Expert Systems With Applications 328 (2026) 132943

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An observer-enhanced evolutionary search framework for solving the simplified diversified top-k s-plex problem

Jiejiang Chen [a], Shuping Xu [a], Jun Wu [b,*], Weiguo Sheng [a,*]

[a] School of Information Science and Technology, Hangzhou Normal University, Hangzhou, 311121, China
[b] School of Computer Science, Nanjing University of Information Science & Technology, Nanjing, 210044, China

ARTICLE INFO

Keywords:
Combinatorial optimization
Top-k
s-plex
Evolutionary search

ABSTRACT

Identifying multiple large and diversified cohesive subgraphs plays a crucial role in a wide range of real-world applications. The Simplified Diversified Top-k s-Plex (S-DTKSP) problem aims to identify k s-plexes that collectively maximize vertex coverage, generalizing the classical maximum s-plex problem while significantly increasing computational complexity. However, existing algorithms often suffer from limited scalability or suboptimal solution quality on large or dense graphs. To address these challenges, we propose an Evolutionary Search Algorithm with Observer Individual (EA-OI) and its specialized instantiation for the S-DTKSP problem, termed EA-OIS. Unlike traditional evolutionary algorithms, EA-OI introduces an observer individual that continuously monitors the evolutionary process and selectively retains promising solution components, thereby mitigating premature information loss and enabling a dual-mode search mechanism that balances exploration and exploitation. To further enhance performance, two complementary initialization strategies — Tri-Tier Vertex Prioritization (TTVP) and Dynamic Prioritized Weighted Selection (DPWS) — are designed to generate high-quality and diverse initial solutions. In addition, the concept of Neighbor Coverage Density (NCD) is introduced, together with a two-layer scoring function to guide solution updates during evolution. Extensive experiments on 139 benchmark graphs comprising 2224 instances show that EA-OIS achieves new best-known results on 1284 instances and outperforms the state-of-the-art Iterated Local Search algorithm on 1557 instances, demonstrating its effectiveness and robustness.

1. Introduction

Mining multiple large and diverse cohesive subgraphs is a fundamental yet challenging task in complex network analysis (Hastad, 1996; Wolfe, 1997). In many practical scenarios, such as social network analysis (Balasundaram et al., 2011; Zhu et al., 2020), biological network exploration (Etzion & Ostergard, 2002; Strickland et al., 2005), and recommendation systems (Boginski et al., 2006; Dorndorf et al., 2008), it is often insufficient to identify only a single optimal subgraph; instead, a set of high-quality subgraphs with broad vertex coverage is required (Fan et al., 2013; Hao et al., 2020; Zou et al., 2010). Motivated by this need, Wu et al. introduced the Simplified Diversified Top-k s-Plex (S-DTKSP) problem (Wu et al., 2023; Wu & Yin, 2021), which aims to select k s-plexes that collectively maximize vertex coverage. The s-plex model, a well-known relaxation of the clique that tolerates missing edges, provides robustness to noise and incomplete data while retaining strong cohesion properties (Chang & Yao, 2024).

The S-DTKSP problem generalizes the classical Maximum s-Plex Problem (Gao et al., 2024; Pullan, 2021) by shifting the focus from identifying a single largest s-plex to discovering multiple diversified s-plexes. This generalization significantly increases computational complexity (Dondi et al., 2021; Ghanbarpour et al., 2020; Xu et al., 2024). In fact, S-DTKSP has been proven NP-hard (Wu et al., 2023), and a number of solution approaches have been proposed, which can be broadly categorized into exact, approximation, and heuristic algorithms. Wu et al. formulated an Integer Linear Programming (ILP) model (Adouani et al., 2024) that is capable of producing optimal solutions on small-scale graphs when solved using commercial solvers such as CPLEX (Wu et al., 2023). However, due to the exponential growth of the search space, exact methods quickly become infeasible as graph size increases.

To address scalability, several approximation algorithms have been developed (Conte et al., 2017; Wu & Yin, 2021). Among them, EnumMax and EnumFast are representative enumeration-based frameworks that differ mainly in their s-plex retention strategies (Wu et al., 2023). EnumMax achieves a $(1 - 1/e)$ approximation ratio by exhaustively enumerating candidate s-plexes, whereas EnumFast significantly reduces memory consumption through an integrated enumertion-selection mechanism, at the cost of a lower approximation ratio of 0.25.

* Corresponding authors.
E-mail addresses: chenjj016@hznu.edu.cn (J. Chen), xushuping@stu.hznu.edu.cn (S. Xu), wujun@nuist.edu.cn (J. Wu), w.sheng@ieee.org (W. Sheng).

https://doi.org/10.1016/j.eswa.2026.132943
Received 3 March 2026; Received in revised form 29 April 2026; Accepted 18 May 2026
Available online 26 May 2026

In the heuristic category, the Iterated Local Search (ILS) algorithm (Hamza et al., 2023; Paquete & Stützle, 2018) currently represents the state of the art. ILS employs multiple neighborhood update operators together with tabu-based diversification strategies to effectively explore the solution space (Wu et al., 2023). Despite their effectiveness, existing algorithms – whether exact, approximation, or heuristic – often suffer from limited scalability on large or dense graphs and frequently produce suboptimal solutions in practice. These limitations motivate the development of more efficient and robust algorithms for solving the S-DTKSP problem.

In this paper, we propose an Evolutionary Search Algorithm with Observer Individual (EA-OI), a novel and general heuristic framework, together with its specialized instantiation for the S-DTKSP problem, namely EA-OIS. Unlike traditional evolutionary algorithms, EA-OI introduces an observer individual that does not directly participate in population operations but continuously monitors the evolutionary process. By assimilating population updates and selectively retaining promising solution components, the observer individual helps preserve valuable search information and mitigates premature information loss. As a result, the final solution can be generated either from the evolving population or from the observer itself, forming a dual-mode search mechanism that effectively balances exploration and exploitation.

To further enhance the performance of EA-OIS, we design two complementary initialization strategies: Tri-Tier Vertex Prioritization (TTVP) and Dynamic Prioritized Weighted Selection (DPWS). TTVP categorizes vertices into three priority tiers based on historical solution quality, thereby guiding the initialization process toward structurally significant regions of the graph. DPWS adopts a power-law-based probability function to dynamically generate selection weights, ensuring both high-quality initial individuals and sufficient population diversity.

For the solution update phase, we introduce the concept of Neighbor Coverage Density (NCD), which quantifies the potential impact of modifying a vertex on future solution expansion. Building upon NCD, a two-layer scoring function is developed to evaluate and guide vertex updates during the evolutionary process. We evaluate EA-OIS on 139 benchmark graphs comprising 2224 instances. Experimental results demonstrate that the proposed algorithm achieves new best-known results in 1284 instances, outperforms the state-of-the-art ILS algorithm in 1557 instances, and performs comparably in the remaining cases.

The remainder of this paper is structured as follows. Section 2 formally defines the S-DTKSP problem and presents the necessary notation. Section 3 introduces the EA-OI framework. Section 4 details the TTVP and DPWS initialization strategies, while Section 5 elaborates on the NCD-based scoring mechanism. Section 6 provides the complete algorithmic workflow. Section 7 reports comparative and ablation experiments, and Section 8 concludes the paper with future research directions.

2. Preliminaries

Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, \dots, v_n\}$ denotes the vertex set and $E = \{e_1, \dots, e_m\}$ denotes the edge set. An undirected edge between vertices u and v is represented by both (u, v) and (v, u). Vertex u (resp. v) is said to be adjacent to or a neighbor of v (resp. u). The set of neighbors of vertex v in G is denoted by $N_G(v) = \{u \in V \mid (u, v) \in E\}$, and the degree of v is given by $deg_G(v) = |N_G(v)|$. We use $N[v]$ to denote the closed neighborhood of v, i.e., $N[v] = N(v) \cup \{v\}$. The maximum vertex degree in G is denoted by Δ_G.

For a vertex subset $C \subseteq V$, the subgraph of G induced by C is denoted by $G[C] = (C, \{(u, v) \in E \mid u, v \in C\})$. A subset C is called an s-plex if every vertex in C is nonadjacent to at most $s - 1$ other vertices in $G[C]$. A vertex $v \in C$ is said to be saturated if $|N(v) \cap C| = |C| - s$. The set of all saturated vertices in C is denoted by $D(C) = \{v \mid v \in C, |N(v) \cap C| = |C| - s\}$. For simplicity, we refer to an s-plex by its vertex set and omit the subscript G when the context is clear. Given a collection of s-plexes $S = \{C_1, C_2, \dots\}$, the coverage of S, denoted $cov(S)$, is the set of vertices covered by S, i.e.,

$$cov(S) = \bigcup_{C_i \in S} C_i$$

Given an undirected graph $G = (V, E)$, the S-DTKSP problem aims to identify a set of k s-plexes $S = \{C_1, C_2, \dots, C_k\}$ that maximizes the overall coverage $|cov(S)|$. The problem can be formulated as the following Integer Linear Programming (ILP) model:

$$\text{Maximize} \quad f(G, s, k) = \sum_{v_i \in V} y_i \tag{1}$$

subject to

$$\sum_{v_j \in V \setminus N[v_i]} x_{jh} \le (s-1)x_{ih} + \bar{d}_i(1 - x_{ih}), \qquad \forall v_i \in V, \quad 1 \le h \le k \tag{2}$$

$$y_i \le \sum_{h=1}^{k} x_{ih}, \qquad \forall v_i \in V \tag{3}$$

$$x_{ih} \in 0, 1, \qquad \forall v_i \in V, \quad 1 \le h \le k \tag{4}$$

$$y_i \in 0, 1, \qquad \forall v_i \in V \tag{5}$$

where binary variable x_{ih} indicates whether vertex v_i belongs to the hth s-plex ($x_{ih} = 1$ if true, 0 otherwise), and $\bar{d}_i = |V \setminus N[v_i]|$ represents the degree of v_i in the complement graph $\bar{G} = (V, \bar{E})$. Variable y_i is a binary indicator denoting whether vertex v_i is covered by at least one s-plex ($y_i = 1$ if covered, 0 otherwise).

3. Evolutionary algorithm with an observer individual

Evolutionary Algorithms (EAs), as a class of metaheuristic optimization techniques, have been widely employed to solve diverse and complex optimization problems (He & Liu, 2025; Yue et al., 2025). In the context of Top-k subgraph mining, Wu et al. proposed a hybrid EA to address the diversified Top-k weighted clique problem (Wu et al., 2022), while Xue et al. developed an EA integrated with a post-reduction strategy to tackle the diversified Top-k clique problem (Xue et al., 2024). Although these methods target different problem variants, they share a common search paradigm: the exploration capability of the algorithm is enhanced through population-based evolutionary operations, while its exploitation capability is strengthened via local search refinement applied to individual solutions.

Within this paradigm, the population consists of multiple individuals, each encoding a feasible solution (i.e., a set of k subgraphs). The algorithm iteratively alternates between population evolution and local search until termination, as illustrated in Fig. 1.

The local search component typically employs a subgraph replacement strategy, in which a newly discovered subgraph attempts to replace one within the current individual. However, two major limitations can be observed:

- **Lack of continuity** – local search phase is interrupted by population operations, disrupting the refinement process.
- **Limited information sharing** – subgraphs discovered during local search are utilized solely to improve the current individual and are not disseminated across the population.

These limitations restrict the algorithm's capacity to fully exploit its search potential and lead to the loss of valuable intermediate information.

To overcome these drawbacks, we propose a novel EA framework termed the Evolutionary Algorithm with an Observer Individual (EA-OI). In addition to the standard population, EA-OI introduces an auxiliary observer individual with the following characteristics:

- It does not participate in any population operations such as selection, crossover, or mutation.
- It receives every new subgraph generated during local search and updates itself through a selective replacement mechanism.

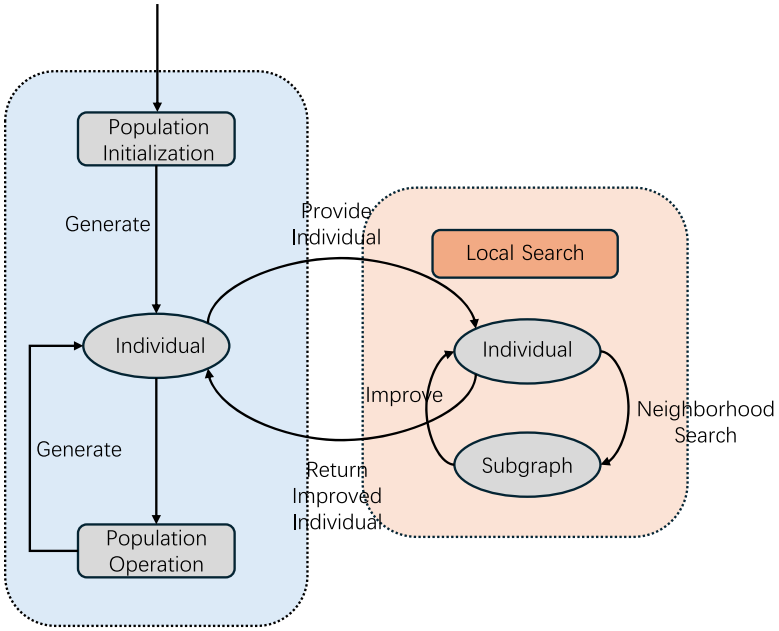


Fig. 1. EA search framework.

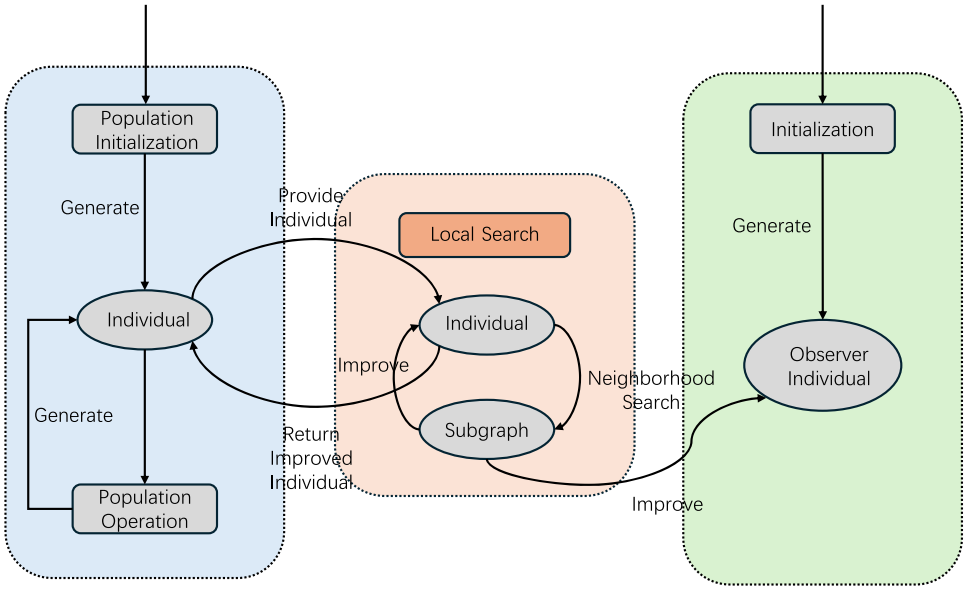


Fig. 2. EA-OI search framework.

