解可滿足性問題之新區域搜尋演算法

曾怜玉

國立中興大學資訊網路與多媒體研究所

國立中興大學資訊科學與工程學系

Email: lytseng@cs.nchu.edu.tw

林祐安

國立中興大學資訊科學與工程學系

Email: alimen@gmail.com

摘要－可滿足性問題(satisfiability problem)為計算複雜度 NP 是否等於 P 問題的核心問題，並在人工智慧、硬體設計、VLSI 測試與驗證、最佳化問題等領域有諸多應用。本研究中使用一改良的區域搜尋演算法來解可滿足性問題。針對目前為止所發展的演算法的弱點，以下列方法加以克服：1. 提出新的適應度函數以加強對搜尋之引導；2. 並於區域搜尋演算法中加入重設條件，捨棄陷於區域最佳解附近的解以達到系統化的搜尋。我們應用本論文所提出之演算法於 Gottlieb、Marchiori 與 Rossi 於 2002 年所匯整的題組中，和其他演算法進行比較，獲得了不錯的成果。

關鍵詞－區域搜尋演算法、可滿足性問題。

一、前言

可滿足性問題(satisfiability problem, SAT)為一決定性問題(decision problem)，並可簡述如下：對一給定的布林表示式 $\Phi : B^n \to B, B = \{0,1\}$ 而言，是否存在有一組解 $X = (x_1, x_2, \ldots, x_n) \in B^n$ 而滿足(satisfy)此一布林表示式，意即使 $\Phi(X) = 1$？此一問題自 1971 年經 Stephen Cook 證明為 NP-Complete 問題後，便一直處於計算複雜度 NP 是否等於 P 問題的核心，近三十年來引起相當多的討論與研究。此外可滿足性問題，在諸如人工智慧、硬體設計、VLSI 測試與驗證、最佳化問題等領域中，都有廣泛的應用。

截至目前為止，對於所有解可滿足性問題的演算法，依概念不同，可分為兩大類：complete algorithms 與 incomplete algorithms。前者以 Davis-Putnam algorithm，一個以 branch-and-bound 為基本概念的演算法為首，以依序決定各個變數之值的方式，有效走訪並檢驗所有可能的解；此類型演算法之優點在於，能夠確實的驗證一布林表示式是否存有可滿足之解，但相反的，由於其最壞情況的時間複雜度仍然為指數複雜度，面對變數較多的題目將相當吃力。因應前述弱點，泛指區域搜尋與基因演算法等啟發式演算法(heuristic algorithms)為基礎的 incomplete algorithms 被大量提出，此類演算法由一組或多組任意解出發，運用不同的策略，以一次替換，或稱「翻動(flip)」一個或多個變數之值的方式在解空間中搜尋，以期獲得可滿足布林表示式之解。

有鑑於以區域搜尋為基礎的演算法在限制滿足問題(constraint satisfaction problem, CSP)的成功[1]，Gu [13]與 Selman、Levesque、Mitcheel [20]兩組人於 1992 年分別提出簡單而有效率 SAT 1.1 與 Greedy SAT(GSAT)兩種演算法，正式將區域搜尋的概念引進解可滿足性問題的演算法當中。同年稍晚，Gent 與 Walsh 歸納整理並提出了一個泛用的區域搜尋演算法框架，GenSAT 架構[8]，使得接下來的數年內，以區域搜尋為基礎的演算法包括 Walk SAT (WSAT) [21]、Historic SAT (HSAT) [9]等如雨後春筍般出現，也帶動了屬相同性質的變動鄰域搜尋(variable

neighborhood search) [14]、禁忌搜尋(tabu search) [18]、模擬退火(simulated annealing) [3]為基礎的演算法的投入。但是，以區域搜尋為基礎的演算法其搜尋時間過長、容易陷入區域最佳解等缺點仍未解決。

