**國立臺灣大學工學院工業工程研究所**

**柔性計算法與應用期末報告**

**Animal Food Chain Based**

**Particle Swarm Optimization**

**食物鏈型態優化粒子團演算法**

**授課老師：楊烽正 副教授**

**學生：R08546036 工業工程學研究所 趙上涵**

**目次**

[**I. Introduction** 3](#_Toc61810734)

[**II. Methods and Problems** 4](#_Toc61810735)

[**III. Requirements and Functionality Implementation of the System** 11](#_Toc61810736)

[**IV. Numerical or Example Tests** 12](#_Toc61810737)

[**V. Conclusion and Discussion** 13](#_Toc61810738)

# **I. Introduction**

本次所挑選的論文研究對象為由Rutgers, The State University of New Jersey的Ayca Altay及Istanbul Technical University於2011年所發表的 Animal Food Chain Based Particle Swarm Optimization一文，於論文開頭便提到傳統上粒子團演算法 (Particle Swarm Optimization Algorithm)常會面臨的幾個問題，首先為被區域最佳解給侷限住，導致無法進一步求出更為理想的解；二為過早的收斂，演算法無法繼續嘗試並迭代更好的解可能演算法解；最後則為太著重於整個解領域的搜尋，導致未能於具有潛力的解領域做徹底的搜尋並找出適合的解 (Exploration overwhelming the exploitation)。

在這樣的著眼點上，作者首先介紹了受到大自然捕獵行為啟發、改進後的兩個粒子團演算法：捕獵者與獵物演算法 (Predator Prey PSO) 及狩獵演算法 (The Hunting Search)，並進而介紹自己在修正前述兩項演算法後得出的食物鏈演算法(Animal Food Chain Based PSO)作為本篇論文的主要主題。在於本次報告中，學生將依序建立起前述三者的演算法邏輯，並使用老師於課堂上所提供的Benchmark Problem作為比較傳統粒子團演算法及前述三個改進後的演算法的方式。

在傳統的粒子團演算法中，粒子團的粒子分別會受到自身在搜尋過程中找尋到的最佳解、以及整個群體在歷個代次搜尋到的最好解影響，並更新粒子自己的解內容，從而推導出新的解組合以及測試是否存在更為好的解答案。本篇論文的三個演算法中，皆是將這樣的演算法架構作為基礎發展而來的，所主要處理解的同樣為連續型優化問題，每個粒子團的的粒子則分別代表一個可能的解，並可以依序這個解計算出相對應的目標值。

# **II. Methods and Problems**

在於本部分，學生將會分別為狩獵者與獵物演算法、狩獵演算法、食物鏈演算法做演算法介紹。

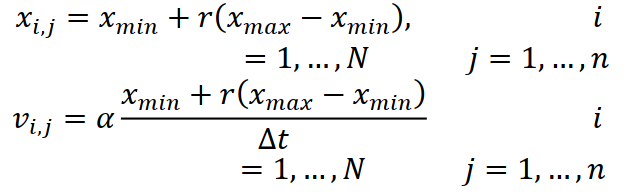
**一、狩獵者與獵物演算法 (Predator Prey PSO)**

本演算法師法大自然中捕獵者及及獵物的樣態，將粒子團區分為狩獵者及獵物兩個族群，兩個族群有著不同的解迭代及更新方式。捕獵者將會依據周遭的獵物的最佳解來決定解的更新方向；獵物則除了受到獵物群體最佳解的影響之外，同時將可能受到捕獵者的影響（取決於獵物是否害怕以及與捕獵者的距離決定），摘要來說，本演算法在傳統PSO的Cognitive Coefficient和Social Coefficient之外主要新增了幾個參數：

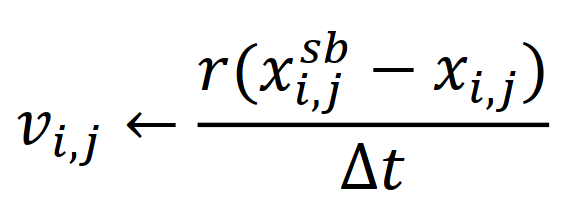
1. **獵物感到害怕機率 (Fear Probability)：**為獵物受到周遭捕獵者影響的機率，若將本參數設置薇零，則獵物群體的搜尋解方式將會和一般粒子團演算法的搜尋方式沒有差異。
2. **獵物參數 (Prey Coefficient)：**如果本參數設定值大於 Cognitive Coefficient（即獵物追尋自己的最佳解經驗的比例參數）和 Social Coefficient（獵物追尋整個群體搜尋過的最佳解參數），則整個粒子團將因為捕獵者的存在而成為發散 (diverge) 並在可行解區域進行隨機的搜尋。
3. **距離參數A和距離參數B (Distance Coefficient) ：**距離參數A和**Fear Probability有類似的效果，**距離參數B則有著相反的效應。

