國土防衛地面兵力部署最佳化—最短路徑網路攔截問題之應用

賴智明[a]　陳秀靜[b*]

[a] 國防大學資源管理及決策研究所
[b] 陸軍第四地區支援指揮部補給油料庫

論文編號: NM-44-01-09
DOI: 10.29496/JNDM.202411_45(2).0002
來稿 2023 年 1 月 15 日→第一次修訂 2023 年 2 月 27 日→第二次修訂 2023 年 4 月 22 日→同意刊登 2023 年 5 月 12 日

摘要

本研究以國土防衛的角度切入，採用網路攔截問題，建構第四道防線，以增加敵軍進攻的困難性，並提高戰略持久的效果。假設敵軍的作戰目標是尋求從登陸點至首都之間最小總時長的進攻路徑，因此以最短路徑網路攔截問題作為基礎，透過雙層規劃建構「防禦者-侵略者」模型，模型的上層是我軍伏擊點選擇的最佳化問題，下層是敵軍進攻的最短路徑問題。為提高求解效率與品質，本研究提出巢式序列架構的求解方法，採用簡群演算法，並加入伏擊最短路徑的區域搜索機制以提高演算法性能。最後，本研究以 20 個隨機問題及 24 個實例問題進行演算法參數設定、演算法驗證與實證分析。

關鍵詞：網路攔截、雙層規劃、巢式序列、簡群演算法、Dijkstra 演算法

* 聯絡作者：陳秀靜 email: chenshioujing14@gmail.com

Optimizing Ground Forces Deployment for National Defense – Application of Shortest Path Network Interdiction Problem

Lai, Chyh-Ming [a] **Chen, Shiou-Jing** [b*]

[a] **Graduate School of Resources Management and Decision Science, National Defense University, *Taiwan, R.O.C.***

[b] **Supply Oil Depot, Army 4th Regional Support Command, *Taiwan, R.O.C.***

Abstract

This study applies the network interception problem to the construction of the fourth line of defense for national security from the perspective of national defense. The enemy's objective is to seek the shortest total time attack path from the landing point to the capital city, and the interception actions are taken to increase the difficulty of achieving this objective for strategic endurance. Therefore, the shortest path network interception problem is adopted as the basis to construct the "defender-attacker" model using game theory. The top layer of the model is an optimization problem for choosing the ambush points for the military, and the bottom layer is the shortest path problem for enemy attack, which belongs to a two-layer binary integer programming problem. To balance solution efficiency and quality, this study proposes a nested sequence architecture based on simplified swarm optimization for solving the problem. To improve algorithm performance, a local search mechanism for the shortest ambush path is also proposed. Finally, algorithm parameter settings, verification, and empirical analysis are conducted on 20 random problems of different sizes and 24 example problems.

Keywords: Network Interdiction, Bi-level Programming, Nested Sequential, Simplified Swarm Optimization, Dijkstra Algorithm

[*] Corresponding Author: Chen, Shiou-Jing email: chenshioujing14@gmail.com

一、緒論

1.1 研究背景

當臺海戰爭爆發時，軍事戰略的指導原則是「防衛固守，重層嚇阻」，依此制定「戰力防護、濱海決勝、灘岸殲敵」的防衛作戰構想（Hsi-min and Lee, 2020）。該構想旨在離岸90海哩內的海峽地區建立多層次的防禦系統，透過聯合火力的逐層攻擊，削弱敵軍的作戰能力，直至在敵軍到達海岸線前擊敗之，以達到戰略持久的效果，最終實現「迫敵奪島任務失敗」的作戰目標。

在國土防衛戰爭中，中共的目標是「迅速佔領臺灣，不讓第三方有機會介入」。一旦登陸成功，其作戰目標將以政治和經濟中心為主要目標，指揮和管轄情報系統為次要目標，逐一佔領臺灣各地，實現對臺灣全島政局和民眾治理權的掌控。為了阻止中共的計劃，我方軍隊可以試圖將防禦深度從原本的灘岸延伸至臺灣本島上，並通過由近250萬後備軍人組成的國土防衛軍，形成防空、濱海和灘岸之後的第四道防線。我軍可以利用小群多點的原則，在敵軍可能的進攻路線上設置多個伏擊點，逐步削弱敵軍的戰鬥力。這不僅從空間上實質增加了敵軍奪島的戰術難度和傷亡成本，更從精神上鼓舞全民共同防禦的決心和凝聚總體戰力，使其無法輕易掌控臺灣（Hunzeker et al., 2018）。

縱深作戰中的第四道防線，是指將國土防衛軍部署在敵軍必經之處，例如交通樞紐、城鎮、叢林和山區等，採用高機動、小規模的伏擊方式，以阻擋或遲滯敵軍長驅直入奪取關鍵目標（如臺北市），從而實現游擊戰的目的。此舉可以將戰場推延至本島，發揮主場優勢，對敵軍造成最大的阻礙和威脅，以空間換取時間，爭取國際支援的寶貴時間。透過主場優勢和軍民同心的整體戰力，這並不是一個新的防禦概念（陳勝昌，2013），而游擊戰的效果已在近年伊拉克戰爭和阿富汗戰爭中得到了充分驗證。伊拉克和阿富汗都在戰局不利的情況下，整合非正規軍力，採用游擊戰的方式對抗美軍的入侵，使美軍陷入城鎮、山區等局部戰鬥的泥淖之中，體現了非正規軍隊採取游擊戰對抗正規軍的驚人效果（Ricks, 2007; Strachan, 2019）。

1.2 研究動機與目的

本研究旨在探討如何在敵軍登陸後的情況下，透過建構第四道防線，延續國土防衛作戰，發揮全民整體戰力並展現護國意志。雖然許多學者已就此議題發表研究，然而從軍事作業研究的角度，仍缺乏量化模型來探討第四道防線的建構相關議題（吳傳國，2009；葉紘胥，2017；Sheng and Gao, 2016），為本研究動機之一。此外，臺海情勢具有作戰縱深淺、反應時間短的特殊性，且地理環境錯綜複雜、變化多端，使得國軍需進行龐大的計算才能制定適當的戰爭預防、前置準備和作戰策略等，已超出一般作戰參謀的能力和負荷（蔡宗憲，2020）。因此，本研究動機之二在於建構適當的量化工具，以加速參謀的戰場情報分析並協助作戰指揮官做出決策。

網路攔截（network interdiction）主要探討如何利用有限的兵力，透過游擊戰的方式在路網中阻擋敵軍，避免其進攻並奪取目標（Wollmer, 1964）。在軍事防禦中，攔截行動是指先發制人，限制敵軍的作戰行動，使其無法發揮預期的戰鬥能力。攔截的方式包

括拘束其增援部隊、遲滯其補給鏈或截斷其通訊網路等。過去相關的研究主要著重在網路結構方面，根據網路的種類、攔截的標的（如路網中的城市、橋樑或路段）以及通訊網路中的訊號鏈或轉接器等，來進行攔截（Bidgoli and Kheirkhah, 2018; Smith and Song, 2020; Wood, 2010）。通常的做法是先識別需要攔截的目標，例如關鍵邊緣（arc）或節點（nodal），再進行有限的攔截行動，以阻斷敵軍的進攻路線、補給鏈或通訊網路等，以達到限制或降低其作戰能力的目的。

本研究聚焦於國土防衛作戰中應用網路攔截的概念，將遏止敵軍登陸的作戰範圍擴大至臺灣本島，透過國土防衛軍中近250萬後備軍人組成的第四道防線，以小群多點的原則，在敵軍潛在攻擊路線上設置有限的伏擊點，逐步削弱敵軍戰力（Hunzeker et al., 2018），並基於網路攔截問題透過雙層規劃建構數學模型。模型中，我方軍隊扮演上層領導者的角色，負責攔截敵軍的進攻行動，而敵方軍隊則扮演下層追隨者的角色，依據我方軍隊的攔截行動來計劃進攻策略。敵方軍隊的目標是尋找從登陸點至首都之間的最短入侵路徑，而我方軍隊的目標則是透過攔截行動增加敵軍入侵的困難度（例如使用伏擊方式拖延敵軍前進或迫使其繞路），從而建構出「防禦者-侵略者」雙層二元整數規劃模型。在此模型中，上層問題為我方軍隊選擇最佳的伏擊點組合，下層問題則為敵方軍隊選擇最短路徑問題。此為本研究目的之一。

Wood（1993）證明即使在所有邊成本相同的情況下，網路攔截問題仍是NP-complete問題，而雙層規劃也被證實屬於NP-hard問題。因此，求解網路攔截問題的最佳解是一項極具挑戰的任務。自1970年以來，學者提出了多種雙層規劃的求解方法，包括分支定界法（branch-and-bound）（Bard and Falk, 1982）、Karush-Kuhn-Tucker方法（Bard and Moore, 1990）、懲罰函數法（penalty function）（White and Anandalingam, 1993）和班德氏分解法(Benders decomposition）（Pan et al., 2003）。然而，雙層規劃問題的內在複雜性使傳統的解法耗時費力（Israeli, 1999）。因此，學者們陸續提出了基於進化式演算法的求解架構，包括巢式序列、協同進化、單層轉換和多目標轉換等，力求在合理時間內求得近似最佳解。與其他三種方法相比，巢式序列的求解架構是當前的主流趨勢之一（Said et al., 2022; Sinha et al., 2017; Talbi, 2013）。巢式序列的特點是在上、下層問題中採用適當的求解方法（精確演算法或進化式演算法），以巢狀方式求解雙層規劃的上、下層問題。在求解過程中，需要為上層問題的每個候選解給予下層最佳化後的解，因此其運算效率主要取決於下層問題的複雜度。

大量研究的結果顯示，簡群演算法（Simplified Swarm Optimization, SSO）在參數調整、運算效率等方面具有顯著優勢，使其在組合最佳化問題的求解方面表現優異（Lai and Yeh, 2016）。本研究希望通藉由本議題，擴展簡群演算法的應用領域，推廣國內演算法的發展（Yeh, 2009）。因此，本研究以群演算法為基礎，設計並開發一個巢式求解架構，以求解本研究提出的雙層二元整數規劃問題，以協助決策者在可接受的運算時間內制定適當的國土防禦策略，是為本研究目的之二。

二、文獻探討

2.1 網路攔截問題

網路攔截問題（network interdiction problem）最早源自於1950年代所建立的最大流最小割定理（Ford and Fulkerson, 1956）。網路攔截問題早期的應用側重於防禦策略，主要被定義為通過掌握當前對手的決策來最佳化自身的網路系統安全，進而阻止蓄意攻擊引起的事件，是尋找網路系統漏洞的共通議題之一。在求解最佳化網路攔截問題的文獻中，通常會以雙層規劃的架構建模，視為領導者與追隨者之間的賽局（leader-follower game）（Smith and Lim, 2008）。