以基因演算法為基礎的演算法，早在 1989 年便由 de Jong 與 Spears [5]提出，但是由於效能比不上針對問題特性發展的 complete algorithms，因此沉寂了相當的一段時間。中途雖歷經在基因表示法(chromosome representation)、適應度函數(fitness function)等基本設定上的討論，但一直到 1997 年，Eiben 與 van der Hauw 提出 SAWEA [6]，於傳統基因演算法中加入自動調適的適應度函數，才再度引起眾人的注意，陸續發展出 RFEA [10]、RFEA2 [11]、FlipGA [17]、ASAP [19]等等不同的演算法；其中又以 FlipGA 結合基因演算法與區域搜尋演算法的作法最受矚目。

混合區域搜尋的基因演算法，又稱基因區域搜尋演算法(genetic local search algorithm)或瀰集進化演算法(memetic algorithm)，是近年來新興的概念：在基因演算的父代與子代之間，對部分或全部的染色體執行區域搜尋演算法來改良染色體品質，以期求得更佳解。除 FlipGA 與 ASAP 外，Hao、Lardeux 與 Saubion 三人於 2003 年所提出、並於 2007 年改良的 GASAT 演算法也同屬此類[15][16]。

綜觀 incomplete algorithms，其實仍存在有下列主要缺點：1.搜尋不夠系統化，有可能在區域最佳解附近花費過多時間搜尋；2.目前所慣用之適應度函數對於引導搜尋的能力不夠，導致無法有效獲得可滿足之解；3.演算法的泛用性不足，對於變數數量較少的題組有效的演算法，在變數數量較多的題組中則表現不佳，反之亦然。因應以上缺點，本研究由目前已知最有效之區域搜尋演算法 WSAT 出發，並提出下列方法加以克服：1.提出新的適應度函數以加強對搜尋之引導，縮短搜尋時間；2.並於區域搜尋演算法中加入重設條件，捨棄陷於區域最佳解附近的解以達到更有效率的搜尋。

本論文結構如下：第二節為問題之定義；第三節簡介區域搜尋演算法 WSAT；於第四、五節介紹本研究所提之改良方法；並於第六節將本研究所提出之各項概念整合，一覽整個區域搜尋演算法；最後於第七節列出本方法在各題組之實驗成果並與其他演算法做一比較；第八節則總結本論文研究成果，並建議後續之研究方向。

二、問題定義

可滿足性問題定義如下：對一指定的布林表示式 $\Phi: B^n \to B, B = \{0,1\}$ 而言，是否存在有一組解 $X = (x_1, x_2, \ldots, x_n) \in B^n$ 使得 $\Phi(X) = 1$？

在不影響複雜度的情況下，為求格式之簡潔，所有可滿足性問題之布林表示 Φ 式皆轉為一種稱作 conjunctive normal form (CNF)的標準格式。此格式定義如下：令 $C = \{C_1, C_2, \ldots, C_m\}$ 為布林變數 $x_1, x_2, \ldots, x_n$ 所構成 m 個子句(clause)之集合，且每個子句 C_i 為 $|C_i|$ 個字母(literal) l 以 OR 運算子串聯，即：

$$C_i = \bigvee_{j=1}^{|C_i|} l_{ij}$$

字母 l 可為任意變數 x_j 或其相反—x_j 其中之一種；子句與子句之間則以 AND 運算子串聯，意即：

$$\Phi = \bigwedge_{i=1}^{m} C_i = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{|C_i|} l_{ij}$$

換言之，若有一解組解 X 能夠使所有子句滿足，意即使所有子句之運算結果為 1，則此組解必能滿足題目所給定的布林表示式。

三、WSAT 演算法與 minimizing breaks

WSAT 演算法於 1994 年由 Selman、Kautz、Cohen 三人所提出，並於 1997 年時改良[22]。此一演算法可敘述如下：

1. 初始化，以亂數隨機產生一組解。
2. 隨機選擇一未滿足之子句。
3. 隨機產生一介於 0 與 1 之間的數 r，並與事先決定好的參數 p 比較：若 r > p，則在子句中隨機選擇一變數，否則，評估子句內所有的變數並選擇得分最高或最低者翻動。
4. 當尋獲一可滿足布林表示式之解，或翻動次數達到預先設定的上限，程式即終止。