**演算法主要執行步驟則分為幾個步驟：**

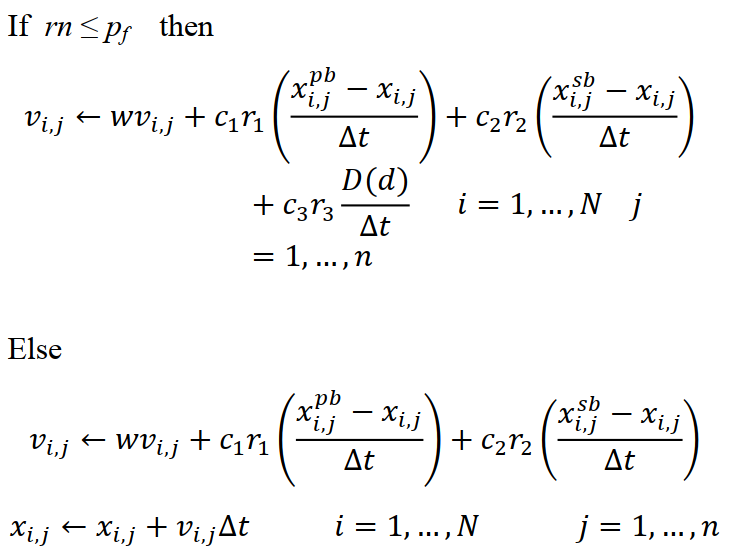
1. 在區分完捕獵者及獵物兩類粒子後，分別給予初始解和初始速度，如下方公式所示。

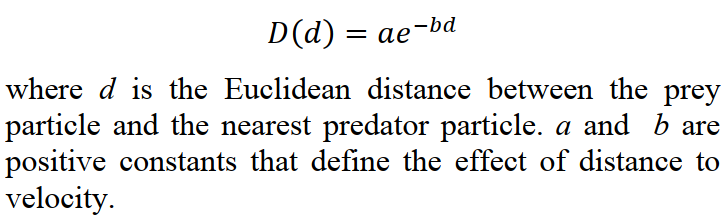


1. 計算目標值，並更新捕獵者群和獵物群分別的粒子最佳解及群體最佳解。
2. 更新捕獵者群體的速度值，並使用該速度值更新解。相對於傳統PSO所使用的為整個群體的最佳解作為更新標準，本演算法中的捕獵者使用的為「獵物」群體的最佳解作為目標對象以更新最佳解，其所模仿的即為大自然中捕獵者追逐獵物的樣態。



1. 更新獵物群體的速度值，並使用該速度值更新解。在這裡前述提到的Fear Probability參數即派上用場，更新每個獵物粒子的解時，將會隨機生成一亂數，而假若此亂數較Fear Probability為小，則認定為該獵物為恐懼的狀態，此時與最近捕獵者的距離參數便會被納入並用以更新解。





1. 持續反覆直到該結果收斂

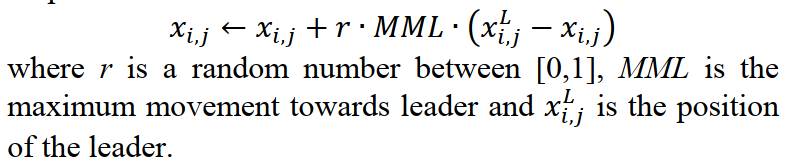
**二、狩獵演算法(The Hunting Search)**

另一個基於傳統粒子團演算法並導入大自然捕食概念的則為狩獵演算法。本演算法師法的為自然界中狩獵群體在團體狩獵上的行為，然而與大自然情不同的，在於本演算法中並不是以「粒子」作為獵物的代表，而是將我們所要搜尋的「最佳解」作為粒子狩獵群體的目標。摘要來說，本演算法中，每個代次最為佳的解將會被視為狩獵群體的「領導者」，其他粒子將以這個狩獵領導作為移動目標進行移動，而在狩獵群體收斂到一定程度後，除了原先搜尋到的最佳解外，其餘的粒子將會打散並重新開始這樣的狩獵及尋找解的流程。在於這個演算法中，主要相較於傳統的PSO新增了幾個參數：