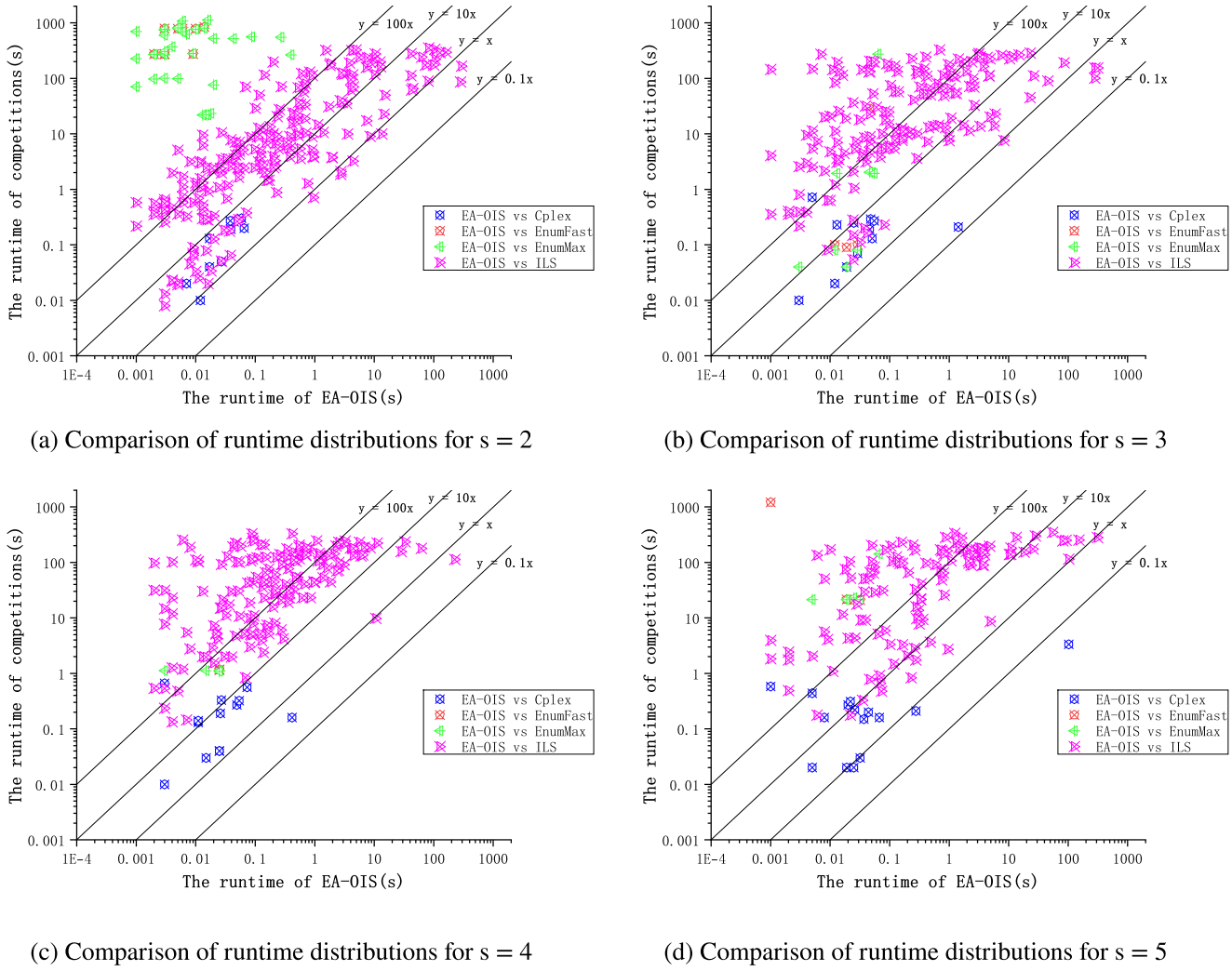
The search framework of EA-OI is illustrated in Fig. 2, where the main EA continues along its conventional evolutionary trajectory while each subgraph identified during local search is concurrently delivered to the observer individual for potential incorporation.

Compared with the conventional EA framework, EA-OI offers the following advantages:

- **Continuous utilization of local search outcomes:** every newly discovered subgraph contributes to the observer individual, ensuring uninterrupted accumulation of search information.
- **Limited runtime overhead:** the observer individual is inexpensive to maintain, and it operates independently, without interfering with population dynamics, while still benefiting from the search direction guided by the population. Its computational overhead is further analyzed in Section 6.

This design allows EA-OI to preserve the exploration-exploitation balance of the original EA while the observer individual plays two complementary roles: it acts as a long-term memory that persistently retains high-quality subgraphs across the entire search history without risk of loss through population operations, and as an auxiliary intensification path that accumulates the best components encountered by all population individuals, often converging to a solution superior to any single individual in the population. The final solution is selected as the best individual among the entire population and the observer. The pseudocode of EA-OI is presented in Algorithm 1.

The EA-OI framework serves as a general-purpose evolutionary architecture that can be adapted to various optimization problems traditionally solved using EAs. In Algorithm 1, we illustrate its application to the Top-k subgraph problem. The core local search process (Lines 5–8) focuses on subgraph mining and individual updating, while other components can be customized according to the problem characteris-



(a) Comparison of runtime distributions for s = 2

(b) Comparison of runtime distributions for s = 3

(c) Comparison of runtime distributions for s = 4

(d) Comparison of runtime distributions for s = 5

Fig. 3. Comparison of runtime distributions between EA-OIS and its competitors.

Algorithm 1: Evolutionary algorithm with an observer individual (EA-OI).

Input: undirected graph
Output: optimal solution found
$Pop \leftarrow$ *initialize* the population;
$S_o \leftarrow$ *initialize* the observer individual;
while *the cut-off conditions are not met* **do**
  $S_{new} \leftarrow$ *population operation* based on Pop;
  while *the cut-off conditions for local search are not met* **do**
    $C \leftarrow$ mine a new subgraph based on S_{new};
    update S_{new} with C;
    update S_o with C;
  update Pop with S_{new};
return best solution among $Pop \cup \{S_o\}$

tics. In particular, the generic EA-OI elements — the observer individual, selective replacement, and dual-mode output — are designed to be problem-agnostic and are potentially transferable to other evolutionary frameworks. The S-DTKSP-specific elements (introduced below) — TTVP's degree-based tiers (relying on Lemmas 1-2), DPWS's missing-edge grouping, and the AddSet/SwapSet operators — require adaptation for other subgraph models. The transferable elements — the NCD two-layer scoring and DPWS weighting scheme — apply broadly to any top-k subgraph problem with a vertex-coverage objective. This flexibility enables EA-OI to be efficiently extended and maintained for a wide range of optimization tasks.

In the following sections, we present the proposed heuristic search strategies for efficiently solving the S-DTKSP. These strategies are elaborated from two key perspectives: individual initialization and individual update.

4. Individual initialization

In the EA-OI framework, each individual represents a candidate solution (Fang et al., 2024; Sheng et al., 2024); therefore, the method used to generate individuals plays a crucial role in determining the algorithm's overall performance. The generation of individuals can be divided into two stages: individual initialization and individual update. This section focuses on the initialization process, while the update procedure will be discussed in the next section. Since each individual consists of k s-plexes, initialization can be decomposed into k separate instances of s-plex construction. To achieve effective initialization, we propose two complementary strategies — Tri-Tier Vertex Prioritization (TTVP) and Dynamic Prioritized Weighted Selection (DPWS) — to guide both the initial vertex selection and the subsequent vertex expansion during s-plex initialization.

4.1. Tri-tier vertex prioritization

We first establish two lemmas that describe fundamental properties of the s-plex structure:

Lemma 1. *Given an undirected graph $G = (V, E)$, for any vertex $v \in V$, the maximum size of an s-plex containing v cannot exceed $deg_G(v) + s$.*

Proof. By the definition of an s-plex, each vertex can be nonadjacent to at most $s - 1$ other vertices. For any s-plex C that includes vertex v,

$$deg_G(v) + s - 1 + 1 \geq |C| \implies |C| \leq deg_G(v) + s$$

□

Lemma 2. *Given an undirected graph $G = (V, E)$ and a s-plex C of G, for a vertex sequence $\{v_1, v_2, \ldots, v_{|C|}\}$ where $v_i \in C$ and $deg_G(v_i) \leq deg_G(v_j)$ for $1 \leq i < j \leq |C|$, the following holds:*

$$|C| \leq deg_G(v_1) + s \leq deg_G(v_2) + s \leq \ldots, \leq deg_G(v_{|C|}) + s$$

The proof follows directly from Lemma1. Hence, the vertex with the smallest degree in an s-plex largely determines the upper bound of the s-plex size. Based on this insight, we categorize vertices according to their degrees and assign them different selection priorities.

Specifically, we use the first constructed individual as a reference, we identify the largest and smallest s-plexes, denoted $C_{\max}$ and $C_{\min}$, to establish upper and lower degree bounds:

$$UB = |C_{max}| - s, LB = |C_{min}| - s.$$

An intermediate threshold is defined as $MB = \lfloor (UB + LB)/2 \rfloor$. Vertices are then grouped into three categories according to their degrees:

- $\mathcal{V}_1$: Vertices with degree $LB \leq deg_G(v) < MB$;
- $\mathcal{V}_2$: Vertices with degree $MB \leq deg_G(v) < UB$;
- $\mathcal{V}_3$: Vertices with degree $deg_G(v) \geq UB$.

Vertices with higher degrees possess greater expansion potential and are therefore more likely to be selected as initial vertices. Vertices with degrees smaller than LB are excluded from consideration. For each category, we define selection probabilities p_1, p_2, and p_3, which are tuned empirically (as described in Section 7).

Compared with traditional random or purely greedy initialization methods, TTVP offers the following advantages:

1. **Retention of greedy benefits** — Vertices with higher degrees retain a higher likelihood of selection.
2. **Enhanced diversity** — Vertices within each category are chosen randomly with equal probability, promoting structural diversity.
3. **Empirical reliability** — The upper and lower bounds for categorization are adaptively determined based on the characteristics of the first individual.

4.2. Dynamic prioritized weighted selection

Once the initial vertices have been selected, the remaining vertices of an s-plex are constructed through sequential expansion. The order in which vertices are added critically influences the quality of the resulting s-plex. Following the recommendation of Pullan (2021), we partition all candidate vertices — those satisfying the addition constraints — into s groups according to the number of edges they can still miss:

$$T_i = \{v \in N(C) \mid |C \setminus N(v)| = s - i, D(C) \setminus N(v) = \emptyset\}$$

where C denotes the currently constructed s-plex and $D(C)$ is the set of saturated vertices in C. The index i satisfies $1 \leq i \leq s$; a larger i indicates higher extension potential. For example, when $i = s$, the vertices in T_i are adjacent to all vertices in C. Prioritizing high-i vertices offers two benefits:

1. Such vertices are less likely to become saturated, as they still possess a larger margin for missing edges.
2. Their inclusion expands the future search space, allowing for greater flexibility in subsequent vertex additions.

A native greedy policy that always chooses the highest-priority vertex tends to cause premature convergence (Song & Zhang, 2024; Zheng et al., 2025). To mitigate this, we propose the DPWS approach, which combines a power-based weighting scheme with roulette-wheel selection.

Specifically, a weight function is assigned to each vertex group T_i as:

$$w(i) = i^{\alpha}$$

where α is a dynamic control parameter that evolves over time. Initially set to α_{init}, the value of α decreases by $\frac{\alpha_{init}}{20}$ every 18 s according to:

$$\alpha = \alpha_{init} - (search\_time \div 18) \times \frac{\alpha_{init}}{20}$$

The 18-s interval is designed to partition the total 360-s search budget into 20 equal phases, providing a structured schedule for α adjustment. The $\frac{\alpha_{init}}{20}$ decrement ensures that α decreases progressively from α_{init} to near zero over the course of the search, gradually shifting the search mode from exploitation-dominant to exploration-dominant. The selection probability for each set is then computed using roulette-wheel sampling:

$$\Gamma(i) = \frac{w(i)}{\sum_{j=1}^{s} w(j)}$$

This design yields two desirable characteristics:

- Vertices with higher remaining connectivity (larger i) are more likely to be selected early in the search.
- As the search proceeds, probability differences between groups gradually diminish, converging to uniform probabilities.

Consequently, DPWS promotes greedy exploitation in the early stages and diversified exploration later on, achieving a balance between convergence speed and solution diversity.

Algorithm 2: Dynamic prioritized weighted vertex selection (DPWS).

Input: s-plex C, dynamic parameter α
Output: expanded vertex v

```
1  for i ← 1 to s do
2      T_i ← ∅;
3  foreach u ∈ AddSet(C) do
4      m ← s − |C \ N(u)|;
5      T_m ← T_m ∪ {u};
6  for i ← 1 to s do
7      Γ(i) ← i^α / Σ_{j=1}^{s} j^α;
8  T ← select a vertex set according to Γ function;
9  v ← select a random vertex in T;
10 return v
```

The pseudocode of the DPWS method is presented in Algorithm 2. The algorithm first partitions vertices based on their admissible missing edges (Lines 3–5), computes their selection probabilities (Lines 6-7), and finally selects a vertex set according to these probabilities, returning a randomly chosen vertex from the selected set.

5. Individual update

Before introducing the individual update mechanism, we first define the concept of a private vertex set within an individual.

Definition 1. Private Vertex Set. Given an individual $S = \{C_1, C_2, \ldots, C_k\}$, for each s-plex $C_i \in S$, the private vertex set of C_i, denoted $priv(C_i, S)$, is defined as the set of vertices that appear exclusively in C_i, i.e.,

$$priv(C_i, S) = C_i \setminus cov(S \setminus C_i).$$

Based on this definition, we employ a greedy replacement strategy for individual updating. Specifically, for the current individual S, the algorithm first generates a new s-plex C' using a tabu search procedure and temporarily incorporates it into S to form a pseudo-individual S' containing $k+1$ s-plexes:

$$S' = S \cup \{C'\}.$$

Next, the algorithm removes the s-plex C'' with the smallest private vertex set, i.e.,

$$C'' = argmin_{C \in S'} |priv(C, S')|.$$

This step ensures that the updated individual maintains exactly k s-plexes while preserving solution feasibility.

The key to this process lies in obtaining high-quality s-plexes via tabu search (He et al., 2024; Zhong et al., 2025). In the work by Wu et al. (2023), a solver for the maximum s-plex problem was directly employed to extract the largest possible s-plex. While this approach ensures large s-plexes in terms of vertex count, it does not necessarily improve the overall objective of the S-DTKSP, as it ignores the influence of the existing s-plexes in the current individual. To better evaluate the contribution of vertices during the s-plex improvement process, we propose a Two-Layer Scoring Function Based on the Neighbor Coverage Density (TLNCD), which allows the tabu search to dynamically assess vertex importance based on both direct and neighborhood-level coverage information.

Definition 2. Vertex Coverage. Given an individual $S = \{C_1, C_2, \ldots, C_k\}$, the vertex coverage of a vertex $v \in V$, denoted $cv(v, S)$, is defined as the number of s-plexes in S that contain v:

$$cv(v, S) = |\{C_i \in S : v \in C_i\}|$$

A smaller $cv(v, S)$ value indicates that v appears less frequently within S. When $cv(v, S) = 0$, the vertex is uncovered, implying that adding an s-plex containing v will directly increase the overall coverage of S. Therefore, in the tabu search procedure, vertex coverage serves as the primary evaluation criterion, and is defined as the first-layer scoring function:

$$score_1(v) = cv(v, S).$$

However, relying solely on $score_1$ is insufficient, since multiple vertices may share identical coverage values. To address this limitation, we introduce a second-layer scoring function based on neighbor coverage density, which considers the coverage characteristics of adjacent vertices.

Definition 3. Neighbor Coverage Density. Given an individual S, the neighbor coverage density of a vertex $v \in V$, denoted $ncv(v, S)$, is defined as the average vertex coverage of its neighbors:

$$ncv(v, S) = \frac{1}{deg_G(v)} \sum_{u \in N(v)} cv(u, S)$$

For isolated vertices with $deg_G(v) = 0$, the NCD value is not used in the search procedure, since such vertices do not participate in neighborhood-based expansion. Accordingly, the second-layer scoring function is defined as:

$$score_2(v) = ncv(v, S)$$

Here, $score_1$ and $score_2$ quantify, respectively, (1) the direct coverage frequency of a vertex and (2) the average coverage frequency of its neighbors. A smaller value of either score indicates a vertex with higher potential to improve the overall coverage of the individual.