發展至今，網路攔截問題衍生多種型態，以目標函數可區分為三種類型：（一）最大流量網路攔截問題、（二）最大可靠路徑網路攔截問題與（三）最短路徑網路攔截問題（Smith and Song, 2020; Xiang and Wei, 2020）。無論何種類型，領導者通常先採取行動，攔截網路中部份的邊或節點，追隨者再依據攔截後的網路連通狀況，最佳化其目標函數，此目標函數則決定了該問題屬於何種類型的網路攔截問題。例如，最小化起點與終點之間的路徑距離（最短路徑網路攔截）。按照問題建模方式則可區分為兩種：（一）確定性與（二）隨機性（Ramirez-Marquez, 2010）。也可依議題的具體要求加以定義網路中起點與終點是否固定及其數量，發展不同的模型。

其中，最短路徑網路攔截問題由Israeli and Wood（2002）首次提出，該問題中追隨者的目標是尋找一條從來源節點s到匯集節點t之間最短長度（或最短時間、最低成本等）的路徑，而領導者則是使用有限的資源攻擊網路，破壞某些邊或增加其長度，進而增加追隨者的最短路徑長度，與本研究之議題最為貼近。

2.2 雙層規劃

雙層規劃問題（bilevel programming problem）可以追溯到1934年，Stackelberg在關於市場經濟的專著中首次提出（Von Stackelberg, 1934）。其基本模型中，是假設在不平衡的經濟背景條件下，兩個決策者如何控制決策變數，以尋求最佳化各自的收益（目標函數），但其決策內容皆會影響另一方。其中所謂的領導者（leader）是根據追隨者（follower）最佳化後的決策結果，使其目標函數最小化，且領導者的決策會影響追隨者問題的可行解和目標函數，而追隨者的決策對領導者的收益同時具有相當的影響力（Bard, 2013）。

雙層規劃問題涉及兩個層次的最佳化問題，其中一個嵌套在另一個最佳化問題之中。外部最佳化問題通常稱為領導者（上層）的最佳化問題，而內部最佳化問題則被稱為跟隨者（或下層）的最佳化問題。兩個層次都有自己的目標和限制條件，並且有其各自決策變數：領導者（上層）決策變數和跟隨者（下層）決策變數。下層最佳化時，是針對其下層決策變數進行求解，而上層決策變數則充當參數。當上層求解時，下層的最佳化問題是上層最佳化問題的限制式之一，換句話說，只有那些使下層達到最佳解且滿足上層限制式的解，才可視為可行解（Sinha et al., 2017）。通常雙層規劃問題可以數學式表示如下，其中$x \in \mathbb{R}^{n_1}$與$y \in \mathbb{R}^{n_2}$：

$$\min_{x \in X, y} F(x, y) \tag{1a}$$

$$s.t.\ G(x, y) \leq 0 \tag{1b}$$

$$\min_{y} f(x, y) \tag{1c}$$

$$s.t.\ g(x, y) \leq 0 \tag{1d}$$

雙層規劃問題的決策變數分屬上、下層，即上層變數$x \in \mathbb{R}^{n_1}$和下層變數$y \in \mathbb{R}^{n_2}$；相同的，函數$F: \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$和$f: \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$分別是上層和下層的目標函數；函數$G: \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_1}$和$g: \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_2}$則分別是上層和下層限制條件。上層限制條件涉及兩個層級的決策變數（亦即x與y），並且扮演特定的角色，必須透過間接的方式（例如上層的決策變數）影響下層的決策內容，因為上層無法直接採取約束的方式，限制下層決策者的決策內涵（Colson et al., 2005）。

2.3 簡群演算法

在簡群演算法中，每組解$X_i^t = \left(x_{i,1}^t, x_{i,2}^t, \dots, x_{i,j}^t, \dots\right)$都被編碼為一組可計算適應函數值的向量，在更新迭代的過程中，每組解就其過去歷程中，具有最佳適應函數值的解稱為$pBest$，以$P_i = \left(p_{i,1}, p_{i,2}, \dots, p_{i,j}, \dots\right)$表示；在所有解群體中，具有最佳適應函數值的解稱為$gBest$，以$G = (g_1, g_2, \dots, g_j, \dots)$表示。與其他進化式演算法最不同之處在於其特殊的更新機制（update mechanism），其基本概念是，每組解向量中的變數$x_{i,j}^t$是依據一個介於$(0, 1)$之間的隨機數$\rho_{i,j}^t$位於參數C_g、C_p與C_w所定義的機率區間決定，從當前$gBest$（作為全局搜索）、$pBest$（作為區域搜索）、解本身及一組隨機可行解所調和構成，以保持解群體的多樣性（Lai, 2019; Yeh, 2009, 2012a, 2012b），其更新機制數學模式如公式（2）所示，其中$x_{i,j}^t$為第t次迭代，第i組解的第j個變數，x是介於上、下界限制的隨機變量。

$$x_{ij}^t = \begin{cases} g_j & \text{if} \quad \rho_{ij}^t \in [0, C_g] \\ p_{ij} & \text{if} \quad \rho_{ij}^t \in [C_g, C_p] \\ x_{ij}^{t-1} & \text{if} \quad \rho_{ij}^t \in [C_p, C_w] \\ x & \text{if} \quad \rho_{ij}^t \in [C_w, 1] \end{cases} \tag{2}$$

另針對求解連續變數，Yeh在2015年提出改進的簡群演算法（improved simplified swarm optimization, iSSO），修改後的更新機制如公式（3）所示（Yeh, 2015），其中u_j如公式（4）所示，$Nvar$是變數數量，x_j^{min}及x_j^{max}則是第j個變數的上、下界。

$$x_{i,j}^t = \begin{cases} g_j + \rho_{[-0.5,0.5]} \cdot u_j & \text{if} \quad \rho_{i,j}^t \in [0, C_g] \\ x_{i,j}^{t-1} + \rho_{[-0.5,0.5]} \cdot u_j & \text{if} \quad \rho_{i,j}^t \in [C_g, C_w] \\ x_{i,j}^{t-1} + \rho_{[-0.5,0.5]} \cdot \left(x_{i,j}^{t-1} - g_j\right) & \text{if} \quad \rho_{i,j}^t \in [C_w, 1] \end{cases} \tag{3}$$

$$u_j = \frac{x_j^{min} - x_j^{max}}{2 \cdot Nvar} \quad (4)$$

簡群演算法的更新機制是以c_g、c_p、c_w與c_r分別表示解更新到全域最佳解、區域最佳解、解本身與隨機變數等四種結果的機率參數，並定義$C_g = c_g$，表示解更新到全域最佳解之機率區間的設定參數；$C_p = c_g + c_p$，表示解更新到區域最佳解之機率區間的設定參數；$C_w = c_g + c_p + c_w = 1 - c_r$，表示解更新到解本身之機率區間的設定參數。以$Nsol$及$Nitr$分別表示群體大小及總迭代次數，完整演算步驟如表1所示。

表 1　簡群演算法的演算步驟

SSO Procedure	
Initialization Phase:	
Step 0.	Randomly generate X_i^0, calculate $F(X_i^0)$, define $pBest\ P_i$ and $gBest\ G$, and let $t = 1$ for $i = 1, 2, \dots, Nsol$.
Evolution Phase:	
Step 1.	Let $i = 1$.
Step 2.	Update X_i^{t-1} to X_i^t based on Eq.（2） and calculate $F(X_i^t)$.
Step 3.	If $F(X_i^t)$ is better than $F(P_i)$, let $P_i = X_i^t$; otherwise, go to **Step 5**.
Step 4.	If $F(P_i)$ is better than $F(G)$, let $G = P_i$.
Step 5.	If $i < Nsol$, let $i = i + 1$ and go to **Step 2**.
Step 6.	If $t < Nitr$, let $t = t + 1$ and go to **Step 1**. Otherwise, halt and $gBest$ is the final solution.

2.4 Dijkstra 演算法

Dijkstra演算法是求解最短路徑問題的精確演算法之一，由荷蘭計算機科學家Edsger Dijkstra於1956年構思、1959年發表，旨在找到任何起始節點和終止節點之間的最短路徑，可解決加權圖中不包含負權重邊的最短路徑問題（Dijkstra, 1959），其時間複雜度為$O(V^2)$，演算效率優於其他精確演算法，如Floyd-Warshall演算法（$O(V^3)$）與Bellman-Ford演算法（$O(V \cdot E)$）等（Di Caprio et al., 2022）。

當圖中起點$s \in V$與終點$t \in V$定義後，Dijkstra演算法不但可以求解兩點之間權重最低的路徑（即最短路徑）及其長度，亦可進一步查找從起點s到其他節點$v \in V$的最短路徑長度$L(v)$。在有向加權圖$G = (V, E, \omega)$中，有一個關於權重（即長度或成本）的函數$\omega: E \to N$，表示圖中每個邊$(u, v) \in E$具有一個非負整數的權重值$\omega(u, v) \in N$。當$\omega(u, v) = 0, u \in V$及$\omega(u, v) = \infty, (u, v) \notin E$時，可將該函數擴展為$\omega: V \times V \to N \cup \{\infty\}$。Dijkstra演算法的計算程序如表2所示（Deepa et al., 2018）。

表 2　Dijkstra 演算法的計算程序

Dijkstra Algorithm Procedure
Input: Edge-weighted graph, $G = (V, E, \omega)$ with (extended) weight function $\omega : V \times V \to N$, and a source vertex $s \in V$.
Output: Function $L : V \to N \cup \{\infty\}$, such that for all $v \in V$, $L(v)$ is the length of the shortest path from s to v in G.

Algorithm:
Initialize $S \leftarrow \{s\}$; $L(s) \leftarrow 0$;
Initialize $L(v) \leftarrow \omega(s, v)$, for all $v \in V - \{s\}$;
while $(S \neq V)$ do
$u \leftarrow \arg \min_{z \in V-S} \{L(z)\}$
$S \leftarrow S \cup \{u\}$
for all $v \in V - S$ such that $(u, v) \in E$ do
$L(v) \leftarrow \min\{L(v), L(u) + \omega(u, v)\}$
end for
end while
Output function $L(.)$.