與其他區域搜尋演算法相比，WSAT 有兩項獨創之處：一是參考摸擬退火演算法，在搜尋過程中加入 random walk 機制；二則是評估函數的創新。

在 1997 年 Selman 等人所提出的論文中，一共提出了六種不同的評估函式，其中最重要者，便為 minimizing breaks 評估函式。

一般而言，對一解 X，翻動其中任意一變數 x_j 將可能造成未滿足且含有 x_j 之子句轉變為滿足，但也有可能使得含有 x_j 且已滿足之子句轉為不滿足，將此兩者的數量以前者減去後者，即可得出翻動變數 x_j 所能夠改進解 X 的量。minimizing breaks 之概念，相較之下有些許不同，只記錄翻動 x_j 後，原為滿足且含有 x_j 之子句轉變為不滿足的子句數。此一方法十分簡單卻很有效率，常做為後繼演算法比較之對象。

本研究則以 WSAT 演算法與 minimizing breaks 評估函式出發，加入第四節之輔助性適應度函數與第五節之重設條件，加以改良，提出一新的區域搜尋演算法。

四、輔助性適應度函數 f_{lit}

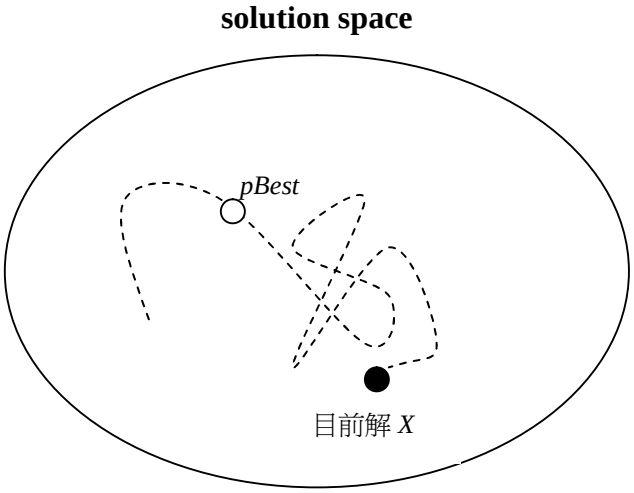
由 Eiben 與 van der Hauw [7]的研究結果可知，單純以計算已滿足之子句數的適應度函數 f_{MAXSAT}，對於解可滿足性問題之引導是不足的，因此，本研究提出一新的輔助性適應度函數 f_{lit} 以加強搜尋。此一函數可以以下列方式表示：

$$f_{Total}(X) = (m - f_{MAXSAT}(X)) + f_{lit}(X)$$

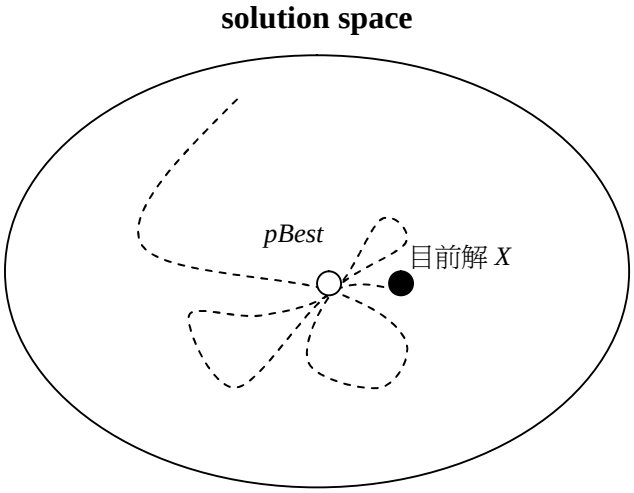
$$f_{lit}(X) = \frac{\sum_{i=1}^{m} t_i}{\sum_{i=1}^{m} |C_i| + 1}$$

