Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the manuscript number!)

Detecting Critical Nodes in Sparse Graphs via "Reduce-Solve-Combine" Memetic Search

Yangming Zhou
Data-Driven Management Decision Making Lab, Shanghai Jiao Tong University, Shanghai 200030, China
Sino-US Global Logistics Institute, Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China
yangming.zhou@sjtu.edu.cn

Jiaqi Li
Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
y30211039@mail.ecust.edu.cn

Jin-Kao Hao*(Corresponding author)
LERIA, Université d'Angers, Angers 49045, France
jin-kao.hao@univ-angers.fr

Fred Glover*(Corresponding author)
Entanglement, Inc., Boulder, Colorado 80302, USA
fred@entanglement.ai

This study considers a well-known critical node detection problem that aims to minimize a pairwise connectivity measure of an undirected graph via the removal of a subset of nodes (referred to as critical nodes) subject to a cardinality constraint. Potential applications include epidemic control, emergency response, vulnerability assessment, carbon emission monitoring, network security and drug design. To solve the problem, we present a "reduce-solve-combine" memetic search approach that integrates a problem reduction mechanism into the popular population-based memetic algorithm framework. At each generation, a common pattern mined from two parent solutions is first used to reduce the given problem instance, then the reduced instance is solved by a component-based hybrid neighborhood search that effectively combines an articulation point impact strategy and a node weighting strategy, and finally an offspring solution is produced by combining the mined common pattern and the solution of the reduced instance. Extensive evaluations on 42 real-world and synthetic benchmark instances show the efficacy of the proposed method, which discovers 9 new upper bounds and significantly outperforms the current state-of-the-art algorithms. Investigation of key algorithmic modules additionally discloses the importance of the proposed ideas and strategies. Finally, we demonstrate the generality of the proposed method via its adaptation to solve the node-weighted critical node problem.

Key words: Critical Node Problem; Memetic Search; Instance Reduction; Reduce-Solve-Combine; Heuristic Search.

1. Introduction

Given an undirected graph $G=(V,E)$ with vertex (or node) set V and edge set E, critical node detection problems (CNDPs) (Arulselvan et al. 2009, Naoum-Sawaya and Buchheim 2016, Zhou et al. 2019, Baggio et al. 2021, Zhou et al. 2021a, Salemi and Buchanan 2022) aim to identify a subset of nodes (referred to as *critical nodes*) $S \subseteq V$ whose removal enhances (decreases) the graph connectivity of the residual graph $G[V \setminus S]$ evaluated by a given connectivity measure σ. According to different cases of $|S|$ and σ, CNDPs can be divided into two categories: K-vertex-CNDP and β-connectivity-CNDP. The former is to optimize (minimize or maximize) the connectivity measure σ, such that no more than K nodes are deleted (i.e., $|S| \leq K$), while the latter aims to minimize the set of deleted nodes, such that the connectivity measure σ is bounded by a given threshold β (Zhou et al. 2023d). A detailed taxonomy of CNDPs is provided in (Zhou et al. 2021a). In addition, node-weighted CNDPs (Chen et al. 2020, Zhou et al. 2021c) and distance-based CNDPs (Salemi and Buchanan 2022, Zhou et al. 2023c) have been receiving increasing attention in the literature.

The critical node problem (CNP) (Arulselvan et al. 2009, Zhou et al. 2019, Baggio et al. 2021) is a fundamental CNDP, which belongs to the category of K-vertex-CNDPs. It seeks a set $S \subseteq V$ of at most K nodes, the deletion of which minimizes the total pairwise connectivity in $G[V \setminus S]$. Formally, the objective function $f(S)$ of CNP can be written as follows:

$$f(S) = \sum_{i=1}^{M} \frac{|\mathcal{C}_i|(|\mathcal{C}_i|-1)}{2} \tag{1}$$

where $\mathcal{C}_i$ is a connected component, and M is the total number of connected components in the residual graph $G[V \setminus S]$. Hence, the residual graph $G[V \setminus S]$ is composed of M connected components, i.e., $\sum_{i=1}^{M} \cup \mathcal{C}_i = G[V \setminus S]$. From (1), we observe that a good solution of CNP should generate a residual graph that maximizes the number of connected components while simultaneously minimizing the variance in the component sizes.

CNP has a wide spectrum of applications in many fields. For example, the overall transmissibility of a virus can be limited by identifying only a specific number of people to be

vaccinated in epidemic control (Doostmohammadian et al. 2020). Emergency response can be modeled as a CNP by identifying some critical nodes that can be used to plan good emergency evacuations on a disaster case (Vitoriano et al. 2011). Besides epidemic control and emergency response, CNP is also a convenient model for other applications such as vulnerability assessment (Nguyen et al. 2013, Zhou et al. 2021b), carbon emission monitoring (Zhang et al. 2020), network security (Mugisha and Zhou 2016) and drug design (Abbas et al. 2021).

CNP is known to be NP-hard on general graphs (Arulselvan et al. 2009). Its solution space grows exponentially with its size. As indicated in (Veremyev et al. 2014a), CNP can be solved exactly on medium sparse graphs with up to 1500 nodes under the time limit 50000 seconds. However, CNP instances from real-world applications can be considerably larger. To deal with such instances, computationally efficient heuristic algorithms have been developed to provide high-quality solutions in reasonable computation time. To the best of our knowledge, two population-based memetic algorithms with fixed or variable population represent the current state-of-the-art for solving CNP (Zhou et al. 2019, 2021a). However, these algorithms are time-consuming, which becomes a handicap when they are applied on large graphs. This motivates us to incorporate a strategy based on the divide-and-conquer principle that divides an original problem into several subproblems to solve them separately, followed by combining the solutions of the subproblems to yield an overall solution of the original problem. To take advantage of both the memetic search framework and the divide-and-conquer principle, this work introduces a "reduce-solve-combine" memetic algorithm that integrates a problem reduction mechanism into the popular memetic algorithm framework.

The main contributions of the work are summarized as follow:

- The proposed algorithm, called Instance Reduction-based Memetic Search (IRMS), incorporates a "reduce-solve-combine" mechanism within the popular population-based memetic algorithm framework. At each generation, a common pattern mined from two parent solutions is first used to reduce the original instance, then the reduced instance is solved, and an offspring solution is finally obtained by combining the mined common pattern and the solution of the reduced instance. In addition, a component-based hybrid neighborhood search that combines the articulation point impact and node weighting strategies is developed to ensure an effective local optimization. It is worth noting that the reduced instance

can be approximately solved by a heuristic solver or optimally solved by an exact solver in the "reduce-solve-combine" module. Moreover, this module is of a generic nature, which can be combined with other heuristic algorithms to improve their search performances.

- The proposed IRMS algorithm achieves a high level of performance on the 42 synthetic and real-world benchmark instances commonly used in the literature that compares very favorably with the state-of-the-art CNP algorithms, finding new upper bounds for 9 instances. Our experimental results also disclose the superiority of IRMS over the most recent FPBS (Zhou et al. 2022), which is based on mining patterns from a set of high-quality solutions with a time-consuming frequent itemset mining algorithm to guide the offspring solution construction. Experimental analyses on key algorithmic modules of the proposed algorithm are performed to identify the elements underlying the effectiveness of the proposed ideas and techniques.
- We also demonstrate the inherent generality of the proposed IRMS algorithm by an application to solve the node-weighted critical node problem, where this generalized version of IRMS performs significantly better than the best-performing algorithm for the node-weighted problem in terms of both the best and average values.

The rest of this paper is organized as follows. After a brief review of previous studies on CNP and instance reduction techniques in Section 2, we present IRMS for CNP in Section 3. Section 4 conducts experimental studies of the proposed algorithm and compares its results with those of the state-of-the-art methods. The generalization of IRMS that adapts it to the node-weighted CNP is presented in Section 5. Key issues of IRMS are analyzed in Section 6, followed by conclusions in Section 7.

2. Related Work

2.1. Previous Studies on CNP

The critical node problem (CNP) has been shown to be NP-hard (Arulselvan et al. 2009) and has attracted widespread research attention (Arulselvan et al. 2009, Pullan 2015, Zhou et al. 2019, Veremyev et al. 2019, Zhou et al. 2021a). Existing solution approaches can be divided into two categories: exact and heuristic algorithms. Exact algorithms can theoretically guarantee the optimality of their obtained solutions. For example, Arulselvan et al. (2009) presented the first integer programming model with $O(|V|^2)$ variables and $O(|V|^3)$ constraints for CNP and used CPLEX to solve the model. Di Summa et al. (2012) further proposed two improved formulations: an extended formulation of (Arulselvan et al.

2009) and a quadratic programming reformulation considering the complete form of CNP. Both of them are solved with the branch-and-cut framework. Due to the large number of constraints (i.e., $O(|V|^3)$), the exact algorithms could solve CNP to optimality only for small sparse graphs with up to 150 nodes. Veremyev et al. (2014b) developed an improved compact linear reformulation with only $O(|V|^2)$ constraints, which was able to provide exact solutions for CNP with up to 1200 nodes, and further developed a general integer programming framework for solving different CNDPs (Veremyev et al. 2014a). Rezaei et al. (2021) proposed an efficient exact iterative algorithm (EIA-CNDP) to solve a CNDP whose objective is to minimize the size of the largest connected component. In addition, they provided a comprehensive survey on both exact and heuristic algorithms for solving different CNDPs.

To deal with large instances, heuristic algorithms are required to solve CNP approximately in an affordable computation time. Existing heuristics for CNP can be grouped into two categories: local search and population-based methods. Local search methods manipulate only a single candidate solution of the given problem in each search step. For example, Arulselvan et al. (2009) presented an early heuristic that starts with an independent set and is coupled with a 2-exchange local search. Ventresca (2012) proposed a simulated annealing (SA) algorithm for CNP using a combinatorial unranking-based problem representation. Pullan (2015) developed a multi-start greedy algorithm for CNP (CNA1 for short). Addis et al. (2016) proposed several hybrid heuristic algorithms by combining the two basic greedy rules (i.e., add-back and remove) with some flavor of local search. Based on two smart and computationally efficient neighborhoods, Aringhieri et al. (2016b) presented two metaheuristic algorithms for CNP based on the iterated local search (ILS) and variable neighborhood search (VNS) frameworks. More recently, de San Lázaro et al. (2021) proposed an improved VNS algorithm and Wang and Di (2022) proposed a cluster expansion method called CEMCNP for CNP. CEMCNP utilizes a strategy similar to that of the multi-start ILS algorithm, supplemented by integrating a contraction mechanism to greedily alleviate the effect of vertex scale without loss of accuracy and an incremental cluster expansion approach to iteratively separate the graph into many disconnected components whose sizes are kept within reasonable bounds.

In contrast to the preceding local search methods, population-based methods often maintain a population of candidate solutions that are manipulated and evaluated during the

search process. For instance, Ventresca (2012) proposed a population-based method for CNP called the population-based incremental learning (PBIL) algorithm, which employed an unranking-based problem representation and obtained higher quality solutions than SA. However, SA found solutions faster than PBIL. Aringhieri et al. (2016a) designed a general evolutionary algorithm framework for different classes of CNDPs, which followed a simple genetic algorithm framework that made use of greedy rules to repair and correct the solution during the reproduction and mutation phases. Purevsuren et al. (2017) combined a greedy randomized adaptive search procedure (GRASP) with exterior path-relinking for CNP. Following this, Zhou et al. (2019) presented a memetic algorithm for CNP (MACNP for short) and subsequently proposed a variable population memetic search (VPMS) (Zhou et al. 2021a), employing a strategic population sizing mechanism to dynamically adjust the population size during the search process. These memetic algorithms adopt a strategy for combining solutions to create new ones similar to that of the path relinking approach (Glover 1997), which invites the relationship between these approaches to be investigated more fully. Unlike the general local search and population-based algorithms, Nabli and Carvalho (2020) proposed a multi-agent reinforcement learning framework to learn to solve a multilevel budgeted combinatorial problem. A case study on the multilevel CNP show this learning algorithms outperforms classical reinforcement learning algorithms.

To the best of our knowledge, most state-of-the-art results of the CNP instances were obtained by memetic algorithms (i.e., MACNP and VPMS), which rely on a fixed or variable population of candidate solutions to explore the search space. However, these algorithms are very time-consuming and have trouble to effectively solve very large instances. It is necessary to propose computationally efficient algorithms capable to deal with such large instances.

2.2. Previous Studies on Instance Reduction

To solve large scale instances, instance reduction is a useful strategy for a number of difficult combinatorial optimization problems, as shown in various studies in the literature (Zheng and Xue 2010, Wu and Hao 2012, Chen and Hao 2014, Delgadillo et al. 2016, Kenny et al. 2018, de Holanda Maia et al. 2020, Zhang et al. 2021, Le et al. 2022). In the following, we briefly review some representative instance reduction methods fitting different categories.