資料來源：Deepa et al.（2018）

三、模型建構

3.1 問題描述

本研究採用最短路徑網路攔截問題作為基礎，考慮伏擊效果對路徑選擇之影響，以節點為攔截標的，探討單一固定起點與終點的網路，透過雙層規劃建構「防禦者-攻擊者」模型（如圖 1 所示），發展適合國土防衛作戰之網路攔截模型。模型中，上層結構為我軍伏擊點選擇的最佳化問題，下層結構為敵軍進攻的最短路徑問題。

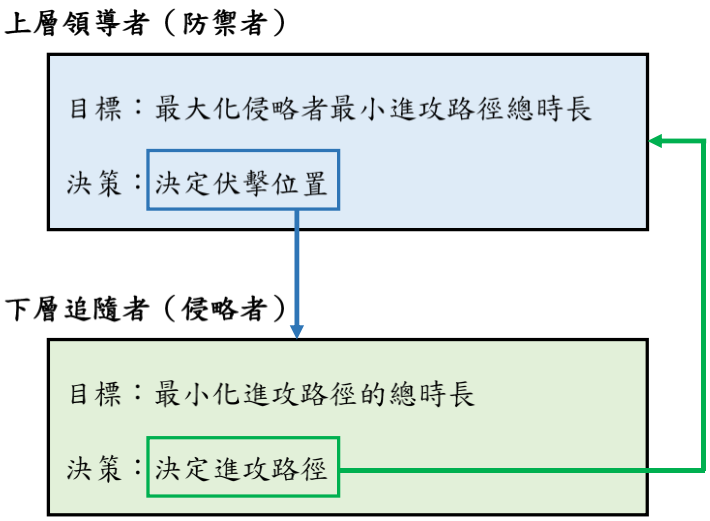


圖 1　本研究建構之「防禦者-攻擊者」模型

3.2 符號說明

本研究提出之數學模型所需符號定義如下：

nCt　可選擇伏擊的地點總數，在圖上以節點表示可伏擊點。

I'　所有節點的集合。$I' = \{s, 1,2, \dots, nCt, t\}$

I　除了起點 s 跟終點 t 以外所有節點的集合。$I = I'/\{s, t\}$

A　鄰接矩陣，是$|I'| \times |I'|$大小的(0, 1)矩陣，0 表示節點之間沒有邊相連，1 表示節點之間有邊相連。

nEg　鄰接矩陣中邊的總數。

$D(i)$　離開節點 i 所有邊的集合。$\forall i \in I'$

$E(i)$　進入節點 i 所有邊的集合。$\forall i \in I'$

m　可供我軍部署的部隊總數。

coe　依網路的節點數給定部隊總數m之規則參數。

d_{ij}　節點 i 與 j 之間的距離。$\forall i, j \in I'$

α　敵軍的行軍速度，單位為公里/小時。

λ　遲滯時長，表示敵軍經過伏擊點所延遲的時間，單位為小時。

w_{ij}　節點 i 與 j 之間邊的權重，代表時間，單位為小時。$\forall i, j \in I'$

δ　敵軍進攻路徑的總時長。

$$x_i = \begin{cases} 1，部隊於節點i伏擊。 \\ 0，則否。 \end{cases} \quad \forall i \in I$$

$$y_i = \begin{cases} 1，敵軍進攻路徑包含節點i。 \\ 0，則否。 \end{cases} \quad \forall i \in I$$

$$a_{ij} = \begin{cases} 1，敵軍進攻路徑包含節點i到j的邊。 \\ 0，則否。 \end{cases} \quad \forall i, j \in I'$$

3.3 模型說明

行軍是軍隊機動的基本方法，意旨沿著想定的路線進行有組織的移動，其形式（徒步、乘車或兩者結合）與強度（常行軍、急行軍或強行軍）會依據任務、敵情、地形、部隊規模及能力而定。敵軍在進攻途中，遭遇我軍游擊部隊伏擊時，其作戰態勢由行軍轉為遭遇戰，將嚴重遲緩其行軍速度，為簡化問題，以α表示敵軍的行軍速度，以λ表示敵軍經過伏擊點所致之延遲時間。

本研究假設敵軍登陸後的作戰目的是迅速抵達終點，因此，將選擇最小總時長之路徑做為進攻路線，故本研究將邊(i, j)的權重w_{ij}定義為途經邊(i, j)所需的總時長，計算如公式（5）所示。其中d_{ij}/α是依據邊(i, j)的距離d_{ij}及敵軍的行軍速度α所計算的行軍時長；λ是遲滯時長，表示部署伏擊點於節點 j 所造成的遲滯效果。敵軍進攻路徑所需的總時長δ可以公式（6）計算。

$$w_{ij} = d_{ij}/\alpha + \lambda \cdot x_j, \forall i,j \in I' \quad (5)$$

$$\delta = \sum_{i=1}^{|I'|}\sum_{j=1}^{|I'|} a_{ij} w_{ij} \quad (6)$$

本研究提出國土防衛第四道防線兵力部署最佳化問題，並以網路攔截問題建構雙層規劃模型，數學模型說明如下：

$$\max\ f \quad (7)$$

$$s.t.$$

$$\sum_{i=1}^{|I|} x_i \le m \quad (8)$$

$$x_i \in \{0,1\}, \forall i \in I \quad (9)$$

$$f = \min \delta \quad (10)$$

$$s.t.$$

$$\sum_{(i,j)\in D(i)} a_{ij} - \sum_{(k,i)\in E(i)} a_{ki} = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases}, \forall i,j,k \in I' \quad (11)$$

$$\sum_{j=1}^{|I|} a_{ij} = y_i, \forall i \in I \quad (12)$$

$$y_i \in \{0,1\}, \forall i \in I \quad (13)$$

$$a_{ij} \in \{0,1\}, \forall i,j \in I' \quad (14)$$

上層目標函數（7）為藉由伏擊點的選擇最大化敵軍最小進攻路徑總時長。限制式（8）為資源限制，限制所有部隊進駐的節點數不超過可供部署的部隊總數m。限制式（9）定義上層決策變數為二元變數。公式（10）是下層目標函數，可解釋為敵軍依上層的決策（伏擊點的位置），最小化其進攻路徑的總時長。限制式（11）限制路徑需從起點s開始至終點t結束，且途中不重複任一節點。限制式（12）限制敵軍進攻路徑不可經過未包含在路徑上的節點，且該路徑所包含的節點皆僅有一個進入的邊。限制式（13）定義下層決策變數為二元變數。限制式（14）則定義輔助決策變數為二元變數。

四、研究方法

本研究依據問題提出三種求解方法，第一種是兩階段的精確解法，命名為Enumeration-Dijkstra Algorithm（EDA），目的在驗證模型的數學式是否正確、評估模型的時間複雜度與驗證第二種方法在求解小問題的精確程度。EDA 的上層是以隱舉法列

舉所有伏擊點的組合，下層則以 Dijkstra 演算法依據各伏擊點組合，求解最短路徑問題。第二種是基於簡群演算法與 Dijkstra 演算法所發展的求解方法，在彌補精確解法運算成本過高的缺點，稱之 SSO-Dijkstra Bilevel Evolutionary Algorithm（SDBEA）。第三種是為了提高 SDBEA 的運算能力，依據問題性質設計特定的伏擊最短路徑之區域搜索機制，該機制命名為 Shortest-path-based Ambushing Scheme（SAS），以加速簡群演算法收斂與提高求解品質，稱之 $SDBEA_{SAS}$。三種種方法的上、下層之間的差異，請見表 3。

表 3　求解方法

名稱	EDA	SDBEA	$SDBEA_{SAS}$
上層結構	Enumeration	SSO	SSO ＋ SAS
下層結構	Dijkstra Algorithm	Dijkstra Algorithm	Dijkstra Algorithm

4.1 Enumeration-Dijkstra Algorithm

EDA 為兩階段的精確解法，第一階段是以隱舉法列舉出上層所有可行解集合（伏擊點選擇策略）；第二階段，將上層解集合逐一透過 Dijkstra 演算法求解下層最短路徑問題，再依上層的目標函數從所有解集合中找出最佳解，求解流程如圖 2 所示，其中上層可行解的總數為$nsol$，X_i及Y_i分別表示上層第i組解及該組解所對應的下層精確解，$\boldsymbol{X} = \{X_1, X_2, \dots, X_{nsol}\}$則表示上層伏擊點選擇最佳化問題的所有可行解集合。

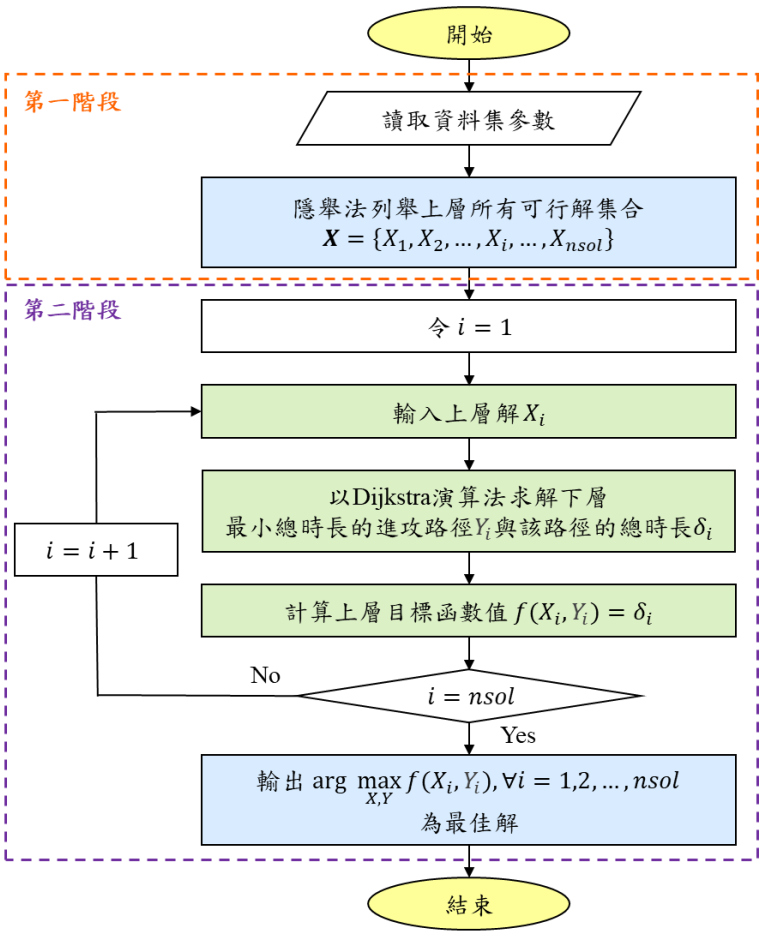


圖 2　EDA 之求解流程圖

4.2 SSO-Dijkstra Bilevel Evolutionary Algorithm

SDBEA 的上層結構是以簡群演算法求解我軍伏擊點選擇的最佳化問題，上層解代表防禦者的伏擊決策。在演算法初始時，會隨機產生一群數量為$Nsol$的解$\boldsymbol{X^0}$，群體中每組解X_i的編碼是以向量方式編成，X_i的向量長度為nCt，是可選擇伏擊的地點總數，$X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,nCt})$。向量中任一元素$x_{i,j}$，$j = 1, 2, \dots, nCt$，是介於$[0, 1]$的隨機實數，其數值大小代表該節點被選為伏擊點的優先順序(如圖 3 上半部，數值越大、順序越優先)，依可供我軍部署的部隊總數m，從順序中選取前m個節點設置伏擊點，續將每組解X_i解碼為$X_i' = (x_{i,1}', x_{i,2}', \dots, x_{i,nCt}')$(如圖 3 下半部)，其中$X_i'$內各元素$x_{i,j}'$為二元整數，$x_{i,j}' = 1$表示部隊於節點$j$設置伏擊點，$x_{i,j}' = 0$表示節點$j$無設置伏擊點。假設可供我軍部署的部隊總數$m = 3$時，選取前兩個優先順序的節點設置伏擊點，分別為節點 4、5 及 8，故$x_{i,4}' = 1$、$x_{i,5}' = 1$及$x_{i,8}' = 1$，其餘為 0。