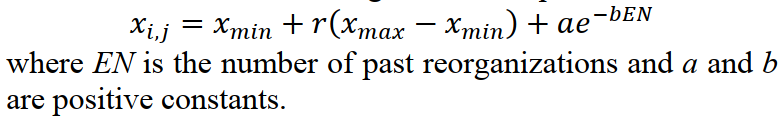
1. **朝向狩獵領導者移動的最大幅度 (Maximum movement towards leader – MML)：**將會影響每次粒子朝向狩獵領導移動的幅度，通常較小的訓練代次會設置較大的MML參數，反之亦然。
2. **狩獵群體大小 (Hunting group size) ：**本參數和傳統PSO的群粒子數量沒有差異。

**演算法主要執行步驟則分為幾個步驟：**

1. 隨機給予每一個捕獵者（粒子）一個初始解及初始速度。
2. 計算目標值，並更新每個粒子的最佳解及群體最佳解，並將擁有最佳解的粒子視為「狩獵領導」。
3. 每一個粒子依照「狩獵領導」的位置更新自己的解方位，並往領導者的位置前進。所移動的距離則交由亂數及MML參數決定。



1. 檢查狩獵群體是否過於收斂，若過於收斂，除了狩獵領導外其餘粒子重新建立自己解，其中新的解將會與過去的「重新建立」次數和相關參數有關。如下列所示：

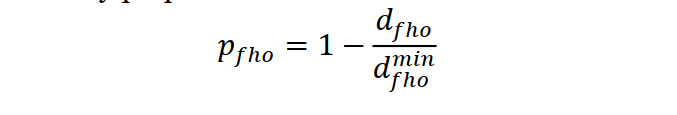


1. 持續反覆直到解收斂。

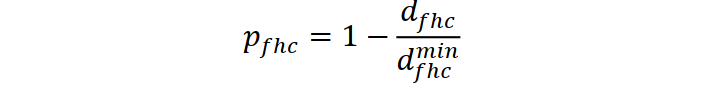
**三、食物鏈演算法(Animal Food Chain Based PSO)**

最後欲介紹的即是本次論文作者所提出的食物鏈演算法 (Animal Food Chain Based PSO)。本演算法是基於狩獵者與獵物演算法 (Predator Prey PSO) 的，然而本演算法導入了食物鏈分層的概念，將粒子團分為草食者 (Herbivores)、雜食者 (Omnivores)以及肉食者 (Carnivores) 三類，其中草食者位於食物鏈的最底層、為被獵食者；雜食者則同時扮演了捕食及被捕食的角色；肉食者則為最終的捕食者。三個族群的數量的分布，則參考了大自然界中因為自然環境資源多寡而形成的草食、肉食動物數量分布，將環境自然資源富庶從低至高分為Harsh（草食：雜食：肉食＝10:3:1）、Average（草食：雜食：肉食＝25:6:1）及Calm（草食：雜食：肉食＝40:10:1）三種比率，並以此分別配出相應的族群分布。在於這個算法中，主要新增了幾個參數：

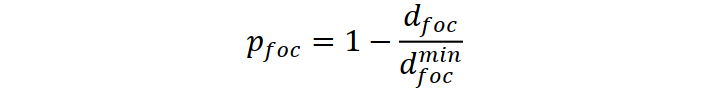
1. **草食動物對雜食動物的恐懼參數 (Fright factor of herbivores to omnivores - )：**為草食動物對雜食動物的恐懼程度，其算法為1減掉「周遭最近捕食者距離」除以「最小面對該類捕食者恐懼距離」。