The divide-and-conquer strategy is a popular and general approach to solve large-scale problems. The main idea is to decompose the original large problem into smaller subproblems that can be solved individually. For the arc routing problem, a commonly used

divide-and-conquer strategy is to divide the task into subsets, and then solve the subproblems induced by the task subsets separately. For example, Zhang et al. (2021) proposed a novel problem decomposition operator called the route cutting off operator, and integrated it within two state-of-the-art divided-and-conquer algorithms to solve large scale capacitated arc routing problems.

Reduce-and-solve methods represent a closely related derivative of divide-and-conquer approaches illustrated by the work of Zheng and Xue (2010) who employ formal calculation rules to divide a discrete optimization problem into subproblems with smaller search spaces and accompanied by efficient implicit algorithms to incrementally construct a complete solution from the solutions to the subproblems. Chen and Hao (2014) developed a reduce-and-solve heuristic approach for the multiple-choice multidimensional knapsack problem, which combined problem reduction techniques with the CPLEX solver. Its basic idea is to employ some dedicated heuristics to fix a number of groups and variables in order to obtain a reduced critical subproblem which is then solved by the CPLEX solver.

Similarly, Wu and Hao (2012) proposed an effective approach called EXTRACOL to coloring large graphs. It first applied a preprocessing procedure to extract large independent sets from the graph, and then used a memetic algorithm to color the residual graph. However, if an independent set extracted in the preprocessing is not part of the optimal coloring, it can never be repaired. To cope with this issue, the authors developed an extraction and expansion method called E2COL, which integrates an expansion phase to allow the coloring process to reconsider each extracted independent set on a one-by-one basis (Wu and Hao 2013). They further extended E2COL by proposing additional strategies, resulting in an improved extraction and expansion algorithm named IE^{2}COL (Hao and Wu 2012) which employed a forward independent set extraction strategy to reduce the initial graph, followed by a backward coloring process which uses extracted independent sets as new color classes for intermediate subgraph coloring.

Inspired by the concept of immunization by vaccination derived from artificial immune systems, Montiel et al. (2013) proposed a reduce-optimize-expand method to improve existing discrete optimization algorithms for large-scale traveling salesman problems (TSPs) composed of three steps. The first step decreases the problem complexity by a heuristic for reducing the number of nodes of the original problem. The next step applies an exact or heuristic algorithm to obtain an intermediate solution which is expanded in the third step

to produce a final solution. Based on the reduce-optimize-expand method, Delgadillo et al. (2016) presented an intelligent strategy with a fuzzy logic classifier to obtain systematic reductions of TSP instances. In related work, de Holanda Maia et al. (2020) proposed a MineReduce approach to improve a heuristic for the heterogeneous fleet vehicle routing problem by performing problem reductions.

Blum et al. (2016) proposed an alternative hybrid metaheuristic framework called construct, merge, solve & adapt (CMSA) for combinatorial optimization problems that similarly employs reduced problem instances and works in three phases. First, it generates a reduced sub-instance of the original problem instance by a process that ensures a solution to the sub-instance is also a solution to the original instance. Second, it applies an exact solver to the reduced sub-instance to obtain a high-quality solution of the original instance. Finally, it makes use of the results of the exact solver as feedback for the next algorithm iteration. The method has been successfully applied to solve the minimum common string partition problem and minimum covering arborescence problem.

Kenny et al. (2018) presented a problem reduction metaheuristic called merge search (MS). It consists of three main modules: an initial solution construction heuristic, a SA-based local search to quickly generate a population of neighbouring solutions, and a merge operation that uses information from all solutions in the population to produce a reduced subproblem, which is then solved by a mixed integer programming solver. The method has been further extended with an improved population generation and variable aggregation heuristics for the constrained pit problem in (Kenny et al. 2019). The main difference between MS and CMSA is that CMSA generates solutions from scratch, while MS starts with a single initial seed solution and uses local search to generate a population of neighbouring solutions to the initial seed solution.

Mihic et al. (2018) presented a general purpose local search approach called randomized decomposition (RD) for solving hard, nonlinear, nonconvex mathematical programs. RD utilizes a novel decomposition to partition the solution space into random subspaces and then find a local optimum in each subspace independently. In addition to the quadratic assignment problem, the RD decomposition method has been successfully applied to a wide range of other problems, including revenue management (Cooper and Homem-de-Mello 2007) and traveling salesman problems (Subramanyam and Gounaris 2018).

As mentioned previously, memetic algorithms (i.e., MACNP (Zhou et al. 2019) and VPMS (Zhou et al. 2021a)) are the best performing heuristic algorithms for CNP. In this work, we present an instance reduction-based memetic search (IRMS) approach for CNP, where a "reduce-solve-combine" instance reduction mechanism is integrated into the well-known memetic algorithm to join the merits of these two strategies for solving large and hard CNP applications.

3. Instance Reduction-based Memetic Search for CNP

Our instance reduction-based memetic search algorithm for CNP is based on the following key modules.

3.1. Solution Representation and Evaluation

Given a CNP instance with an integer K, any subset $S \subset V$ of size K is a feasible solution, i.e., $|S| = K$. A candidate solution S can be represented by $S = \{v_{S(1)}, v_{S(2)}, \ldots, v_{S(K)}\}$ such that $S(i)$ is the index of node i in V or equivalently a binary vector of size $|V|$ such that exactly K variables receive the value of 1 and the other $|V| - K$ variable receive 0. The solution space Ω contains all possible subsets of K nodes, i.e., $\Omega = \{S \subset V : |S| = K\}$. The size of Ω is given by $\binom{|V|}{K} = \frac{|V|!}{K!(|V|-K)!}$ and increases rapidly with increases in $|V|$ and K. According to (1), the solution cost of S can be evaluated by a modified depth first search (DFS) algorithm (Hopcroft and Tarjan 1973) in $O(|V| + |E|)$.

3.2. General Scheme

The general scheme of the proposed IRMS approach is illustrated in Figure 1. One distinct feature of IRMS is that a common pattern mined from only two parent solutions is used to guide the instance reduction, and the reduced instance is then solved by a fast local search heuristic. Finally, a promising offspring solution is obtained by directly combining the common pattern and the solution of the reduced instance. From the perspective of algorithm architecture, IRMS is composed of five modules (see Algorithm 1): 1) population Initialization (Section 3.3), 2) component-based hybrid neighborhood search (Section 3.4), 3) common pattern mining (Section 3.5), 4) "reduce-solve-combine" mechanism (Section 3.6), and 5) population updating (Section 3.7).

The IRMS approach starts with an initial population of λ high-quality solutions (line 1). At each generation, it randomly selects two parent solutions S^F and S^M from the population P (line 4). A common pattern mining procedure is then applied to find a

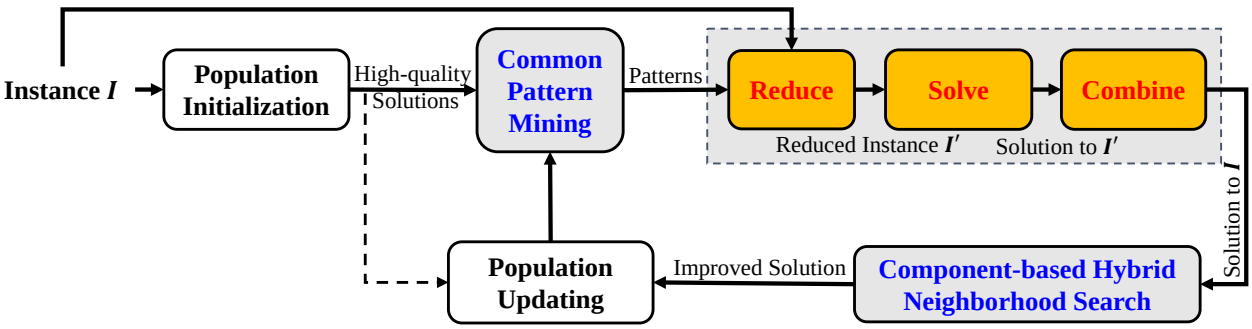


Figure 1 **Diagram of the Proposed IRMS Approach**

common pattern ζ between S^F and S^M (line 5). An offspring solution S is then generated by the "reduce-solve-combine" (RSC) mechanism (line 6), which is further improved by a component-based hybrid neighborhood search (CHNS) procedure (line 7). Finally, a population updating procedure is used to accept or discard the offspring solution (line 11). The process repeats until a given stopping condition is satisfied, such as a time limit $\hat{t}$ or a given number of generations.

Algorithm 1 Pseudo Code of IRMS Approach for CNP

Input: A CNP instance I (i.e., an undirected graph G with an integer K), population size λ, selection probability θ, and maximal idle iteration count $\hat{\xi}$

Output: The best found solution S^*

1: $P \leftarrow$ **PopulationInitialization**(λ) /* Build an initial population */
2: $S^* \leftarrow \arg\min\{f(S_i) : i = 1, 2, \ldots, \lambda\}$ /* Record the best solution S^* */
3: **while** a stopping condition is not satisfied **do**
4: Randomly select two solutions S^F and S^M from P
5: $\zeta \leftarrow$ **CommonPatternMining**(S^F, S^M) /* Mine common pattern between two parents */
6: $S \leftarrow$ **RSC**(I, ζ) /* Construct an offspring solution */
7: $S' \leftarrow$ **CHNS**$(S, \theta, \hat{\xi})$ /* Improve the solution */
8: **if** $f(S') \leq f(S^*)$ **then**
9: $S^* \leftarrow S'$
10: **end if**
11: $P \leftarrow$ **PopulationUpdating**(P, S') /* Update the population */
12: **end while**
13: **return** The best found solution S^*

3.3. Population Initialization

The initial population is composed of λ diverse and high-quality solutions, where each solution is created as follows. A random solution is first generated, which is then improved to a high-quality local optimum by the component-based neighborhood search procedure (Zhou et al. 2019). Then the improved solution is added into the population P if the solution does not duplicate any existing solution in the current population, and the process repeats until λ different high-quality solutions are obtained.

3.4. Component-based Hybrid Neighborhood Search

3.4.1. Basic Idea: To perform local optimization, we propose a component-based hybrid neighborhood search (CHNS), which effectively combines the node weighting and articulation point impact strategies. As shown in Algorithm 2, CHNS starts from a candidate solution S, and then iteratively improves it by adding a new node to S (lines 4-11) and greedily removing a node from S (lines 12-13). CHNS stops when the idle iteration count ξ (i.e., the number of iterations without improvement) reaches an allowed maximal value $\hat{\xi}$.

DEFINITION 1 (**Large Connected Component**). A connected component $\mathcal{C}$ is considered as a large connected component if its size is greater than $(\hat{\chi}+\check{\chi})/2$, where $\hat{\chi} = \max_{i\in\{1,\ldots,M\}} |\mathcal{C}_i|$ and $\check{\chi} = \min_{i\in\{1,\ldots,M\}} |\mathcal{C}_i|$ present the size of largest and smallest connected components in the residual graph $G[V \setminus S]$, respectively.

At each iteration, a large connected component $\mathcal{C}$ is selected randomly. Then, CHNS employs a hybrid node selection strategy to select a node v from $\mathcal{C}$ and combines the node weighting and articulation point impact strategies in a probabilistic way. That is, a node v is selected by the articulation point impact strategy with a selection probability θ $(0 \leq \theta \leq 1)$, and otherwise is selected by a node weighting strategy as in (Zhou et al. 2019). CHNS can be considered as an improved form of component-based neighborhood search (CBNS) (Zhou et al. 2019) that includes a way to remove a node from a large connected component $\mathcal{C}$ with the articulation point impact strategy described next.