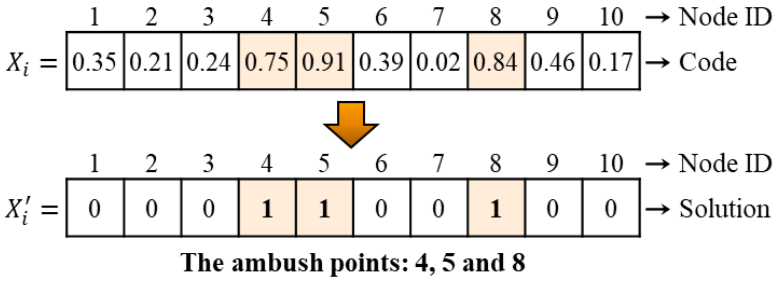


圖 3　上層解編碼方式

完成初始解編碼後，進入巢式序列的下層結構，透過下層結構的 Dijkstra 演算法求解最短路徑問題。當上層簡群演算法的任一組解X_i^t在計算適應函數值時，會先啟動下層 Dijkstra 演算法求解相對應之最小總時長的進攻路徑Y_i^*與該路徑的總時長δ_i，故適應函數值可以$F(X_i^t, Y_i^*)$表示。Dijkstra 演算法的計算程序如第二章的表 2 所示，輸入一個網路圖$G = (V, E, \omega)$，圖中每個邊$(u, v) \in E$具有一個非負數的權重值$w_{uv} \in \omega$，w_{uv}是依上層X_i^t伏擊點選擇的位置，由公式（4）計算途經邊(u, v)所需的總時長，再透過表 2 程序求解出該網路圖中起點s到終點t的最小進攻路徑總時長δ_i，定義如公式（5），並令$F(X_i^t, Y_i^*) = \delta_i$，回傳至上層結構，以評估解$X_i^t$的優劣。完整 SDBEA 演算流程如圖 4 所示。

4.3 區域搜索機制

本研究所提出的巢式序列架構之求解方法，其上層結構是以簡群演算法求解組合最佳化問題，為了提高演算法的性能，可在更新階段加入區域搜索機制，其目的是在針對具有潛力或特定的決策空間，依據問題性質進行局部的解搜索，試圖找到更好的解（Pirlot, 1996）。以巢式序列架構求解雙層規劃時，上層每組解X會對應到一組下層的最佳解Y^*，即在該組伏擊點部署策略下，敵軍所選擇最小總時長的進攻路徑，以$Path$表示。此時，若能在敵軍的當前進攻路徑上設置伏擊點，必能對其構成更大的遲滯效果，因此，本研究提出伏擊最短路徑的區域搜索機制，命名為 Shortest-path-based Ambushing Scheme（SAS）。SAS 的核心概念是，在上層簡群演算法更新階段時，解X會從下層所回饋最小總時長的進攻路徑$Path$上，選擇一個節點v_{new}成為新的伏擊點，再從原來的伏擊點Apt

中選擇一個伏擊點v_{old}，無論v_{new}或v_{old}皆透過隨機方式選擇，最後將被選中的兩個節點之編碼作互換，將v_{new}取代v_{old}成為新的伏擊點，即完成該組解的更新。完整 SAS 步驟如表 4 所示。

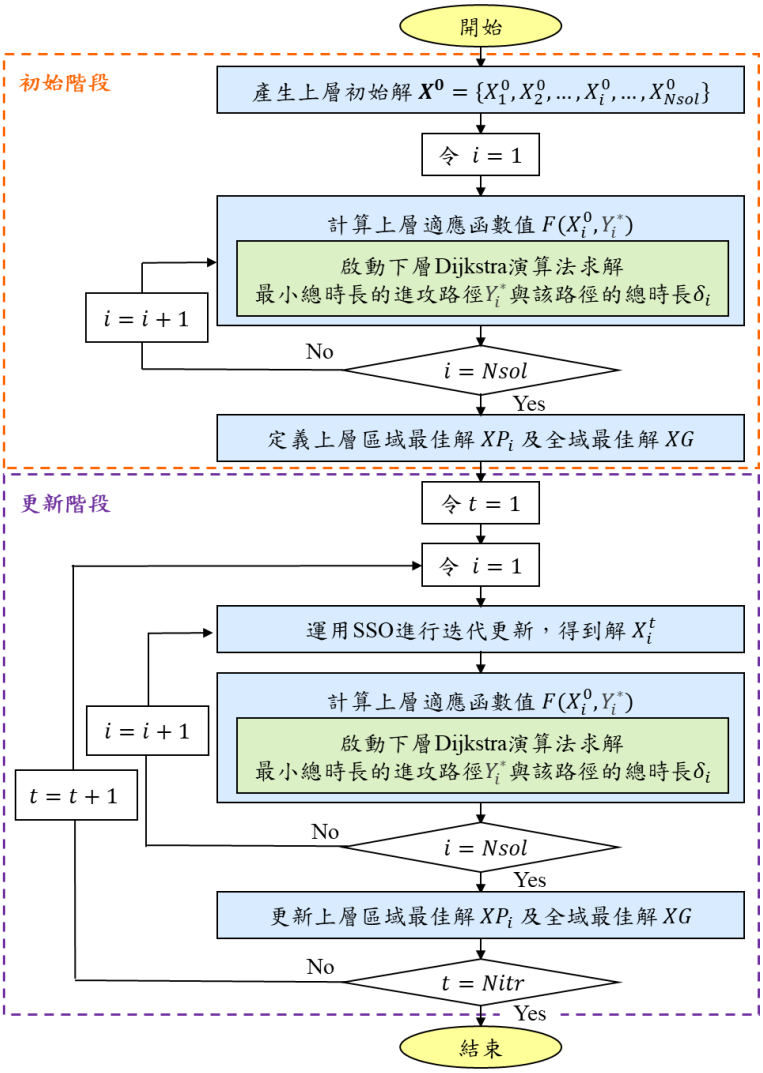


圖 4　SDBEA 之演算流程圖

表 4　伏擊最短路徑的區域搜索機制之步驟

Shortest-path-based Ambushing Scheme Procedure	
Step 1.	Let Apt_i be the ambush point set selected by $X_i^t = [x_{i,j}^t]_{1\times Nvar}$, and $path_i$ be its corresponding shortest path found by lower-level method.
Step 2.	Randomly select v_{new} and v_{old} from $Path_i$ and Apt_irespectively.
Step 3.	Copy X_i^t to X_i^{t+1}, let $x_{i,v_{new}}^{t+1} \leftarrow x_{i,v_{old}}^t$ and $x_{i,v_{old}}^{t+1} \leftarrow x_{i,v_{new}}^t$.
Step 4.	Output $X_i^{t+1} = [x_{i,j}^{t+1}]_{1\times Nvar}$.

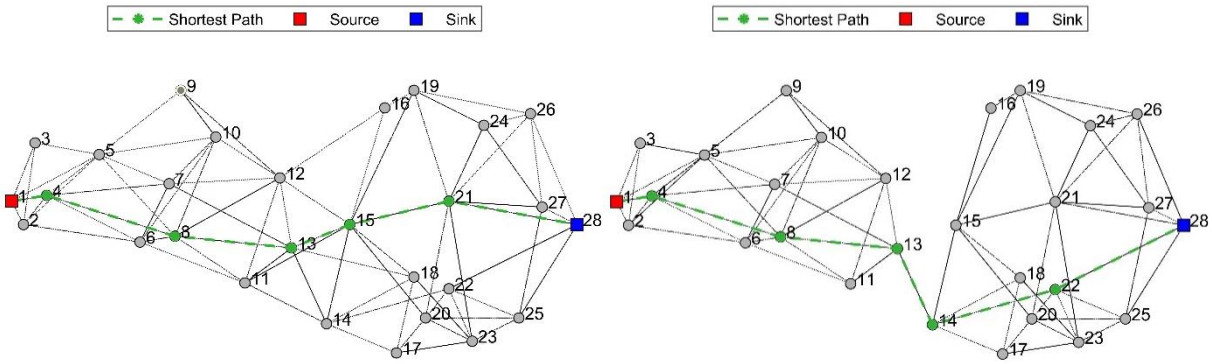
五、實證與分析

本研究設計不同規模的 20 個隨機問題及 24 個實例問題，進行模型驗證、實證分析、演算法參數設定與演算法之間的比較。首先，以 EDA 及 SDBEA 求解隨機問題驗證本研究所提出的模型，並將 SDBEA 的求解結果與 EDA 求得的精確解作比較，驗證 SDBEA 在小規模問題上找到精確解的能力。再透過實驗設計定義 SDBEA 的參數，最後求解大量實例問題進行演算法的比較，並以統計檢定驗證本研究所提出的求解方法與其他演算法之間在效能上的差異程度，最後基於結果的分析綜整我軍的部署策略。本章所有演算法的運算均使用 Matlab(R2021a)語言撰寫，電腦硬體規格為 Intel Core i5- 1035G1 CPU @ 1.19GHz 處理器，搭配 8G 記憶體及 Windows 10 作業系統。

5.1 問題說明

5.1.1 隨機問題

本研究提出之最短路徑網路攔截問題的複雜程度取決於網路規模，例如（網路的節點數，邊的數量），可以表示為$(|I'|, nEg)$，與部隊總數m的大小等數量因子。因此以$(|I'|, nEg) = (28, 72)$與$m = 0$~9，設計 10 個隨機問題。此外，為了分析特殊情形「隘道」，對決策之影響，刻意刪除上述問題中的部分邊成為具有隘道之路網，形成另外 10 個對照問題（無隘道路網 vs.有隘道路網，如圖 5 所示）。其他參數定義如下：敵軍的行軍速度$\alpha = 10$、敵軍每經過伏擊點的遲滯時長$\lambda = 5$。