Based on the two-layer scoring mechanism, the vertex selection rule during tabu search is defined as follows:

Vertex Selection Rule. When selecting a vertex to add to an s-plex during tabu search, the vertex with the smallest $score_1$ is chosen. In case of a tie, the vertex with the smallest $score_2$ is selected. If the tie still persists, one vertex is selected uniformly at random from the tied candidates.

This hierarchical evaluation ensures that vertex selection not only promotes the inclusion of underrepresented vertices but also encourages balanced neighborhood coverage, ultimately enhancing the diversity and quality of the constructed s-plexes.

6. EA-OI for S-DTKSP problem

Section 3 introduced the general framework of the EA-OI algorithm, which can be adapted to various optimization problems (Gao et al., 2011; Lyu et al., 2022; Pachuau et al., 2023). For the S-DTKSP problem, we designed each component of the algorithm specifically for this context, integrating the individual initialization and update strategies presented in Sections 4 and 5. The resulting algorithm, denoted as EA-OIS, is detailed in Algorithm 3.

Algorithm 3: EA-OIS algorithm.

Input: undirected graph $G = (V, E)$, population size p_{size}, cutoff-time $maxtime$, search depth $maxstep$, parameters k and s
Output: best individual solution found S^*

1 $S^* \leftarrow \emptyset$, $Pop \leftarrow \emptyset$, $S_o \leftarrow \emptyset$;
2 **for** $i \leftarrow 1$ **to** p_{size} **do**
3 $S_i \leftarrow \emptyset$;
4 **for** $j \leftarrow 1$ **to** k **do**
5 **if** $i = 1$ **then**
6 $C \leftarrow$ ConstructSplex$(G, S_i, s, mode1)$;
7 $C \leftarrow$ TabuSearch(G, S_i, C);
8 $S_i \leftarrow S_i \cup \{C\}$, $S_o \leftarrow S_o \cup \{C\}$;
9 **else**
10 $C \leftarrow$ ConstructSplex$(G, S_i, s, mode2)$;
11 $C \leftarrow$ TabuSearch(G, S_i, C);
12 $S_i \leftarrow S_i \cup \{C\}$, $S_o \leftarrow$ UpdateIndividual(S_o, C);
13 $Pop \leftarrow Pop \cup \{S_i\}$;
14 $S^* \leftarrow \underset{S \in \{S^*, S_i, S_o\}}{\arg\max} |cov(S)|$;
15 **while** $time < maxtime$ **do**
16 select S_1, S_2 randomly from Pop;
17 $S_{new} \leftarrow$ Crossover(S_1, S_2);
18 **for** $step \leftarrow 0$ **to** $maxstep$ **do**
19 $C \leftarrow$ ConstructSplex$(G, S_{new}, s, mode2)$;
20 $C \leftarrow$ TabuSearch(G, S_{new}, C);
21 $S_{new} \leftarrow$ UpdateIndividual(S_{new}, C);
22 $S_o \leftarrow$ UpdateIndividual(S_o, C);
23 $S^* \leftarrow \underset{S \in \{S^*, S_{new}, S_o\}}{\arg\max} |cov(S)|$;
24 $S_{\min} \leftarrow \underset{S \in \{S_1, S_2\}}{\arg\min} |cov(S)|$;
25 $Pop \leftarrow Pop \setminus \{S_{\min}\} \cup \{S_{new}\}$;
26 **return** S^*

In Algorithm 3, each individual in the initial population is constructed by mining k s-plexes (Lines 1–14). During this phase, the initial vertices of the s-plexes in the first individual are chosen randomly, while for all subsequent individuals, vertex selection follows the TTVP method

introduced in Section 4.1. The input flags $mode1$ and $mode2$ distinguish these two selection modes.

After initialization, tabu search is applied immediately to improve each s-plex. The observer individual is initialized from the first completed individual (Line 8), which provides a feasible starting solution. Afterwards, it is updated independently using newly generated subgraphs during the search (Line 12). The update mechanism follows the strategy described in Section 5, where the s-plex with the smallest private vertex set ($priv$) is replaced.

During the population search phase (Lines 15–25), the algorithm alternates between crossover and update operations. In each iteration, two individuals (S_1 and S_2) are randomly selected as parents, and a new offspring S_{new} is generated via a merge-and-delete crossover. Specifically, the two parents are merged to form an intermediate solution S' containing $2k$ s-plexes. Then, the k s-plexes with the smallest private vertex sets are removed from S' to produce the offspring S_{new}. This offspring is refined using tabu search and subsequently used to update both the current individual and the observer individual (Lines 19–21). Finally, the algorithm retains the best solution among all individuals in the population and the observer. Here, the outer loop is controlled by the global cutoff time, whereas *maxstep* limits the number of local search iterations applied to each current individual.

Algorithm 4: TabuSearch function.

Input: undirected graph $G = (V, E)$, individual S, s-plex C
Output: improved s-plex C^*

1 $C^* \leftarrow C$, $tabu\_list = \emptyset$;
2 **for** $step \leftarrow 1$ **to** 1000 **do**
3 **if** $AddSet(C) \neq \emptyset$ **then**
4 $v \leftarrow$ select a vertex from $AddSet(C)$ by **Vertex Selection Rule**;
5 $C \leftarrow C \cup \{v\}$;
6 $C^* \leftarrow C$;
7 **else if** $SwapSet(C) \setminus tabu\_list \neq \emptyset$ **then**
8 $v \leftarrow$ select a vertex from $SwapSet(C) \setminus tabu\_list$ by **Vertex Selection Rule**;
9 $u \leftarrow$ the corresponding swap vertex of v in C;
10 $C \leftarrow C \cup \{v\} \setminus \{u\}$;
11 add u into $tabu\_list$ with tabu tenure $10 + |SwapSet(C) \setminus tabu\_list|$;
12 **else**
13 break;
14 **return** C^*

The TabuSearch function aims to enhance the quality of each s-plex through localized neighborhood exploration (Algorithm 4), which includes two basic operations:

- **Add** — directly inserts a new vertex into the current s-plex.
- **Swap** — replaces an existing vertex in the s-plex with one outside it.

The candidate sets $AddSet$ and $SwapSet$ are maintained dynamically where:

$$AddSet(C) = \{v \in N(C) \mid |N(v) \cap C| \geq |C| + 1 - s, D(C) \setminus N(v) = \emptyset\};$$

$$SwapSet(C) = \{v \in N(C) \mid |N(v) \cap C| \geq |C| - s, |D(C) \setminus N(v)| = 1\} \cup \{v \in N(C) \mid |N(v) \cap C| = |C| - s, |D(C) \setminus N(v)| = 0\}.$$

Since the quality of a single s-plex does not solely determine the overall solution quality, the search depth is fixed as a moderate limit of 1000 iterations. At each iteration, the algorithm first attempts an add operation (Lines 2–6). If no vertex can be added, it performs a swap operation (Lines 7–11). If neither is possible, the search terminates early. Vertices added to the s-plex are unrestricted by tabu constraints, while vertices removed during swap operations are inserted into the tabu list with dynamically adjusted tenure $10 + |SwapSet(C) \setminus tabu\_list|$. This value combines a fixed base of 10 with an adaptive component. The base value prevents short-term cycling and is consistent with prior s-plex local search practice (Pullan, 2021). The adaptive term scales tenure with the available swap space: a larger effective swap set allows a longer tenure without over-restricting search, whereas a shrinking swap set triggers shorter tenure to preserve flexibility. Moreover, if a swap move yields an s-plex strictly larger than C^*, the tabu restriction on the corresponding vertex is overridden regardless of its tabu status.

Complexity Analysis. The computational cost of EA-OIS is mainly dominated by the TabuSearch procedure. Let $n = |V|$, let Δ_G denote the maximum degree of G, and let ℓ denote the maximum size of an s-plex maintained during the search. By Lemma 1, $\ell \leq \Delta_G + s$, indicating that the effect of s mainly enters through the size bound of the maintained s-plexes. In each iteration of TabuSearch, the candidate sets AddSet(C) and SwapSet(C) are maintained by scanning the candidate neighborhood $N(C)$, and the feasibility check for each candidate vertex can be bounded by $O(\Delta_G)$. Therefore, one iteration of TabuSearch requires $O(|N(C)| \cdot \Delta_G)$ time, giving a worst-case bound of $O(n \cdot \Delta_G)$ since $|N(C)| \leq n$. In practical large-scale graphs, $|N(C)|$ is typically much smaller than the full vertex set, so the observed runtime is substantially better than this worst-case estimate. By contrast, UpdateIndividual inserts a newly generated s-plex and removes the one with the smallest private vertex set among the $k + 1$ candidates; the vertex coverage values are maintained incrementally, and this procedure can be completed in $O(k\ell)$, equivalently $O(k(\Delta_G + s))$. Overall, TabuSearch is the main computational bottleneck, whereas UpdateIndividual contributes a moderate linear cost in k. Since the observer individual is maintained by the same UpdateIndividual procedure and does not participate in selection or crossover, it only introduces a minor overhead; empirically, observer-related operations account for 2.26% of the total runtime on average, measured across all 2224 instances.

7. Experimental evaluation

In this section, we first describe the benchmark instances used in our experiments, the four state-of-the-art competitors for the S-DTKSP problem, and the experimental setup, then present extensive experiments to evaluate the performance and robustness of our proposed EA-OIS algorithm across diverse graph types. Finally, we conduct further analyses to assess the individual contributions of the key components in EA-OIS, examining their impact on solution quality, computational efficiency, and stability.

7.1. The benchmark

For our experiments, we utilized 139 real-world graphs sourced from the work of Wu et al. (2023), which were previously used to validate algorithmic performance on the S-DTKSP problem. These graphs represent various network types, including biological networks, collaboration networks, Facebook networks, interaction networks, infrastructure networks, Amazon recommendation networks, scientific computation networks, social networks, technological networks, and web networks. Many of these graphs are large, containing over 1,000,000 vertices and 10,000,000 edges, posing significant challenges for the algorithms. For each graph, we used the same problem parameter settings as in Wu et al. (2023). Specifically, the parameter k took values from the set $\{10, 20, 30, 40\}$, and the parameter s took values from $\{2, 3, 4, 5\}$. This resulted in the generation of $139 \times 4 \times 4 = 2224$ distinct instances of the S-DTKSP problem for our experiments.

7.2. Experimental setup

We compare our EA-OIS algorithm against four existing algorithms for solving the S-DTKSP problem, including: a linear integer program-

Table 1
Parameters used in EA-OIS and their final values.

Parameters	Description	Range	Final values
p_1	Probability of selecting vertices from $\mathcal{V}_1$	{0.00, 0.05, 0.10, 0.15}	0.10
p_2	Probability of selecting vertices from $\mathcal{V}_2$	{0.10, 0.15, 0.20, 0.25}	0.20
p_3	Probability of selecting vertices from $\mathcal{V}_3$	$1 - p_1 - p_2$	0.70
α_{init}	Initial exponent coefficient	{3, 5, 7, 9}	5
p_{size}	Population size	{5, 10, 15, 20}	10

ming algorithm Cplex (using the formulations in equations (1)-(5)), two approximation algorithms EnumFast and EnumMax, and a local search algorithm ILS (Wu et al., 2023). Recent population-based methods for diversified Top-k clique-related problems also indicate the promise of evolutionary search for diversified subgraph optimization (Wu et al., 2022; Xue et al., 2024; Zheng et al., 2024). Since these methods are developed for clique-based structures, their direct adaptation to S-DTKSP is nontrivial; therefore, the present comparison focuses on algorithms specifically designed for S-DTKSP.

Among these, ILS dominates in performance across the majority of test cases. For the comparison, Cplex is implemented using the commercial solver CPLEX (version 22.1.1), while EnumFast, EnumMax, ILS, and our proposed EA-OIS are implemented in C++ and compiled with g++ using the -O3 optimization flag. All experiments were run on a Centos 7.9 operation system on a server equipped with dual AMD EPYC 9654 processors (2.40 GHz, 192 cores) and 755 GB of DDR5-4800 memory. The parameter setting for the comparison algorithms were consistent with the original papers. Cplex, EnumFast, and EnumMax are deterministic algorithms and do not rely on random seeds. Each of these algorithms is executed once per instance, with a cutoff time of 3600 s. In contrast, ILS and EA-OIS are heuristic algorithms whose performance depends on random seeds. For these two algorithms, each instance is solved 10 times independently with random seeds from 1 to 10, and each run is limited to 360 s. This protocol ensures a fair total computation budget.

Our EA-OIS algorithm has five critical parameters: the selection probabilities p_1, p_2, and p_3 for the vertex sets in the TTVP strategy; the initial exponent coefficient α_{init} for the DPWS strategy; the population size p_{size}. The value of p_3 is computed as $1 - p_1 - p_2$. For the remaining parameters, we use the automatic tuning tool irace (López-Ibáñez et al., 2016) to optimize them. Specifically, 10% of the 2224 instances (i.e., 222 instances) were randomly selected as the test set, and a total of 20,000 runs were performed, with each run limited to 360 s. The final results from irace are shown in Table 1. A detailed sensitivity analysis is provided in Section 7.5.

7.3. Comparisons with existing algorithms for the S-DTKSP

7.3.1. Summarizing the comparative results

Table 2 summarizes the performance of all algorithms on 139 graphs. The values reported in the table indicate the number of instances (out of 139) for which a given algorithm achieves the best solution, while the values in parentheses denote the number of instances where the algorithm obtains a solution that is strictly better than those produced by all other algorithms. It can be observed that, for every combination of the problem parameters (s, k), the proposed EA-OIS algorithm demonstrates stable and highly efficient performance. Under the shared experimental environment described above, EA-OIS achieves strictly better $sbest$ values than all competing algorithms on 1284 instances — referred to hereafter as new best-known results under these controlled conditions — and matches or exceeds all competitors on 2203 of 2224 instances in total.

When the problem parameters s and k are relatively small, the competing algorithms are also able to deliver satisfactory performance. In particular, the ILS algorithm attains the best solutions for most instances under these settings. However, as the problem parameters increase and the problem difficulty becomes more challenging, the performance of ILS degrades significantly. The remaining algorithms, such as Cplex as well as EnumFast and EnumMax, are constrained by their respective solution strategies and are able to achieve the best results only on a very limited number of instances.

Table 2
Summary of comparison between EA-OIS and its competitors.

s	k	Cplex	EnumFast	EnumMax	ILS	EA-OIS
2	10	8(1)	14(0)	32(0)	93(1)	**137(36)**
2	20	4(0)	15(1)	26(0)	78(1)	**137(53)**
2	30	6(0)	15(0)	24(0)	67(1)	**138(63)**
2	40	6(0)	15(0)	22(0)	55(0)	**139(75)**
3	10	7(0)	1(0)	7(0)	72(0)	**139(63)**
3	20	6(0)	2(0)	4(0)	55(0)	**139(80)**
3	30	6(0)	3(0)	4(0)	41(1)	**138(94)**
3	40	7(0)	4(0)	4(0)	36(2)	**137(98)**
4	10	8(2)	0(0)	3(0)	73(1)	**136(63)**
4	20	7(1)	2(0)	2(0)	51(1)	**137(85)**
4	30	7(0)	2(0)	2(0)	45(1)	**138(92)**
4	40	7(0)	4(0)	2(0)	39(1)	**138(98)**
5	10	7(0)	0(0)	3(1)	52(1)	**137(84)**
5	20	7(0)	3(0)	4(0)	42(1)	**138(96)**
5	30	9(1)	4(0)	4(0)	37(1)	**137(100)**
5	40	8(0)	6(0)	3(0)	34(1)	**138(104)**

7.3.2. Experimental results on 48 representative graphs

Among the 139 graphs, 48 representative graphs were selected, and the detailed solution results obtained by all algorithms are presented in Tables 3–6. The selection criterion was based on the solvability of the instances by the Cplex solver — that is, only graphs for which Cplex successfully produced solutions under at least one combination of parameters s and k were included (Wu et al., 2023).