3.4.2. Articulation Point Impact Strategy: Let $\mathcal{C} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ be a large connected component, where $V_{\mathcal{C}}$ and $E_{\mathcal{C}}$ denote the node set and edge set in $\mathcal{C}$, respectively. The articulation point impact strategy aims to select a node whose removal maximally decreases the connectivity of $\mathcal{C}$ (see Algorithm 3). To quickly find such a node in $\mathcal{C}$, we design a

Algorithm 2 Pseudo Code of CHNS

Input: A solution S, selection probability θ, and maximal idle iteration count $\hat{\xi}$

Output: The best solution S^* found

1: $S^* \leftarrow S$
2: $\xi \leftarrow 0$ /* Idle iteration count */
3: **while** $\xi < \hat{\xi}$ **do**
4: Randomly select a large connected component $\mathcal{C}$
5: Generate a random probability $p \in (0,1)$
6: **if** $p < \theta$ **then**
7: $v \leftarrow$ **ArticulationPointImpact**$(\mathcal{C})$ /* Executed with the pre-defined selection probability θ */
8: **else**
9: $v \leftarrow$ **NodeWeighting**$(\mathcal{C})$ /* Executed with the probability $1-\theta$ */
10: **end if**
11: $S \leftarrow S \cup \{v\}$
12: $u \leftarrow \arg\min\{f(S \setminus \{w\}) - f(S) | w \in S\}$
13: $S \leftarrow S \setminus \{u\}$
14: **if** $f(S) < f(S^*)$ **then**
15: $S^* \leftarrow S$
16: $\xi \leftarrow 0$
17: **else**
18: $\xi \leftarrow \xi + 1$
19: **end if**
20: **end while**
21: **return** The best solution S^* found

TarjanInComponent procedure based on the algorithm of (Tarjan 1972) that traverses every node in a component in only one round with time complexity $O(|V_{\mathcal{C}}| + |E_{\mathcal{C}}|)$ (line 11). The detailed pseudo code of TarjanInComponent is provided in Algorithm 1 of the online supplement (Zhou et al. 2023b). The articulation point impact strategy starts its search from a root of the large connected component $\mathcal{C}$. Note that any node in $\mathcal{C}$ can be considered as a root node.

A conversion from an original connected component to a DFS tree is illustrated in Figure 2, where Figure 2(a) and (b) respectively present a connected component of nine nodes and a corresponding DFS tree rooted at node a. As shown in Figure 2(b), black solid arrows indicate tree edges that are taken when visiting unvisited nodes, while red dashed arrows denote back edges taken when visiting visited nodes. After removing the

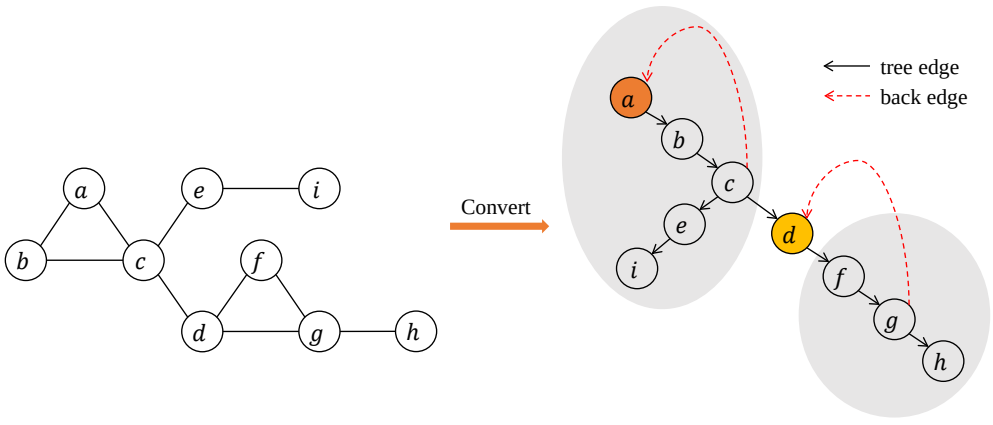
Algorithm 3 Pseudo Code of Articulation Point Impact Strategy

Input: A large connected component $\mathcal{C}$

Output: The selected node v^*

$root \leftarrow$ a random node in $\mathcal{C}$ /* Randomly select a node as the root node */
// Initialize all arrays
$Count \leftarrow 0$ /* Time stamp */
for all nodes $v \in \mathcal{C}$ **do**
  $dfn[v] = \psi[v] = 0$
  $\gamma[v] = \eta[v] = 1$
end for
// Calculate the impact of each node
TarjanInComponent$(\mathcal{C}, root, Count, \gamma, \eta)$
for each node $v \in V_{\mathcal{C}}$ **do**
  if v is an articulation point **then**
    $\psi[v] += \frac{(Count-\eta[v])(Count-\eta[v]-1)}{2}$
  else
    $\psi[v] += \frac{(Count-1)(Count-2)}{2}$
  end if
  $v^* \leftarrow \arg\min\{\psi[v] : v \in V_{\mathcal{C}}\}$
end for
return The selected node v^*

node d, the two shaded areas that correspond to two resulting connected components are obtained.



(a) A component (b) A DFS tree rooted at node a

Figure 2 **An Illustrative Example of a Conversion between (a) an Original Component and (b) the Corresponding DFS Tree**

Unlike the approach of (Ventresca and Aleman 2015), our articulation point impact strategy builds a DFS tree and evaluates its nodes simultaneously. To achieve this, two arrays are defined to record the states of nodes in the tree, i.e., time stamps (dfn) and trace values (low). The former presents the time stamp when a node is first visited (i.e., the traverse sequence number), while the latter records the trace value of the node (i.e., the smallest time stamp of the current node's neighbour visited). Therefore, we can obtain dfn, low, γ and η values of the above-mentioned example, as summarized in Table 1.

Table 1 Related Parameter Values of the Example Shown in Figure 2

Node	a	b	c	d	e	f	g	h	i
Time Stamp (dfn)	1	2	3	6	4	7	8	9	5
Trace Value (low)	1	1	1	3	3	6	6	8	4
γ	9	8	7	4	2	3	2	1	1
η	1	1	7	4	2	3	2	1	1

Definition 2 (**Articulation Point**). An articulation point (or cut vertex) of a graph is a node whose deletion with associated edges makes the original graph disconnected, or more precisely, increases the number of connected components in the graph.

From a DFS tree, we have two simple observations:

Remark 1. A leaf node is not an articulation point.

Remark 2. A root node with at least two subtrees is an articulation point.

Suppose u is an internal node (i.e., neither a leaf node nor a root node), v is an arbitrary node, and $e(u,v)$ is the edge between nodes u and v. We calculate low values according to the following rules,

- If $e(u,v)$ is a tree edge of the graph, then we have $low[u] = \min\{low[u], low[v]\}$;
- If $e(u,v)$ is a back edge of the graph and v is not the parent of u, then we have $low[u] = \min\{low[u], dfn[v]\}$.

To evaluate each node in a component, we define an impact function ψ that calculates the impact of removing a node in a recursive way as in (Ventresca and Aleman 2015). The impact function value $\psi(v)$ of a node $v \in \mathcal{C}$ is calculated based on two auxiliary parameters γ and η, where γ identifies the number of nodes that are descendants of v (i.e., $\phi(v)$) including v, and η indicates the sum of the node v and nodes of all new components which come from v's subtrees if removing v. It is worth noting that γ and η have the same value if v is an articulation point and is not the root node of a component, i.e.,

$\gamma[v] = \eta[v] = \phi(v) + 1$. Let $\delta(v)$ be the set of children nodes of v in the DFS tree, and $\phi(v)$ denote the total number of nodes that are descendants of v. Then $\phi(v)$ can be recursively computed as follows,

$$\phi(v) = \sum_{w \in \delta(v)} \begin{cases} \phi(w), & \text{if } w \text{ is a root or an internal node} \\ 1, & \text{if } w \text{ is a leaf node} \end{cases} \tag{2}$$

Once an articulation point v is removed, the component $\mathcal{C}$ is divided into two parts: ancestors and descendants. The descendants consist of multiple children subtrees that can be transformed into a series of new components. Therefore, the contribution of v's descendants to the objective function value can be calculated as follows:

$$\sum_{t_i \in T(v)} \frac{|t_i|(|t_i|-1)}{2} \tag{3}$$

where $|t_i|$ is the number of nodes in the children subtree $t_i \in T(v)$, and $T(v)$ is the children subtree set of v. Accordingly, the increment in the objective function can be simplified as

$$\psi[v] = f'(\mathcal{C} \setminus \{T(v) \cup \{v\}\}) + \sum_{t_i \in T(v)} \frac{|t_i|(|t_i|-1)}{2} \tag{4}$$

where the first part $f'(\mathcal{C} \setminus \{T(v) \cup \{v\}\})$ computes the total pairwise connectivity of the ancestors, while the second part presents the total pairwise connectivity of the descendants.

After traversing the component $\mathcal{C}$, a node v^* with the minimum ψ value is added to the solution S. Note that the chosen node must be the node whose removal will maximally decrease the objective function $f(S)$. Once a node v is added into S based on the node weighting or articulation point impact strategies, i.e., $S \leftarrow S \cup \{v\}$, a node u whose removal minimally deteriorates the objective function $f(S)$ is removed from S, i.e., $S \leftarrow S \setminus \{u\}$. The graph changes along with the add and remove operations, accompanied by disintegration of old components and regeneration of new components.

3.4.3. Node Weighting Strategy: The node weighting strategy is an effective diversification technique for local search that has been successfully used to solve many combinational optimization problems, such as boolean satisfiability (Thornton et al. 2004) and vertex cover (Cai et al. 2011) problems. Our node weighting strategy employs an idea similar to that of the node weighting scheme used in (Zhou et al. 2019). Each node of a large connected component is associated with a positive integer number as its weight. Weights

are initially set to 1. At each step, we select the node v from $\mathcal{C}$ with the largest weight (breaking ties randomly) to add to S. Then, the weights of the remaining nodes in $\mathcal{C}$ are incremented. Once an exchange operation is made between $v \in \mathcal{C}$ and $u \in S$, we set the weight of u to 1. As the search processes, the "hard to remove" nodes of a large connected component will have larger weights, and thus have a higher chance to be selected and removed from the large connected component in subsequent search.

3.5. Common Pattern Mining

Identifying common patterns that frequently appear in a set of high-quality solutions can be naturally modeled as a frequent pattern mining task (Grahne and Zhu 2005), where common patterns refer to frequent patterns. CNP is a typical subset selection problem, its solution is usually represented as a set of removed nodes. Based on this characteristic, the problem of performing frequent pattern mining on a set of high-quality CNP solutions is reduced to mine frequent itemsets that often appear together. Note that an itemset is composed of multiple removed nodes, and each removed node is an item. Besides itemsets, frequent patterns can also be represented as complex entities such as subsequences and substructures.

To mine useful information from high-quality solutions, considerable efforts have been made to hybridize frequent pattern mining with metaheuristic algorithms (Ribeiro et al. 2006, Plastino et al. 2014, Arnold et al. 2021, Zhou et al. 2022). A pioneer algorithm called DM-GRASP (Ribeiro et al. 2006) was proposed to solve the set packing problem, where a data mining procedure is first applied to mine useful patterns from an elite set of solutions, and then the mined pattern is used to guide the search of GRASP. MDM-GRASP (Plastino et al. 2014) improved DM-GRASP by performing data mining as soon as the elite set becomes stable (i.e., no change occurs in the elite set throughout a given number of iterations) and whenever the elite set has been changed and has become stable again, instead of performing data mining only once. Recently, Zhou et al. (2022) presented a frequent pattern based search (FPBS) method for the quadratic assignment problem, where frequent patterns mined from the population by FPmax* algorithm are used to guide the offspring solution construction. Although numerous frequent pattern mining algorithms are available in the literature (Luna et al. 2019), they are time consuming.

To quickly find frequent itemsets, we focus on mining common elements from only two parent solutions, randomly selecting two parent solutions S^F and S^M from the population

at each generation. We observe that if all mined common elements between S^F and S^M are used to guide instance reduction, the size of the reduced instance tends to become very small. To avoid this problem and keep some randomness, we inherit each common element with a random probability. Therefore, the resulting frequent itemset ζ has no more than $\beta|S^F \cap S^M|$ elements, i.e., $|\zeta| \leq \beta|S^F \cap S^M|$, where β $(0 < \beta \leq 1)$ is a proportional factor.

3.6. "Reduce-Solve-Combine" Mechanism

Inspired by the divide-and-conquer strategy, we propose a "reduce-solve-combine" (RSC) method to generate promising offspring solutions. RSC consists of three main stages, as shown in Algorithm 4. At the reduction stage, the original instance I is reduced to I' based on a common pattern ζ (i.e., a l-itemset, where $l = |\zeta|$). All nodes of the pattern ζ and all edges associated with the nodes are deleted from the original instance I (lines 1-4). At the solution stage, from a random solution S, we apply the fast local search heuristic CHNS of Section 3.4 to find an improved solution S' for the reduced instance I' (lines 5-6). The improved solution S' can be treated as a part of the final offspring solution S^o. At the combination stage, a feasible solution of the original instance I is obtained by combining the common pattern ζ and the local optimal solution S' of I', i.e., $S^o \leftarrow \zeta \cup S'$ (line 7). For some types of graphs, the reduced instance I' is usually small, and can be solved by a fast exact algorithm instead of a heuristic (i.e., CHNS). Consequently, IRMS can be implemented as a matheuristic (Boschetti and Maniezzo 2022, Archetti et al. 2017).

