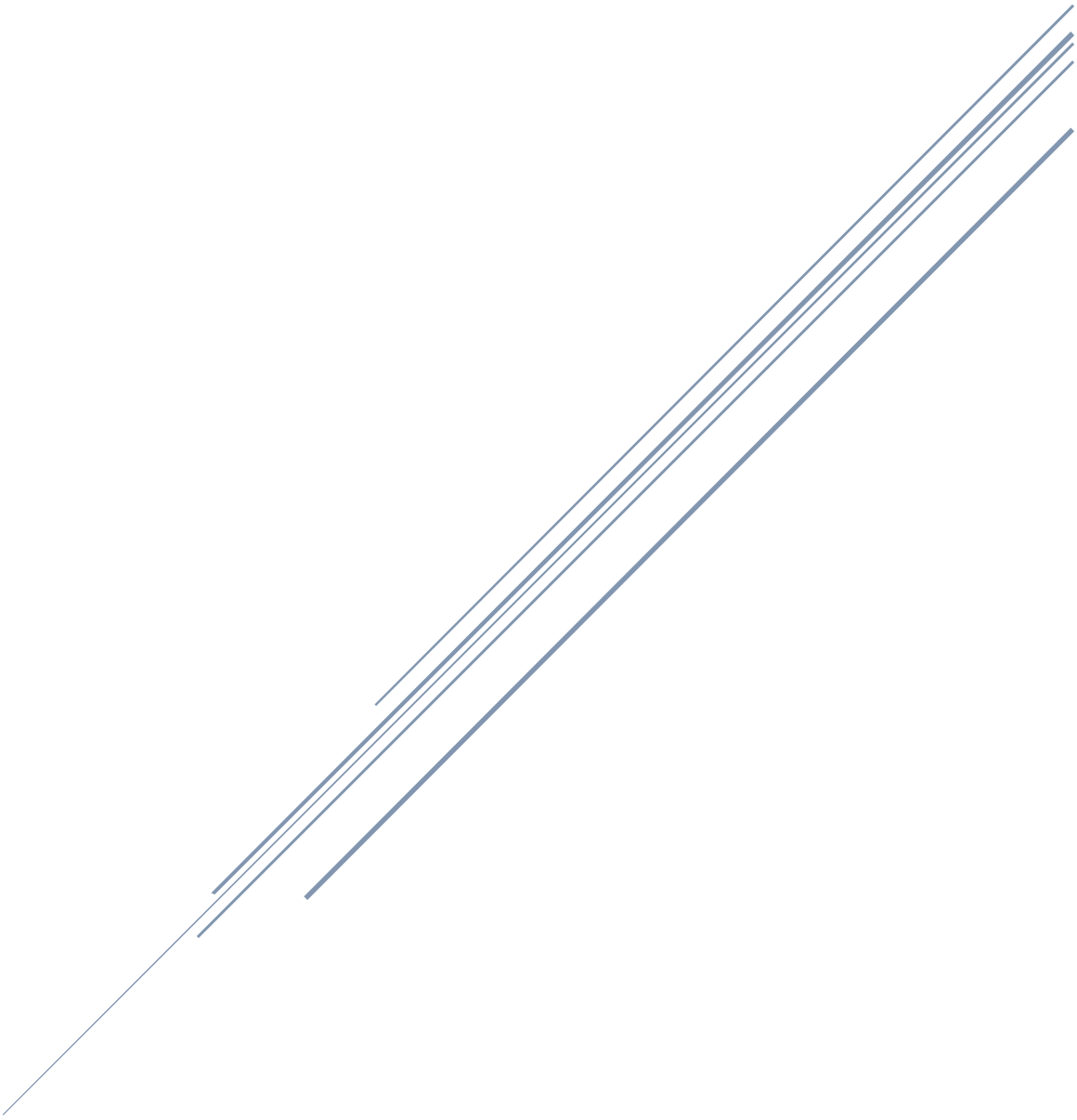
Term Project Proposal －Vertex Cover

DECISION PROCEDURE WITH APPLICATION TO CRYPTO VERIFICATION

Term Project Proposal



姓名：江冠緯
學號：B11009003

目錄

1. 摘要 (Abstract)