Given the similarity between the EnumFast and EnumMax algorithms, and the fact that both exhibit competitive performance in only a small number of cases, we report the superior of the two under the unified label "Enum" for conciseness. For both Cplex and Enum, the obtained solutions are denoted as "sol". Since ILS and EA-OIS are stochastic heuristics, each instance was independently executed 10 times with distinct random seeds. For these two algorithms, the best solution among the 10 runs is denoted as "$sbest$", the average solution value is denoted by "$savg$", and the average computational time required to obtain the best solution is denoted by "$tavg$". Instance where an algorithm failed to produce any feasible solution within the cutoff time are marked with "-".

The proposed EA-OIS algorithm achieves the best performance on almost all instances, with only a few exceptions. Specifically, when $s = 2$ and $k = 10$, Cplex obtains the best solution on *web-polblogs*, and when $k = 20$, the Enum algorithm achieves the best solution on *rt-retweet*. When $s = 4$ and $k = 10$, Cplex produces the best solution on *ia-infect-hyper* and *scc_fb-forum*, and when $k = 20$, Cplex again achieves the best solution on *rt-retweet*. In addition, when $s = 5$ and $k = 30$, Cplex obtains the best solution on *scc_fb-forum*. On the other small instances where Cplex can verify optimal or near-optimal solutions, EA-OIS achieves competitive results, while its main advantage becomes more evident on larger and harder instances where exact methods are no longer practical. When comparing the two heuristic algorithms, ILS and EA-OIS, it can be observed that EA-OIS significantly outperforms ILS in terms of both the best solution value and the average solution value. Even for those instances where both algorithms achieve identical solution quality (i.e., the same $sbest$ and $savg$), EA-OIS still requires less computational time.

Fig. 3 further presents a detailed comparison of the runtime between EA-OIS and the competing algorithms. Each point in the figure corresponds to one graph instance for which the two algorithms obtain the same solution quality. It can be clearly seen that EA-OIS requires sub-

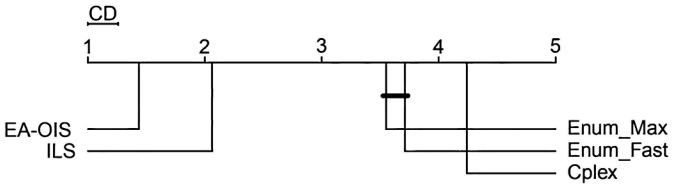


Fig. 4. Critical difference plot for $s = 2$.

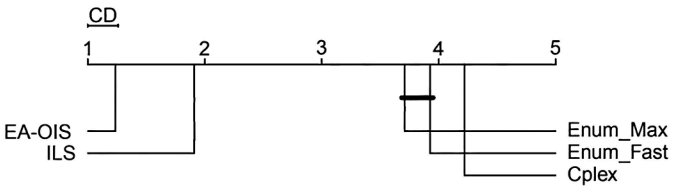


Fig. 5. Critical difference plot for $s = 3$.

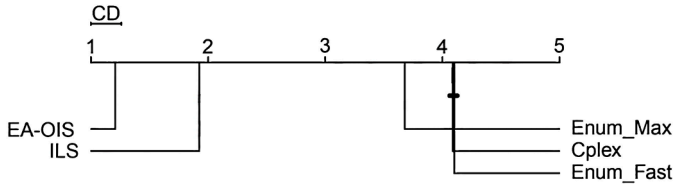


Fig. 6. Critical difference plot for $s = 4$.

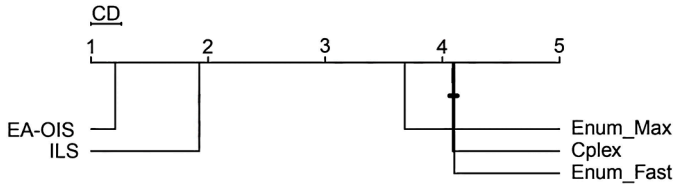


Fig. 7. Critical difference plot for $s = 5$.

stantially less computational time than the competing algorithms on the vast majority of instances.

A closer examination of the results reveals that EA-OIS's advantages are most pronounced under three conditions. First, on large and dense graphs (e.g., *scc_fb-messages*, *scc_reality*), ILS frequently stagnates due to limited diversification, whereas the observer individual in EA-OIS continuously accumulates promising subgraphs across the entire search, mitigating premature convergence. Second, the performance gap widens as s and k increase: larger parameter values expand the solution space significantly, amplifying the benefit of EA-OIS's dual-mode search. For instance, under $s = 5$, $k = 40$, EA-OIS establishes new best-known results on over 104 out of 139 graphs (Table 2). Third, on structurally irregular sparse graphs (e.g., *inf_power*, *tech_routers_rf*), TTVP's degree-based vertex prioritization provides a more effective initialization than random or purely greedy methods. By contrast, on small or near-complete graphs (e.g., *soc_karate*), all algorithms readily reach the global optimum, leaving little room for differentiation.

7.3.3. Critical difference analysis

We evaluate the statistical differences among all algorithms using a two-stage non-parametric procedure. The ranked metric is the best solution value ($sbest$) obtained per instance. Each test unit is one instance, defined as a unique triple (graph, k, s); for ILS and EA-OIS the best result over 10 independent runs is used, while Cplex, EnumFast, and EnumMax contribute their single deterministic result. This yields 556 test units per value of s (139 graphs × 4 values of k). A Friedman test (Friedman, 1937) is first applied with the null hypothesis that all five algorithms perform equivalently in terms of average rank. The null hypothesis is rejected in all four cases ($s = 2, 3, 4, 5$) at significance level $\alpha = 0.05$, justifying pairwise post-hoc analysis. All pairwise comparisons are then performed using the Nemenyi post-hoc test (Garcia & Herrera, 2008), whose results are presented as critical difference (CD) diagrams in Figs. 4–7. In each diagram, the horizontal axis shows average ranks (lower is better); two algorithms are connected by a bar if and only if their rank difference does not exceed the critical difference threshold, i.e., they are not significantly different at $\alpha = 0.05$. In all four diagrams, EA-OIS achieves the lowest average rank and is not connected to any competitor, confirming that EA-OIS is statistically superior to all baselines across every value of s.

7.4. Ablation study

The EA-OIS algorithm integrates four novel strategies, namely the Evolutionary Algorithm with an Observer Individual, Tri-Tier Vertex Prioritization, Dynamic Prioritized Weighted Selection, and Two-Layer Scoring Function Based on Neighbor Coverage Density. To evaluate the contribution of each strategy, we performed an ablation study by successively removing one component from EA-OIS, resulting in five variants: EA-OIS1 through EA-OIS5. The configurations are described as follows:

- **EA-OIS1:** Removes the observer individual, adopting only the conventional evolutionary framework.
- **EA-OIS2:** Removes population maintenance and retains only the observer individual, thereby maintaining a single solution throughout the search. This variant is structurally analogous to a single-solution iterative search.
- **EA-OIS3:** Replaces the Tri-Tier Vertex Prioritization strategy with random vertex initialization.
- **EA-OIS4:** Replaces the Dynamic Prioritized Weighted Selection with random vertex selection.
- **EA-OIS5:** Replaces the Two-Layer Scoring Based on Neighbor Coverage Density with the scoring mechanism from the ILS algorithm.

To provide a more informative understanding of the contribution of each component, we complement the win/lose statistics in Table 7 with average degradation analysis and scale-aware evaluation. Specifically, Table 7 reports, for each EA-OIS variant, the numbers of instances on which EA-OIS performs better or worse than that variant in terms of both the best solution value ($sbest$) and the average solution value ($savg$). While these counts reflect the frequency of superiority, they do not quantify the magnitude of performance differences. Therefore, Table 8 further reports the average relative performance loss (ARPL) of each variant with respect to EA-OIS over all 2224 instances, computed based on $savg$ as follows:

$$\mathrm{ARPL} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{savg^{\text{EA-OIS}}(i) - savg^{\text{variant}}(i)}{savg^{\text{EA-OIS}}(i)} \times 100\%$$

where $\mathcal{I}$ denotes the set of all test instances. The component-wise analysis is presented below.

Dual-mode framework (EA-OIS1 and EA-OIS2). Removing the observer individual (EA-OIS1) or reducing the algorithm to an observer-only single-solution search (EA-OIS2) leads to consistent performance degradation, with overall ARPL values of 0.2526% and 0.2041%, respectively. For both variants, the loss is much more pronounced on large graphs (0.6550% and 0.5723%) than on small or medium graphs, indicating that the collaboration between the evolving population and the observer becomes increasingly important as graph scale grows.

Initialization strategies (EA-OIS3 and EA-OIS4). Replacing TTVP with random initialization (EA-OIS3) yields an overall ARPL of 0.3266%, while replacing DPWS with uniform random selection (EA-OIS4) causes the largest relative degradation among all ablated variants, with an ARPL of 0.4422%. Their impact is especially evident on large graphs, where the ARPL rises to 0.9340% for EA-OIS3 and 1.1277% for

Table 3
Experimental results of EA-OIS and its comparison algorithms for $s = 2$.