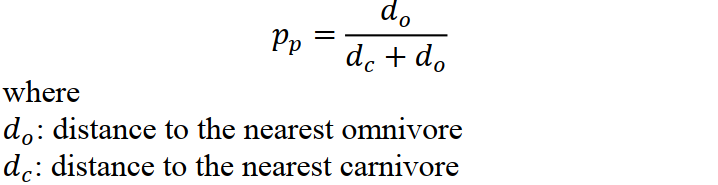
1. **草食動物對肉食動物的恐懼參數 (Fright factor of herbivores to carnivores- )：**為草食動物對肉食動物的恐懼程度，其算法為1減掉「周遭最近捕食者距離」除以「最小面對該類捕食者恐懼距離」。



1. **雜食動物對肉食動物的恐懼參數 (Fright factor of omnivores to carnivores - )：**為雜食動物對雜食動物的恐懼程度，其算法為1減掉「周遭最近捕食者距離」除以「最小面對該類捕食者恐懼距離」。



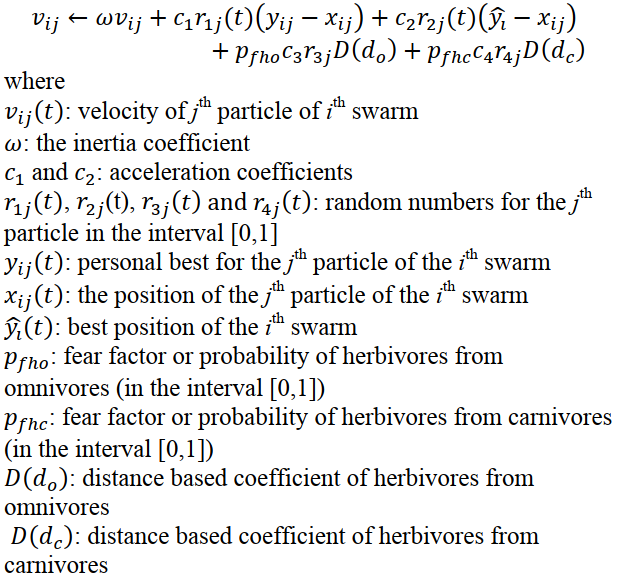
1. **雜食動物成為捕獵者機率 (Predator probability- )：**為雜食動物成為「捕獵者」的機率 ，其算法為「周遭最近雜食動物的距離」除以「周遭最近肉食動物的距離加上周遭最近雜食動物的距離」



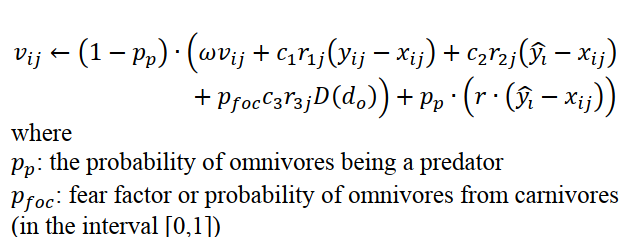
1. **環境參數 (Environmental factor)：**如上一段所闡述，共有三種環境比率可供選擇，將環境自然資源富庶從低至高分為Harsh（草食：雜食：肉食＝10:3:1）、Average（草食：雜食：肉食＝25:6:1）及Calm（草食：雜食：肉食＝40:10:1）三種比率。

**本演算法在這樣的架構及參數下，主要按照幾個步驟執行，如下所示：**

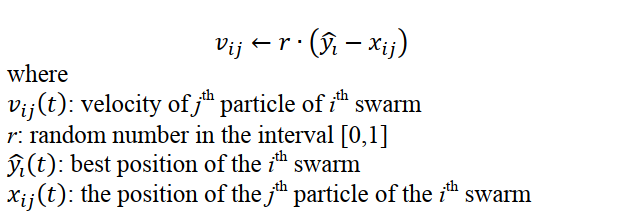
1. 依照所選擇的環境樣態，分配出相應的草食、雜食、肉食族群的粒子數量，並仿造傳統PSO的方式生成初始解。
2. 計算草食群體、雜食群體、肉食群體分別的最佳解。
3. 各群體分別更新解，依據各自的更新公式，分別為：
   1. 草食性動物