本提案旨在使用滿足性問題 (SAT) 解決 Vertex Cover 問題。Vertex Cover 問題是一個經典的 NP-完整問題，目標是判斷給定一個無向圖 $G=(V,E)$ 、整數 k、每個頂點 v_i的權重 w_i 以及權重上限 W，判斷是否存在一個大小為 k 的頂點集合 S，使得圖中每條邊至少有一個端點在此集合中，且 S 的總權重 $\sum_{v_i \in S} w_i \leq W$。我們將問題轉換為 SAT 問題，通過生成合取範式 (CNF) 公式，使用現代 SAT 求解器 (例如 Glucose) 進行求解。轉換過程涉及邊覆蓋約束和集合大小約束。本計畫將使用 Python 和 pysat 庫實現，並提供兩個問題實例進行測試。預計結果將展示 SAT 技術在解決組合問題中的有效性，並提供可重現的實現方案。

2. 問題定義 (Problem Statement)

Vertex Cover 問題是一個廣泛研究的 NP-Complete，在計算理論和 SAT 競賽中常作為基準問題。給定一個無向圖 G=(V,E)，其中 V 是頂點集合，E 是邊集合，每個頂點 $v_i \in V$ 具有一個非負整數權重 w_i，以及權重上限 W 與一個整數 k，問題要求判斷是否存在一個頂點子集 $S \subseteq V$，滿足以下條件：

1. |S| ≤ K (集合大小不超過 k)。
2. 對於每條邊 (u,v)∈E，至少有一個端點 u ∈ S 或 v ∈ S (即 S 是一個頂點覆蓋)。

3. S 的總權重 $\sum_{v_i \in S} w_i \leq W$。

為了適應 SAT 編碼，我們假設 W是給定的上限，並將問題轉換為判斷是否存在一個**大小恰為 k** 頂點覆蓋 S 滿足上述條件。本計畫的目標是將

Vertex Cover 問題轉換為 SAT 問題，並使用 SAT 求解器解決中等規模的實例（例如 50 個頂點，100 條邊，權重範圍在 [1, 10]，W = 50）。將提供詳細的轉換過程、實現方案和測試實例，以符合 SAT 競賽的要求。

3. Reduction 概述 (Overview of the Reduction)

Vertex Cover 問題到 SAT 的轉換是一個標準的 NP-Complete 問題轉換。我們將問題編碼為一個合取範式（CNF）命題公式，該公式可滿足當且僅當圖 G 存在一個大小為 k 且總權重不超過 W 的頂點覆蓋。

3.1 變數定義

對於每個頂點 $v_i \in V$（其中 $i = 1, 2, \ldots, |V|$），我們定義一個命題變數 x_i，表示頂點 v_i 是否被選擇進入頂點覆蓋 S。若 x_i 為真，則 $v_i \in S$；若 x_i 為假，則 $v_i \notin S$。

為了編碼 "at-most-k" 和 "at-least-k" 約束，我們引入輔助變數來實現基數約束的分解：

- 對於 "at-most-k"，我們引入輔助變數 $s_{i,j}$（其中 i=1, 2, …, | V |, j = 1, 2, … , k+1），用於表示前 i 個頂點中選擇的頂點數是否超過 j。若 $s_{i,j}$ = true，則表示前 i 個頂點中選擇的頂點數 $\geq j$。
- 對於 "at-least-k"，我們引入輔助變數 $t_{i,j}$（其中 i = 1, 2, …, | V | − k + 1），用於表示前 i 個頂點中未選擇的頂點數是否超過 j。

