m is sufficiently large, respectively. Thus, when m is rather small, MA1, which employs NM as its global search, has a much smaller evaluation time than SS and MA2, which employ random global search. When m is large, the employed local search algorithm dictates the evaluation time of a memetic algorithm: SS and MA1, employing exhaustive local search, have their evaluation times linearly scale with respect to m as the exhaustive local search does on all the four types of problems; MA2, employing



(a) b1-easy



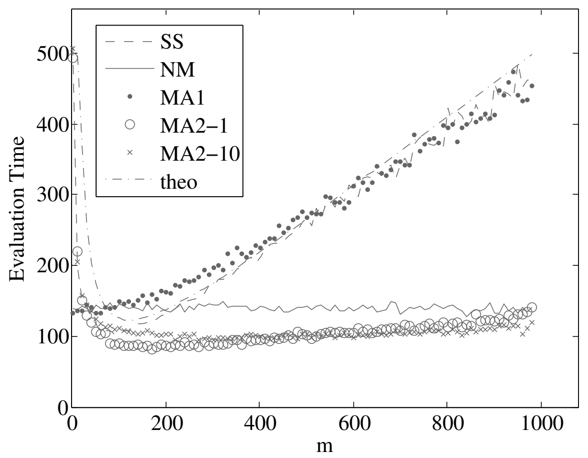
(b) b1-hard

Fig. 6. The behavior of the proposed representative archetypes for memetic algorithms on unimodal PDLQBCs.

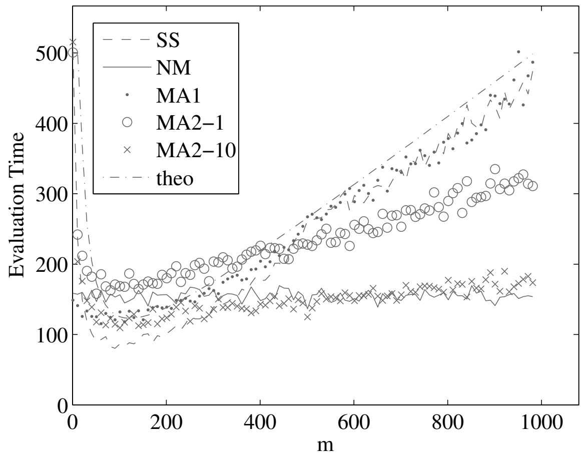
NM as local search, has its evaluation time logarithmically increases with respect to m on easy problems and linearly increases with respect to m on hard problems as NM local search does. The shapes and collaborative behavior of all the subject algorithms basically consist with the modeling approach in [9]. This implies that the proposed *local search zone* concept could be adopted to model the collaborative behavior of more realistic algorithm-problem complexes than the one proposed in [9].

C. Memetic Algorithms on Unimodal Problems

In Fig. 6(a), NM captures the landscape well and achieves an ideal evaluation time about the order of the logarithm of the size of the search space. On these smooth unimodal problems, both MA1, employing NM as global search, and MA2, employing NM as local search, have their best performance at some values of m better than NM; and as both the global search algorithm and the local search algorithm



(a) b10-easy



(b) b10-hard

Fig. 7. The behavior of the proposed representative archetypes for memetic algorithms on multi-modal PDLQBCs.

of SS perform worse than NM global search and NM local search, its collaborating performance consequently is not comparable to NM and the rest of MAs. This implies that when an optimization algorithm is efficient for a problem, properly collaborating it with a global search algorithm or a local search algorithm could further slightly improve its performance. Note also that on smooth unimodal problems, MA2 which employs NM local search is much robust to m, corresponding to the local search criterion, as NM local search scales logarithmically with respect to m. Contrastively, MA1 employing exhaustive local search performs better than NM only in a small range of m, thus a more delicate setting of m is required. This may also suggest that when having an efficient problem-specific algorithm at hand for an unimodal problem, either utilizing the algorithm alone or properly employing it as local search will provide an acceptable and robust solution to the problem. Taking CMA-ES as an example of efficient heuristic algorithm, on the Sphere function of BBOB 2010, algorithms which employ CMA-ES and have their exploration-exploitation synergies adjusted perform well [10].

In Fig. 6(b), as the landscape is rough, NM cannot perform well as it does on b1-easy problems and may be considered as a heuristic algorithm that cannot properly capture the landscape of problem. Under this circumstance, when NM is employed as local search, it may not find the minimum of a quasi-basin in one local search and thus requires more global searches and local searches to revisit a quasi-basin. Contrastively, though exhaustive local search linearly scales as m increases, it is more efficient than NM on small rough quasi-basins for it guarantees the discovery of the minimum of the quasi-basin in one local search. When m, corresponding to the size of the local search zone, is large enough for an employed global search to enter effortlessly and small enough to require little exhaustive local search time, a better performance than NM could be attained by collaborating a global search and exhaustive local search. This is why the best evaluation times of SS and MA1 outperform those of NM and MA2.

Note that as MA1 employs NM as its global search, it is more efficient than SS when m is small and thus has better evaluation times than SS does. Note also that MA2 does not benefit from collaborating random global search and NM local search as their best performance could not surpass NM's performance. With a bad initial local search step size, MA2-1 even exhibits the severely performance degradation than MA2-10 does on these problems. The rough landscapes make the initial local search step influential to MA2. Overall, these observations on b1-hard problems suggest that on rough unimodal problems, memetic algorithms which employ heuristic global search and exploitive local search like MA1 does may have a good chance of achieving good performance with an elaborated collaborating mechanism. Or, in another perspective, when heuristic algorithms cannot properly capture the landscape of a unimodal problem, carefully coordinating it with exploitive local search algorithms is a good choice to pursue salient performance. For example, the Artificial Bee Colony algorithm, one of the best algorithms for the Rastrigin function of BBOB 2010, is an exploitive heuristic algorithm [10].