* 1. 雜食性動物



* 1. 肉食性動物



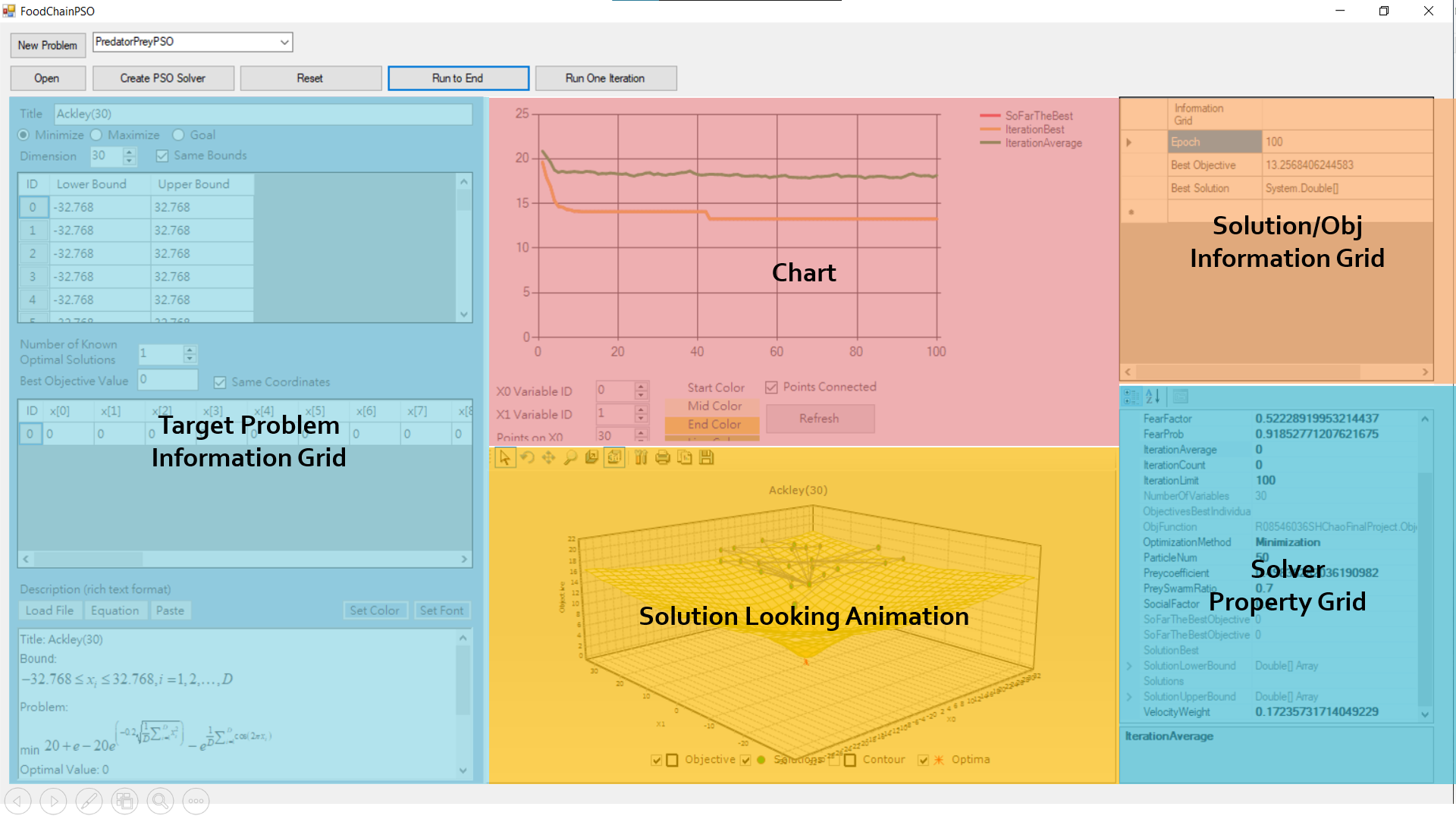
1. 持續迭代至演算法收斂。

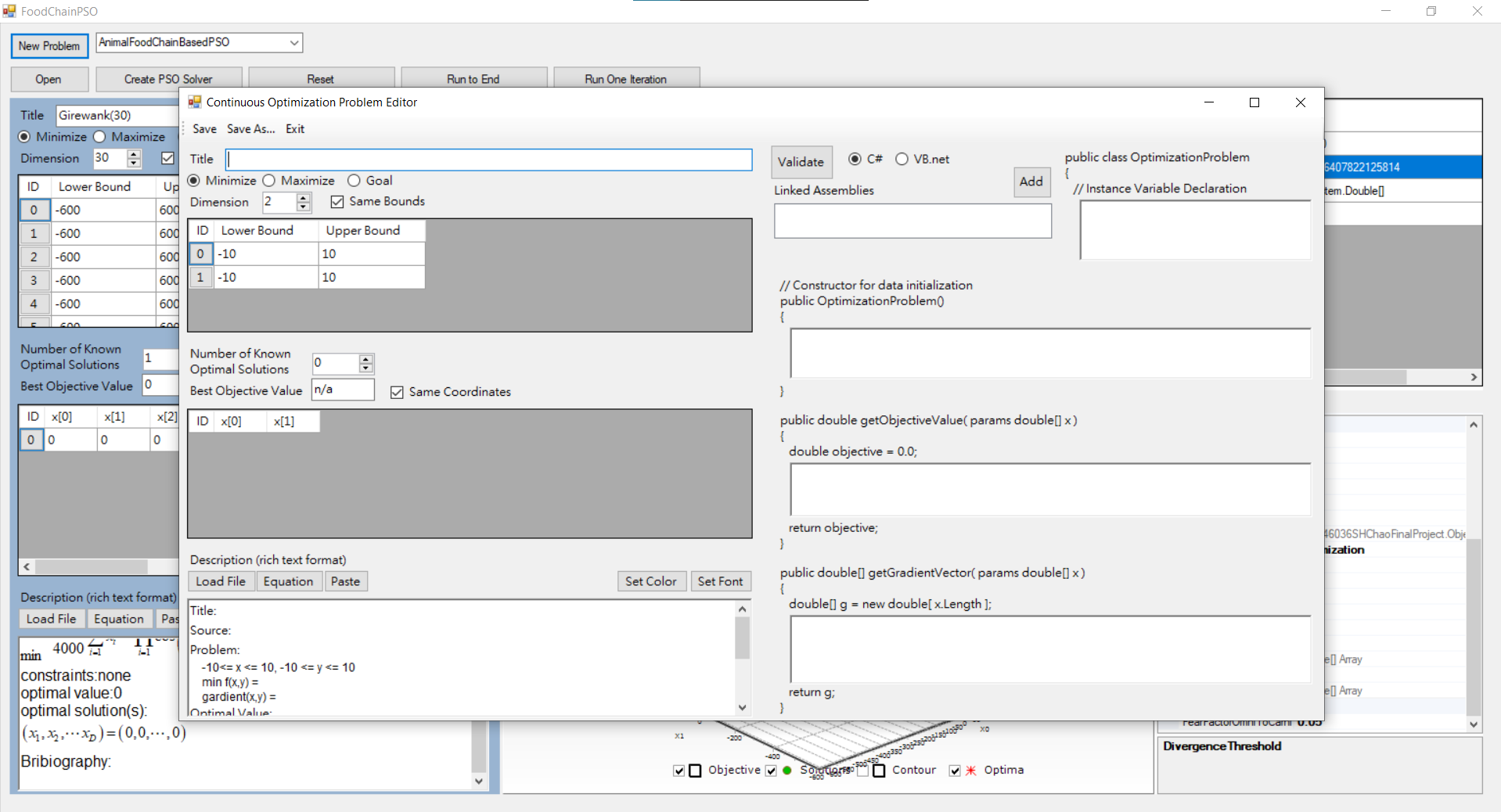
# **III. Requirements and Functionality Implementation of the System**

本次演算法求解系統的設計，在流程上需要使用者先以讀入/建立所需要求解的連續型優化問題，並選擇所欲套用的演算法來建立求解器 (Solver)，在調整完所需參數後，進行求解器的重整。最後可選擇以迭代一次或迭代至設定上限的方式進行求解。本系統便會按照使用者所設定的參數及設置對讀入的連續型優化問題進行求解，並將求解的結果以動畫、折線圖及資訊框的方式陳列於程式介面中。

本次設計界面如下圖所示，除了設置可讀入、建立求解問題的按鈕外，並有配置Panel用以作為問題本身資訊及求解過程的展示，使使用者能夠在求解器求解問題的過程中，即時觀測求解狀況並決定是否更換參數或以更多的迭代次數來求解問題。