（1）無隘道路網　　（2）有隘道路網

圖 5　隨機問題示意圖

5.1.2 實例問題

實例題組是參考 WGS84 座標系統（World Geodetic System 1984），取臺灣本島重要的城鎮與交通要道，以敵軍登陸點區分 6 種規模的題組（如圖 6 所示），登陸點分別是金山、林口、壯圍、鹿港、布袋及加祿堂，揮軍北上直指總統府（終點），隨登陸點越往南移，題組包含的節點與邊越多。此外，每一個題組給定 4 個不同的部隊總數m，形成 24 個實例問題，資料集細節如表 5 所示。

表 5　實例問題之資料集

題組編號	起點 s	終點 t	網路的節點數 $\|I'\|$	邊的數量 nEg	部隊總數 m
1	金山	總統府	45	68	5, 6, 8, 9
2	林口		90	159	9, 13, 16, 18
3	壯圍		180	323	18, 25, 32, 36
4	鹿港		270	477	27, 38, 49, 54
5	布袋		360	643	36, 50, 65, 72
6	加祿堂		500	879	50, 70, 90, 100

*部隊總數m給定規則說明：

$$coe = \left[\frac{1}{10}, \frac{1.4}{10}, \frac{1.8}{10}, \frac{2}{10}\right], m = |I'| \times coe$$

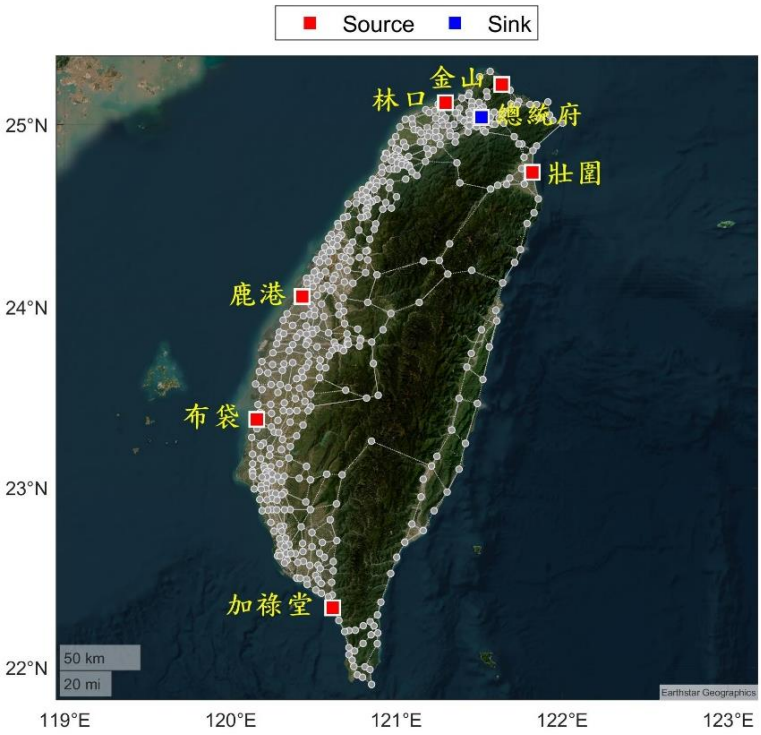


圖 6　實例問題示意圖

5.2 演算法參數設定

為了有效發揮本研究所提出之演算法 SDBEA$_{SAS}$ 的求解效果，針對其上層簡群演算法更新機制及區域搜索機制（SAS）的參數進行全因子實驗。簡群演算法的更新機制參數包括C_g、C_p與C_w，分別定義 3 個水準，$C_g = (0.4, 0.5, 0.6)$、$C_p = (0.75, 0.8, 0.85)$與$C_w = (0.9, 0.95, 0.99)$。另本研究提出的 SAS，則以參數C_s表示解群體進入該機制的比例，當一組解所擲出的隨機數$nr < C_s$時，才以 SAS 更新該組解，否則不進入 SAS 程序。經過本研究前測結果給定 5 個水準，$C_s = (0, 0.1, 0.4, 0.7, 1)$，0 代表演算法沒有啟用 SAS，0.1 至 1 代表演算法解群體啟用 SAS 的比例。依據上述四種參數定義的水準，共有 135 個實驗組合，每一種實驗組合以 SDBEA$_{SAS}$ 演算法各別執行 10 次，實驗次數共計 1350 次，求解實例資料集各規模題組中m分別為 10、20、40、60、80 及 100 的問題，其他演

算法相關參數定義如下： $Nsol = 30$、$Nitr = 300$。

以 SDBEA$_{SAS}$ 求解 6 個問題，為檢視各因子在求解效能與效率上是否具有差異，使用 ANOVA 分析，顯著水準設定為 0.05，分析結果如表 6 所示（粗體字表示具顯著性）。每個問題執行 1350 次實驗的平均適應函數值及平均運算時間分別以F_{avg}及T_{avg}表示，為利於觀察實驗結果，將F_{avg}及T_{avg}標準化至[0, 1]尺度，並繪製成主效果圖（圖 7），結果分析如下：

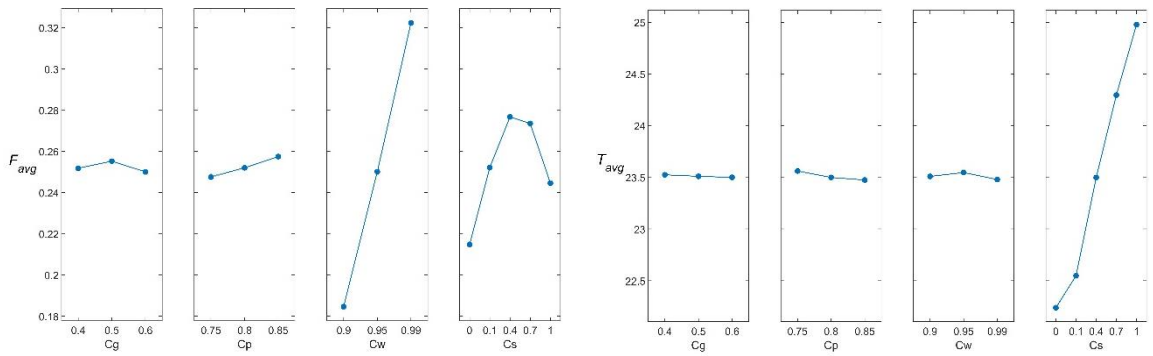
（一）從 ANOVA 分析結果顯示，四個因子（C_g、C_p、C_w與C_s）對適應函數值F皆有顯著影響，單從主效果圖（圖 7）來看，初步認定四個因子的最佳組合為$(C_g, C_p, C_w, C_s) = (0.5, 0.85, 0.99, 0.4)$時，可達到較好的求解效果。其中，$C_s = 0.4$ 與 0.7 之間，效果沒有顯著差異。

（二）簡群演算法更新機制的參數C_g、C_p與C_w對運算時間T皆不顯著，僅區域搜索機制的C_s有顯著差異，從主效果圖（圖 7）來看，運算時間T會隨著C_s的數值越大而顯著增加，顯然選擇$C_s = 0.4$有利於降低演算時間。

（三）ANOVA 分析結果亦顯示，四個因子之間對適應函數值F具有顯著的交互作用，因交互作用圖不易辨識，另繪製所有因子對適應函數值F的組合績效圖，如圖 8 所示，其最佳組合為$(C_g, C_p, C_w, C_s) = (0.5, 0.85, 0.99, 0.4)$，此結果與上述分析一致，故本研究的 SDBEA$_{SAS}$ 採用此參數組合進行求解。

表 6　ANOVA 分析結果

Group	F					T				
Source	DF	SS	MS	F value	P value	DF	SS	MS	F value	P value
C_g	2	0.0062	0.0031	4.17	**0.0157**	2	0.13	0.067	0.11	0.9003
C_p	2	0.0219	0.0110	14.69	**0.0000**	2	1.81	0.907	1.42	0.2424
C_w	2	4.2719	2.1360	2863.13	**0.0000**	2	1.01	0.504	0.79	0.4547
C_s	4	0.6782	0.1695	227.25	**0.0000**	4	1434.50	358.625	561.20	**0.0000**
C_g*C_p	4	0.0032	0.0008	1.08	0.3650	4	4.88	1.220	1.91	0.1065
C_g*C_w	4	0.0245	0.0061	8.21	**0.0000**	4	4.80	1.200	1.88	0.1120
C_g*C_s	8	0.1078	0.0135	18.06	**0.0000**	8	2.87	0.358	0.56	0.8106
C_p*C_w	4	0.0044	0.0011	1.48	0.2053	4	2.35	0.588	0.92	0.4513
C_p*C_s	8	0.0088	0.0011	1.48	0.1596	8	3.75	0.469	0.73	0.6620
C_w*C_s	8	1.8204	0.2276	305.02	**0.0000**	8	5.44	0.680	1.06	0.3857
C_g*C_p*C_w	8	0.0049	0.0006	0.82	0.5843	8	8.27	1.033	1.62	0.1154
C_g*C_p*C_s	16	0.0060	0.0004	0.50	0.9468	16	13.10	0.819	1.28	0.2006
C_g*C_w*C_s	16	0.0469	0.0029	3.93	**0.0000**	16	11.96	0.747	1.17	0.2857
C_p*C_w*C_s	16	0.0082	0.0005	0.69	0.8064	16	5.47	0.342	0.54	0.9296
C_g*C_p*C_w*C_s	32	0.0314	0.0010	1.31	0.1138	32	18.52	0.579	0.91	0.6199
Error	1215	0.9064	0.0008			1215	776.43	0.639		
Total	1349	7.9512				1349	2295.29			