Figure 3 illustrates the basic idea of the RSC mechanism applied to an original instance I of 12 nodes and 16 edges with a common pattern $\zeta = \{f, g\}$. Figure 3(b) presents the reduced instance I' with 10 nodes and 9 edges by deleting all nodes of ζ and their associated edges from I. Figure 3(c) shows a high-quality local optimal solution $S' = \{c, h\}$ of I' found by CHNS. Figure 3(d) gives the feasible solution $S^o = \{f, g, c, h\}$ that is obtained by combining the common pattern $\zeta = \{f, g\}$ and the solution $S' = \{c, h\}$ of I'.

3.7. Population Updating

Following (Fu and Hao 2015, Zhou et al. 2019), we use a rank-based population updating strategy to manage the population P. Once an improved offspring solution S^o is obtained, we first tentatively insert it into P, i.e., $P' \leftarrow P \cup \{S^o\}$. All $\lambda + 1$ individuals in P' are then

Algorithm 4 Pseudo Code of RSC Procedure

Input: The CNP instance I, common pattern ζ, selection probability θ, and maximal idle iteration count $\hat{\xi}$

Output: An offspring solution S^o

// **Reduction stage**

1: **for** all nodes $v \in \zeta$ **do**

2: $E(v) \leftarrow \{$all edges associated with $v\}$

3: $I' \leftarrow I \setminus \{v, E(v)\}$

4: **end for**

// **Solution stage**

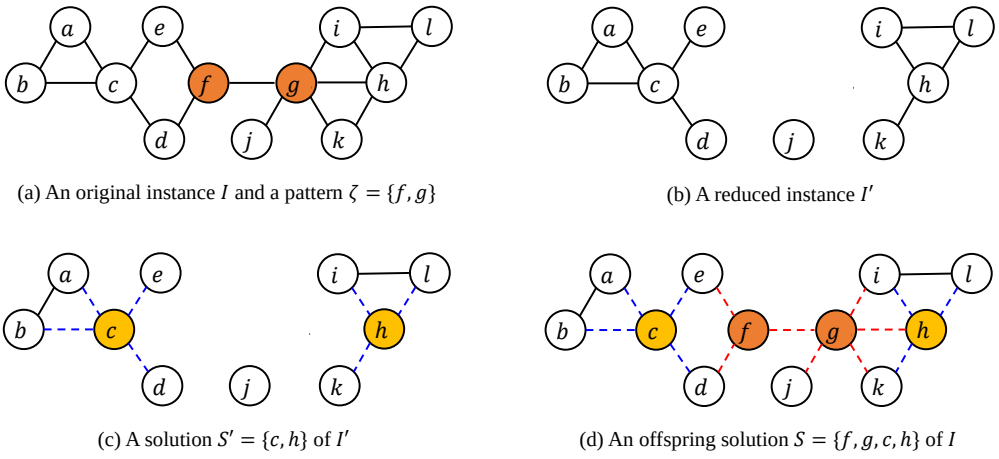
5: Randomly generate an initial solution S' of I'

6: $S' \leftarrow$ **CHNS**$(S', \theta, \hat{\xi})$

// **Combination stage**

7: $S^o \leftarrow \zeta \cup S'$

8: **return** An offspring solution S^o



(a) An original instance I and a pattern $\zeta = \{f, g\}$

(b) A reduced instance I'

(c) A solution $S' = \{c, h\}$ of I'

(d) An offspring solution $S = \{f, g, c, h\}$ of I

Figure 3 **An Illustrative Example of the RSC Mechanism**

evaluated by a combined score function $\Psi(S_i, P')$ that simultaneously considers the solution quality and solution distance. The combined score function $\Psi(S_i, P')$ can be formally defined as follows:

$$\Psi(S_i, P') = \alpha * \Phi(f(S_i)) + (1 - \alpha) * \Phi(D(S_i, P')) \tag{5}$$

where $\Phi(f(S_i))$ and $\Phi(D(S_i, P'))$ represent the rank of solution S_i with respect to its objective value f and average distance D to the population P', respectively. Note that we rank the solutions of P' in terms of solution quality and average distance in descending order and ascending order, respectively. The parameter α is a weighting coefficient between

solution quality and solution distance, which is empirically set to $\alpha = 0.6$. Afterwards, the worst solution S^w with respect to the combined score function is identified, i.e., $S^w \leftarrow \arg_{S_i \in P'} \min \Psi(S_i, P')$. Finally, if S^o is different from S^w, we replace S^w with S^o; otherwise we discard S^o.

3.8. Discussion

IRMS enhances the canonical memetic algorithm framework (Neri and Cotta 2012, Zhou et al. 2023c,a) with the "reduce-solve-combine" mechanism, which benefits from the merits of both the instance reduction technique and the population-based algorithm for solving large and hard CNP instances. As indicated by the review of Section 2.1, MACNP is one of the best-performing heuristic algorithms for CNP (Zhou et al. 2019). IRMS distinguishes itself from MACNP in the following four features. First, IRMS is an enhanced algorithm that integrates an instance reduction mechanism into a memetic algorithm, while MACNP is only a canonical memetic algorithm. Second, IRMS generates an offspring solution based on the "reduce-solve-combine" mechanism instead of the backbone-based crossover operator used in MACNP. Third, IRMS employs a component-based hybrid neighborhood search (CHNS) to perform local optimization, which reinforces the component-based neighborhood search used in MACNP by an articulation point impact strategy. Accordingly, CBNS can be considered as a special case of CHNS, where only the node weighting strategy is applied. Finally, IRMS can use both heuristic and exact solvers to solve the reduced instances in IRMS.

Compared to existing studies on combining frequent pattern mining with metaheuristics (Ribeiro et al. 2006, Plastino et al. 2014, Arnold et al. 2021, Zhou et al. 2022), IRMS employs frequent patterns mined from high-quality solutions to guide the instance reduction instead of relying on solution construction processes. Moreover, IRMS performs instance reduction by referencing to nodes common to only two high-quality solutions instead of derived from multiple high-quality solutions as in the existing (time-consuming) frequent pattern mining algorithms (e.g., FPmax* (Grahne and Zhu 2005)).

The "reduce-solve-combine" (RSC) mechanism is only an algorithmic component of IRMS, while the "reduce-optimize-expand" (ROE) framework is an algorithm framework (Montiel et al. 2013). RSC distinguishes itself from ROE in the three aspects. Firstly, RSC reduces the original instance by directly removing some nodes, while ROE generates a

reduced instance by creating fewer new nodes to represents a set of removed nodes. Secondly, RSC directly combines the removed nodes and the solution of the reduced instance to obtain a feasible solution of the original instance. While ROE expands the solution of the reduced instance by inserting the discarded nodes in an additional (heuristic) way. Thirdly, once a feasible solution of the original instance is obtained, RSC employs a local optimization procedure to further improve it to a high-quality solution.

3.9. Computational Complexity of IRMS

To analyze the computational complexity of IRMS, we consider each main module of Algorithm 1. IRMS begins its search from a high-quality initial population generated by the population initialization procedure in $O(\lambda K(|V|+|E|)\tilde{\xi})$, where λ denotes the population size, and $\tilde{\xi}$ is the total number of iterations used in CBNS.

At each generation of the main loop of Algorithm 1, IRMS sequentially executes five search procedures: parent selection, common pattern mining, RSC, CHNS and population management. The parent selection procedure only takes time $O(1)$. A common pattern (i.e., a set of nodes) between two parent solutions can be found in $O(K)$. The RSC mechanism consists of three phases: reduction, solution and combination. Both reduction and combination phases can be executed in $O(|V|)$, and the solution phase uses CHNS to solve the reduced instance, whose complexity is $O(\xi K(|V|+|E|))$, where ξ is the total number of iterations used in CHNS. Hence, the total complexity of IRMS at each generation is $O(\xi K(|V|+|E|))$.

4. Computational Studies

4.1. Benchmark Instances and Experimental Settings

Our IRMS algorithm was implemented in C++, and compiled using GNU gcc 7.3.0 with the "-O3" option on an Intel Xeon 8269CY 16-core processor with 2.5 GHz and 32 GB RAM under the linux system. Please refer to Zhou et al. (2023b) for the instances, codes and results of the experiments. Our experiments are conducted on two sets of widely used benchmark instances.

- **Synthetic benchmark set** consists of 16 graphs belonging to 4 groups with different characteristics. They are generated according to four classes of commonly used complex network models: Barabási-Albert (BA), Erdős-Rényi (ER), Forest-Fire (FF), and Watts-Strogatz (WS). BA graphs are scale-free networks and proved to be the easiest to process.

ER graphs are random graphs. FF graphs reproduce the behaviour of how a fire spreads through a forest, with a scale-free structure like BA graphs but a denser structure. WS graphs are designed to mimic a dense small-world structure and are the most challenging to solve. The detailed characteristics of these graphs can be found in (Ventresca and Aleman 2014).

- **Real-world benchmark set** is composed of 26 real-world networks from various practical applications, such as social networks, transportation networks, communication networks, biological networks, and power networks. Their details are summarized in (Aringhieri et al. 2016a).

In the following experiments, we employ the well-known two-tailed sign test (Demšar 2006) to check the statistical difference between the compared algorithms on each comparison indicator. At a significance level of 0.05, the critical value is $\mathrm{CV}_{0.05}^{42} = N/2 + 1.96\sqrt{N}/2 \approx 27$, where N is the total number of benchmark instances, i.e., $N = 42$. This implies that algorithm A statistically outperforms algorithm B if A wins in at least 27 out of 42 instances.

4.2. Parameter Sensitivity Analysis

Our computational results are obtained by running IRMS with the parameter settings provided in Table 2. The parameter $\hat{\xi}$ identifies the allowable maximal idle iteration count used in CHNS. Since CHNS can be considered as an improved CBNS, we set $\hat{\xi} = 1000$ as for CBNS (Zhou et al. 2019). For the values of the other three parameters, i.e., population size (λ), selection probability (θ) and proportional factor (β), are determined according to common practice in heuristic algorithm design by testing a limited number of parameter configurations on representative problem instances (Cordeau et al. 2006). To identify an appropriate value for a given parameter, we allow the chosen parameter to vary, while fixing the values of other parameters.

Table 2 Parameter Settings of Our IRMS Method

Parameter	Description	Considered Values	Final Value	Section
λ	Population Size	{2,3,4,5,6,7,8,9,10}	5	3.2
$\hat{\xi}$	Maximal Idle Iteration Count	{1000}	1000	3.4
θ	Selection Probability	{0.2,0.3,0.4,0.5,0.6,0.7,0.8}	0.3	3.4
β	Proportional Factor	{0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}	0.9	3.5

Our parameter sensitivity analysis is conducted on a set of 10 representative instances with different sizes and variable levels of difficulty, selected from both synthetic and real-world instance sets, i.e., BA5000, ER941, FF500, WS250, TreniR, open-flights, H3000a, H4000, powergrid and OClinks. In our experiment, each parameter value varies within a range specified in the column "Considered Values" in Table 2, while the other parameters are fixed to the "Final Values". The total time budget for tuning is specified to be 30 executions of IRMS for each selected instance with a limit to 100 generations.

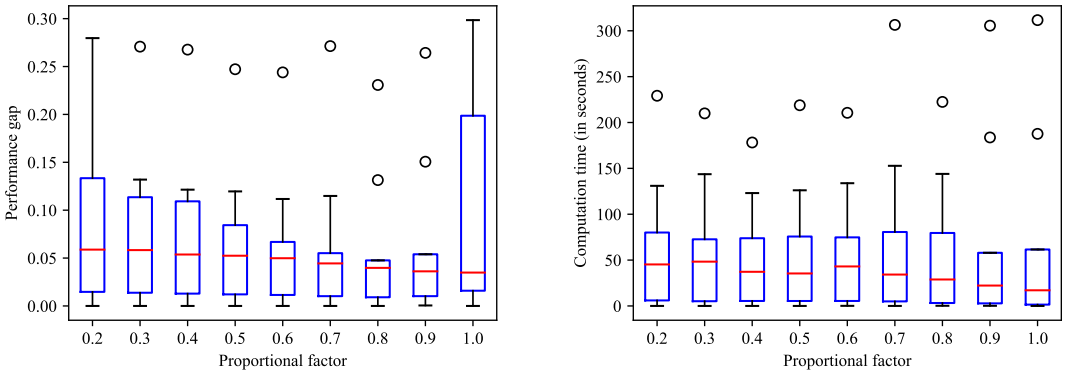


Figure 4 Boxplots of IRMS with Different β Values

We take the parameter sensitivity analysis of the proportional factor β as an example. Figure 4(a) and (b) show the box plots of IRMS with different $\beta \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ values in terms of the average performance gap $\Delta\overline{f}$ and the average computation time $\overline{t}$, respectively. We calculate the performance gap as $\Delta\overline{f} = \frac{\overline{f}-\text{BKV}}{\text{BKV}}$, where BKV indicates the best known value. From Figure 4(a), we observe that IRMS with $\beta = 0.9$ yields the best performance in terms of $\Delta\overline{f}$. We also observe that large β values ($\beta > 0.8$) have a better performance than small ones ($\beta \leq 0.8$) in terms of $\overline{t}$, as shown in Figure 4(b). To make a reasonable compromise between the solution quality and computation time, we adopt $\beta = 0.9$ in IRMS.