圖：求解系統主介面及供使用者建立自訂問題的選框





# **IV. Numerical or Example Tests**

在於本次演算法測試中，我分別就Branin(2)、Easom(2)、Ackley(2)、Ackley(30)、Girewank(30) 幾個問題，使用傳統的粒子團演算法及本篇論文所提到的三個演算法進行試解，為求測試數據的穩定，本次測試在每個演算法上，我皆施測三次，並以測試數據平均值作為最後的判斷依據，另為了使每一個演算法都有足夠時間收斂，我在每次測試中皆是採取200個迭代次數的設置。

測試結果如下表所示，可以發現在Branin(2)、Ackley(2) 問題中，食物鏈演算法 (Animal Food Chain Based PSO) 皆有最好的表現，並在Easom(2)及Ackley(30)問題中有著次佳的結果。從這個結果來看，我們可以說食物鏈演算法是一個在傳統PSO方法上改進後並有不錯表現的演算法，並在低維連續型及高維連續型優化問題上皆有不錯的求解效果。而同時間我們並觀測到Predator Prey PSO與The Hunting Search表現亦可圈可點，亦是在處理連續型優化問題上值得去嘗試的求解演算法。

表：各演算法對各問題的求解效果比較

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Problem Name | Algorithm Name | Test 1 | Test 2 | Test 3 | Average | Known Optima |
| Branin(2) | Traditional PSO | 0.0038 | 0.0044 | 0.0074 | 0.0052 | 0 |
| Predator Prey PSO | 0.0000 | 0.0046 | 0.0087 | 0.0045 |
| The Hunting Search | 0.0395 | 0.1464 | 0.0223 | 0.0694 |
| Animal Food Chain Based PSO | 0.0005 | 0.0014 | 0.0008 | **0.0009** |
| Easom(2) | Traditional PSO | -0.9995 | -0.9708 | -0.7736 | -0.9146 | -1 |
| Predator Prey PSO | -1.0000 | -0.8974 | -1.0000 | -0.9658 |
| The Hunting Search | -0.9995 | -0.9932 | -0.9939 | **-0.9956** |
| Animal Food Chain Based PSO | -0.9788 | -0.9586 | -0.9649 | -0.9674 |
| Ackley(2) | Traditional PSO | 0.0147 | 2.7709 | 2.5846 | 1.7901 | 0 |
| Predator Prey PSO | 3.2398 | 0.0291 | 0.3322 | 1.2004 |
| The Hunting Search | 0.2833 | 1.1036 | 2.9856 | 1.4575 |
| Animal Food Chain Based PSO | 2.5809 | 0.6720 | 0.0063 | **1.0864** |
| Ackley(30) | Traditional PSO | 12.5445 | 12.3469 | 10.4512 | **11.7809** | 0 |
| Predator Prey PSO | 17.1693 | 17.7447 | 17.9689 | 17.6276 |
| The Hunting Search | 19.9804 | 19.9632 | 20.0720 | 20.0052 |
| Animal Food Chain Based PSO | 16.1664 | 16.6744 | 18.1622 | 17.0010 |
| Girewank(30) | Traditional PSO | 36.6239 | 48.1839 | 40.1376 | 41.6485 | 0 |
| Predator Prey PSO | 48.0291 | 17.9928 | 36.4985 | 34.1734 |
| The Hunting Search | 15.4686 | 15.7609 | 12.2242 | **14.4846** |
| Animal Food Chain Based PSO | 55.7128 | 50.5273 | 64.6408 | 56.9603 |

# **V. Conclusion and Discussion**

經過本次演算法的實作，我們觀察到食物鏈演算法 (Animal Food Chain Based PSO) 作為一個以粒子團演算法作為基礎的演算法，在求解連續性優化問題上的確有不錯的效果。然而，我也同時間觀察到，在與本篇論文中所設置的族繁不及備載的參數中，論文本身卻不大能可以給出一個較為推薦的參數調教及設定的方向，使得在面對不同的連續性優化問題上，使用者可能必須嘗試多種不通的參數組合，方能最後得出最適合的求解參數配置。

作為結尾，我想本次的演算法實作確實讓我看到了在啟發式演算及粒子團演算法上的多樣性及可能性，並了解到作為傳統粒子團演算法的改進，食物鏈演算法的確提供了一種不同的可能性在更好及有效率的對連續性問題進行求解。