（1）F_{avg}主效果圖　　（2）T_{avg}主效果圖

圖 7　主效果圖

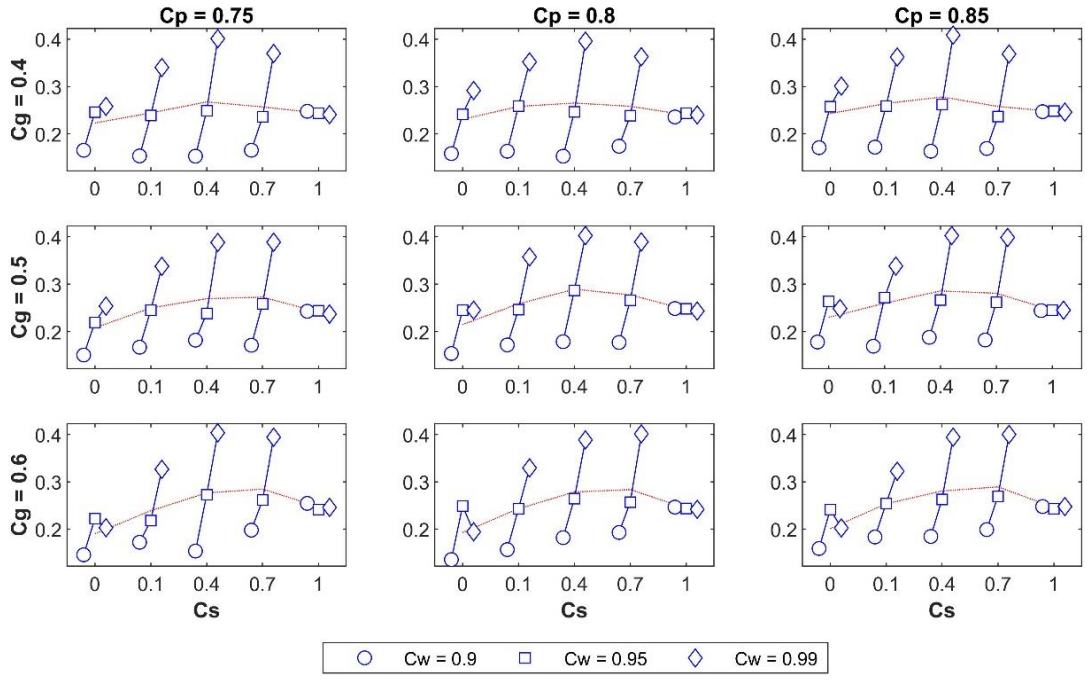


圖 8　因子的組合績效圖

5.3 隨機問題求解與比較

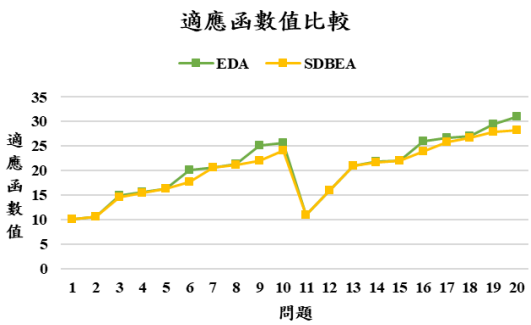
為了驗證本研究所提出的模型及演算法的效能，以 EDA 及 SDBEA$_{SAS}$ 分別求解 20 個隨機問題，其中 SDBEA$_{SAS}$ 具有隨機性因此每個問題運算 10 次，並定義$Nsol = 30$、$Nitr = 100$。由於 EDA 求得之解為精確解，因此將目標函數值δ及運算時間T做為驗證之基準，供驗證 SDBEA$_{SAS}$ 之用。SDBEA$_{SAS}$ 的績效指標包括運算 10 次後的最佳適應函數值（f_{best}）、平均適應函數值（f_{avg}）、標準差（f_{std}）、平均運算時間（T_{avg}）、成功找到精確解的機率Rate = (成功次數/10) × 100%及與精確解間的差距值$Gap = (f_{best} - \delta) \times 100/\delta$。求解結果如表 7 及圖 9 所示。

從表 7 及圖 9 的運算時間 T 可發現，EDA 的求解效率會隨著可供我軍部署的部隊總數 m 越大、複雜程度增加時，使上層隱舉出的決策組合成指數增長，大幅增加整個雙層規劃問題求解所需的時間（0.1053~497.7393 秒），如果問題持續擴大，EDA 的求解效率將不符合運算成本，也無法應用在實務上。反觀 SDBEA$_{SAS}$ 的求解結果發現，其平均

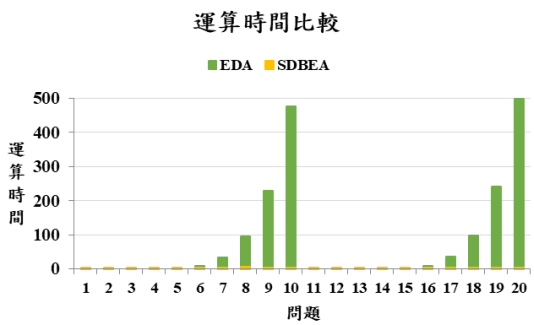
運算時間為 4.5139 秒，比 EDA 的 86.4404 秒減少 95%的時間成本。更進一步分析發現，SDBEA$_{SAS}$ 的平均適應函數值與精確解相比僅差 3.75%，且其中 19 個問題 Gap 均為 0，表示該方法在 10 次運算中都能找到最佳解，足以證明本研究提出的 SDBEA$_{SAS}$ 在尋解能力及效率上有一定的優異程度。比較無隘道（1～10 題）與有隘道（11～20）的結果，可以發現有隘道設計的網路得到的解優於無隘道問題，有此可知隘道的形成有利於防禦者的伏擊，可用較少的部隊數量即可達到直接攔截的效果。

表 7　隨機問題求解結果

問題編號	EDA		SDBEA$_{SAS}$				
	δ	T	f_{avg}	f_{std}	T_{avg}	Gap	Rate (%)
1	9.9865	0.1205	9.9865	0.0000	4.3144	0	100
2	10.5374	0.1053	10.5374	0.0000	4.2190	0	100
3	14.9865	0.1823	14.6232	1.1489	4.4859	0	90
4	15.5681	0.4735	15.4635	0.2205	4.6692	0	80
5	16.3535	1.9311	16.2719	0.2581	4.5798	0	90
6	20.0451	8.9713	17.7390	1.2348	4.6464	0	10
7	20.5681	32.6112	20.5620	0.0130	4.5555	0	80
8	21.3535	95.8146	21.0838	0.4582	5.2651	0	20
9	25.0451	228.7507	21.9983	0.7711	4.6418	-11	0
10	25.5681	474.9763	24.0483	1.7387	4.3398	0	10
11	11.0024	0.1739	11.0024	0.0000	4.2599	0	100
12	16.0024	0.1058	16.0024	0.0000	4.4343	0	100
13	21.0024	0.1579	21.0024	0.0000	4.4988	0	100
14	21.7552	0.5312	21.6369	0.2494	4.4912	0	80
15	22.0876	2.3702	22.0715	0.0509	4.4772	0	90
16	26.0024	9.9672	23.9242	1.4392	4.4679	0	20
17	26.7552	34.8553	25.8318	1.3156	4.5221	0	40
18	27.0876	96.8860	26.6849	0.4882	4.4952	0	50
19	29.3876	242.0848	27.8692	1.3349	4.4870	0	40
20	31.0024	497.7393	28.3167	1.7822	4.4277	0	20
平均	20.6049	86.4404	19.8328	0.6252	4.5139	-1	61



（1）適應函數值比較圖



（2）運算時間比較圖

圖 9　隨機問題求解結果比較圖

5.4 實例問題求解與比較

5.4.1 演算法比較

為驗證 SDBEA$_{SAS}$ 的運算效能，另外以基因演算法（GA）與粒群演算法（PSO）分別撰寫 GDBEA 與 PDBEA，再加上未加入區域搜索機制 SAS 的 SDBEA，透過實例問題的求解進行比較。四種演算法上、下層結構與參數設定如表 8 所示。

表 8　各演算法求解架構及參數表

演算法名稱	GDBEA	PDBEA	SDBEA	SDBEA$_{SAS}$
上層結構	GA	PSO	SSO	SSO + SAS
下層結構	Dijkstra Algorithm			
相關參數	$Nsol = 30$ $Nitr = 300$ $Pc = 0.7$ $Pm = 0.3$ $gamma = 0.4$ $mu = 0.05$	$Nsol = 30$ $Nitr = 300$ $w = 1$ $c1 = 5$ $c2 = 5$	$Nsol = 30$ $Nitr = 300$ $Cg = 0.5$ $Cp = 0.85$ $Cw = 0.99$	$Nsol = 30$ $Nitr = 300$ $Cg = 0.5$ $Cp = 0.85$ $Cw = 0.99$ $Cs = 0.4$

四種演算法分別求解 24 個實例問題，每個問題均獨立運算 30 次，紀錄的求解結果，包括最差適應函數值（F_{worst}）、平均適應函數值（F_{avg}）、最佳適應函數值（F_{best}）、適應函數值的標準差（F_{std}）及平均運算時間（T_{avg}），並將 24 個問題求解的績效指標繪製成折線圖（圖 10，圖中 SDBEA$_{SAS}$ 以 SDBEA*表示），結果分析詳述如下：

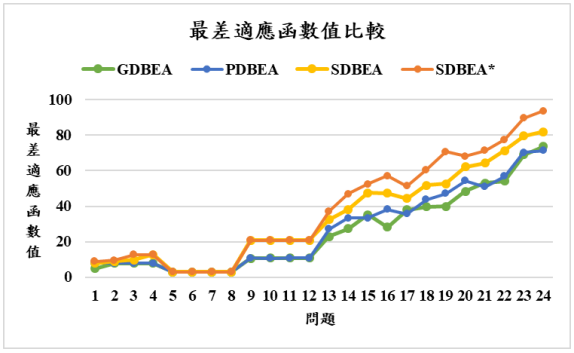
（一）最差適應函數值（F_{worst}）：即各演算法運算 30 次中最小的適應函數值。SDBEA$_{SAS}$ 表現最佳，SDBEA 次之，PDBEA 與 GDBEA 表現最差。

（二）平均適應函數值（F_{avg}）：即各演算法運算 30 次平均的適應函數值。SDBEA$_{SAS}$ 表現最佳，SDBEA 次之，PDBEA 僅第 21 及 24 題略差於 GDBEA，餘均排序第三，而 GDBEA 表現最差。

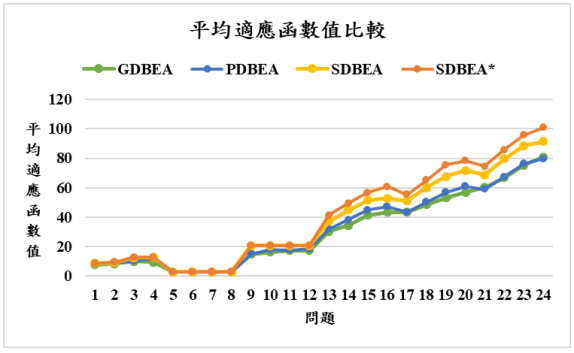
（三）最佳適應函數值（F_{best}）：即各演算法運算 30 次中最大的適應函數值。在第 1 至 12 題的結果中，四種演算法的結果幾乎一致，隨著問題複雜度越大 SDBEA$_{SAS}$ 及 SDBEA 逐漸與 PDBEA 及 GDBEA 拉開差距，SDBEA$_{SAS}$ 的表現又優於 SDBEA，而 PDBEA 與 GDBEA 平分秋色。