4.3. Comparison between IRMS and FPBS

As indicated in (Zhou et al. 2022), the frequent pattern based search (FPBS) method tries to construct an offspring solution guided by frequent itemsets which are minded from a set of high-quality solutions by the pattern mining procedure FPmax* (Grahne and Zhu 2005). Unlike FPBS, IRMS uses the mined common elements to guide the instance

reduction, where the common elements are quickly identified from only two high-quality solutions. To demonstrate the superiority of IRMS, we first adapt FPBS for CNP, and then experimentally compare it with IRMS. We independently solve each instance 30 times with different random seeds, and set the time limit of each run at $\hat{t} = 3600$ seconds.

Detailed comparative results between IRMS and FPBS are summarized in Table 3, where columns 1-5 present for each instance its name (Instance), number of nodes ($|V|$), K value, $K/|V|$ value, and best known value (BKV) reported in the literature. Columns 6-8 describe the detailed results of IRMS, i.e., the best result ($\hat{f}$) found during 30 runs, average result ($\overline{f}$), and average computation time to attain the best result ($\overline{t}$) at each run. Similarly, columns 9-11 provide the results of FPBS. The better values of the compared results in terms of $\hat{f}$ and $\overline{f}$ are indicated in bold. In addition, we also count the number of instances in which IRMS's solution are better (#Wins), equal (#Ties), and worse (#Loses) in terms of each indicator compared to BKV and FPBS.

It is worth noting that 1) the tested instances have been studied for a long time since year 2009 (Arulselvan et al. 2009); 2) the current best known results have been improved progressively by a number of algorithms (Arulselvan et al. 2009, Ventresca 2012, Pullan 2015, Addis et al. 2016, Aringhieri et al. 2016a,b, Purevsuren et al. 2017, Zhou et al. 2019, Veremyev et al. 2019, Zhou et al. 2021a, de San Lázaro et al. 2021, Wang and Di 2022), and 3) no single algorithm can attain all best known results. Thus, it is challenging to further improve the current best known results, even by a small order.

From Table 3, we observe that IRMS demonstrates an excellent performance by finding new upper bounds for seven instances (marked by '$\star$'), and matching the best-known upper bounds on 22 instances. Compared to FPBS, IRMS finds better results in terms of $\hat{f}$ on 26 out of 42 instances, and matches best known values on the remaining 16 instances except for the instance astroph. For the $\overline{f}$ performance indicator, IRMS also shows a better performance by attaining 28 better results and 14 equal results. At a significance level of 0.05, IRMS is significantly better than FPBS both in terms of $\hat{f}$ (i.e., $34.0 > \mathrm{CV}^{42}_{0.05}$) and $\overline{f}$ (i.e., $35.0 > \mathrm{CV}^{42}_{0.05}$).

4.4. Comparison with State-of-the-art Algorithms

To further evaluate IRMS, we conduct a detailed comparison with four state-of-the-art algorithms, i.e., CAN1 (Pullan 2015), MACNP (Zhou et al. 2019), VPMS (Zhou et al.

Table 3 Comparison between IRMS and FPBS on Synthetic and Real-world Benchmarks under $\hat{t} = 3600$ Seconds

					FPBS°			**IRMS**		
Instance	$\|V\|$	K	$K/\|V\|$	BKV	$\hat{f}$	$\overline{f}$	$\overline{t}$	$\hat{f}$	$\overline{f}$	$\overline{t}$
BA500	500	50	0.10	195*	**195**	**195.0**	0.0	**195**	**195.0**	0.0
BA1000	1000	75	0.08	558*	**558**	**558.0**	28.8	**558**	**558.0**	3.1
BA2500	2500	100	0.04	3704*	**3704**	**3704.0**	3.6	**3704**	**3704.0**	3.9
BA5000	5000	150	0.03	10196*	**10196**	**10196.0**	20.8	**10196**	**10196.0**	16.8
ER235	235	50	0.21	295*	**295**	**295.0**	5.3	**295**	**295.0**	6.5
ER466	466	80	0.17	1524.0	**1524**	1551.9	2222.2	**1524**	**1524.0**	83.7
ER941	941	140	0.15	5012.0	5122.0	5303.7	1724.0	**5012**	**5020.0**	520.0
ER2344	2344	200	0.09	902498.0	1006653.0	1035422.4	1451.9	**920748**	**944406.9**	3146.7
FF250	250	50	0.20	194*	**194**	**194.0**	0.0	**194**	**194.0**	0.0
FF500	500	110	0.22	257*	**257**	**257.0**	1.4	**257**	**257.0**	1.4
FF1000	1000	150	0.15	1260*	**1260**	**1260.0**	95.5	**1260**	**1260.0**	22.2
FF2000	2000	200	0.10	4545*	**4545**	4545.5	1810.3	**4545**	**4545.0**	207.3
WS250	250	70	0.28	3083.0	3339.0	3542.2	1411.6	**3085**	**3179.0**	2013.2
WS500	500	125	0.25	2072.0	2088.0	2123.2	2076.1	**2072**	**2080.1**	297.7
WS1000	1000	200	0.20	109677.0	257569.0	280878.6	1715.6	**138098**	**145969.1**	1963.2
WS1500	1500	265	0.18	13098.0	13769.0	14256.5	1620.1	**13098**	**13112.9**	2028.7
Bovine	121	3	0.02	268.0	**268**	**268.0**	0.0	**268**	**268.0**	0.0
Circuit	252	25	0.10	2099.0	**2099**	**2099.0**	13.8	**2099**	**2099.0**	1.3
Ecoli	328	15	0.05	806.0	**806**	**806.0**	0.0	**806**	**806.0**	0.4
USAir97	332	33	0.10	4336.0	5444.0	5444.0	0.1	**4336**	**4648.0**	668.8
HumanDi	516	52	0.10	1115.0	**1115**	**1115.0**	0.7	**1115**	**1115.0**	0.1
TreniR	255	26	0.10	918.0	**918**	**918.0**	0.0	**918**	**918.0**	2.5
EU_fli	1191	119	0.10	348268.0	350762.0	350887.1	1223.9	**348268**	**348295.7**	998.0
openfli	1858	186	0.10	26783.0	29130.0	29778.2	1647.0	**27198**	**28757.5**	1695.8
yeast1	2018	202	0.10	1412.0	**1412**	**1412.0**	187.1	**1412**	**1412.0**	37.8
H1000	1000	100	0.10	306349.0	322615.0	328173.6	1697.8	**306349**	**308951.9**	2165.2
H2000	2000	200	0.10	1242739.0	1331626.0	1360981.3	1721.2	**1236503**⋆	**1254481.6**	3028.8
H3000a	3000	300	0.10	2840690.0	3062331.0	3108832.5	1568.7	**2804579**⋆	**2849985.8**	3088.5
H3000b	3000	300	0.10	2837584.0	3064784.0	3104320.8	2111.0	**2801186**⋆	**2842174.8**	3164.3
H3000c	3000	300	0.10	2835369.0	3077676.0	3101625.7	1609.9	**2801692**⋆	**2840618.6**	3066.0
H3000d	3000	300	0.10	2828492.0	3054775.0	3100897.9	1903.3	**2816590**⋆	**2864256.5**	2940.0
H3000e	3000	300	0.10	2843000.0	3070679.0	3114306.9	2228.6	**2836177**⋆	**2877807.4**	2715.4
H4000	4000	400	0.10	5038611.0	5541031.0	5591268.4	1489.2	**5021551**⋆	**5110687.5**	3042.0
H5000	5000	500	0.10	7964765.0	8720111.0	8778198.6	1615.3	**8029837**	**8188900.3**	2741.0
powergr	4941	494	0.10	15862.0	16097.0	16182.9	1651.9	**15866**	**15886.6**	3021.6
Oclinks	1899	190	0.10	611253.0	616684.0	618350.7	1782.1	**614467**	**614467.6**	1038.9
facebook	4039	404	0.10	420334.0	1567137.0	1602706.4	2183.0	**719722**	**741314.4**	2852.1
grqc	5242	524	0.10	13591.0	13673.0	13708.7	2463.1	**13594**	**13613.0**	3201.9
hepth	9877	988	0.10	106276.0	122109.0	132995.7	2181.8	**115133**	**119766.8**	3159.4
hepph	12008	1201	0.10	6155877.0	11300876.0	11957556.6	1432.2	**9401029**	**9781789.8**	3077.4
astroph	18772	1877	0.10	53963375.0	61896814.0	62977189.9	1880.7	**57592461**	**58649781.0**	3271.4
condmat	23133	2313	0.10	2298596.0	9950262.0	10890742.9	2254.0	**9670268**	**10789125.6**	2286.2
#Wins|Ties|Loses				7|22|13	26|16|0	28|14|0	–	–	–	–

Notes. "∗" presents the optimal solution, "⋆" indicates the improved best upper bounds, "∘" indicates an application of the FPBS (Zhou et al. 2022) method for CNP.

2021a) and CEMCNP (Wang and Di 2022). To the best of our knowledge, these four methods are the best-performing algorithms for CNP in the literature, and they attain the best known values available except for the facebook and condmat instances[1]. Since the source code and executable program of CEMCNP are not available to us, we have re-implemented

[1] For the facebook and condmat instances, the best known values were reported in (Aringhieri et al. 2016a) (Table 5), which were reached only by the ILS-N_1-FC algorithm of (Aringhieri et al. 2016b).

the method based on its pseudo code (Wang and Di 2022). Detailed comparisons between the reported results and computational results of CEMCNP are summarized in the online supplement (Zhou et al. 2023b). To guarantee a fair comparison, we run IRMS and these four algorithms (with their source codes) on the same computational platform and under the same time limit $\hat{t}$. Table 4 summarizes the detailed results between IRMS and these state-of-the-art algorithms on synthetic and real-world benchmark instances under the time limit $\hat{t} = 3600$ seconds.

Table 4 Comparison between IRMS and State-of-the-art Algorithms on Synthetic and Real-world Benchmarks under $\hat{t} = 3600$ Seconds