Besides the performance relevant observations on unimodal problems, we explain the intriguingly increasing deviations between SS and $theo$ with respect to m in Figs. 6(a) and 6(b). These deviations may be mainly due to the landscape regularity introduced by the Lipschitz continuity. In these unimodal problems, the minimum always lies around the middle points of the only basin. Any local search starts at a subthreshold point in the basin will find the minimum within half of the size of subthreshold points. Similarly, MA1 which employs the exhaustive local search also exhibits this deviation. As MA1's initial local search points are the better point of simplexes, they are generally better than SS's. On b1-easy problems with smooth landscapes, MA1's initial local search points are generally closer to the local minimum than SS's. This makes MA1 deviate more from $theo$ than SS in Fig. 6(a). On the other hand, as shown in Fig. 6(b), on b1-hard problems with rough landscapes, better initial local points do not guarantee closer distances to the local minimum and do not help much to reduce the local time.

D. Memetic Algorithms on Multi-modal Problems

In Fig. 7(a), NM performs worse than on b1-hard problems. The reason for this performance degradation may be twofold. Firstly, the multi-modal landscapes may induce deceptive unimodal information and thus interfere the fast convergence of NM. Secondly, as explorability is not a main concern of NM, it may repeatedly visiting the same quasi-basin leaving other quasi-basins, where the minimum may reside in, unexplored. On these multi-modal problems, MA2 exhibits the best performance as it employs random global search, capable of effectively exploring the basins, and NM local search, capable of efficiently exploiting an encountered quasi-basin. Though m has smaller impact on MA2, a feature of NM local search, when the m is properly set to some values that guarantee both small global search time and local search time, MA2 still could further achieve its best performance as indicated in this figure. SS also has its best performance outperforming NM on these multi-modal problems as a result of good collaboration between the random global search and exhaustive local search.

Note that MA1 does not perform well as its collaboration of NM global search and exhaustive local search may not suit this type of problems well. All of these observations on b10-easy problems suggest that on smooth multi-modal problems, coordinating random global search with an efficient heuristic algorithm may provide a good optimization solution which is in some degree robust to the local search criterion. Furthermore, these observations also imply that when an efficient problem-specific local search is at hand for multi-modal problems, coordinating it with explorative global search algorithm may provide salient performance than other kinds of hybrids. In recent studies, the IPOP-CMA-ES, which benefits from the efficiency of CMA-ES and the explorative restart and increase population size mechanism, may be a good example of MA2-type algorithms. It performs well on the Weierstrass function of BBOB 2010 [10].

In Fig. 7(b), NM has its worst performance among the four types of problems due to the roughness and the multi-modality of the landscapes explained in b1-hard and b10-easy problems. On these problems, all the MAs (considering MA2-1 and MA2-10 as one) with a not-too-bad coordination of global search and local search achieve better performance than NM does. This implies that when the problem is extremely hard, rough and multi-modal, any MAs with proper design could outperform a mediocre heuristic algorithm. Note that SS achieved the best performance among MAs. This suggests that memetic algorithms employing an explorative global search algorithm like random search and an exploitive local search algorithm like exhaustive search, with an elaborated coordination, have a chance to achieve better performance than other combinations of memetic algorithms on hard problems.

Another interesting observation on the algorithmic performance is that SS and MA1 perform better on b10-hard than on b10-easy. This may be due to the larger deviations among depths of the basins induced by the landscape roughness as illustrated in Fig. 2(b). Within some value of m, the number of quasi-basin viewed by the SS and MA1 on b10-hard problems is less than that on b10-easy problems and thus requires less evaluation time. Note that though with this virtual reduction on the number of quasi-basins, the landscape roughness still make the b10-hard problems harder than b10-easy for NM and MA2 which lack of the exploitability of exhaustive local search as depicted in Fig. 7(b).

VI. Conclusions

In this work, we made an attempt to investigate when and what kind of memetic algorithms perform well. We extended our previous framework to illustrate the collaborative behavior of three representative archetypes of memetic algorithms and compare them with a representative heuristic algorithm, Nelder-Mead method, on four representative types of problems. On each type of problems, the success and failure of each algorithm are analyzed and discussed. This study may provide a reasonable and systematic explanation to why some memetic algorithms on certain problems outperform heuristic algorithms and attain salient performance, and we, as researchers on memetic algorithms, thus could gain insights into the design principals of memetic algorithms.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2.

References

[1] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation – past, present & future," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, 2010.

[2] W. E. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, University of California, 1994.

[3] M. W. S. Land, "Evolutionary algorithms with local search for combinatorial optimization," Ph.D. dissertation, University of California, 1998.

[4] D. Sudholt, "On the analysis of the (1+1) memetic algorithm," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006 (GECCO-2006)*, 2006, pp. 493–500.

[5] ——, "The impact of parametrization in memetic evolutionary algorithms," *Theoretical Computer Science*, vol. 410, no. 26, pp. 2511–2528, 2009.

[6] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, University of the West of England, 2002.

[7] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.

[8] J. Smith, "Coevolving memetic algorithms: A review and progress report," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 6–17, 2007.

[9] J.-Y. Lin and Y.-P. Chen, "Analysis on the collaboration between global search and local search in memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 608–623, 2011.

[10] A. Auger, S. Finck, N. Hansen, and R. Ros, "BBOB 2010: Comparison tables of all algorithms on all noiseless functions," Inria, Tech. Rep. RT-388, 2010. [Online]. Available: http://hal.inria.fr/inria-00516689/en/