Instance	$k = 10$						$k = 20$						$k = 30$						$k = 40$					
	CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS	
	sol	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*
bio-celegans	57	57	**59**(59)	2.59	**59**(59)	<0.01	101	100	99(99)	1.63	**106**(106)	75.91	130	123	130(130)	3.00	**144**(143.9)	244.36	162	151	154(154)	9.95	**175**(175)	187.08
bio-diseasome	76	**78**	**78**(78)	0.09	**78**(78)	<0.01	127	**133**	**133**(133)	0.10	**133**(133)	<0.01	174	181	**182**(182)	0.15	**182**(182)	<0.01	214	221	**222**(222)	0.14	**222**(222)	<0.01
bio-yeast	33	42	**46**(46)	0.09	**46**(46)	<0.01	64	71	**86**(86)	0.07	**86**(86)	<0.01	90	101	**126**(126)	0.06	**126**(126)	<0.01	111	132	**166**(166)	0.13	**166**(166)	<0.01
ca-CSphd	29	**39**	**39**(39)	0.33	**39**(39)	0.02	54	**69**	**69**(69)	0.28	**69**(69)	0.02	78	**99**	**99**(99)	0.30	**99**(99)	0.01	106	**129**	**129**(129)	0.29	**129**(129)	0.02
ca-Erdos992	–	**59**	**59**(59)	1.69	**59**(59)	<0.01	–	108	**109**(109)	0.74	**109**(109)	<0.01	–	148	151(151)	1.23	**152**(152)	3.38	–	188	191(191)	1.43	**192**(192)	2.48
ca-GrQc	62	**246**	**246**(246)	0.33	**246**(246)	<0.01	82	**349**	**349**(349)	0.44	**349**(349)	<0.01	–	**424**	**424**(424)	0.47	**424**(424)	<0.01	–	486	**493**(493)	0.69	**493**(493)	0.01
ca-netscience	68	**69**	**69**(69)	0.04	**69**(69)	<0.01	114	**119**	**119**(119)	0.05	**119**(119)	<0.01	155	140	**161**(161)	0.09	**161**(161)	<0.01	188	167	**200**(200)	0.28	**200**(200)	<0.01
ia-Email-univ	54	**86**	**86**(86)	0.07	**86**(86)	<0.01	95	145	**148**(148)	0.15	**148**(148)	<0.01	119	178	207(207)	0.78	**208**(208)	0.02	149	221	258(258)	2.93	**259**(259)	0.03
ia-enron-only	61	57	61(61)	0.64	**62**(62)	<0.01	97	82	96(96)	4.01	**101**(101)	<0.01	125	103	112(112)	9.58	**128**(127.2)	342.76	141	120	112(112)	1.02	**143**(143)	0.09
ia-fb-messages	41	55	**56**(56)	0.31	**56**(56)	<0.01	75	101	**106**(106)	1.15	**106**(106)	0.01	99	131	152(151.4)	134.00	**154**(154)	0.07	135	158	192(191.9)	150.93	**196**(195.9)	70.71
ia-infect-dublin	111	121	**122**(122)	0.13	**122**(122)	<0.01	175	187	**189**(189)	0.11	**189**(189)	<0.01	235	232	237(237)	0.63	**242**(241.3)	186.57	273	233	274(274)	4.99	**283**(283)	8.16
ia-infect-hyper	81	68	72(71.8)	129.38	**84**(82.6)	216.06	112	99	86(86)	84.08	**113**(112.9)	248.27	**113**	109	86(86)	78.34	**113**(113)	0.04	**113**	110	86(86)	79.56	**113**(113)	0.04
inf-power	24	**52**	**52**(52)	0.56	**52**(52)	<0.01	47	**92**	**92**(92)	0.58	**92**(92)	<0.01	–	**132**	**132**(132)	0.56	**132**(132)	<0.01	–	**172**	**172**(172)	0.58	**172**(172)	<0.01
rt-retweet	31	**33**	32(32)	0.07	**33**(33)	<0.01	60	**82**	52(52)	0.37	61(61)	<0.01	**81**	73	54(54)	0.26	**81**(81)	<0.01	**96**	81	54(54)	0.27	**96**(96)	<0.01
rt-twitter-copen	39	37	**44**(44)	0.04	**44**(44)	<0.01	68	65	**83**(83)	2.40	**83**(83)	<0.01	99	93	**114**(114)	7.82	**114**(114)	0.04	123	126	**144**(144)	1.97	**144**(144)	<0.01
scc_enron-only	**151**	145	136(132.5)	360.01	**151**(151)	0.02	**151**	146	141(133.4)	360.01	**151**(151)	0.04	**151**	149	136(133.1)	360.01	**151**(151)	0.07	**151**	148	135(132.4)	360.01	**151**(151)	0.06
scc_fb-forum	**382**	378	364(363.4)	59.56	**382**(381.1)	268.15	401	398	384(383.4)	59.74	**402**(401.6)	223.61	420	418	404(403.4)	61.39	**422**(421.7)	167.39	434	438	424(423.4)	62.46	**442**(441.6)	254.41
scc_fb-messages	976	1024	978(978)	23.05	**1028**(1027)	220.83	1039	1065	998(998)	23.77	**1072**(1071.7)	197.89	959	1087	1018(1018)	47.67	**1095**(1093.2)	183.67	1063	1107	1038(1038)	69.67	**1115**(1113.3)	358.29
scc_infect-hyper	**113**	**113**	107(107)	0.01	**113**(113)	<0.01	**113**	**113**	109(107.4)	360.01	**113**(113)	0.01	**113**	**113**	109(107.2)	360.01	**113**(113)	0.02	**113**	**113**	108(107.1)	360.01	**113**(113)	0.03
scc_reality	–	50	1872(1867.4)	356.94	**1885**(1883.9)	288.68	–	99	1964(1949.3)	318.81	**2004**(2003.6)	176.84	–	123	2005(1984.6)	346.08	**2086**(2085.2)	331.20	–	156	2027(2005.8)	349.99	**2152**(2151.1)	255.38
scc_retweet	–	20	398(398)	57.24	**409**(409)	81.04	–	40	427(425.6)	179.93	**461**(459.7)	301.12	–	60	446(445.2)	143.87	**497**(494.5)	327.54	–	80	466(465.3)	178.41	**526**(524.7)	306.98
scc_rt_alwefaq	–	**42**	39(39)	16.07	**42**(42)	0.40	–	62	59(59)	15.99	**63**(62.1)	102.29	–	82	79(79)	16.44	**83**(82.8)	147.88	–	102	99(99)	16.27	**103**(103)	0.43
scc_rt_assad	20	**29**	28(28)	0.07	**29**(29)	<0.01	40	**49**	48(48)	0.06	**49**(49)	<0.01	60	**69**	68(68)	0.07	**69**(69)	0.02	80	**89**	88(88)	0.06	**89**(89)	<0.01
scc_rt_bahrain	–	31	31(31)	0.72	**33**(33)	0.02	–	51	51(51)	0.69	**53**(53)	<0.01	–	71	71(71)	0.73	**73**(73)	0.01	–	91	91(91)	0.72	**93**(93)	<0.01
scc_rt_damascus	20	29	31(31)	0.78	**32**(32)	<0.01	–	49	51(51)	0.79	**52**(52)	<0.01	–	69	71(71)	0.80	**72**(72)	<0.01	–	89	91(91)	0.80	**92**(92)	<0.01
scc_rt_dash	–	**27**	26(26)	0.26	**27**(27)	0.27	–	**47**	46(46)	0.29	**47**(47)	0.09	–	**67**	66(66)	0.26	**67**(67)	0.04	–	**87**	86(86)	0.26	**87**(87)	0.02
scc_rt_gop	–	**21**	**21**(21)	0.18	**21**(21)	<0.01	–	**41**	**41**(41)	0.18	**41**(41)	<0.01	–	**61**	**61**(61)	0.19	**61**(61)	<0.01	–	**81**	**81**(81)	0.20	**81**(81)	<0.01
scc_rt_http	–	**22**	**22**(22)	0.30	**22**(22)	<0.01	–	**42**	**42**(42)	0.27	**42**(42)	<0.01	–	**62**	**62**(62)	0.29	**62**(62)	<0.01	–	**82**	**82**(82)	0.32	**82**(82)	<0.01
scc_rt_israel	–	**21**	**21**(21)	0.17	**21**(21)	<0.01	–	**41**	**41**(41)	0.17	**41**(41)	<0.01	–	**61**	**61**(61)	0.19	**61**(61)	<0.01	–	**81**	**81**(81)	0.18	**81**(81)	<0.01
scc_rt_ksa	–	**25**	**25**(25)	0.33	**25**(25)	<0.01	–	**45**	**45**(45)	0.32	**45**(45)	<0.01	–	**65**	**65**(65)	0.31	**65**(65)	<0.01	–	**85**	**85**(85)	0.33	**85**(85)	<0.01
scc_rt_lebanon	–	**20**	**20**(20)	<0.01	**20**(20)	<0.01	–	**40**	**40**(40)	<0.01	**40**(40)	<0.01	–	**60**	**60**(60)	0.01	**60**(60)	<0.01	–	**80**	**80**(80)	0.02	**80**(80)	<0.01
scc_rt_libya	–	**23**	**23**(23)	0.16	**23**(23)	<0.01	–	**43**	**43**(43)	0.19	**43**(43)	<0.01	–	**63**	**63**(63)	0.19	**63**(63)	<0.01	–	**83**	**83**(83)	0.15	**83**(83)	<0.01
scc_rt_obama	**20**	**20**	**20**(20)	<0.01	**20**(20)	<0.01	–	**40**	**40**(40)	<0.01	**40**(40)	<0.01	–	**60**	**60**(60)	0.01	**60**(60)	<0.01	–	**80**	**80**(80)	0.01	**80**(80)	<0.01
scc_rt_occupy	20	31	31(31)	0.91	**32**(32)	0.01	–	50	51(51)	0.88	**52**(52)	0.01	–	70	71(71)	0.89	**72**(72)	0.01	–	90	91(91)	0.86	**92**(92)	0.06
scc_rt_occupywallstnyc	–	54	54(54)	8.40	**55**(55)	1.14	–	74	74(74)	8.31	**75**(75)	0.48	–	94	94(94)	8.46	**95**(95)	0.11	–	114	114(114)	8.30	**115**(115)	0.12
scc_rt_oman	–	**23**	**23**(23)	0.22	**23**(23)	<0.01	–	**43**	**43**(43)	0.20	**43**(43)	<0.01	–	**63**	**63**(63)	0.23	**63**(63)	<0.01	–	**83**	**83**(83)	0.21	**83**(83)	<0.01
scc_rt_p2	–	**22**	**22**(22)	0.39	**22**(22)	<0.01	–	**42**	**42**(42)	0.39	**42**(42)	<0.01	–	**62**	**62**(62)	0.40	**62**(62)	<0.01	–	**82**	**82**(82)	0.39	**82**(82)	<0.01
scc_rt_tcot	–	**23**	**23**(23)	0.20	**23**(23)	<0.01	–	**43**	**43**(43)	0.20	**43**(43)	<0.01	–	**63**	**63**(63)	0.22	**63**(63)	<0.01	–	**83**	**83**(83)	0.19	**83**(83)	<0.01
scc_rt_tlot	–	**21**	**21**(21)	0.13	**21**(21)	<0.01	–	**41**	**41**(41)	0.13	**41**(41)	<0.01	–	**61**	**61**(61)	0.13	**61**(61)	<0.01	–	**81**	**81**(81)	0.11	**81**(81)	<0.01
scc_rt_uae	–	**23**	**23**(23)	0.59	**23**(23)	<0.01	–	**43**	**43**(43)	0.56	**43**(43)	<0.01	–	**63**	**63**(63)	0.58	**63**(63)	<0.01	–	**83**	**83**(83)	0.55	**83**(83)	<0.01
scc_rt_voteonedirection	**21**	**21**	**21**(21)	0.09	**21**(21)	<0.01	40	**41**	**41**(41)	0.11	**41**(41)	<0.01	60	**61**	**61**(61)	0.08	**61**(61)	<0.01	80	**81**	**81**(81)	0.09	**81**(81)	<0.01
soc-karate	**30**	**30**	**30**(30)	1.06	**30**(30)	<0.01	**34**	32	32(31.5)	122.48	**34**(34)	<0.01	**34**	**34**	32(31.5)	124.83	**34**(34)	<0.01	**34**	**34**	32(31.5)	122.37	**34**(34)	<0.01
soc-wiki-Vote	52	60	63(63)	3.51	**64**(64)	0.02	91	97	111(111)	64.15	**114**(114)	4.02	115	134	151(151)	75.50	**155**(155)	5.68	145	160	191(191)	34.49	**195**(195)	1.80
tech-routers-rf	61	**104**	**104**(104)	0.07	**104**(104)	<0.01	42	155	164(164)	16.64	**166**(166)	0.55	89	205	214(214)	15.77	**217**(217)	0.34	100	220	264(264)	12.60	**267**(267)	1.06
web-edu	60	**147**	**147**(147)	0.26	**147**(147)	<0.01	90	**189**	**189**(189)	0.39	**189**(189)	<0.01	110	**229**	**229**(229)	0.35	**229**(229)	<0.01	130	**269**	**269**(269)	0.36	**269**(269)	<0.01
web-google	86	**135**	**135**(135)	0.09	**135**(135)	<0.01	143	**215**	**215**(215)	0.11	**215**(215)	<0.01	175	**271**	**271**(271)	0.19	**271**(271)	<0.01	224	314	315(315)	0.78	**316**(316)	0.01
web-polblogs	**70**	58	66(66)	2.50	68(67.2)	257.61	93	99	108(108)	9.66	**113**(113)	3.35	135	120	146(145.4)	226.45	**153**(153)	0.77	166	149	177(175.8)	176.40	**192**(191.8)	238.84
web-spam	21	121	142(142)	8.01	**143**(142.1)	188.21	46	206	238(238)	24.91	**241**(240.9)	113.59	–	272	325(325)	73.33	**328**(328)	25.57	–	364	402(401.1)	54.72	**408**(408)	19.19

Table 4
Experimental results of EA-OIS and its comparison algorithms for $s = 3$.

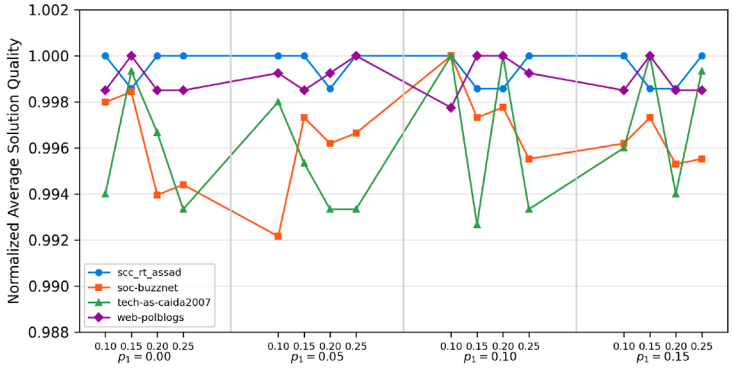
Instance	$k = 10$						$k = 20$						$k = 30$						$k = 40$					
	CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS	
	sol	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*
bio-celegans	65	61	71(70.8)	85.76	**72**(72)	14.48	111	106	124(124)	44.30	**125**(125)	0.20	154	149	130(130)	224.31	**175**(175)	105.81	198	181	214(213.3)	148.53	**221**(220.1)	332.58
bio-diseasome	73	**78**	**78**(78)	0.13	**78**(78)	<0.01	128	129	**134**(134)	0.19	**134**(134)	<0.01	179	144	**182**(182)	0.23	**184**(184)	<0.01	215	193	**234**(234)	0.20	**234**(234)	<0.01
bio-yeast	35	40	**57**(57)	0.21	**57**(57)	<0.01	78	80	**107**(107)	0.19	**107**(107)	<0.01	108	120	**126**(126)	0.63	**157**(157)	0.01	132	160	**200**(200)	3.89	**200**(200)	0.03
ca-CSphd	40	40	**42**(42)	0.07	**42**(42)	<0.01	80	80	**82**(82)	0.08	**82**(82)	<0.01	118	120	**99**(99)	0.07	**122**(122)	<0.01	156	160	**162**(162)	0.09	**162**(162)	<0.01
ca-Erdos992	–	25	**67**(67)	0.64	**67**(67)	<0.01	–	45	**127**(127)	14.70	**127**(127)	0.10	–	65	151(151)	149.87	**181**(180.6)	148.93	–	85	229(229)	74.23	**231**(231)	14.88
ca-GrQc	30	40	**247**(247)	0.36	**247**(247)	<0.01	60	64	**350**(350)	1.22	**350**(350)	0.01	–	86	**424**(424)	2.17	**428**(428)	0.03	–	104	497(497)	2.23	**498**(498)	0.04
ca-netscience	70	**71**	**71**(71)	0.12	**71**(71)	<0.01	121	125	**130**(130)	0.14	**130**(130)	<0.01	167	169	**161**(161)	0.25	**180**(180)	<0.01	213	176	**223**(223)	8.41	**223**(223)	0.16
ia-Email-univ	51	**94**	**94**(94)	0.15	**94**(94)	<0.01	100	137	**168**(168)	3.63	**168**(168)	0.01	159	200	207(207)	10.19	**238**(238)	1.03	183	242	295(295)	9.35	**300**(299.3)	239.12
ia-enron-only	74	74	73(73)	0.09	**75**(75)	<0.01	116	94	114(113.4)	130.03	**119**(118.8)	319.38	141	114	112(112)	161.84	**143**(143)	60.92	**143**	124	140(140)	0.92	**143**(143)	<0.01
ia-fb-messages	47	58	**71**(71)	7.10	**71**(71)	0.05	91	115	**132**(131.2)	245.37	**132**(132)	3.32	127	133	152(151.4)	191.54	**191**(191)	12.60	170	176	236(234.3)	220.47	**245**(243.5)	189.52
ia-infect-dublin	124	127	**129**(128.1)	140.77	**129**(129)	<0.01	198	202	**206**(205.1)	144.25	**206**(206)	<0.01	246	258	237(237)	12.53	**265**(265)	0.11	296	298	309(308.5)	183.97	**315**(314.4)	269.91
ia-infect-hyper	85	87	82(81.2)	249.08	**92**(90.6)	280.27	**113**	102	106(105.6)	139.98	**113**(113)	1.43	**113**	112	86(86)	179.18	**113**(113)	0.03	**113**	**113**	111(110.7)	182.29	**113**(113)	0.05
inf-power	33	38	**58**(58)	8.53	**58**(58)	0.04	69	58	**108**(108)	8.93	**108**(108)	0.05	–	78	**132**(132)	11.15	**158**(158)	0.01	–	98	**208**(208)	7.26	**208**(208)	0.03
rt-retweet	**42**	40	**42**(42)	0.03	**42**(42)	<0.01	78	68	76(76)	0.25	**79**(79)	0.08	**96**	86	54(54)	0.93	**96**(96)	<0.01	**96**	91	95(95)	0.29	**96**(96)	<0.01
rt-twitter-copen	45	51	**54**(54)	0.15	**54**(54)	<0.01	85	82	**103**(103)	11.31	**103**(103)	<0.01	120	121	**114**(114)	12.91	**143**(143)	0.45	160	156	**183**(183)	30.05	**183**(183)	0.06
scc_enron-only	**151**	**151**	**151**(150.7)	201.01	**151**(151)	0.05	**151**	**151**	**151**(150.7)	204.01	**151**(151)	0.05	**151**	**151**	136(133.1)	194.52	**151**(151)	0.01	**151**	**151**	**151**(150.7)	197.76	**151**(151)	0.06
scc_fb-forum	**385**	30	374(372.3)	319.31	**385**(384.8)	147.68	**414**	60	404(402.3)	299.37	**414**(413.5)	214.95	433	90	404(403.4)	300.07	**445**(444.8)	126.03	465	120	464(462.3)	308.44	**475**(474.6)	146.03
scc_fb-messages	862	27	1012(1011.6)	184.16	**1033**(1031.5)	340.99	1004	47	1045(1043.9)	264.00	**1081**(1079.6)	280.81	1044	67	1018(1018)	265.75	**1113**(1111.7)	232.33	1097	87	1105(1103.7)	271.51	**1141**(1139.5)	300.70
scc_infect-hyper	**113**	**113**	110(109.3)	360.01	**113**(113)	<0.01	**113**	**113**	110(109.2)	360.01	**113**(113)	0.01	**113**	**113**	109(107.2)	360.01	**113**(113)	0.02	**113**	**113**	110(109.5)	360.01	**113**(113)	0.03
scc_reality	–	22	1855(1836.5)	353.13	**1889**(1889)	65.92	–	42	1930(1906.7)	359.60	**2021**(2020.3)	187.25	–	62	2005(1984.6)	354.69	**2110**(2108.7)	234.22	–	82	1978(1951.9)	348.53	**2180**(2178)	227.99
scc_retweet	–	21	404(401)	326.57	**419**(418.4)	194.30	–	41	438(434.7)	359.93	**470**(469.1)	295.90	–	61	446(445.2)	355.38	**511**(508.9)	289.70	–	81	496(494)	57.51	**544**(541.4)	348.84
scc_rt_alwefaq	–	30	50(49.6)	233.06	**54**(54)	0.23	–	60	80(79.6)	207.03	**84**(84)	8.54	–	79	79(79)	196.84	**114**(114)	3.69	–	100	140(139.4)	203.02	**144**(144)	21.21
scc_rt_assad	36	30	39(38.1)	61.76	**40**(39.8)	253.97	60	60	69(68.1)	62.79	**70**(69.9)	253.30	90	90	68(68)	64.07	**100**(99.9)	199.94	120	120	129(128.1)	62.08	**130**(129.8)	274.95
scc_rt_bahrain	–	30	44(43.1)	110.86	**47**(47)	0.79	–	53	74(73.2)	229.44	**81**(81)	6.13	–	74	71(71)	230.26	**111**(111)	40.51	–	94	135(133.2)	213.58	**141**(141)	6.38
scc_rt_damascus	30	30	37(37)	89.64	**38**(38)	0.72	–	60	67(67)	88.23	**68**(68)	3.03	–	90	71(71)	87.70	**98**(98)	3.22	–	120	**128**(127.1)	259.92	**128**(128)	7.44
scc_rt_dash	–	27	36(35.8)	156.67	**38**(38)	24.15	–	47	66(65.8)	157.96	**68**(68)	4.04	–	67	66(66)	140.04	**98**(98)	4.71	–	87	126(125.8)	111.30	**128**(128)	8.35
scc_rt_gop	–	30	32(31.1)	275.25	**33**(33)	9.29	–	60	62(61.1)	276.48	**63**(63)	1.98	–	88	**61**(61)	271.29	**93**(93)	1.97	–	108	122(121.1)	274.33	**123**(123)	8.13
scc_rt_http	–	28	**31**(31)	0.38	**31**(31)	<0.01	–	48	**61**(61)	0.35	**61**(61)	<0.01	–	68	**62**(62)	0.35	**91**(91)	<0.01	–	88	**121**(121)	0.38	**121**(121)	<0.01
scc_rt_israel	–	30	33(32.8)	112.14	**35**(35)	2.53	–	60	63(62.8)	111.56	**65**(65)	1.83	–	87	**61**(61)	89.87	**95**(95)	2.41	–	107	123(122.8)	110.75	**125**(125)	1.66
scc_rt_ksa	–	28	35(34.1)	99.96	**36**(36)	5.16	–	48	65(64.1)	99.79	**66**(66)	5.36	–	68	**65**(65)	163.67	**96**(96)	5.59	–	88	125(124.1)	100.79	**126**(126)	3.84
scc_rt_lebanon	–	30	**32**(31.1)	106.58	**32**(32)	1.27	–	60	61(61)	28.01	**62**(62)	2.25	–	90	**60**(60)	27.76	**92**(92)	1.30	–	115	121(121)	28.28	**122**(122)	1.72
scc_rt_libya	–	30	34(34)	117.47	**37**(36.3)	37.21	–	51	64(64)	96.18	**67**(66.1)	256.66	–	71	**63**(63)	96.11	**96**(96)	5.51	–	91	124(124)	91.24	**126**(126)	10.34
scc_rt_obama	30	30	31(31)	68.53	**32**(32)	6.27	–	60	61(61)	68.52	**62**(62)	1.24	–	90	**60**(60)	67.90	**92**(92)	4.95	–	97	121(121)	68.27	**122**(122)	1.90
scc_rt_occupy	30	30	40(39.3)	125.83	**43**(43)	2.73	–	60	70(69.3)	125.35	**74**(74)	5.39	–	90	71(71)	131.26	**104**(103.6)	194.60	–	120	130(129.1)	83.90	**134**(134)	44.04
scc_rt_occupywallstnyc	–	30	62(61.4)	220.73	**66**(66)	144.94	–	60	92(91.4)	220.28	**96**(96)	188.63	–	89	94(94)	199.52	**126**(126)	216.77	–	109	152(151.4)	228.42	**156**(156)	160.35
scc_rt_oman	–	30	32(31.9)	98.42	**33**(33)	5.17	–	55	62(62)	123.26	**63**(63)	8.33	–	76	**63**(63)	122.06	**93**(93)	8.48	–	96	122(122)	123.78	**123**(123)	14.27
scc_rt_p2	–	30	35(33.3)	55.49	**36**(36)	1.96	–	52	65(63.3)	55.65	**66**(66)	1.35	–	72	**62**(62)	55.55	**96**(96)	5.39	–	92	125(123.4)	55.77	**126**(126)	11.27
scc_rt_tcot	–	30	34(33.5)	203.06	**36**(36)	5.41	–	54	64(63.4)	167.61	**66**(66)	1.54	–	75	**63**(63)	203.64	**96**(96)	4.84	–	95	124(123.6)	216.35	**126**(126)	9.74
scc_rt_tlot	–	30	32(32)	72.90	**33**(33)	3.76	–	60	62(62)	73.59	**63**(63)	13.04	–	90	**61**(61)	72.67	**93**(93)	2.13	–	112	122(122)	73.11	**123**(123)	6.27
scc_rt_uae	–	30	32(32)	68.87	**34**(34)	31.14	–	53	62(62)	68.97	**64**(64)	20.18	–	73	**63**(63)	68.40	**94**(94)	1.57	–	92	122(122)	68.45	**124**(124)	4.92
scc_rt_voteonedirection	**31**	30	**31**(31)	52.58	**31**(31)	0.36	60	60	**61**(61)	53.93	**61**(61)	1.23	90	90	**61**(61)	51.75	**91**(91)	0.41	120	120	**121**(121)	53.49	**121**(121)	1.32
soc-karate	**34**	**34**	**34**(34)	0.10	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.04	**34**(34)	<0.01	**34**	**34**	32(31.5)	0.03	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.04	**34**(34)	<0.01
soc-wiki-Vote	53	64	**76**(76)	25.51	**76**(76)	0.08	94	115	134(134)	87.73	**136**(136)	7.42	135	153	151(151)	68.26	**188**(187.7)	239.85	176	188	234(233.7)	168.83	**238**(238)	174.11
tech-routers-rf	58	40	**115**(115)	0.22	**115**(115)	<0.01	60	80	184(184)	22.69	**186**(186)	56.62	100	120	214(214)	154.21	**246**(246)	76.75	121	160	297(296.3)	240.72	**301**(301)	46.52
web-edu	77	40	**147**(147)	0.38	**147**(147)	<0.01	107	80	**197**(197)	0.40	**197**(197)	<0.01	137	116	**229**(229)	0.38	**247**(247)	<0.01	167	141	**297**(297)	0.39	**297**(297)	<0.01
web-google	68	**135**	**135**(135)	0.11	**135**(135)	<0.01	138	198	**217**(217)	0.17	**217**(217)	<0.01	195	259	**271**(271)	2.72	**274**(274)	<0.01	202	295	**324**(324)	2.95	**324**(324)	<0.01
web-polblogs	69	**79**	78(78)	4.01	**79**(79)	0.06	109	128	131(130.1)	300.95	**134**(133.4)	326.33	–	165	146(145.4)	135.39	**182**(181.4)	313.17	192	182	218(216.6)	328.46	**226**(224.7)	284.66
web-spam	31	32	160(159.9)	147.27	**161**(160.2)	275.08	61	52	269(268.3)	237.14	**272**(272)	5.96	91	72	325(325)	167.37	**368**(367.1)	130.10	–	92	448(446.4)	141.53	**458**(456)	298.35