			CAN1°		MACNP°		VPMS°		CEMCNP°		**IRMS**	
Instance	K	BKV	$\hat{f}$	$\overline{f}$	$\hat{f}$	$\overline{f}$	$\hat{f}$	$\overline{f}$	$\hat{f}$	$\overline{f}$	$\hat{f}$	$\overline{f}$
BA500	50	195*	**195**	**195.0**	**195**	**195.0**	**195**	**195.0**	**195**	**195.0**	**195**	**195.0**
BA1000	75	558*	**558**	558.1	**558**	**558.0**	**558**	**558.0**	**558**	**558.0**	**558**	**558.0**
BA2500	100	3704*	**3704**	**3704.0**	**3704**	**3704.0**	**3704**	**3704.0**	**3704**	**3704.0**	**3704**	**3704.0**
BA5000	150	10196*	**10196**	**10196.0**	**10196**	**10196.0**	**10196**	**10196.0**	**10196**	**10196.0**	**10196**	**10196.0**
ER235	50	295	**295**	**295.0**	**295**	**295.0**	**295**	**295.0**	297	302.8	**295**	**295.0**
ER466	80	1524	**1524**	1524.4	**1524**	**1524.0**	**1524**	**1524.0**	1569	1630.6	**1524**	**1524.0**
ER941	140	5012	5102	5221.7	**5012**	**5014.5**	**5012**	5030.6	5363	5635.3	**5012**	5020.0
ER2344	200	902498	993035	1010337.9	**905472**	**922882.6**	909510	938362.3	1012527	1060618.6	920748	944406.9
FF250	50	194*	**194**	**194.0**	**194**	**194.0**	**194**	**194.0**	**194**	**194.0**	**194**	**194.0**
FF500	110	257*	262	265.2	**257**	**257.0**	**257**	**257.0**	**257**	258.6	**257**	**257.0**
FF1000	150	1260*	1262	1266.1	**1260**	1260.1	**1260**	**1260.0**	**1260**	**1260.0**	**1260**	**1260.0**
FF2000	200	4545*	4548	4551.9	**4545**	4545.6	**4545**	**4545.0**	4546	4552.5	**4545**	**4545.0**
WS250	70	3083	3491	3820.7	**3083**	3114.5	**3083**	**3090.6**	4203	5447.9	3085	3179.0
WS500	125	2072	2086	2102.5	**2072**	2084.1	2081	2084.9	2085	2193.0	**2072**	**2080.1**
WS1000	200	109677	138212	159031.0	115075	**136374.1**	**114066**	140033.7	154899	169877.2	138098	145969.1
WS1500	265	13098	13784	13997.7	13103	13203.5	**13098**	13216.4	13664	27810.6	**13098**	**13112.9**
Bovine	3	268	**268**	**268.0**	**268**	**268.0**	**268**	**268.0**	**268**	**268.0**	**268**	**268.0**
Circuit	25	2099	**2099**	**2099.0**	**2099**	**2099.0**	**2099**	**2099.0**	2101	2188.7	**2099**	**2099.0**
Ecoli	15	806	**806**	**806.0**	**806**	**806.0**	**806**	**806.0**	**806**	808.8	**806**	**806.0**
USAir97	33	4336	**4336**	**4336.0**	**4336**	4372.1	**4336**	5175.4	**4336**	5149.5	**4336**	4648.0
humanDi	52	1115	**1115**	**1115.0**	**1115**	**1115.0**	**1115**	**1115.0**	**1115**	**1115.0**	**1115**	**1115.0**
TreniR	26	918	**918**	**918.0**	**918**	**918.0**	**918**	**918.0**	**918**	**918.0**	**918**	**918.0**
EU_fli	119	348268	**348268**	348334.3	**348268**	353026.5	**348268**	350180.1	350762	357502.7	**348268**	**348295.7**
openfli	186	26783	29534	30149.7	28560	28808.2	**26874**	**28283.6**	29481	31377.7	27198	28757.5
yeast1	202	1412	1416	1418.6	**1412**	**1412.0**	**1412**	**1412.0**	**1412**	1414.3	**1412**	**1412.0**
H1000	100	306349	315911	318845.9	306960	311398.7	307117	310827.9	330493	337217.0	**306349**	**308951.9**
H2000	200	1242739	1274815	1294724.6	1251076	1272246.2	1242907	1258529.7	1324988	1344914.2	**1236503**⋆	**1254481.6**
H3000a	300	2840690	2914000	2927166.6	2885246	2922682.4	2849192	2877853.3	2962661	3042695.4	**2804579**⋆	**2849985.8**
H3000b	300	2837584	2902347	2926677.7	2870707	2927682.0	2839130	2863625.1	2968601	3035272.7	**2801186**⋆	**2842174.8**
H3000c	300	2835369	2899932	2926225.4	2863929	2909535.2	2837599	2861449.2	2956916	3008793.3	**2801692**⋆	**2840618.6**
H3000d	300	2828492	2899196	2930428.5	2865406	2924250.5	2833030	2870559.0	2967747	3030236.3	**2816590**⋆	**2864256.5**
H3000e	300	2843000	2919830	2937228.7	2875727	2934267.0	2845660	**2876424.3**	3012046	3043889.5	**2836177**⋆	2877807.4
H4000	400	5038611	5196850	5241007.7	5211868	5322566.9	5098049	5186904.9	5261850	5383301.3	**5021551**⋆	**5110687.5**
H5000	500	7964765	8181618	8221915.3	8369202	8506623.6	8078843	8217193.6	8206499	8348589.5	**8029837**	**8188900.3**
powergr	494	15862	16141	16276.4	15882	15917.5	15957	16014.3	15965	16084.9	**15866**	**15886.6**
Oclinks	190	611253	**611352**	614711.5	614467	614728.9	612314	615030.6	622237	626701.5	614467	**614467.6**
facebook	404	420334	**675630**	754777.4	711505	785554.5	713625	794649.1	794938	857245.1	719722	**741314.4**
grqc	524	13591	15544	15874.9	13599	13632.4	13628	13666.4	13666	13701.7	**13594**	**13613.0**
hepth	988	106276	136635	221421.8	124269	132192.5	115168	120040.4	**109504**	**111495.6**	115133	119766.8
hepph	1201	6155877	10059965	10724290.3	11114770	11972968.7	9664750	10580257.6	**8464199**	10530358.9	9401029	**9781789.8**
astroph	1877	53963375	60078332	61679547.3	61667966	62931865.7	59811734	61171775.9	59476077	60449655.3	**57592461**	**58649781.0**
condmat	2313	2298596	14140443	15233424.7	10101966	10829853.5	11141111	12186050.0	**3756761**	**4632078.6**	9670268	10789125.6
#Wins|Ties|Loses		–	26|14|2	31|10|1	17|21|4	23|14|5	16|20|6	21|16|5	26|13|3	31|9|2	–	–

Notes. "∗" presents the optimal solution, "⋆" indicates the improved best upper bounds, and "○" indicates the results of CNA1, MACNP and VPMS are obtained by executing their source codes, while the results of CEMCNP are obtained by executing our re-implemented CEMCNP algorithm, which are slightly different from their reported results.

From Table 4, we observe that IRMS competes very favorably with these state-of-the-art algorithms, by attaining seven new upper bounds and matching 22 best-known upper bounds. At a significance level of 0.05, IRMS significantly outperforms CAN1 both in terms of $\hat{f}$ (i.e., $33.0 > \mathrm{CV}^{42}_{0.05} = 27.0$) and $\overline{f}$ (i.e., $36.0 > \mathrm{CV}^{42}_{0.05}$). For the comparison between MACNP and IRMS, we can obtain similar observations. That is, IRMS is significantly better than MACNP in terms of $\hat{f}$ (i.e., $27.5 > \mathrm{CV}^{42}_{0.05}$), and it also outperforms MACNP in terms of $\overline{f}$ (i.e., $30.0 > \mathrm{CV}^{42}_{0.05}$). Compared to VPMS, IRMS wins in 26.0 instances in terms of $\hat{f}$, which is just slightly smaller than the critical value $\mathrm{CV}^{42}_{0.05} = 27.0$. For the $\overline{f}$ indicator, IRMS significantly outperforms VPMS, i.e., $29.0 > \mathrm{CV}^{42}_{0.05}$. For the comparison between IRMS and CEMCNP, we find that IRMS significantly outperforms CEMCNP both in terms of $\hat{f}$ (i.e., $32.5 > \mathrm{CV}^{42}_{0.05}$) and $\overline{f}$ (i.e., $35.5 > \mathrm{CV}^{42}_{0.05}$). These observations show that IRMS is highly effective compared to the state-of-the-art algorithms.

To further demonstrate the performance of IRMS, we report detailed results of IRMS under the longer time limit $\hat{t} = 7200$ seconds in Table 1 of the online supplement (Zhou et al. 2023b). We observe that IRMS improves its results with this extended time limit by finding two new upper bounds for the instances H5000 and grqc.

Finally, we note that IRMS has trouble to attain the best known results for facebook and condmat. On the other hand, we observe that these instances are challenging for almost all algorithms, except the ILS-N_1-FC algorithm of (Aringhieri et al. 2016b), which reported the current best known results for these two instances (under the time limits of 3000 and 16000 seconds, respectively), but does not perform so well on a number of other instances, as shown in Table 5 of (Aringhieri et al. 2016a).

5. Application to the Node-weighted Critical Node Problem

To show that our IRMS method may be applied to solve other optimization problems, we consider the node-weighted critical node problem (NWCNP) (Chen et al. 2020, Zhou et al. 2021c), which consists of minimizing the pairwise connectivity of a given node-weighted graph by removing a subset of nodes subject to a budgetary constraint. We start with an introduction of NWCNP, followed by reporting a detailed comparative results between IRMS and existing methods.

5.1. Node-weighted Critical Node Problem

NWCNP is a node-weighted version of CNP, which aims to minimize the objective function (1) as well as CNP, and simultaneously satisfies the following budgetary constraint (6).

$$\text{Subset to} \sum_{i=1}^{|S|} w(v_{S(i)}) \leqslant K \tag{6}$$

where the terms $w(v_{S(i)}) > 0$ are positive weights associated with each node and $K > 0$ is a predefined budget limit.

Note that CNP can be considered as a special case of NWCNP where the weight of each node is set to 1, i.e., $w(v_i) = 1, \forall v_i \in V$. As pointed out by Zhou et al. (2021c), an optimal solution of CNP is not necessarily optimal for NWCNP, and NWCNP is at least as challenging computationally as CNP (Arulselvan et al. 2009). Recently, some efforts have been devoted to solve it. For example, Chen et al. (2020) studied CNP in undirected weighted networks, and proposed a mixed-integer quadratic programming model and a greedy algorithm. Zhou et al. (2021c) introduced an iterative local search algorithm (ILS-NWCNP for short) by iterating through a late acceptance-based local search and a destructive-constructive perturbation, which achieved the state-of-the-art results for NWCNP.

5.2. Computational Results on NWCNP

As benchmark instances for NWCNP are not available, we generated new weighted instances starting from the widely used synthetic and real-world CNP benchmarks. The only information required to be added is the weighting information for each node in the sparse graph. Following (Zhou et al. 2021c), we adopt a random weighting scheme to assign a weight to each node given by $w(v_i) \in [0.2, 3], \forall v_i \in V$. Please refer to Zhou et al. (2023b) for both our implemented programs and generated benchmark instances.

To adapt IRMS to solve NWCNP, some modifications to main algorithmic modules (e.g., solution initialization and CHNS) of IRMS are necessary. According to the budgetary constraint (6), a feasible solution of NWCNP must satisfy the constraint dictating that the total weight of all removed nodes should be no more than K. Therefore, an initial solution is constructed by iteratively removing nodes from the graph until the total weight of all removed nodes is large than K. Note that a feasible solution of NWCNP is not necessary to have K nodes. In addition, at each iteration, CHNS selects a removed node according to both impact function value ψ and node weight w instead of value ψ only.

We use IRMS-NWCNP to denote the resulting IRMS algorithm for NWCNP. Note that the source code of ILS-NWCNP is available to us, which is responsible for achieving most of state-of-the-art results for NWCNP in the literature. While for CEM-NWCNP, it is a new adaption of CEMCNP (Wang and Di 2022) for NWCNP. Correspondingly, the resulting algorithm for NWCNP is denoted as CEM-NWCNP. Therefore, we focus on experimentally comparing IRMS-NWCNP with both ILS-NWCNP and CEM-NWCNP and report the results in Table 5.

From Table 5, we observe that IRMS-NWCNP outperforms ILS-NWCNP in terms of both $\hat{f}$ and $\bar{f}$, finding better $\hat{f}$ values for 37 instances, and equal $\hat{f}$ values for 2 out of the 5 remaining instances. In terms of $\bar{f}$, IRMS-NWCNP achieves better results than ILS-NWCNP except for three instances (i.e., WS1000, WS1500 and facebook). At a significance level of 0.05, IRMS-NWCNP significantly outperforms ILS-NWCNP in terms of both $\hat{f}$ (i.e., $38 > CV_{0.05}^{42}$) and $\bar{f}$ (i.e., $39 > CV_{0.05}^{42}$). Compared to CEM-NWCNP, IRMS-NWCNP also shows better performance. At a significance level of 0.05, IRMS-NWCNP is significantly better than CEM-NWCNP in terms of $\hat{f}$, i.e., $34.5 > CV_{0.05}^{42}$. While for the performance indicator $\bar{f}$, IRMS-NWCNP also significantly outperforms CEM-NWCNP (i.e., $37.5 > CV_{0.05}^{42}$).

6. Experimental Analysis

In this section, we perform additional testing to gain a deeper understanding of IRMS, by conducting three groups of experiments: 1) to study the run-time distributions of IRMS and state-of-the-art algorithms, 2) to investigate the benefit of the component-based hybrid neighborhood search procedure, and 3) to confirm the effectiveness of the "reduce-solve-combine" mechanism.