（四）適應函數值的標準差（F_{std}）：在求解所有問題中，SDBEA$_{SAS}$ 的標準差均最小，可見其求解的穩健性相較其他三種演算法高。

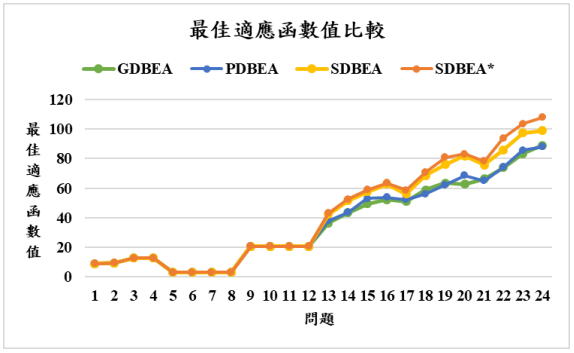
（五）平均運算時間（T_{avg}）：從前一小節可知，本研究設計之區域搜索機制 SAS，會給演算法帶來額外的運算成本，因此有加入區域搜索機制的 SDBEA$_{SAS}$，運算時間皆大於其他三種演算法，但其運算時間與最快的演算法相比，差距均不超過 2 秒鐘，從求解的效率上來看，其增加的運算成本是可接受的。



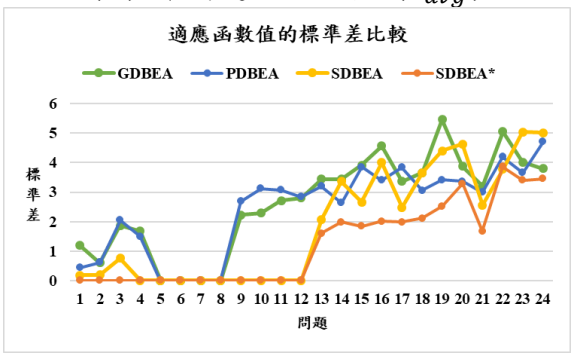
（1）最差適應函數值（F_{worst}）



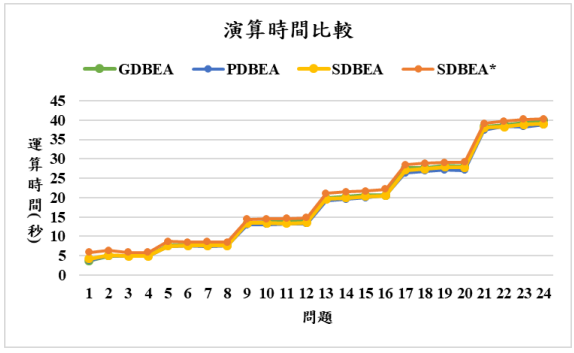
（2）平均適應函數值（F_{avg}）



（3）最佳適應函數值（F_{best}）



（4）適應函數值的標準差（F_{std}）



（5）平均運算時間（T_{avg}）

圖 10　各演算法求解實例問題之績效指標

綜合以上比較結果，本研究提出的 $SDBEA_{SAS}$ 求解的效能與穩健性方面，相較其他三種演算法(GDBEA、PDBEA 與 SDBEA)突出，同時 SDBEA 的求解效能也優於 PDBEA 與 SDBEA。推估其效能較為優異的原因，可能是 SDBEA 的參數是經過實驗設計定義，其他則無。PDBEA 的求解效能在大部分問題中都優於 GDBEA，推測是本研究採實數編碼，相較於基因演算法，粒群演算法更適於求解連續最佳化問題。$SDBEA_{SAS}$ 的求解品質優於 SDBEA 則證明了本研究提出的區域搜索機制能確實發揮作用，有效提高演算法的求解品質。

5.4.2 結果分析

前述比較結果說明，$SDBEA_{SAS}$ 優於其他三種演算法，因此，以 $SDBEA_{SAS}$ 的最佳解進行分析，並將求解結果以圖像呈現。因全數實例問題共有 24 個，受限於篇幅，考量邊與節點的數量，因此以挑選地圖範圍有利於說明的題組，故以「金山」為登陸點的第 1 至 4 題進行分析，如圖 11 所示。

（1）第 1 題

（2）第 2 題

（3）第 3 題

（4）第 4 題

圖 11　實例問題求解結果示意圖（第 1 至 4 題）

從圖 11（1）可知，當$m = 5$時，最短路徑上已設置兩個伏擊點，其他所有可行路徑上則設置至少一個伏擊點，敵軍為減少與我軍正面交鋒的次數，選擇相對較長的路徑（$\delta = 9.7929$）。當部隊數量增加至$m = 6$時（圖 11（2）），此時不論敵軍選擇哪條進攻路徑，都會遭受至少兩次的伏擊，故敵軍會選擇以最短路徑進攻。當部隊數量持續增加時（$m = 8$或$m = 9$），比較圖 11（3）（$\delta = 13.7614$）與圖 11（4）（$\delta = 14.3915$）的求解結果發現，隨著部隊數量m的增加，造成敵軍繞路的效果越趨明顯。

5.4.3 關鍵節點分析

另外在圖 11 中，羅列第 1 至 4 題我軍的部署位置，將各題重複設置伏擊點的節點，定義為關鍵節點。透過此實例的求解與分析可知，當敵軍從金山登陸直取台北地區時，我軍部署的關鍵節點為「竹子湖」、「陽明山」、「翡翠灣」及「內湖」等節點，這些關鍵節點在建軍備戰上有其特殊意涵。戰時，應優先集結可用兵力於上述節點上設置伏擊點，方可對敵軍的有生力量造成最大傷害或遲滯其奪取首都；平時，可針對上述關鍵節點，考量伏擊戰術作為與可用地形，建構有利於我軍伏擊的堅固工事，並納入國軍固安作戰計畫於每年漢光演習中實施實兵操演。

5.4.4 部署策略分析

同樣採用從「金山」登陸的第 1 個規模之題組，重新定義部隊總數$m = 0\sim11$，設計 12 個實例問題。相關參數定義如下：敵軍的行軍速度$\alpha = 10$、敵軍每經過伏擊點的遲滯時長$\lambda = 5$。透過 $\text{SDBEA}_{\text{SAS}}$ 的求解不同m值之下，我軍與敵軍的最佳決策，並觀察求解結果並分析，綜整第四道防線的部署模式如下：

（一）在沒有設置伏擊點的情況下（$m = 0$），敵軍會選擇起點與終點之間的最短路徑為其進攻路徑，可在最短時間內抵達最終目標（如圖 12（1））。

（二）當部隊數量不足以直接攔截敵軍時，我軍會選擇在最短路徑上設置伏擊點，迫使敵軍為躲避伏擊而選擇繞道而行（如圖 12（2）），遲滯進攻所需的時間。

（三）當部隊數量增加至足以直接攔截敵軍時，伏擊點的選擇會傾向在每條可行路徑上

皆設置一個伏擊點（如圖 12（2）至（4）），使敵軍無法透過繞路來躲避與我軍正面交鋒，進而給予最大的遲滯效果。

（四）假設敵軍當前所選擇的路徑上已設置N個伏擊點（如圖 12（7）），我方每新增一個可供我軍部署的部隊時，通常會部署在敵軍當前所選擇的路徑上（如圖 12（8）與（10）），或部署在僅有$N-1$個伏擊點的路徑上（如圖 12（11）），使敵軍不論選擇哪條路徑進攻，都會遭受至少為N的伏擊次數。

（五）所有可行路徑上皆設置相同數量之伏擊點時，敵軍最終會選擇以最短路徑奪取最終目標（如圖 12（4）與（7））。

從上述分析可知，當可用於伏擊的部隊數量不足時，應優先部署於最短路徑上，再依可行路徑長度逐一部署伏擊點，迫使敵軍為躲避與我軍正面交鋒而選擇更長的路徑；當我兵力充裕時，應將所有可行路徑上皆設置相同數量之伏擊點，不論敵軍選擇哪條進攻路徑，均可對其造成最大的遲滯效果。

（1）$m=0,\delta=2.7783$

（2）$m=1,\delta=3.7614$

（3）$m=2,\delta=4.7929$

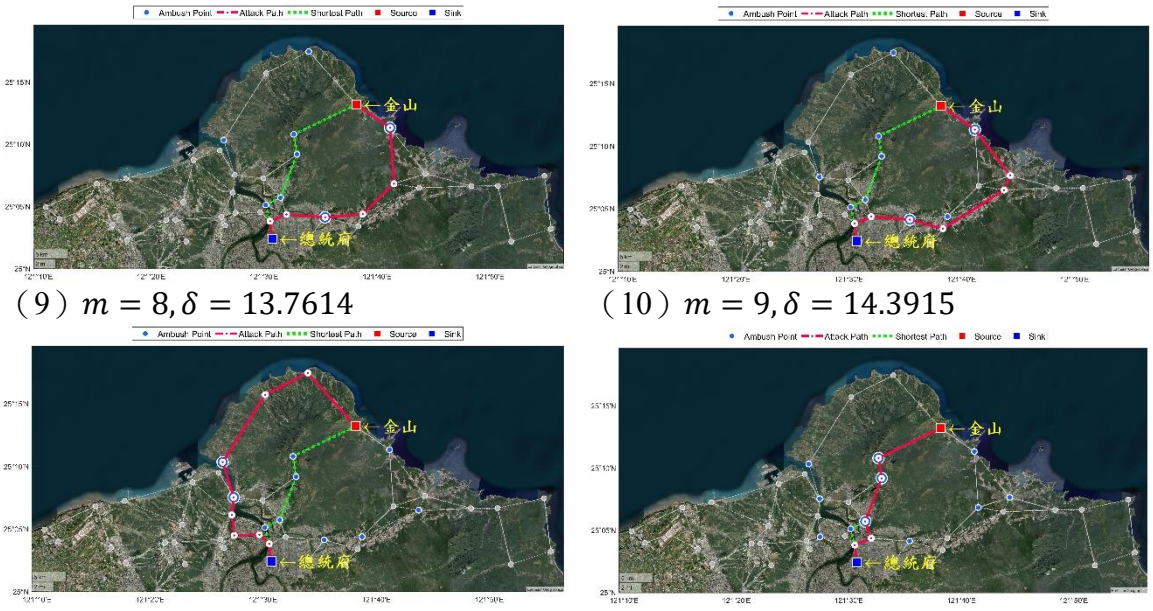
（4）$m=3,\delta=7.7783$

（5）$m=4,\delta=8.7614$

（6）$m=5,\delta=9.7929$

（7）$m=6,\delta=12.7783$

（8）$m=7,\delta=12.8542$



（9）$m = 8, \delta = 13.7614$　（10）$m = 9, \delta = 14.3915$

（11）$m = 10, \delta = 14.8962$　（12）$m = 11, \delta = 17.8137$

圖 12　實例問題求解結果示意圖（$m = 0$~11）

5.4.5 統計檢定

為檢視 SDBEA$_{SAS}$ 是否顯著優於其他演算法，採用 Friedman's test 及 Holm's test，對所有演算法在相同迭代次數$Nitr$及解群體$Nsol$的參數設定下，運算 30 次的平均適應函數值F_{avg}進行差異檢定，兩種檢定的顯著水準均設置為 0.05，檢定結果整理如表 9 所示（**粗體字**表示具顯著性），詳述如下：