Table 5
Experimental results of EA-OIS and its comparison algorithms for $s = 4$.

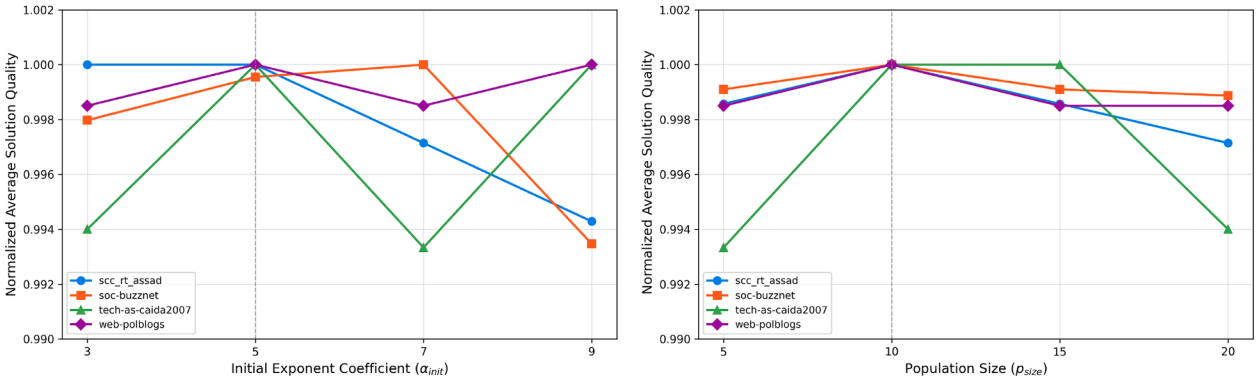
Instance	$k=10$						$k=20$						$k=30$						$k=40$					
	CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS	
	sol	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*
bio-celegans	73	64	**82**(81.8)	181.58	**82**(82)	3.98	130	110	143(142.6)	160.63	**146**(145.1)	348.84	184	153	199(197.6)	227.14	**205**(204.2)	332.86	237	216	247(246)	300.71	**258**(256.8)	355.00
bio-diseasome	76	50	**81**(81)	0.26	**81**(81)	<0.01	135	98	**149**(148.8)	156.35	**149**(149)	28.27	198	145	**209**(208.9)	179.48	**209**(209)	61.89	245	184	268(267.9)	164.37	**269**(269)	180.58
bio-yeast	50	32	**66**(66)	0.18	**66**(66)	<0.01	92	62	**126**(126)	0.24	**126**(126)	<0.01	120	92	**186**(186)	1.25	**186**(186)	<0.01	181	122	**246**(244.8)	328.08	**246**(246)	0.09
ca-CSphd	47	29	**53**(53)	0.14	**53**(53)	<0.01	93	62	**103**(103)	0.14	**103**(103)	<0.01	137	92	**153**(153)	0.14	**153**(153)	<0.01	178	118	**203**(203)	0.17	**203**(203)	<0.01
ca-Erdos992	–	31	**78**(78)	14.63	**78**(78)	<0.01	–	61	147(145.8)	148.02	**148**(148)	0.50	–	91	207(206.1)	199.73	**211**(211)	2.91	–	121	267(265.3)	301.93	**272**(271.9)	127.43
ca-GrQc	40	32	**249**(249)	0.48	**249**(249)	<0.01	–	62	**356**(356)	2.90	**356**(356)	0.02	–	92	**437**(437)	29.98	**437**(437)	0.04	–	122	**517**(517)	30.56	**517**(517)	0.05
ca-netscience	69	**77**	**77**(77)	0.20	**77**(77)	<0.01	134	125	**146**(146)	3.90	**146**(146)	0.13	191	168	**206**(206)	5.13	**206**(206)	0.07	245	217	259(259)	31.69	**260**(260)	55.29
ia-Email-univ	68	34	**106**(106)	0.55	**106**(106)	<0.01	115	64	**189**(189)	2.92	**189**(189)	<0.01	149	94	264(263.9)	79.46	**266**(266)	1.67	225	124	334(333.3)	267.44	**337**(337)	1.68
ia-enron-only	82	70	**83**(83)	6.41	**83**(83)	0.15	128	108	128(127)	151.99	**132**(131.2)	350.22	**143**	122	142(142)	35.23	**143**(143)	0.07	**143**	138	142(142)	7.37	**143**(143)	<0.01
ia-fb-messages	57	32	**82**(81.2)	271.12	**82**(82)	0.09	113	62	149(147.8)	234.81	**154**(154)	28.22	146	92	208(206.7)	321.18	**222**(221.2)	110.11	201	122	266(264.3)	89.48	**285**(284.5)	206.60
ia-infect-dublin	134	75	137(137)	0.41	**138**(138)	<0.01	202	126	223(223)	8.49	**224**(224)	<0.01	268	176	287(286.8)	105.24	**290**(290)	13.33	324	214	337(336)	223.96	**346**(345.1)	258.91
ia-infect-hyper	**98**	86	89(87.8)	238.28	97(96.5)	295.93	**113**	112	**113**(111.9)	329.73	**113**(113)	0.42	**113**	112	**113**(113)	15.00	**113**(113)	0.03	**113**	112	**113**(113)	15.15	**113**(113)	0.03
inf-power	44	32	**71**(71)	1.54	**71**(71)	0.02	85	62	**132**(132)	10.71	**132**(132)	0.05	130	92	**192**(192)	10.94	**192**(192)	0.09	–	122	**252**(252)	11.11	**252**(252)	0.03
rt-retweet	**52**	48	**52**(52)	0.46	**52**(52)	<0.01	**95**	72	93(93)	4.77	93(93)	0.04	**96**	94	**96**(96)	0.11	**96**(96)	<0.01	**96**	**96**	**96**(96)	0.09	**96**(96)	<0.01
rt-twitter-copen	53	48	**64**(64)	33.03	**64**(64)	0.44	103	78	**123**(121.6)	99.89	**123**(123)	2.53	146	106	172(171.8)	176.69	**176**(175.1)	74.55	189	136	222(221.5)	178.73	**225**(225)	99.68
scc_enron-only	**151**	146	**151**(151)	102.83	**151**(151)	0.01	**151**	147	**151**(151)	103.57	**151**(151)	0.01	**151**	147	**151**(151)	103.35	**151**(151)	0.05	**151**	149	**151**(151)	101.03	**151**(151)	0.05
scc_fb-forum	**393**	31	382(380.1)	58.36	389(388.3)	170.87	**429**	61	422(420)	59.51	**429**(428.1)	170.36	457	91	462(460)	58.73	**469**(468.7)	83.28	493	121	502(499.9)	59.51	**510**(508.9)	339.71
scc_fb-messages	890	31	1016(1011.8)	321.93	**1036**(1035.2)	42.83	1024	61	1057(1051.8)	322.72	**1088**(1086.3)	283.04	1077	91	1097(1092.9)	341.13	**1128**(1126.4)	299.51	1100	121	1134(1128)	342.44	**1169**(1167.5)	304.25
scc_infect-hyper	**113**	**113**	111(110.1)	360.01	**113**(113)	<0.01	**113**	**113**	111(110.4)	360.01	**113**(113)	0.02	**113**	**113**	110(110)	0.05	**113**(113)	0.03	**113**	**113**	111(110.2)	360.01	**113**(113)	0.03
scc_reality	–	32	1836(1815)	340.16	**1897**(1896.1)	227.24	–	62	1909(1877.8)	324.15	**2033**(2032)	264.50	–	92	1941(1902.8)	343.67	**2131**(2129.4)	327.10	–	122	1952(1923.7)	354.44	**2206**(2202.3)	347.61
scc_retweet	–	31	406(401.8)	53.88	**423**(422)	166.06	–	61	446(442.6)	52.70	**479**(477.4)	323.56	–	91	486(483)	53.10	**523**(520.3)	332.93	–	121	526(522.6)	53.77	**564**(561.4)	288.42
scc_rt_alwefaq	–	31	57(57)	56.64	**60**(59.2)	188.14	–	61	97(97)	57.24	**100**(99.3)	193.42	–	91	137(137)	58.21	**140**(139.2)	296.20	–	121	178(177.1)	221.53	**180**(179.3)	231.69
scc_rt_assad	40	31	**48**(47.9)	121.45	**48**(48)	0.11	84	61	**88**(87.9)	123.08	**88**(88)	0.22	120	91	**128**(127.9)	124.10	**128**(128)	0.34	160	121	**168**(167.9)	124.44	**168**(168)	0.15
scc_rt_bahrain	–	31	50(49.4)	189.16	**52**(51.1)	70.29	–	61	90(89.4)	205.03	**91**(91)	0.96	–	91	130(129.3)	150.16	**132**(132)	95.32	–	121	170(169.4)	180.78	**172**(171.3)	263.18
scc_rt_damascus	40	31	47(46.3)	108.67	**48**(48)	35.90	–	61	87(86.3)	74.95	**88**(88)	1.45	–	91	127(126.3)	75.23	**128**(128)	2.83	–	121	167(166.3)	75.10	**168**(168)	76.96
scc_rt_dash	–	31	44(43.9)	106.75	**45**(45)	4.60	–	61	84(83.9)	106.31	**85**(85)	3.72	–	91	124(123.9)	108.29	**125**(125)	4.07	–	121	164(163.9)	106.28	**165**(165)	2.31
scc_rt_gop	–	31	**41**(41)	44.30	**41**(41)	0.09	–	61	**81**(81)	44.26	**81**(81)	0.14	–	91	**121**(121)	44.64	**121**(121)	0.99	–	121	**161**(161)	44.35	**161**(161)	0.14
scc_rt_http	–	31	**41**(41)	0.28	**41**(41)	<0.01	–	61	**81**(81)	0.27	**81**(81)	<0.01	–	91	**121**(121)	0.29	**121**(121)	<0.01	–	121	**161**(161)	0.29	**161**(161)	<0.01
scc_rt_israel	–	31	**41**(41)	23.04	**41**(41)	0.38	–	61	**81**(81)	23.03	**81**(81)	0.61	–	91	**121**(121)	23.20	**121**(121)	0.28	–	121	**161**(161)	23.01	**161**(161)	0.14
scc_rt_ksa	–	31	**43**(42.9)	92.24	**43**(43)	0.03	–	61	**83**(82.9)	91.51	**83**(83)	1.33	–	91	**123**(122.9)	92.11	**123**(123)	0.29	–	121	**163**(162.9)	92.06	**163**(163)	0.28
scc_rt_lebanon	–	31	**40**(40)	<0.01	**40**(40)	<0.01	–	61	**80**(80)	<0.01	**80**(80)	<0.01	–	91	**120**(120)	0.02	**120**(120)	<0.01	–	121	**160**(160)	0.02	**160**(160)	<0.01
scc_rt_libya	–	31	**43**(43)	5.45	**43**(43)	<0.01	–	61	**83**(83)	5.36	**83**(83)	0.02	–	91	**123**(123)	5.14	**123**(123)	0.02	–	121	**163**(163)	5.16	**163**(163)	0.12
scc_rt_obama	**40**	31	**40**(40)	<0.01	**40**(40)	<0.01	–	61	**80**(80)	<0.01	**80**(80)	<0.01	–	91	**120**(120)	0.01	**120**(120)	<0.01	–	121	**160**(160)	0.02	**160**(160)	<0.01
scc_rt_occupy	40	31	49(48.1)	234.58	**51**(51)	1.23	–	61	89(88.1)	232.75	**91**(91)	0.44	–	91	129(128.1)	232.33	**131**(131)	3.80	–	121	169(168.1)	234.08	**171**(171)	11.85
scc_rt_occupywallstnyc	–	31	70(69.1)	308.78	**71**(71)	0.20	–	61	109(109)	8.03	**111**(111)	1.18	–	91	149(149)	8.06	**151**(151)	13.65	–	121	189(189)	8.34	**191**(191)	0.32
scc_rt_oman	–	31	**42**(42)	116.97	**42**(42)	0.55	–	61	**82**(82)	116.04	**82**(82)	0.45	–	91	**122**(122)	131.55	**122**(122)	1.75	–	121	**162**(162)	131.12	**162**(162)	1.22
scc_rt_p2	–	31	**42**(41.5)	139.05	**42**(42)	0.26	–	61	**82**(81.5)	138.23	**82**(82)	0.42	–	91	**122**(121.4)	104.80	**122**(122)	0.49	–	121	**162**(161.4)	105.94	**162**(162)	1.15
scc_rt_tcot	–	31	**42**(42)	53.34	**42**(42)	0.52	–	61	**82**(82)	52.79	**82**(82)	1.38	–	91	**122**(122)	53.79	**122**(122)	0.11	–	121	**162**(162)	54.23	**162**(162)	0.48
scc_rt_tlot	–	31	**41**(41)	0.13	**41**(41)	<0.01	–	61	**81**(81)	0.13	**81**(81)	<0.01	–	91	**121**(121)	0.15	**121**(121)	<0.01	–	121	**161**(161)	0.13	**161**(161)	<0.01
scc_rt_uae	–	31	42(41.1)	108.36	**43**(43)	8.26	–	61	82(81.1)	110.15	**83**(83)	1.08	–	91	122(121.1)	110.50	**123**(123)	4.52	–	121	162(161.1)	109.47	**163**(163)	2.45
scc_rt_voteonedirection	40	31	**41**(41)	31.44	**41**(41)	<0.01	80	61	**81**(81)	31.23	**81**(81)	0.37	120	91	**121**(121)	31.50	**121**(121)	0.22	160	121	**161**(161)	31.49	**161**(161)	<0.01
soc-karate	**34**	**34**	**34**(34)	0.03	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.01	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.02	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.03	**34**(34)	<0.01
soc-wiki-Vote	73	38	**86**(85.2)	146.35	**86**(86)	0.53	118	70	152(150.3)	280.31	**156**(155.9)	145.19	156	100	210(209.8)	209.78	**217**(216.3)	234.38	218	130	268(266.5)	95.06	**276**(275.2)	81.11
tech-routers-rf	40	32	**125**(125)	0.83	**125**(125)	<0.01	80	62	202(201.8)	145.23	**207**(206.4)	198.44	120	92	273(271.3)	224.00	**277**(276.8)	215.12	160	122	333(331.6)	282.38	**341**(341)	74.24
web-edu	87	32	**147**(147)	0.54	**147**(147)	<0.01	121	62	**207**(207)	0.55	**207**(207)	<0.01	161	92	**267**(267)	0.54	**267**(267)	<0.01	201	122	**327**(327)	0.53	**327**(327)	<0.01
web-google	77	34	**136**(136)	0.13	**136**(136)	<0.01	159	64	**220**(220)	0.24	**220**(220)	<0.01	173	94	**282**(282)	1.48	**282**(282)	<0.01	219	123	**342**(342)	2.76	**342**(342)	<0.01
web-polblogs	82	50	89(88.9)	151.02	**90**(89.6)	195.38	138	85	150(149.1)	182.56	**154**(154)	14.95	185	116	201(200.2)	157.83	**212**(211.3)	308.27	224	137	251(249.2)	169.07	**264**(262.7)	291.73
web-spam	41	32	**171**(170.2)	244.10	**171**(171)	1.07	82	62	289(287.6)	162.71	**293**(292.5)	167.01	121	92	389(388.7)	267.30	**401**(400.1)	133.57	–	122	479(478)	210.21	**497**(496.6)	210.26