6.1. Run-time Distributions of IRMS and State-of-the-art Algorithms

To further compare IRMS with the three state-of-the-art algorithms, we employ time-to-target (TTT) plots (Aiex et al. 2007) to show the algorithmic run-time distributions on representative instances. We execute each algorithm 100 times for each instance and record the computation time to obtain a solution at least as good as a given target value at each run. The 100 computation times are sorted in ascending order, and a probability $p_i = \frac{i-0.5}{100}$ is associated with the i-th sorted computation time t_i. A TTT plot is then obtained by plotting these 100 points $(t_i, p_i), i = 1, 2, \ldots, 100$. Figure 5 presents the TTT plots of IRMS

Table 5 Comparison between IRMS-NWCNP and Reference Algorithms on Weighted Synthetic and Real-world Benchmarks under $\hat{t} = 3600$ Seconds

		ILS-NWCNP°			CEM-NWCNP			**IRMS-NWCNP**		
Instance	K	$\hat{f}$	$\overline{f}$	$\overline{t}$	$\hat{f}$	$\overline{f}$	$\overline{t}$	$\hat{f}$	$\overline{f}$	$\overline{t}$
BA500	50	283	286.7	65.0	278	280.7	1.1	**269**	**271.4**	1946.1
BA1000	75	820	835.8	703.4	819	829.3	3.2	**815**	**823.4**	2098.5
BA2500	100	4910	4973.2	1153.1	**4784**	4909.7	55.8	4825	**4884.4**	2544.0
BA5000	150	13722	13907.4	162.6	**13346**	**13640.3**	535.3	13672	13797.1	2600.0
ER235	50	724	1495.7	680.4	922	1591.0	0.2	**616**	**638.0**	2081.8
ER466	80	19104	23745.6	92.8	19870	27351.1	0.6	**4274**	**5117.9**	1989.0
ER941	140	63079	78285.9	1407.5	55194	78522.3	35.7	**38654**	**43229.0**	3073.9
ER2344	200	1397614	1491688.6	2885.2	1495253	1629017.5	69.5	**1334743**	**1355718.0**	2845.2
FF250	50	479	514.8	196.3	522	547.0	0.1	**466**	**475.2**	1661.5
FF500	110	532	556.4	356.9	561	580.4	1.5	**514**	**525.5**	2992.4
FF1000	150	2469	2509.1	1918.0	**2345**	2494.5	11.5	2374	**2402.1**	2912.4
FF2000	200	8715	8822.4	824.0	**7837**	**8212.2**	180.2	8435	8535.9	2731.7
WS250	70	8801	10270.9	414.2	9724	14414.2	205.8	**8304**	**8468.5**	1408.9
WS500	125	5924	6553.2	785.4	13810	25821.4	4.2	**4679**	**5049.9**	3081.1
WS1000	200	**222092**	**228611.7**	2048.3	277755	311422.1	628.7	247187	260977.6	2721.2
WS1500	265	**78142**	**95715.1**	2577.0	236725	344796.0	58.9	134518	158567.2	2409.3
Bovine	3	**954**	954.6	237.5	**954**	**954.0**	0.0	**954**	**954.0**	0.1
Circuit	25	3498	4254.6	395.8	4718	7253.7	0.1	**3160**	**3204.0**	1826.1
Ecoli	15	**1348**	1360.0	759.4	1368	1591.4	0.3	**1348**	**1348.0**	0.5
USAir97	33	9484	9545.0	865.5	9441	9955.0	0.7	**9313**	**9345.1**	978.3
humanDi	52	1843	1867.2	575.7	1727	1826.1	2.0	**1693**	**1700.7**	1824.5
TreniR	26	647	648.8	962.9	627	695.3	0.6	**597**	**634.7**	2537.0
EU_flights	119	395636	399652.6	955.0	400991	407205.3	75.3	**386809**	**388392.8**	2144.6
openflights	186	91072	95801.6	2093.8	95097	103457.7	183.6	**87420**	**87939.6**	2474.7
yeast1	202	3569	3700.2	1344.7	3320	3495.9	206.2	**3317**	**3373.0**	2917.6
H1000	100	345971	353678.8	1005.0	381567	398257.1	168.7	**317208**	**325771.1**	2908.5
H2000	300	1412216	1444356.3	490.4	1574504	1608151.6	743.4	**1331567**	**1356273.3**	2889.0
H3000a	300	3146558	3234620.3	1654.6	3581933	3690692.4	899.9	**3098206**	**3133268.5**	2755.0
H3000b	300	3169598	3224242.6	1214.0	3614080	3679776.8	1056.7	**3047488**	**3077125.0**	2991.0
H3000c	300	3180822	3224308.1	1892.6	3523237	3650634.8	759.3	**3074841**	**3118908.1**	2553.1
H3000d	300	3159049	3205091.0	1519.1	3499408	3635271.7	970.8	**3089828**	**3124039.0**	2875.4
H3000e	300	3156667	3221055.5	1160.7	3630227	3660257.7	1282.0	**3097630**	**3145617.3**	2877.8
H4000	400	5650385	5712357.8	2014.2	6435143	6539854.0	1428.0	**5563646**	**5629050.2**	2868.9
H5000	500	8915291	8999620.4	2593.4	10190504	10268278.5	1724.9	**8799566**	**8950070.3**	2973.4
powergrid	494	36639	41493.8	2019.5	**28702**	**30541.8**	2479.8	36395	37144.5	2793.2
Oclinks	190	798262	807393.5	1991.5	812214	826705.5	604.8	**775689**	**782060.6**	2851.2
facebook	404	**1444026**	**1469417.4**	3095.0	1444559	2011511.2	2224.1	1570192	1585472.6	2862.8
grqc	524	47604	52548.5	1720.8	**41914**	**46216.3**	2381.1	44664	47360.7	2747.7
hepth	988	8016122	8225523.5	3436.0	8924436	9908367.9	3524.0	**6934032**	**7070319.2**	2714.5
hepph	1201	25775334	26101767.5	3399.9	28084670	30582642.5	3361.6	**23594695**	**24177629.2**	2851.7
astroph	1877	82976199	84203164.2	3568.2	96874011	100752411.9	3532.4	**77314231**	**78730627.0**	2991.2
condmat	2313	72014730	75003163.3	3555.2	93241452	97411338.5	3580.1	**60805637**	**66817701.5**	3365.8
#Wins|Ties|Loses		37|2|3	39|0|3	–	34|1|7	37|1|4	–	–	–	–

Note. "○" presents the results of ILS-NWCNP are obtained by executing its source code in our computational platform.

and the three state-of-the-art algorithms on four representative instances, i.e., BA1000 (10196), FF1000 (1271), humanDi (1115) and yeast1 (1421). Note that the target value of each instance is indicated in the parentheses after each instance name.

From Figure 5, we observe that IRMS is likely to find a target solution faster than the compared algorithms. For example, for the synthetic instance BA1000, the probability of reaching the target value 10196 in at most 130 seconds is approximately 60% for both CAN1 and VPMS, and 100% for both MACNP and IRMS. For the real-world instance humanDi,

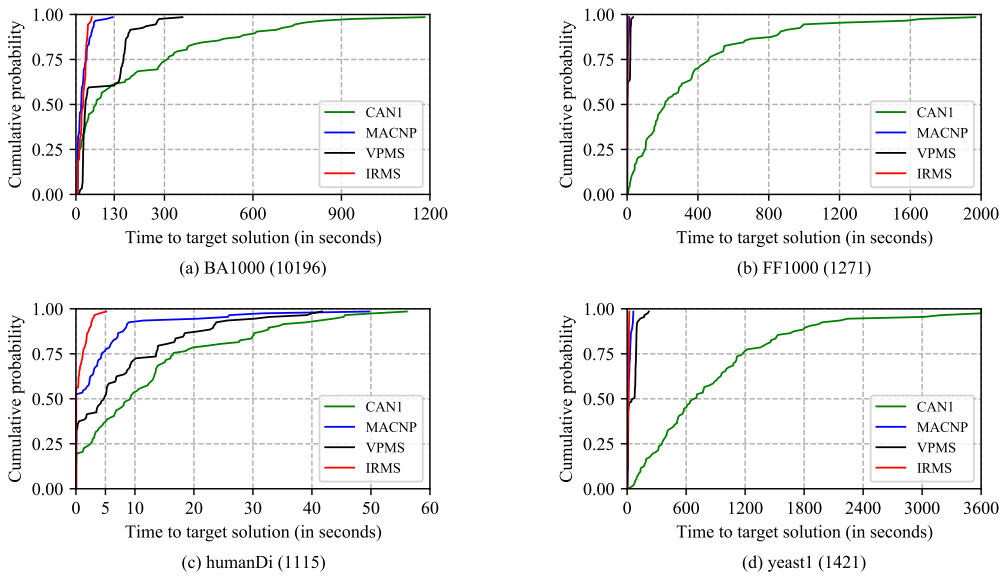


Figure 5 **TTT Plots of IRMS and State-of-the-art Algorithms**

the probability of finding the target value 1115 in at most 5 seconds is approximately 40% for CAN1, 50% for VPMS and 75% for MACNP, while it is at least 95% for IRMS. These observations further confirm that IRMS outperforms the state-of-the-art algorithms.

6.2. Benefit of the Component-based Hybrid Neighborhood Search

As previously noted, IRMS employs component-based hybrid neighborhood search (CHNS) to perform local optimization, using the articulation point impact strategy to improve on the component-based neighborhood search (CBNS) algorithm (Zhou et al. 2019). To show the benefit of CHNS, we experimentally compare IRMS with a variant named IRMS′ that is obtained from IRMS by replacing CHNS with CBNS. That is, IRMS′ selects a node based on the node weighting strategy rather than the articulation point impact strategy during the search. Detailed comparative results between IRMS′ and IRMS on both the synthetic and real-world benchmarks are summarized in Table 6.

Table 6 shows that IRMS dominates IRMS′ by achieving better results on 19 instances and equal results on the 22 remaining instances in terms of $\hat{f}$. IRMS statistically beats IRMS′ on 30.0 instances (i.e., $30.0 > \mathrm{CV}^{42}_{0.05}$) at a significance level of 0.05. For the $\overline{f}$ indicator, IRMS finds better results on 24 instances, and equal results on the 16 remaining instances. At a significance level of 0.05, IRMS significantly outperforms IRMS′ (i.e., $32.0 > \mathrm{CV}^{42}_{0.05}$). These results confirm the benefit of CHNS over CBNS.

Table 6 Comparison between IRMS and IRMS′ on Synthetic and Real-world Benchmarks under $\hat{t} = 3600$ Seconds

			IRMS′			**IRMS**		
Instance	K	BKV	$\hat{f}$	$\overline{f}$	$\overline{t}$	$\hat{f}$	$\overline{f}$	$\overline{t}$
BA500	50	195*	**195**	**195.0**	0.0	**195**	**195.0**	0.0
BA1000	75	558*	**558**	**558.0**	1.3	**558**	**558.0**	3.1
BA2500	100	3704*	**3704**	**3704.0**	5.3	**3704**	**3704.0**	3.9
BA5000	150	10196*	**10196**	**10196.0**	15.8	**10196**	**10196.0**	16.8
ER235	50	295	**295**	**295.0**	2.8	**295**	**295.0**	6.5
ER466	80	1524	**1524**	**1524.0**	651.6	**1524**	**1524.0**	83.7
ER941	140	5012	**5012**	5048.1	1724.0	**5012**	**5020.0**	520.0
ER2344	200	902498	929333	951425.0	3010.4	**920748**	**944406.9**	3146.7
FF250	50	194*	**194**	**194.0**	0.1	**194**	**194.0**	0.0
FF500	110	257*	**257**	**257.0**	0.8	**257**	**257.0**	1.4
FF1000	150	1260*	**1260**	**1260.0**	27.0	**1260**	**1260.0**	22.2
FF2000	200	4545*	**4545**	**4545.0**	115.4	**4545**	**4545.0**	207.3
WS250	70	3083	3324	3600.2	2067.5	**3085**	**3179.0**	2013.2
WS500	125	2072	**2072**	**2079.0**	436.4	**2072**	2080.1	297.7
WS1000	200	109677	244708	274428.2	1567.4	**138098**	**145969.1**	1963.2
WS1500	265	13098	13100	13121.2	1670.6	**13098**	**13112.9**	2028.7
Bovine	3	268	**268**	**268.0**	0.0	**268**	**268.0**	0.0
Circuit	25	2099	**2099**	**2099.0**	0.8	**2099**	**2099.0**	1.3
Ecoli	15	806	**806**	**806.0**	0.0	**806**	**806.0**	0.4
USAir97	33	4336	**4336**	4736.9	1365.0	**4336**	**4648.0**	668.8
HumanDi	52	1115	**1115**	**1115.0**	4.4	**1115**	**1115.0**	0.1
TreniR	26	918	**918**	**918.0**	1.8	**918**	**918.0**	2.5
EU_fli	119	348268	**348268**	349321.2	1683.3	**348268**	**348295.7**	998.0
openfli	186	26783	28718	28880.0	1946.7	**27198**	**28757.5**	1695.8
yeast1	202	1412	**1412**	**1412.0**	51.9	**1412**	**1412.0**	37.8
H1000	100	306349	308299	310940.3	2576.1	**306349**	**308951.9**	2165.2
H2000	200	1242739	1271562	1310345.4	2422.2	**1236503**	**1254481.6**	3028.8
H3000a	300	2840690	2914963	2992166.4	2528.1	**2804579**	**2849985.8**	3088.5
H3000b	300	2837584	2929494	3011990.2	2692.4	**2801186**	**2842174.8**	3164.3
H3000c	300	2835369	2925015	2983251.7	2343.5	**2801692**	**2840618.6**	3066.0
H3000d	300	2828492	2938989	3007547.3	2451.9	**2816590**	**2864256.5**	2940.0
H3000e	300	2843000	2960196	3014327.4	2783.2	**2836177**	**2877807.4**	2715.4
H4000	400	5038611	5313533	5421596.1	2641.3	**5021551**	**5110687.5**	3042.0
H5000	500	7964765	8297260	8575610.7	2392.2	**8029837**	**8188900.3**	2741.0
powergr	494	15862	**15866**	15891.3	3171.9	**15866**	**15886.6**	3021.6
Oclinks	190	611253	**614467**	614651.4	1676.8	**614467**	**614467.6**	1038.9
facebook	404	420334	919158	1258604.9	2991.4	**719722**	**741314.4**	2852.1
grqc	524	13591	13601	13626.6	3147.3	**13594**	**13613.0**	3201.9
hepth	988	106276	116527	**119693.3**	3228.1	**115133**	119766.8	3159.4
hepph	1201	6155877	11353914	11951940.6	2447.0	**9401029**	**9781789.8**	3077.4
astroph	1877	53963375	62237249	63050133.4	2242.8	**57592461**	**58649781.0**	3271.4
condmat	2313	2298596	**9642594**	10902735.5	2233.3	9670268	**10789125.6**	2286.2
#Wins|Ties|Loses	–	–	19|22|1	24|16|2	–	–	–	–

Note. "$*$" presents the optimal solution.