（一）Friedman's test 結果顯示，SDBEA$_{SAS}$ 求解 24 個問題得到的平均排序值 Rank 最小，表示 SDBEA$_{SAS}$ 是求解效果最佳的演算法，其次為 SDBEA，再次之為 PDBEA，最差為 GDBEA，檢定結果顯示各演算法之間的數值存在顯著差異（p-value < 0.05）。

（二）進一步實施 Holm's test 事後檢定，結果顯示 SDBEA$_{SAS}$ 的平均適應函數值顯著優於 GDBEA 及 PDBEA 等兩種演算法，惟與 SDBEA 無顯著差異。

表 9　平均適應函數值統計檢定表

Method	Friedman's test			Holm's test	
	Rank	Statistic	p-value	Statistic	p-value
GDBEA	3.6667	46.0125	**<0.0001**	6.2051	**<0.0001**
PDBEA	3.0000			4.4162	**<0.0001**
SDBEA	1.9792			1.6771	0.0935
SDBEA$_{SAS}$	1.3542				

六、結論與建議

6.1 結論與貢獻

2022 年 2 月俄羅斯兵分三路閃電入侵烏克蘭，堪稱二戰以來歐洲最大規模的戰爭。戰力居劣勢的烏克蘭將兵力分散避免決戰，利用主場優勢創造局部有利條件，以游擊戰、以打帶跑的原則對敵軍進行襲擾，藉由零碎的攻勢發揮滴水穿石的效果，在精神上打擊

敵人士氣，物質上消耗其軍資，時至九月重新奪回超過 8000 平方公里的失土，不僅重創俄羅斯經濟，並影響全球經濟秩序，其守勢策略可為之借鏡（Mbah and Wasum, 2022; Armour et al., 2022）。

基於當前國土防衛作戰為達到戰略持久的效果，本研究提出新型態的防衛作戰策略，探討如何以國土防衛軍建構第四道防線，將國土防衛作戰由灘岸延續至國土上，以空間換取時間。採用最短路徑網攔截問題為基礎，以防禦者的角度建構雙層規劃之量化模型，可同時求解我軍兵力部署最佳化問題及侵略者在此部署情況下的最佳進攻路徑。因此，提出之模型可協助情報部門研判侵略者在登陸之後奪取最後目標的最大可能行動方案；作戰部門則可藉由此模型獲得相應的最佳部署方案，作為戰場整備與兵力部署之決策參考，提高國土防衛的成功公算，為本研究貢獻之一。為加速戰場情報分析，以協助戰場指揮官決心下達，在兼顧求解效率與品質的前提下，本研究提出之模型發展出求解方法 $SDBEA_{SAS}$，此為本研究貢獻之二。

6.2 研究發現

從本研究綜整以下研究發現：

（一）透過隨機及實例問題進行求解，並分析綜整出國土防衛第四道防線兵力部署的模式與策略，請參閱第伍章。

（二）透過戰前整備（地雷鋪設、橋樑破壞）將我國土之路網刻意形成隘道，有助於強化第四道防線。

（三）雙層規劃的求解過程同作戰部門與情報部門在研擬作戰方案時的情境，凸顯兩個決策者間對弈時拉扯之現象。因此可將雙層規劃嵌入國軍的軍事決策程序中，透過最佳化過程研擬相關之軍事決策。

6.3 未來研究方向

網路攔截問題是軍事作業研究中極為重要的研究議題，未來研究方向可針對問題類型、建模方式、攔截的標的與端點的特性加以延伸，例如敵軍採多點登陸，且不同時序發起進攻的可能性，或多個可能登陸點，但不確定起點為何的狀況，依議題發展適切的網路攔截模型，探討在不同登陸想定下的兵力部署策略。本研究發展之 $SDBEA_{SAS}$ 是首個以簡群演算法為基底的雙層進化式演算法，未來仍能將 $SDBEA_{SAS}$ 之架構應用在適當之問題，嘗試發揮其巢式求解架構，更進一步驗證其求解效能。

參考文獻

吳傳國（2009）。嚇阻在國家安全戰略上的運用與實踐—兼論小國嚇阻之道。*國防雜誌，24*（2），7-17。

陳勝昌（2013）。游擊戰理論研究及對國軍之啟示。*國防雜誌，28*（6），69-91。

葉紘胥（2017）。丙種城鎮守備旅在防衛作戰時期任務之研究。*後備動員軍事雜誌半年刊*，103，86-108。

蔡宗憲（2020）。人工智慧與軍事作戰模擬發展。*前瞻科技與管理，10*（1&2），1-6。

Armour, P. G., Berghel, H., Charette, R. N., & King, J. L. (2022). Ukraine aftershocks. *Computer*, *55*(11), 85-93.

Bard, J. F. (2013). *Practical Bilevel Optimization: Algorithms and Application*s. NJ: Springer Science & Business Media.

Bard, J. F., & Falk, J. E. (1982). An explicit solution to the multi-level programming problem. *Computers & Operations Research*, *9*(1), 77-100.

Bard, J. F., & Moore, J. T. (1990). A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, *11*(2), 281-292.

Bidgoli, M. M., & Kheirkhah, A. (2018). An arc interdiction vehicle routing problem with information asymmetry. *Computers & Industrial Engineering*, *115*, 520-531.

Colson, B., Marcotte, P., & Savard, G. (2005). Bilevel programming: A survey. *4OR*, *3*, 87-107.

Deepa, G., Kumar, P., Manimaran, A., Rajakumar, K., & Krishnamoorthy, V. (2018). Dijkstra algorithm application: Shortest distance between buildings. *International Journal of Engineering & Technology*, *7*(4.10), 974-976.

Di Caprio, D., Ebrahimnejad, A., Alrezaamiri, H., & Santos-Arteaga, F. J. (2022). A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights. *Alexandria Engineering Journal*, *61*(5), 3403-3415.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269-271.

Ford, L. R., & Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, *8*, 399-404.

Hsi-Min, L., & Lee, E. (2020). Taiwan's overall defense concept, explained. *The Diplomat, 3*.

Hunzeker, M. A., Lanoszka, A., Davis, B., Fay, M., Goepner, E., Petrucelli, J., & Seng-White, E. (2018). *A question of time: Enhancing Taiwan's conventional deterrence posture*. Center for Security Policy Studies.

Israeli, E. (1999). *System interdiction and defense*. Unpublished master's thesis, Naval Postgraduate School, Monterey, CA.

Israeli, E., & Wood, R. K. (2002). Shortest-path network interdiction. *Networks: An International Journal*, *40*(2), 97-111.

Lai, C. M. (2019). Integrating simplified swarm optimization with AHP for solving capacitated

military logistic depot location problem. *Applied Soft Computing*, *78*, 1-12.

Lai, C. M., & Yeh, W. C. (2016). Two-stage simplified swarm optimization for the redundancy allocation problem in a multi-state bridge system. *Reliability Engineering & System Safety*, *156*, 148-158.

Mbah, R. E., & Wasum, D. F. (2022). Russian-Ukraine 2022 War: A review of the economic impact of Russian-Ukraine crisis on the USA, UK, Canada, and Europe. *Advances in Social Sciences Research Journal*, *9*(3), 144-153.

Pan, F., Charlton, W. S., & Morton, D. P. (2003). A stochastic program for interdicting smuggled nuclear material. In *Network interdiction and stochastic integer programming* (1-19). Boston, MA: Springer.

Pirlot, M. (1996). General local search methods. *European Journal of Operational Research*, *92*(3), 493-511.

Ramirez-Marquez, J. E. (2010). A bi-objective approach for shortest-path network interdiction. *Computers & Industrial Engineering*, *59*(2), 232-240.

Ricks, T. E. (2007). *Fiasco: the American military adventure in Iraq*. UK: Penguin.

Said, R., Elarbi, M., Bechikh, S., & Said, L. B. (2022). Solving combinatorial bi-level optimization problems using multiple populations and migration schemes. *Operational Research*, *22*(3), 1697-1735.

Sheng, Y., & Gao, Y. (2016). Shortest path problem of uncertain random network. *Computers & Industrial Engineering*, 99, 97-105.

Sinha, A., Malo, P., & Deb, K. (2017). A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, *22*(2), 276-295.

Smith, J. C., & Lim, C. (2008). Algorithms for network interdiction and fortification games. In *Pareto optimality, game theory and equilibria* (60-644). New York, NY: Springer.

Smith, J. C., & Song, Y. (2020). A survey of network interdiction models and algorithms. *European Journal of Operational Research*, *283*(3), 797-811.

Strachan, H. (2019). Learning lessons from Afghanistan: Two imperatives. *The US* Army *War College Quarterly: Parameters*, *49*(3), 3.

Talbi, E.-G. (2013). A taxonomy of metaheuristics for bi-level optimization. In Talbi, E. G. 3(Eds), *Metaheuristics for bi-level optimization* (1-39). Berlin, Heidelberg: Springer.

Von Stackelberg, H. (1934). *Marktform and gleichgewicht*. Berlin, Germany: Springer-Verlag.

White, D. J., & Anandalingam, G. (1993). A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, *3*(4), 397-419.

Wollmer, R. (1964). Removing arcs from a network. *Operations Research*, *12*(6), 934-940.

Wood, R. K., 1993. Deterministic network interdiction. *Mathematical and Computer Modelling*, *17*(2), 1-18.

Wood, R. K. (2010). Bilevel network interdiction models: Formulations and solutions. *Network*, 174, 175.

Xiang, Y., & Wei, H. (2020). Joint optimizing network interdiction and emergency facility location in terrorist attacks. *Computers & Industrial Engineering*, *144*, 106480.

Yeh, W.-C. (2009). A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems. *Expert Systems with Applications*, *36*(5), 9192-9200.

Yeh, W.-C. (2012a). Novel swarm optimization for mining classification rules on thyroid gland data. *Information Sciences*, *197*, 65-76.

Yeh, W.-C. (2012b). Simplified swarm optimization in disassembly sequencing problems with learning effects. *Computers & Operations Research*, *39*(9), 2168-2177.

Yeh, W.-C. (2015). An improved simplified swarm optimization. *Knowledge-Based Systems*, *82*, 60-69.