Table 6
Experimental results of EA-OIS and its comparison algorithms for $s = 5$.

Instance	$k = 10$						$k = 20$						$k = 30$						$k = 40$					
	CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS		CplexEnumILS				EA-OIS	
	sol	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*	*sol*	*sol*	*sbest*(*savg*)	*tavg*	*sbest*(*savg*)	*tavg*
bio-celegans	82	38	90(89.5)	127.87	**92**(91.9)	24.34	149	66	160(159)	163.67	**165**(164.2)	331.28	209	102	221(219.3)	154.31	**233**(232.4)	299.94	263	162	274(272.5)	153.32	**294**(293)	335.00
bio-diseasome	86	36	**93**(93)	0.66	**93**(93)	<0.01	159	68	**174**(174)	5.90	**174**(174)	0.08	224	96	246(246)	136.43	**250**(249.3)	335.33	276	162	**309**(307.9)	281.14	**309**(308.3)	310.78
bio-yeast	50	42	**76**(76)	48.68	**76**(76)	0.75	107	82	**146**(146)	73.85	**146**(146)	0.70	150	122	**216**(215.6)	125.28	**216**(216)	0.13	206	162	280(279.2)	264.96	**286**(286)	1.57
ca-CSphd	60	42	**63**(63)	31.11	**63**(63)	0.08	118	82	**123**(123)	28.93	**123**(123)	0.06	176	94	**183**(183)	29.37	**183**(183)	0.04	235	129	**243**(243)	32.70	**243**(243)	0.27
ca-Erdos992	–	41	89(88.3)	196.87	**90**(90)	0.03	–	81	163(161.3)	90.03	**170**(170)	0.30	–	95	234(231.2)	177.54	**243**(243)	55.29	–	162	300(296.2)	345.04	**314**(313.4)	302.17
ca-GrQc	50	42	**251**(251)	1.75	**251**(251)	<0.01	100	82	**366**(365.8)	104.60	**366**(366)	0.11	–	122	**456**(455.8)	103.22	**456**(456)	0.14	–	162	**546**(545.8)	104.34	**546**(546)	0.05
ca-netscience	86	44	**88**(88)	0.16	**88**(88)	<0.01	157	84	**168**(168)	3.28	**168**(168)	0.16	226	122	**238**(237.3)	174.51	**238**(238)	25.55	276	162	294(293.3)	216.15	**298**(296.7)	343.44
ia-Email-univ	62	42	**113**(113)	2.42	**113**(113)	<0.01	125	82	**207**(206.3)	202.01	**207**(207)	0.05	195	122	290(288.9)	130.76	**296**(296)	5.98	234	162	366(363.8)	194.92	**376**(375.2)	330.06
ia-enron-only	85	69	91(91)	20.60	**92**(91.1)	328.36	139	112	136(135.6)	148.01	**142**(140.8)	283.44	**143**	138	**143**(143)	2.03	**143**(143)	<0.01	**143**	**143**	**143**(143)	1.84	**143**(143)	<0.01
ia-fb-messages	66	42	89(88.7)	131.04	**94**(94)	2.73	120	82	163(160.7)	150.70	**175**(175)	142.31	154	96	229(228)	152.40	**252**(251.8)	133.93	221	162	294(293.2)	189.93	**324**(323.5)	298.36
ia-infect-dublin	123	42	142(142)	0.26	**144**(144)	0.05	225	82	235(235)	70.45	**237**(237)	0.04	275	122	305(303.5)	99.19	**313**(312.6)	206.14	337	162	353(351.9)	295.86	**371**(369.3)	327.92
ia-infect-hyper	102	**103**	93(92.3)	170.35	102(100.9)	326.51	**113**	**113**	**113**(113)	15.85	**113**(113)	0.28	**113**	112	**113**(113)	4.34	**113**(113)	0.03	**113**	**113**	**113**(113)	4.29	**113**(113)	0.02
inf-power	54	42	**81**(81)	1.09	**81**(81)	0.01	111	82	156(155.9)	142.48	**158**(158)	7.50	168	100	226(225.9)	136.34	**228**(228)	2.22	–	162	296(295.9)	100.58	**298**(298)	5.99
rt-retweet	**61**	59	60(60)	0.27	**61**(61)	1.38	**96**	**96**	**96**(96)	0.91	**96**(96)	0.07	**96**	**96**	**96**(96)	0.08	**96**(96)	<0.01	**96**	**96**	**96**(96)	0.07	**96**(96)	<0.01
rt-twitter-copen	64	42	**74**(74)	3.92	**74**(74)	<0.01	118	82	140(139.6)	166.15	**143**(143)	5.41	169	96	200(199.4)	298.50	**204**(204)	32.36	219	162	259(258.4)	220.42	**264**(263.2)	285.81
scc_enron-only	**151**	134	**151**(151)	50.54	**151**(151)	0.04	**151**	144	**151**(151)	50.72	**151**(151)	<0.01	**151**	**151**	**151**(151)	51.41	**151**(151)	0.04	**151**	148	**151**(151)	52.08	**151**(151)	0.02
scc_fb-forum	390	41	388(386.7)	32.90	**395**(392.6)	269.52	444	81	438(436.7)	32.74	**445**(443)	234.74	**497**	121	488(486.5)	57.48	494(493.3)	45.80	507	161	537(536.4)	178.74	**544**(542.3)	168.77
scc_fb-messages	1012	41	1009(1005.4)	339.07	**1034**(1033.3)	260.83	1037	81	1059(1055.2)	311.47	**1094**(1091)	196.10	1021	121	1102(1096.7)	350.09	**1144**(1141.8)	233.46	1152	161	1143(1137.2)	358.25	**1194**(1191.3)	299.37
scc_infect-hyper	**113**	**113**	111(110.2)	360.01	**113**(113)	<0.01	**113**	**113**	110(110)	0.04	**113**(113)	0.02	**113**	**113**	111(110.1)	360.01	**113**(113)	0.03	**113**	**113**	111(110.2)	360.01	**113**(113)	0.03
scc_reality	–	42	1832(1793.5)	318.36	**1900**(1895.2)	281.37	–	82	1903(1849)	358.80	**2046**(2042.4)	345.18	–	122	1945(1882.3)	348.11	**2143**(2139)	351.12	–	158	1968(1899.5)	358.92	**2222**(2215.5)	300.94
scc_retweet	–	41	404(402.8)	250.51	**428**(426.2)	139.37	–	81	456(453.3)	358.93	**488**(485.3)	345.13	–	121	506(503.6)	347.32	**538**(535.4)	329.20	–	161	556(553.6)	358.18	**589**(585.9)	347.12
scc_rt_alwefaq	–	41	66(65.7)	141.77	**67**(66.9)	289.06	–	81	116(115.7)	136.67	**117**(117)	10.50	–	121	166(165.7)	137.59	**167**(167)	234.02	–	161	216(215.7)	124.36	**217**(217)	1.62
scc_rt_assad	50	41	57(57)	0.12	**58**(58)	206.46	100	81	107(107)	0.13	**108**(108)	71.46	150	121	157(157)	0.11	**158**(158)	118.17	200	161	207(207)	0.13	**208**(208)	142.01
scc_rt_bahrain	–	41	59(58.9)	230.16	**60**(60)	3.80	–	81	109(108.9)	229.86	**110**(109.8)	53.28	–	121	159(158.9)	252.03	**160**(160)	7.84	–	161	209(208.9)	247.75	**210**(210)	7.27
scc_rt_damascus	50	41	55(55)	38.82	**56**(56)	0.46	–	81	105(105)	38.77	**106**(106)	1.91	–	121	155(155)	39.17	**156**(156)	0.46	–	161	205(205)	39.66	**206**(206)	3.26
scc_rt_dash	–	41	**54**(54)	88.93	**54**(54)	2.34	–	81	**104**(104)	88.81	**104**(104)	3.60	–	121	**154**(154)	91.10	**154**(154)	0.59	–	161	**204**(204)	90.02	**204**(204)	3.76
scc_rt_gop	–	41	50(50)	<0.01	**51**(51)	50.37	–	81	100(100)	<0.01	**101**(101)	51.36	–	121	**150**(150)	0.01	**150**(150)	<0.01	–	161	**200**(200)	0.02	**200**(200)	<0.01
scc_rt_http	–	41	**50**(50)	<0.01	**50**(50)	<0.01	–	81	**100**(100)	<0.01	**100**(100)	<0.01	–	121	**150**(150)	0.01	**150**(150)	<0.01	–	161	**200**(200)	0.02	**200**(200)	<0.01
scc_rt_israel	–	41	**51**(51)	65.13	**51**(51)	0.21	–	81	**101**(101)	64.56	**101**(101)	0.05	–	121	**151**(151)	65.79	**151**(151)	0.05	–	161	**201**(201)	64.51	**201**(201)	0.80
scc_rt_ksa	–	35	**51**(51)	0.24	**51**(51)	<0.01	–	81	**101**(101)	0.24	**101**(101)	<0.01	–	121	**151**(151)	0.29	**151**(151)	<0.01	–	161	**201**(201)	0.26	**201**(201)	<0.01
scc_rt_lebanon	–	41	**50**(50)	<0.01	**50**(50)	<0.01	–	81	**100**(100)	<0.01	**100**(100)	<0.01	–	121	**150**(150)	0.01	**150**(150)	<0.01	–	161	**200**(200)	0.02	**200**(200)	<0.01
scc_rt_libya	–	41	**52**(52)	0.31	**52**(52)	<0.01	–	81	**102**(102)	0.30	**102**(102)	<0.01	–	121	**152**(152)	0.33	**152**(152)	<0.01	–	161	**202**(202)	0.30	**202**(202)	<0.01
scc_rt_obama	**50**	41	**50**(50)	<0.01	**50**(50)	<0.01	–	81	**100**(100)	<0.01	**100**(100)	<0.01	–	121	**150**(150)	0.01	**150**(150)	<0.01	–	161	**200**(200)	0.02	**200**(200)	<0.01
scc_rt_occupy	50	41	57(56.2)	234.53	**58**(58)	3.06	–	81	106(106)	84.96	**109**(108.3)	165.26	–	121	156(156)	91.56	**159**(158.1)	352.15	–	161	207(206.1)	168.28	**208**(208)	34.16
scc_rt_occupywallstnyc	–	41	77(76.2)	207.58	**78**(78)	7.76	–	81	127(126.2)	203.92	**128**(128)	10.26	–	121	177(176.1)	127.02	**179**(178.1)	264.21	–	161	227(226.2)	173.44	**228**(228)	5.39
scc_rt_oman	–	41	51(51)	0.21	**52**(51.4)	120.61	–	81	**101**(101)	0.18	**101**(101)	<0.01	–	121	151(151)	0.18	**152**(151.1)	125.88	–	161	201(201)	0.21	**202**(201.4)	128.53
scc_rt_p2	–	41	**52**(51.2)	168.44	**52**(52)	2.86	–	81	**102**(101.2)	165.59	**102**(102)	0.26	–	121	**152**(151.2)	168.54	**152**(152)	0.72	–	161	**202**(201.3)	142.50	**202**(202)	0.81
scc_rt_tcot	–	41	**52**(51.9)	97.39	**52**(52)	2.16	–	81	**102**(101.9)	96.15	**102**(102)	0.15	–	121	**152**(151.9)	96.65	**152**(152)	0.41	–	161	**202**(201.9)	96.48	**202**(202)	0.17
scc_rt_tlot	–	41	50(50)	<0.01	**51**(51)	313.19	–	81	100(100)	<0.01	**101**(101)	42.15	–	121	**150**(150)	0.01	**150**(150)	<0.01	–	161	200(200)	0.02	**201**(201)	44.31
scc_rt_uae	–	41	**52**(51.2)	137.47	**52**(52)	3.94	–	81	**102**(101.2)	136.43	**102**(102)	2.54	–	121	**152**(151.2)	139.11	**152**(152)	10.27	–	125	**202**(201.2)	140.06	**202**(202)	10.87
scc_rt_voteonedirection	**50**	41	**50**(50)	<0.01	**50**(50)	<0.01	**100**	81	**100**(100)	<0.01	**100**(100)	<0.01	**150**	121	**150**(150)	0.02	**150**(150)	<0.01	**200**	125	**200**(200)	0.02	**200**(200)	<0.01
soc-karate	**34**	**34**	**34**(34)	0.02	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.01	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.02	**34**(34)	<0.01	**34**	**34**	**34**(34)	0.01	**34**(34)	<0.01
soc-wiki-Vote	74	42	95(94.1)	167.69	**96**(96)	2.94	130	71	168(166.8)	127.27	**175**(175)	76.04	179	122	236(235.3)	195.03	**246**(245.8)	202.26	240	162	298(296.8)	129.53	**315**(314)	350.34
tech-routers-rf	50	42	**137**(137)	5.58	**137**(137)	<0.01	101	82	222(221.7)	163.88	**224**(224)	0.99	162	122	297(295.8)	88.36	**304**(304)	9.35	204	162	367(366.1)	264.70	**381**(380.5)	257.63
web-edu	91	42	**148**(148)	0.49	**148**(148)	<0.01	140	82	**219**(218.1)	250.61	**219**(219)	91.71	188	122	**289**(288.1)	252.59	**289**(289)	150.88	238	162	**359**(358.1)	244.05	**359**(359)	81.18
web-google	99	42	**136**(136)	0.14	**136**(136)	<0.01	143	82	**227**(227)	1.01	**227**(227)	<0.01	202	96	300(299.3)	160.20	**304**(303.6)	294.49	240	162	370(369.4)	175.16	**374**(373.2)	296.26
web-polblogs	91	42	100(99.5)	243.27	**102**(102)	9.16	145	82	168(166.2)	230.12	**174**(173.1)	288.19	188	122	225(224)	185.37	**239**(238.3)	283.71	247	162	280(279.1)	145.56	**296**(295.8)	246.91
web-spam	51	42	184(182.9)	153.32	**185**(185)	12.03	100	82	308(306.4)	264.66	**316**(315.4)	128.10	153	122	414(411.9)	218.63	**433**(431.9)	58.37	–	162	508(504.1)	69.24	**537**(536.2)	215.26