其中 m 為子句的數目，t_i 為子句 C_i 中為真的字母之個數，$|C_i|$ 為子句 C_i 中所含有的字母數，並以少者為佳。

此一輔助性之適應度函數，其概念來自 Gottlieb 與 Voss 提出之 RFEA 演算法，其建議可在原有之 f_{MAXSAT} 加上一精練函數以加強搜尋。對 f_{lit} 而言，其概念則來自 Selman 等人所提出之 WSAT 的 minimizing breaks 演算法及 Hoa 等人所提出的 GASAT 演算法，但運算過程更為簡化而易於計算：對所有的已滿足的子句而言，最好的情況是子句中只存在有一個為真的字母，使得其餘的字母能夠滿足更多其他的子句；對於含有 x_j 且已滿足之子句轉為不滿足的情況，表示此一子句中為真的字母只有 x_j，故不應選擇此一 x_j 作為翻動的目標；為達成此一概念，本適應度函數計算一解 X 在布林表示式 Φ 中，所能使之為真的字母數，並除以總字母數加一，使其正規化至 $0 \le f_{lit}(X) < 1$ 之間。



【圖 1】區域搜尋法進入了一個平坦的高原地帶



【圖 2】區域搜尋法在本次最佳解周圍反覆搜尋

在與 minimizing breaks 合用時，f_{lit} 依照貪婪原則，挑出翻動前與後進步最多者；但由於 minimizing breaks 是選擇最少者，故兩者合併時，將 f_{lit} 加上一負號再行加總，即：

$$Score(x_j) = Minimizing\_breaks(x_j) + f_{lit}(flip(X, x_j)) - f_{lit}(X)$$

其中 *flip(A,b)*表示對解 *A* 其中的變數 *b* 作翻動。

五、重設條件

在執行區域搜尋演算法的過程當中，有相當多的機會會陷入區域最佳解而無法脫離，或遇上一直專注於區域最佳解附近搜尋而不自知的情況。在這種情形中，基因演算法或許可以發揮其功能，帶領當前解往較好的區域移動，但仍不足以避免陷入同一個區域最佳解；基因區域搜尋演算法則往往會在區域最佳解附近花費過多的時間進行區域搜尋，壓縮到對解空間其餘區域作搜尋的時間，使得最後成果不佳。

本論文提出三種重設條件，顧名思義，當區域搜尋演算法遇到此三項條件的任一情況時，表示目前解有可能位於一個區域最佳解附近，或位於一個解品質不甚良好的高原區上，在對該區域作短暫的搜尋以確認無全域最佳解存在於附近後，即可終止目前搜尋，以亂數產生新的解，於解空間中的另一區開始搜尋；更甚之，此一區域最佳解會被列入禁忌列表中，以防止將來在同一區浪費時間作搜尋。此三項重設條件詳細設定如下：

第一項重設條件為 Stuck 次數，即未進步次數。如同圖 1 所示，若區域搜尋演算法經過 *stuck-limit* 次的翻動仍未更新本次最佳解 *pBest* 時，則有可能表示搜尋進入了一個平坦的高原區域，附近沒有任何全域或區域最佳解存在，在這種情況下，多對此一區域進行搜尋有很大的可能是浪費時間，故需經重設步驟跳離此一區域。

第二項重設條件為 Loop，即本次最佳解的重複次數。如圖 2 所示，區域搜尋演算法有可能在找到一區域最佳解後，不斷的在其周圍反覆的搜尋而無法跳脫區域最佳解的範圍，此一情況同樣也會造成搜尋時間的浪費，對此，本論文採用

```
Procedure reSAT(Φ, T, Max-flips, p)
    pBest ← T,  stuck ← 0,  loop ← 0,  f ← 0
    for i ← 1 to Max-flips do
        if T satisfies Φ then return (T, f)
        Unsat-clause ← select-a-unsat-clause(Φ, T)
        r ← random([0, 1])          ▽隨機選擇一未滿足之子句
                                    ▽並產生一介於零與一之間的數
        if r < p then V ← random-pick(Unsat-clause)
              ▽若 r < p，則在字句中隨機挑選一變數
        else for each variable in Unsat-clause do
                  score_j ← score(x_j)
                  ▽否則，評估子句內所有的變數
             end
             V ← minimum(score)  ▽取 score 最小之變數爲 V
        end
        T ← flip(T, V)

        if T is in tabu list then
             f ← 1, return (T, f)
        else if T is better than pBest then
             pBest ← T,  loop ← 0,  stuck ← 0
        else if T = pBest then
             loop ← loop+1
             if loop > loop_limit then f ← 1
        else if T is worse than pBest then
             stuck ← stuck+1
             if stuck > stuck_limit then f ← 1
        end

    end
    if f = 1 and pBest is not in tabu list then
        add pBest into tabu list
    end
    return (T, f)
```