為了編碼權重約束 $\sum_{v_i \in S} w_i \leq W$，我們引入輔助變數來實現加權計數：

- 對於每個頂點 v_i 和權重位 j（從 0 到 $\lceil log_2 max(w_i) \rceil$），定義二進制變數 $b_{i,j}$ 表示 w_i 的第 j 位。
- 引入總權重計數變數 $s_{i,j}$（其中 $i = 1, 2, \ldots, |V|$，$j = 0, 1 \ldots, \lceil \log_2 W \rceil + 1$），表示前 i 個頂點選擇的總權重在二進制表示下的第 j 位。

3.2 子句生成

我們需要生成以下約束的子句：

1. **邊覆蓋約束**： 對於每條邊 $(v_i, v_j) \in E$，至少有一個端點必須在頂點覆蓋中。因此，對於每條邊 (v_i, v_j)，我們生成子句：

$$(x_i \vee x_j)$$

這確保至少有一個變數 x_i 或 x_j 為真。

2. **集合大小恰為 k**： 為了確保選擇恰好 k 個頂點，我們需要 "至少 k" 和 "至多 k" 約束：

我們使用更高效的基數約束編碼來實現 "at-most-k" 和 "at-least-k" 約束，而不是直接生成所有子集的子句。

- **至多 k 個頂點**（At-Most-k 約束）： 使用輔助變數 $c_{i,j}$ 實現計數邏輯：
  - **初始化**：對於 i=1：

$$c_{1,1} \equiv x_1 \quad \text{（若第一個頂點被選擇，則選擇數 (≥1)),}$$

$$c_{1,j} \equiv \text{false} \quad \text{（對於（j ≥2））}$$

生成子句：$(\neg c_{1,1} \vee x_1)$, $(c_{1,1} \vee \neg x_1)$, $(\neg c_{1,j})$ （對於（j ≥2））

  - **遞推**：對於 $i \geq 2$ 和 $j = 1, 2, \ldots, k+1$：

$$s_{i,j} \equiv \left(c_{i-1,j} \vee \left(c_{i-1,j-1} \wedge x_i\right)\right),$$

表示前 i 個頂點選擇數 $\geq j$，若前 $i-1$ 個已選擇 $\geq j$，或前 i – 1 個選擇 $\geq j-1$ 且第 i 個被選擇。生成子句：

$$(\neg c_{i,j} \vee c_{i-1,j}), \quad (\neg c_{i,j} \vee c_{i-1,j-1}), \quad (\neg c_{i,j} \vee x_i),$$

$$(c_{i,j} \vee \neg c_{i-1,j} \vee \neg c_{i-1,j-1}), \quad (c_{i,j} \vee \neg c_{i-1,j} \vee \neg x_i).$$

  - **最終約束**：確保總選擇數不超過 k，即 $c_{|V|,k+1} = \text{false}$：

$$(\neg c_{|V|,k+1})$$

- **至少 k 個頂點**（At-Least-k 約束）： 使用輔助變數 $y_i = \neg x_i$ 表示頂點未被選擇，然後對 y_i 應用 "at-most-|V|-k" 約束：
  - 定義 y_i：

$$y_i \equiv \neg x_i,$$

生成子句：

$$(\neg y_i \vee \neg x_i), \quad (y_i \vee x_i).$$

  - 使用輔助變數 $t_{i,j}$ 表示前 i 個頂點中未選擇的數量 $\geq j$：

- **初始化**：對於 i=1：

$$t_{1,1} \equiv y_1, \quad t_{1,j} \equiv \text{false} \quad (\text{對於 (j } \geq 2))$$

生成子句：

$$(\neg t_{1,1} \vee y_1), \quad (t_{1,1} \vee \neg y_1), \quad (\neg t_{1,j}) \quad (\text{對於 (j } \geq 2))$$