(a) Effect of TTVP probabilities p_1 and p_2



(b) Effect of initial exponent coefficient α_{init} (c) Effect of population size p_{size}

Fig. 8. Parameter sensitivity analysis.

Table 7
Comparison of EA-OIS variants.

EA-OIS(vs.)	EA-OIS1		EA-OIS2		EA-OIS3		EA-OIS4		EA-OIS5	
	sbest	*savg*	*sbest*	*savg*	*sbest*	*savg*	*sbest*	*savg*	*sbest*	*savg*
#win	**467**	**743**	**432**	**632**	**443**	**698**	**712**	**719**	**481**	**680**
#lose	57	92	78	156	62	88	30	140	78	158

Table 8
ARPL values averaged over all instances within each category. Graph scale: small $|V| < 1,000$; medium $1,000 \le |V| \le 100,000$; large $|V| > 100,000$.

Variant	ARPL(%)	ARPL-Small graphs(%)	ARPL-Medium graphs(%)	ARPL-Large graphs(%)
EA-OIS1	**0.2526**	0.0669	0.0804	0.6550
EA-OIS2	**0.2041**	0.0098	0.0512	0.5723
EA-OIS3	**0.3266**	0.0152	0.0724	0.9340
EA-OIS4	**0.4422**	-0.0089	0.1736	1.1277
EA-OIS5	**0.3023**	0.0253	0.1772	0.6455

EA-OIS4, showing that both initialization quality and priority-aware expansion are crucial for large-scale instances.

Scoring function (EA-OIS5). Replacing the proposed TLNCD-based two-layer scoring mechanism with the ILS scoring rule (EA-OIS5) results in an overall ARPL of 0.3023%, with positive degradation across small, medium, and large graphs. This suggests that TLNCD consistently improves the search process, particularly when candidate vertices need to be distinguished using both direct coverage and neighborhood-level information.

7.5. Parameter sensitivity analysis

Fig. 8 presents the sensitivity of EA-OIS to its key parameters. All results are averaged over 10 independent runs with $s = 3, k = 20$ on representative instances spanning sparse and dense graph topologies. As shown in Fig. 8(a), solution quality remains stable across most combinations of p_1 and p_2, with the setting $p_1 = 0.10$, $p_2 = 0.20$ consistently delivering competitive performance. Fig. 8(b) shows that $\alpha_{init} = 5$ achieves the best balance between exploitation and exploration: smaller values reduce the priority differentiation among vertex tiers, while larger values lead to overly greedy selection and diminished population diversity. Fig. 8(c) indicates that $p_{size} = 10$ provides a good trade-off between crossover diversity and computational efficiency — smaller populations limit the quality of crossover materials, while larger ones dilute the search budget within the fixed time limit. Overall, the normalized solution quality remains above 0.992 across all tested configurations, confirming that EA-OIS is robust to parameter variation.

8. Conclusion and future work

This work introduced EA-OIS, a novel evolutionary algorithm tailored for the S-DTKSP problem. By incorporating an observer individual, EA-OIS consistently improves search depth and robustness. In conjunction with a Tri-Tier Vertex Prioritization strategy, a Dynamic Prioritized Weighted Selection strategy, and a Two-Layer Scoring Based on Neighbor Coverage Density strategy, the proposed algorithm effectively maintains a balance between exploration and exploitation throughout the search process. Comprehensive experimental evaluations on diverse benchmarks verify the effectiveness and stability of EA-OIS. The results demonstrate its capability to consistently yield high-quality and diversified solutions.

Future research will focus on integrating advanced population diversity preservation mechanisms, adaptive learning-driven search control, and hybrid strategies combining EA-OIS with problem-specific heuristics. These enhancements aim to further improve scalability, convergence efficiency, and solution diversity, particularly for highly complex network instances.

CRediT authorship contribution statement

Jiejiang Chen: Conceptualization, Methodology, Funding acquisition, Software, Writing – review & editing; **Shuping Xu:** Methodology, Software, Writing – review & editing; **Jun Wu:** Data curation, Writing – review & editing, Funding acquisition; **Weiguo Sheng:** Formal analysis, Writing – review & editing, Supervision.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the National Nature Science Foundation of China (grant number 62506109 and 62306149), the Zhejiang Provincial Natural Science Foundation of China under (Grant LQN26F020073), and the Startup Foundation for Introducing Talent of NUIST (grant number 2023r045).

References

Adouani, Y., Masmoudi, M., Jarray, F., & Jarboui, B. (2024). Iterative integer linear programming-based heuristic for solving the multiple-choice knapsack problem with setups. *Expert Systems with Applications*, *242*, 122835.

Balasundaram, B., Butenko, S., & Hicks, I. V. (2011). Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, *59* (1), 133–142.

Boginski, V., Butenko, S., & Pardalos, P. M. (2006). Mining market data: A network approach. *Computers & Operations Research*, *33* (11), 3171–3184.

Chang, L., & Yao, K. (2024). Maximum k-plex computation: Theory and practice. *Proceedings of the ACM on Management of Data*, *2* (1), 1–26.

Conte, A., Firmani, D., Mordente, C., Patrignani, M., & Torlone, R. (2017). Fast enumeration of large k-plexes. In *Proceedings of the 23rd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 115–124).

Dondi, R., Hosseinzadeh, M. M., Mauri, G., & Zoppis, I. (2021). Top-k overlapping densest subgraphs: Approximation algorithms and computational complexity. *Journal of Combinatorial Optimization*, *41* (1), 80–104.

Dorndorf, U., Jaehn, F., & Pesch, E. (2008). Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science*, *42* (3), 292–301.

Etzion, T., & Ostergard, P. R. J. (2002). Greedy and heuristic algorithms for codes and colorings. *IEEE Transactions on Information Theory*, *44* (1), 382–388.

Fan, W., Wang, X., & Wu, Y. (2013). Diversified top-k graph pattern matching. *Proceedings of the VLDB Endowment (PVLDB)*, *6* (13), 1510–1521.

Fang, W., Shen, B., Pan, A., Zou, L., & Song, B. (2024). A cooperative stochastic configuration network based on differential evolutionary sparrow search algorithm for prediction. *Systems Science & Control Engineering*, *12* (1), 2314481.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32* (200), 675–701.

Gao, J., Yu, J., Qiu, H., Jiang, X., Wang, T., & Yang, D. (2011). Holistic top-k simple shortest path join in graphs. *IEEE Transactions on Knowledge and Data Engineering*, *24* (4), 665–677.

Gao, S., Yu, K., Liu, S., & Long, C. (2024). Maximum *k*-plex search: An alternated reduction-and-bound method. arXiv preprint arXiv:2406.00617, .

Garcia, S., & Herrera, F. (2008). An extension on" statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, *9* (12), 2677–2694.

Ghanbarpour, A., Niknafs, K., & Naderi, H. (2020). Efficient keyword search over graph-structured data based on minimal covered r-cliques. *Frontiers of Information Technology & Electronic Engineering*, *21* (3), 448–464.

Hamza, A., Darwish, A. H., & Rihawi, O. (2023). A new local search for the bees algorithm to optimize multiple traveling salesman problem. *Intelligent Systems with Applications*, *18*, 200242.

Hao, F., Pei, Z., & Yang, L. T. (2020). Diversified top-k maximal clique detection in social internet of things. *Future Generation Computer Systems*, *107*, 408–417.

Hastad, J. (1996). Clique is hard to approximate within n/sup 1-/spl epsiv. In *Proceedings of 37th Conference on foundations of computer science* (pp. 627–636). IEEE.

He, Y., Lin, L., Yuan, P., Li, R., Jia, T., & Wang, Z. (2024). Ccss: Towards conductance-based community search with size constraints. *Expert Systems with Applications*, *250*, 123915.

He, Z., & Liu, H. (2025). Dual-stage dual-population diversity maintenance for global and local exploration of constrained multiobjective optimization. *Expert Systems with Applications*, *268*, 126229.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

Lyu, B., Qin, L., Lin, X., Zhang, Y., Qian, Z., & Zhou, J. (2022). Maximum and top-k diversified biclique search at scale. *The VLDB Journal*, *31* (6), 1365–1389.

Pachuau, J. L., Singh, N. B., Singh, L. D., & Saha, A. K. (2023). Solving dense subgraph problems using genetic algorithm. In *Tencon 2023-2023 IEEE Region 10 conference (tencon)* (pp. 674–679). IEEE.

Paquete, L., & Stützle, T. (2018). Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In *Handbook of approximation algorithms and metaheuristics*, (pp. 411–425).

Pullan, W. (2021). Local search for the maximum k-plex problem. *Journal of Heuristics*, *27* (3), 303–324.

Sheng, M., Ding, W., & Sheng, W. (2024). Differential evolution with adaptive niching and reinitialisation for nonlinear equation systems. *International Journal of Systems Science*, *55* (10), 2172–2186.

Song, J., & Zhang, X. (2024). Observer-based adaptive controllers for lur'e multi-agent systems with a dynamic leader. *International Journal of Systems Science*, *55* (1), 33–48.

Strickland, D. M., Barnes, E., & Sokol, J. S. (2005). Optimal protein structure alignment using maximum cliques. *Operations Research*, *53* (3), 389–402.

Wolfe, A. W. (1997). Social network analysis: Methods and applications. *American Ethnologist*, *24*(1), 219–220.

Wu, J., Li, C.-M., Wang, L., Hu, S., Zhao, P., & Yin, M. (2023). On solving simplified diversified top-k s-plex problem. *Computers & Operations Research*, *153*, 106187.

Wu, J., Li, C. M., Zhou, Y., Yin, M., Xu, X., & Niu, D. (2022). Hea-d: A hybrid evolutionary algorithm for diversified top-k weight clique search problem. In *IJCAI* (pp. 4821–4827).

Wu, J., & Yin, M. (2021). Local search for diversified top-k s-plex search problem (student abstract). In *Proceedings of the AAAI Conference on artificial intelligence* (pp. 15929–15930). (*vol. 35*).

Xu, X., Liu, H., Lv, X., Wang, Y., & Li, D. (2024). An efficient and exact algorithm for locally h-clique densest subgraph discovery. *Proceedings of the ACM on Management of Data*, *2* (6), 1–26.

Xue, J., Zheng, J., He, K., Li, C.-M., & Liu, Y. (2024). DiverTEAM: An effective evolutionary algorithm for diversified top-k (weight) clique search problems. In *ECAI 2024* (pp. 4108–4115). *vol. 392*.

Yue, C., Song, J., Liang, J., Liu, M., Yu, K., Lin, H., & Bi, Y. (2025). A multimodal multiobjective evolutionary algorithm based on neighborhood and enhanced special crowding distance. *Knowledge-Based Systems*, *315*, 113340.

Zheng, J., Xue, J., He, K., Li, C.-M., & Liu, Y. (2024). An efficient evolutionary algorithm for diversified top-k (weight) clique search problems. arXiv preprint arXiv:2404.09997, .

Zheng, Z., Xie, Z., Wang, Z., & Hooi, B. (2025). Monte carlo tree search for comprehensive exploration in llm-based automatic heuristic design. arXiv preprint arXiv:2501.08603, .

Zhong, F., Bing, C., & Zhang, S. (2025). Gpo++: A dynamic pooling routing network based on pooling operator for cross-modal retrieval. *Expert Systems with Applications*, (p. 128723).

Zhu, J., Chen, B., & Zeng, Y. (2020). Community detection based on modularity and k-plexes. *Information Sciences*, *513*, 127–142.

Zou, Z., Li, J., Gao, H., & Zhang, S. (2010). Finding top-k maximal cliques in an uncertain graph. In *2010 IEEE 26th International conference on data engineering (ICDE 2010)* (pp. 649–652). IEEE.