【圖 3】 reSAT 演算法

```
Procedure reSAT-test(Φ)
    T ← initial(Φ)
    ▽隨機產生一組初始解
    for i ← 1 to Max-flip do
        (T, f) ← reSAT(Φ, T, flips-limit, p)
        ▽執行區域搜尋演算法
        if T satisfies Φ then return T
        if f = 1 then T ← initial(Φ)
        ▽若 T 需重設，產生另一組解
    end
    return "沒有找到可滿足之解"
```

【圖 4】 測試區域搜尋演算法之虛擬碼

的策略如下：當區域搜尋演算法得到一新的本次最佳解 *pBest* 後，計算其找到同一個解的次數，當次數大於 *loop-limit* 時，則視為陷入區域最佳解之中，準備重設該解。

第三項重設條件為 Tabu，即禁忌搜尋。為了防止反覆對可能沒有全域最佳解的區域進行搜尋，本論文所提出之區域搜尋演算法會在一解因前述兩者原因重設時，將其加入禁忌名單(tabu list)之中；在執行區域搜尋時，若目前解可於禁忌名單中尋獲，則視為重複搜尋之前認定無全域最佳解的區域，並重設該解。在實作上，根據所觀察到的結果，對本論文所提出的適應度函數 f_{TOTAL} 來說，內容相異的兩個染色體，得出相同的適應度值的機率相當低；故利用此一現象，我們可以設計一非常長的禁忌名單而不使系統負荷過重且節省比對搜尋的時間，以確保無全域最佳解的區域不會被重複搜尋。

六、區域搜尋演算法整體架構

圖 3 詳列出本論文所提出之區域搜尋演算法。為了易於記憶，取其反覆重設之意，將此演算法記為 reSAT。區域搜尋演算法 reSAT 可敘述如下：

1. 初始化：將輸入解 X 設為本次最佳解 *pBest*，並將未進步次數 *stuck*、重複次數 *loop* 設為 0、「需重設」旗標 *f* 設為 0。
2. 從布林表示式 Φ 中，隨機選出一未被目前解 X 滿足之子句 C_k。
3. 亂數產生一介於 0 與 1 的數，若此數小於門檻值 *p*，則執行步驟 4，否則執行步驟 5。
4. 依步驟 3 之結果在 C_k 中隨機選擇一變數 x_j。
5. 對 C_k 中所有出現的變數執行評估，並選出分數最小者 x_j。
6. 對目前解 X 中之變數 x_j 執行翻動。
7. 檢查目前解 X 是否存於禁忌列表中，若有，則結束演算法並回傳「需重設」。
8. 檢查目前解 X 是否與本次最佳解 *pBest* 相等，若相等，則重複次數 *loop* 加一，且若重複次數已達上限 *loop-limit*，則將需重設旗標 *f* 設為 1。
9. 檢查目前解 X 是否大於本次最佳解 *pBest*，若大於，則未進步次數 *stuck* 加一，且若未進步次數已達上限 *stuck-limit*，則將需重設旗標 *f* 設為 1。
10. 重複步驟 2 至 9，直到達到停止條件或尋獲全域最佳解為止。

為了使本研究所提出之區域搜尋演算法能夠獨立運作，以利測試其效能，我們用一簡易的演算法外加於區域搜尋演算法之外，來達到測試的目的。圖 4 說明了此一演算法之虛擬碼。