- **遞推**：對於 $i \geq 2$ 和 $j = 1,2,\ldots,|V| - k + 1$：

$$t_{i,j} \equiv \left(t_{i-1,j} \vee \left(t_{i-1,j-1} \wedge y_i\right)\right),$$

生成子句：

$$(\neg t_{i,j} \vee t_{i-1,j}), \quad (\neg t_{i,j} \vee t_{i-1,j-1}), \quad (\neg t_{i,j} \vee y_i),$$

$$(t_{i,j} \vee \neg t_{i-1,j} \vee \neg t_{i-1,j-1}), \quad (t_{i,j} \vee \neg t_{i-1,j} \vee \neg y_i).$$

- **最終約束**：確保未選擇的頂點數不超過 $|V| - k$，即 $t_{|V|,|V|-k+1} = \text{falset}$：

$$(\neg t_{|V|,|V|-k+1})$$

3. 權重約束 $\sum_{v_i \in S} w_i \leq W$:

- **權重二進制表示**： 對於每個頂點 v_i，將權重 w_i 轉換為二進制表示 $b_{i,0}, b_{i,1}, \ldots, b_{i,m}$ $(m = \lceil log_2 \, max(w_i) \rceil)$，生成子句確保 $b_{i,j}$ 與 w_i 一致。

- **總權重計數**： 使用 $s_{i,j}$ 跟踪選擇的總權重：

  - **初始化**：對於 $i = 1$：

$$s_{1,j} \equiv \bigvee_{k=0}^{m} \left(b_{1,k} \wedge x_1 \wedge (\text{第 (k) 位為 } 1)\right)$$

生成子句表示權重的加法。

  - **遞推**：對於 $i \geq 2$ 和 $j = 0,1,\ldots,\lceil \log_2 W \rceil + 1$：

$$s_{i,j} \equiv \left(s_{i-1,j} \vee \left(s_{i-1,j-1} \wedge \bigvee_k \left(b_{i,k} \wedge x_i\right)\right)\right),$$

表示前 i 個頂點的總權重在第 j 位。生成對應的子句。

  - **最終約束**：確保總權重不超過 W，即 $s_{|V|,\lceil \log_2 W \rceil + 1} = \text{false}$:

$(\neg s_{|V|,\lceil \log_2 W \rceil+1})$

最終的 CNF 公式是所有上述子句的合取。該公式可滿足當且僅當圖 G 存在一個大小為 k、總權重不超過 W 的頂點覆蓋。

4. 實作計畫（Implementation Plan）

4.1. 如何設置、編譯和執行程式碼（How to Setup, Compile, and Run the Code）

- 環境設置：
  - 安裝 Python 3.9 或以上版本。
  - 安裝 pysat 庫，該庫提供與 SAT 求解器的接口：

    ```
    pip install python-sat
    ```

  - pysat 庫會自動使用內建的 SAT 求解器（例如 Glucose），無需額外安裝。
- 程式碼儲存：
  - 將程式碼儲存為 vertex_cover_sat.py。
- 執行程式：
  - 通過命令列執行：

    ```
    python vertex_cover_sat.py <input_file> <k>
    ```

其中 <input_file> 是圖的輸入檔案，<k> 是頂點覆蓋的大小。例如：

```
python vertex_cover_sat.py graph.txt 2
```

4.2. 轉換的實現方式（How the Reduction is Implemented）

- **變數生成：**
  - 對於每個頂點 v_i，生成 x_i（映射為 i）。
  - 對於 $y_i = \neg x_i$，映射為 $i+|V|$。
  - 對於 $c_{i,j}$（選擇計數），映射為 $2|V| + (i-1)\cdot(k+1) + j$。
  - 對於 $t_{i,j}$（未選擇計數），映射為 $2|V| + |V|\cdot(k+1) + (i-1)\cdot(|V|-k+1) + j$。
  - 對於 $b_{i,j}$（權重二進制位），映射為 $2|V| + |V|\cdot(k+1) + |V|\cdot(|V|-k+1) + (i-1)\cdot\lceil \log_2 \max(w_i)\rceil + j$。
  - 對於 $s_{i,j}$（總權重計數），映射為 $2|V| + |V|\cdot(k+1) + |V|\cdot(|V|-k+1) + |V|\cdot\lceil \log_2 \max(w_i)\rceil + (i-1)\cdot(\lceil \log_2 W\rceil + 1) + j$