七、研究結果與資料分析

本研究中所使用的題組為 random 3-SAT 問

題。在所有子句之字母數為三的情況下，根據研究指出，子句數量與變數數量與的比值低於 4.3 時，由於子句數量、或限制條件過少，使得題目容易被找到可滿足之解；Back、Eiben 與 Vink[2]，Gottlieb 與 Voss[10]，Marchiori 與 Rossi[17]，de Jong 與 Kosters[4]等四組研究人員，利用 mkcnf 產生器，以強迫有可滿足之解的方式，分別產生四小組題目，並在 Gottlieb 等人發表的論文中，統整為一個大題組[12]。此題組包括題目數量、大小等詳細資料請見表 1。對此題組，我們採用 Gottlieb 等人論文之中的比較方法，在限制條件為 300,000 次翻動的上限下，對每個題目執行 50 次的測試，以比較於各個題目中，尋獲可滿足方程式之解的成功率(success rate, SR)及比較尋獲最佳解所需翻動的平均步數。

本研究以表 2 所列之參數設定進行實驗，並將測試之結果與表 3 列出。

此外，本研究將與 GASAT、SAWEA、RFEA、RFEA2+、FlipGA、ASAP、WSAT 共七種演算法做比較，以下為各演算法簡單的介紹：

GASAT：2006 年，Hoa 等人提出，採用與 FlipGA 相似的，結合基因演算法與區域搜尋演算法的概念，並加入了禁忌搜尋演算法及改良式的 uniform crossover 等方法。

SAWEA：由 Eiben 與 van der Hauw 於 1997 年提出，利用一隨時間改變的適應度函數 f_{SAW}，或稱權重逐步適應法(Stepwise Adaptation of Weights, SAW)，搭配基因演算法做搜尋；若某一子句在搜尋過程中一直沒有被滿足，則此法會將該子句之權重提高，藉以加強對其之搜尋力度。

RFEA2 與 RFEA2+：此兩種演算法由 Gottlieb 與 Voss 於 1998 年提出，利用精練函數將所有的變數加上一權重，並隨時間經過調整權重，以決定在搜尋時，該變數的值應該偏向零或一。

【表 1】 題組之詳細資料

小題	題目數	不同變數量的題目數	變數數量 n	出處
A	12	3	30, 40, 50, 100	參考文獻[2]
B	150	50	50	參考文獻[10]
			75, 100	參考文獻[17]
C	500	100	20, 40, 60, 80, 100	參考文獻[4]

【表 2】 測試區域搜尋演算法之參數設定

每回合執行步數 *flips-limit*	*p*	*stuck-limit*	*loop-limit*
Φ所含變數量之 4 倍	0.3	1000	3

FlipGA：由 Marchiori 與 Rossi 於 1999 年提出，此法先由基因演算法產生子代，再由區域搜尋演算法來改進子代的品質，並重複以上步驟來做搜尋。

ASAP：FlipGA 之變體，於 2000 年提出，不同點在於改進了區域搜尋演算法的做法，及利用一表格儲存目前已知的十個最佳解，用以自動調整各個變數之突變率(mutation rate)。

WSAT：詳見第三節之敘述。

與其他演算法的比較結果列於表 3 中，另為求註記簡潔，表 3 採英文標示，第二行各項為：

【表 3】reSAT 區域搜尋演算法與其他演算法之比較結果

Instances	Suite	A	A	A	A	B	B	B	C	C	C	C	C
	n.b.	3	3	3	3	50	50	50	100	100	100	100	100
	var.	30	40	50	100	50	75	100	20	40	60	80	100
reSAT	SR(%)	**100**	**100**	**100**	**100**	**100**	**99**	**89**	**100**	**100**	**100**	**99**	**90**
	flips	810	414	4947	11260	4757	23125	49891	256	1849	11269	30055	52249
GASAT	SR(%)	99	**100**	91	95	96	83	69	**100**	**100**	97	66	74
	flips	1123	1135	1850	7550	2732	6703	28433	109	903	9597	7153	1533
SAWEA	SR(%)	**100**	93	85	72	-	-	-	**100**	89	73	52	51
	flips	34015	53289	60743	86631	-	-	-	12634	35988	47131	62859	69657
REFA2	SR(%)	**100**	**100**	**100**	99	**100**	95	77	**100**	**100**	99	92	72
	flips	3535	3231	8506	26501	12053	41478	71907	365	3015	18857	50199	68053
REFA2+	SR(%)	**100**	**100**	**100**	97	**100**	96	81	**100**	**100**	99	95	79
	flips	2481	3081	7822	34780	11350	39396	80282	365	2951	19957	49312	74459
FlipGA	SR(%)	**100**	**100**	**100**	89	**100**	82	57	**100**	**100**	**100**	73	62
	flips	25490	17693	127900	116653	103800	29818	20675	1073	14320	127520	29957	20319
ASAP	SR(%)	**100**	**100**	**100**	**100**	**100**	87	59	**100**	**100**	**100**	72	61
	flips	9550	8960	68483	52276	61186	39659	43601	648	16644	184419	45942	34548
WSAT	SR(%)	**100**	**100**	**100**	**100**	95	84	60	**100**	**100**	94	72	63
	flips	1631	3742	15384	19680	16603	33722	23853	334	5472	20999	30168	21331

【表 4】 區域搜尋演算法 reSAT 測試結果

題組	題數	變數量	成功率 SR (%)	平均翻動次數	平均時間 (秒)	平均重設次數	stuck 重設數	loop 重設數	tabu 重設數
A	3	30	100	810.5	0.06	5.57	0	5.34	0.22
A	3	40	100	414.2	0.042	2.08	0	2.03	0.04
A	3	50	100	4947.9	0.52	20.92	0.13	20	0.79
A	3	100	100	11260.7	2.21	19.46	0.61	18.12	0.73
B	50	50	100	4757.9	0.50	19.54	0.16	18.42	0.96
B	50	75	99.32	23125.5	3.48	64.88	0.64	59.23	5.00
B	50	100	88.84	49891.3	9.76	99.68	1.62	89.51	8.55
C	100	20	100	256.7	0.02	2.25	0.01	2.16	0.09
C	100	40	100	1849.8	0.17	9.07	0.08	8.65	0.34
C	100	60	99.70	11269.1	1.39	39.54	0.33	36.47	2.74
C	100	80	99.14	30055.9	4.79	77.54	0.89	70.86	5.78
C	100	100	89.52	52249.0	10.23	104.71	1.69	93.56	9.46

題組 Suite、題數 n.b.、變數量 var.、成功率 SR、平均翻動次數 flips。

在表 3 中可以看到，本論文所提出之 reSAT 區域搜尋演算法與其他演算法相比，尤其是在變數數量等於一百的題目中，是相當具有效能的；並於表 4 中可見，本論文為區域搜尋演算法所提出之三種重設條件為可用的，而且以 loop 條件與 tabu 條件出現之頻率為最高。

八、結論與未來研究方向

在解可滿足性問題的演算法中，常存在有下列主要缺點：1.搜尋不夠系統化，有可能在區域最佳解附近花費過多時間搜尋；2.目前所慣用之適應度函數對於引導搜尋的力道不夠，導致無法有效獲得可滿足之解；3.演算法的泛用性不足，對於變數數量較少的題組有效的演算法，在變數數量較多的題組中則表現不佳，反之亦然。

本論文提出一新的區域搜尋演算法，除了加入三種重設條件，使得搜尋過程中能夠提早預知並避免重複搜尋區域最佳解外，利用「對所有的已滿足的子句而言，最好的情況是子句中只存在有一個為真的字母，使得其餘的字母能夠去滿足更多其他的子句」的概念，加入了新的輔助性適應度函數 f_{lit}，使得所提之區域搜尋演算法 reSAT 搜尋能力更強。

實驗結果顯示，本論文中所提的兩項主要策略，尤其在變數量等於一百的題目中，能夠有效提升解可滿足性問題演算法的效能。

在未來的研究中，為了在能夠更快獲得更好的解，有兩個主要方向可以改進：第一是本文中之適應度函數，雖在本論文中已獲得不錯的成果，但實際上對其理論驗證稍嫌不足，使其仍有改進的空間；第二則是增加一個上層基因演算法，使本演算法對於變數較多的題組也能有好的

結果。另外，也可以往平行運算領域發展，藉以增進解題效率。

九、參考文獻

[1] H.M. Adorf and M.D. Johnston, “A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems,” Proceedings of the International Joint Conference on Neural Networks, vol. 3, pp. 917-924, June 1990.