- **子句生成：**
  - **邊覆蓋約束**：遍歷所有邊$(u,v) \in E$，生成子句 [u, v]。
  - **集合大小約束**：
    - 至多 k：生成 $c_{i,j}$ 的初始化、遞推和最終約束子句。
    - 至少 k：生成 y_i、$t_{i,j}$ 的初始化、遞推和最終約束子句。
  - **權重約束**：
    - 將每個 w_i 轉換為二進制表示，生成對應子句。
    - 初始化 $s_{1,j}$ 並遞推 $s_{i,j}$，生成權重加法子句。
    - 添加最終約束 $(\neg s_{|V|,\lceil \log_2 W \rceil+1})$
- **SAT 求解器交互：**
  - 使用 pysat 庫的 Glucose3 求解器，將生成的子句添加到求解器中，並調用 solve() 方法。
  - 若公式可滿足，則用 getModel ()提取模型，解析選擇的頂點。

4.3. 輸入格式 (The Input Format)

- 輸入檔案採用擴展的 DIMACS 格式，包含權重：
  - 第一行：p edge <num_vertices> <num_edges> <max_weight> <k> <weight_limit>，指定頂點數、邊數、最大權重、集合大小 kkk 和權重上限 WWW。
  - 第二行：w <w_1> <w_2> ... <w_n>，指定每個頂點的權重 wiw_iwi。
  - 後續行：e <v1> <v2> 表示一條邊。
- 範例輸入（4 個頂點，4 條邊，權重範圍 [1, 5]，k=2，W=10）：

```
p edge 4 4 5 2 10
w 2 3 1 4
e 1 2
e 2 3
e 3 4
e 4 1
```

4.4. 輸出的解釋 (The Interpretation of the Output)

- 程式輸出：
  - 若存在大小為 k 的頂點覆蓋：
    - 第一行顯示 “SATISFIABLE”。
    - 第二行列出選擇的頂點，例如：

      SATISFIABLE
      頂點覆蓋：Vertex 1, Vertex 3, 總權重：3
  - 若不存在：

- 顯示 “UNSATISFIABLE”。

- 測試實例：
  - 實例 1：上述環形圖，k=2, w=10：
    - 預期輸出：

```
SATISFIABLE
頂點覆蓋：Vertex 1, Vertex 3, 總權重：3
```

  - 實例 2：一個 6 個頂點的圖，k=2, w=5：

```
p edge 6 7 10 2 5
w 2 3 1 4 2 5
e 1 2
e 1 3
e 1 4
e 2 5
e 3 5
e 4 6
e 5 6
```

    - 預期輸出：

```
UNSATISFIABLE
```

4.5. 實作細節

- 程式使用 Python 實現，主要依賴 pysat 庫的 Glucose3 求解器。
- 程式碼結構：
  - read_graph()：讀取 DIMACS 格式的圖。
  - vertex_cover_to_sat()：生成 SAT 公式的子句。
  - main()：主函數，調用求解器並解析結果。
- 實例測試將包括至少兩個圖：一個可滿足的實例（環形圖，k=2）和一個不可滿足的實例（6 個頂點，k=2）。

5. 參考文獻（References）

1. SAT Competition. (2023). SAT Competition 2023 Benchmarks. Retrieved from https://satcompetition.github.io/.
2. Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). Handbook of Satisfiability. IOS Press. （提供了 SAT 求解技術和轉換方法的詳細背景知識，包括 Vertex Cover 的編碼方法。）