[2] T. Back, A. Eiben and M. Vink, "A Superior Evolutionary Algorithm for 3-SAT," Proceedings of the 7th Annual Conference on Evolutionary Programming, vol. 1477 of LNCS, pp. 125-136, 1998.

[3] A. Beringer, G. Aschemann, H.H. Hoos, M. Metzger and A.Weiss, "GSAT versus Simulated Annealing," Proceedings of the European Conference on Artificial Intelligence, pp. 130-134, 1994.

[5] M. de Jong and W. Kosters, "Solving 3-SAT using adaptive sampling," Proceedings of 10th Dutch/Belgian Artificial Intelligence Conference, pp. 221-228, 1998.

[6] K.A. De Jong and W.M. Spears, “Using Genetic Algorithms to Solve NP-Complete Problems,” Proceedings of the third international conference on Genetic algorithms, pp. 124-132, 1989.

[7] A. Eiben and J. van der Hauw, "Solving 3-SAT with Adaptive Genetic Algorithms," Proceedings of the 4th IEEE Conference on Evolutionary Computation, pp. 81-86, 1997

[11] I.P. Gent and T. Walsh, “Towards an Understanding of Hill-climbing Procedures for SAT,” Proceedings of AAAI-93, pp.28-33, 1993.

[12] I.P. Gent and T. Walsh, “Unsatisfied Variables in Local Search,” in Hybrid Problems, Hybrid Solutions (AISB-95), pp.73-85, 1995.

[13] J. Gottlieb and N. Voss, "Representations, Fitness Functions and Genetic Operators for the Satisfiability Problem," Proceedings of the 5th International Comference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol. 1498, pp. 755-764, 1998.

[14] J. Gottlieb and N. Voss, "Adaptive Fitness Functions for the Satisfiability Problem," Proceedings of the 6th International Comference on Parallel Problem Solving form Nature. Lecture Notes in Computer Science, vol. 1917, pp. 621-630, 2000.

[15] J. Gottlieb, E. Marchiori and C. Rossi, "Evolutionary Algorithms for the Satisfiability Problem," Evolutionary Computation, vol. 10, no. 1, pp. 35-50, 2002.

[16] J. Gu, “Efficient Local Search for Very Large-Scale Satisfiability Problems,” ACM SIGART Bulletin, vol. 3, no. 1, pp. 8-12, Jan. 1992.

[17] P. Hansen and N. Mladenovic, "An Introduction to variable neighborhood search," in Advances and Trends in Local Search Paradigms for Optimization, pp. 433-458, 1999.

[18] J.K. Hao, F. Lardeux and F. Saubion, "Evolutionary Computing for the Satisfiability Problem," Applications of Evolutionary Computing, vol. 2611 of LNCS, pp. 258-267, April 2003.

[19] J.K. Hao, F. Lardeux and F. Saubion, "GASAT: a genetic local search algorithm for the Satisfiability Problem," Evolutionary Computation, vol. 14, no. 2, pp 223-253, June 2006.

[21] E. Marchiori and C. Rossi, "A Flipping Genetic Algorithm for Hard 3-SAT Problems," Proceedings of Genetic and Evolutionary Computation Conference, pp. 393-400, 1999.

[22] B. Mazure, L. Sais and E. Gregoire, "Tabu Search for SAT," Proceedings of the 14th National Conference on Artificial Intelligence, pp. 281-285, 1997.

[26] C. Rossi, E. Marchiori and J. Kok, "An Adaptive Evolutionary Algorithm for the Satisfiability

Problem," Proceedings of ACM Symposium on Applied Computing, pp.463-469, 2000.

[27] B. Selman, H.J. Levesque, and D. Mitchell, “A New Method for Solving Hard Satisfiability Problems,” Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 440-446, 1992.

[29] B. Selman and H. A. Kautz and B. Cohen, “Noise Strategies for Improving Local Search,” Proceedings of the twelfth national conference on Artificial intelligence, vol. 1, pp. 337-343, 1994.

[30] D. McAllester, B. Selman and H. Kautz, "Evidence for Invariants in Local Search," Proceedings of the National Conference on Artificial Intelligence, pp.321-326, 1997.