6.3. Effectiveness of the "Reduce-Solve-Combine" Mechanism

To evaluate the effectiveness of the "reduce-solve-combine" (RSC) mechanism used by IRMS, we compare IRMS with an alternative version called IRMS″ obtained from IRMS by disabling the RSC mechanism and directly constructing an offspring solution based on the frequent pattern and the offspring construction method used in frequent pattern based search (Zhou et al. 2022). Table 7 describes the comparative results between IRMS

and IRMS″ on both the synthetic and real-world instances under the time limit $\hat{t} = 3600$ seconds.

Table 7 Comparison between IRMS (with "reduce-solve-recombine" mechanism) and IRMS″ (without "reduce-solve-combine" mechanism) on Synthetic and Real-world Benchmarks under $\hat{t} = 3600$ Seconds

			IRMS″			**IRMS**		
Instance	K	BKV	$\hat{f}$	$\overline{f}$	$\overline{t}$	$\hat{f}$	$\overline{f}$	$\overline{t}$
BA500	50	195*	**195**	**195.0**	0.0	**195**	**195.0**	0.0
BA1000	75	558*	**558**	**558.0**	0.5	**558**	**558.0**	3.1
BA2500	100	3704*	**3704**	**3704.0**	2.7	**3704**	**3704.0**	3.9
BA5000	150	10196*	**10196**	**10196.0**	26.1	**10196**	**10196.0**	16.8
ER235	50	295	**295**	**295.0**	1.8	**295**	**295.0**	6.5
ER466	80	1524	**1524**	**1524.0**	240.9	**1524**	**1524.0**	83.7
ER941	140	5012	**5012**	5076.0	1929.7	**5012**	5020.0	520.0
ER2344	200	902498	965168	991942.3	2190.0	**920748**	**944406.9**	3146.7
FF250	50	194*	**194**	**194.0**	0.1	**194**	**194.0**	0.0
FF500	110	257*	**257**	**257.0**	0.6	**257**	**257.0**	1.4
FF1000	150	1260*	**1260**	**1260.0**	24.3	**1260**	**1260.0**	22.2
FF2000	200	4545*	**4545**	**4545.0**	388.4	**4545**	**4545.0**	207.3
WS250	70	3083	3090	3275.8	1678.5	**3085**	**3179.0**	2013.2
WS500	125	2072	**2072**	2080.9	1769.3	**2072**	**2080.1**	297.7
WS1000	200	109677	141667	146811.2	2039.8	**138098**	**145969.1**	1963.2
WS1500	265	13098	13858	14144.3	1888.9	**13098**	**13112.9**	2028.7
Bovine	3	268	**268**	**268.0**	0.0	**268**	**268.0**	0.0
Circuit	25	2099	**2099**	**2099.0**	0.5	**2099**	**2099.0**	1.3
Ecoli	15	806	**806**	**806.0**	0.0	**806**	**806.0**	0.4
USAir97	33	4336	**4336**	4674.0	314.4	**4336**	**4648.0**	668.8
HumanDi	52	1115	**1115**	**1115.0**	0.3	**1115**	**1115.0**	0.1
TreniR	26	918	**918**	**918.0**	0.4	**918**	**918.0**	2.5
EU_fli	119	348268	**348268**	348350.8	1705.8	**348268**	**348295.7**	998.0
openfli	186	26783	29118	29497.6	2059.9	**27198**	**28757.5**	1695.8
yeast1	202	1412	**1412**	**1412.0**	52.7	**1412**	**1412.0**	37.8
H1000	100	306349	312592	316868.3	2392.4	**306349**	**308951.9**	2165.2
H2000	200	1242739	1283398	1303858.8	2059.1	**1236503**	**1254481.6**	3028.8
H3000a	300	2840690	2909699	2961854.3	1670.9	**2804579**	**2849985.8**	3088.5
H3000b	300	2837584	2927637	2959051.0	1520.3	**2801186**	**2842174.8**	3164.3
H3000c	300	2835369	2906920	2948294.3	2130.8	**2801692**	**2840618.6**	3066.0
H3000d	300	2828492	2935313	2965419.8	2146.9	**2816590**	**2864256.5**	2940.0
H3000e	300	2843000	2927917	2964465.5	2202.5	**2836177**	**2877807.4**	2715.4
H4000	400	5038611	5246420	5317627.6	1881.6	**5021551**	**5110687.5**	3042.0
H5000	500	7964765	8289948	8388331.1	1929.4	**8029837**	**8188900.3**	2741.0
powergr	494	15862	15938	15990.4	2692.5	**15866**	**15886.6**	3021.6
Oclinks	190	611253	**614467**	**614467.1**	1402.9	**614467**	614467.6	1038.9
facebook	404	420334	755714	805713.6	1894.0	**719722**	**741314.4**	2852.1
grqc	524	13591	13635	13652.6	2764.6	**13594**	**13613.0**	3201.9
hepth	988	106276	126106	137745.1	1652.7	**115133**	**119766.8**	3159.4
hepph	1201	6155877	10255932	11239245.2	2946.7	**9401029**	**9781789.8**	3077.4
astroph	1877	53963375	62169348	63174914.0	2410.7	**57592461**	**58649781.0**	3271.4
condmat	2313	2298596	9943424	10804865.5	2416.2	**9670268**	**10789125.6**	2286.2
#Wins|Ties|Loses	–	–	21|21|0	25|16|1	–	–	–	–

Note. "∗" presents the optimal solution.

As seen from Table 7, IRMS demonstrates a better performance than IRMS″ by obtaining better results on 21 instances, and equal results on the 21 remaining instances in terms of $\hat{f}$. Similar observations apply to the $\overline{f}$ indicator, where IRMS attains the same or better results on all 42 instances except for the instance Oclinks. At a significance level of 0.05,

IRMS significantly outperforms IRMS″ in terms of both the $\hat{f}$ (i.e., $31.5 > \mathrm{CV}^{42}_{0.05}$) and $\overline{f}$ (i.e., $33.0 > \mathrm{CV}^{42}_{0.05}$) indicators, confirming the effectiveness of RSC.

7. Conclusion

Finding an optimal set of nodes, called critical nodes, whose removal will maximally decrease the pairwise connectivity of the remaining graph, is a fundamental critical node detection problem. To solve this problem, we propose an instance reduction-based memetic search (IRMS) method that integrates a "reduce-solve-combine" instance reduction mechanism with the well-known population-based memetic algorithm framework. Extensive experimental results on 42 synthetic and real-world benchmark instances show that IRMS is highly effective compared to the state-of-the-art heuristic algorithms, by discovering 9 new upper bounds. Investigations are also performed that identify the benefit of different search modules and techniques used by the IRMS algorithm. In addition, we report computational results that demonstrate a generalization of IRMS likewise outperforms the previous best algorithm for the node-weighted critical node problem. The updated upper bounds can be useful for future research.

As future work, several potential research directions can be pursued. First, it would be interesting to optimally solve the reduced instance by an exact solver instead of approximately solving by a heuristic solver in the "reduce-solve-combine" module. Second, the "reduce-solve-combine" mechanism being a general-purpose technique for the instance reduction, its generality can be further verified by combining it with other metaheuristics, such as large neighborhood search (Schaap et al. 2022) and path relinking (Wu et al. 2020). Third, it is worth adapting IRMS to solver other problem variants, such as the distance-based critical node problem (i.e., there is a cost associated to each pair of nodes in the residual graph) (Salemi and Buchanan 2022, Zhou et al. 2023c) and connected critical node problem (Hosteins et al. 2022). Finally, this work could benefit exact algorithms. For instance, we can employ IRMS to generate a high-quality initial solution for a given instance, whose objective function value is used as a tight starting upper bound.

Acknowledgments

The authors would like to thank the editors and the anonymous referees for their insightful comments and suggestions which helped us to significantly improve this paper. This work was supported by the National Natural Science Foundation of China under Grant 61903144.

References

Abbas K, Abbasi A, Shi D, Ling N, Yu L, Chen B, Cai S, Hasan Q (2021) Application of network link prediction in drug discovery. *BMC Bioinform.* 22(1):187.

Addis B, Aringhieri R, Grosso A, Hosteins P (2016) Hybrid constructive heuristics for the critical node problem. *Ann. Oper. Res.* 238(1-2):637–649.

Aiex RM, Resende MG, Ribeiro CC (2007) TTT plots: A perl program to create time-to-target plots. *Optim. Lett.* 1(4):355–366.

Archetti C, Boland N, Speranza MG (2017) A matheuristic for the multivehicle inventory routing problem. *INFORMS J. Comput.* 29(3):377–387.

Aringhieri R, Grosso A, Hosteins P, Scatamacchia R (2016a) A general evolutionary framework for different classes of critical node problems. *Eng. Appl. Artif. Intell.* 55:128–145.

Aringhieri R, Grosso A, Hosteins P, Scatamacchia R (2016b) Local search metaheuristics for the critical node problem. *Networks* 67(3):209–221.

Arnold F, Santana Í, Sörensen K, Vidal T (2021) PILS: Exploring high-order neighborhoods by pattern mining and injection. *Pattern Recogn.* 116:107957.

Arulselvan A, Commander CW, Elefteriadou L, Pardalos PM (2009) Detecting critical nodes in sparse graphs. *Comput. Oper. Res.* 36(7):2193–2200.

Baggio A, Carvalho M, Lodi A, Tramontani A (2021) Multilevel approaches for the critical node problem. *Oper. Res.* 69(2):486–508.

Blum C, Davidson PP, López-Ibáñez M, Lozano JA (2016) Construct, merge, solve & adapt A new general algorithm for combinatorial optimization. *Comput. Oper. Res.* 68:75–88.

Boschetti MA, Maniezzo V (2022) Matheuristics: using mathematics for heuristic design. *4OR* 20(2):173–208.

Cai S, Su K, Sattar A (2011) Local search with edge weighting and configuration checking heuristics for minimum vertex cover. *Artif. Intell.* 175(9-10):1672–1696.

Chen W, Jiang M, Jiang C, Zhang J (2020) Critical node detection problem for complex network in undirected weighted networks. *Physica A: Stat. Mech. Appl.* 538:122862.

Chen Y, Hao JK (2014) A "reduce and solve" approach for the multiple-choice multidimensional knapsack problem. *Eur. J. Oper. Res.* 239(2):313–322.

Cooper WL, Homem-de-Mello T (2007) Some decomposition methods for revenue management. *Transp. Sci.* 41(3):332–353.

Cordeau J, Gaudioso M, Laporte G, Moccia L (2006) A memetic heuristic for the generalized quadratic assignment problem. *INFORMS J. Comput.* 18(4):433–443.

de Holanda Maia MR, Plastino A, Penna PHV (2020) Minereduce: an approach based on data mining for problem size reduction. *Comput. Oper. Res.* 122:104995.

de San Lázaro IM, Sánchez-Oro J, Duarte A (2021) Finding critical nodes in networks using variable neighborhood search. *Proc. ICVNS 2021*, 1–13.

Delgadillo FJD, Montiel O, Sepúlveda R (2016) Reducing the size of traveling salesman problems using vaccination by fuzzy selector. *Expert Syst. Appl.* 49:20–30.

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* 7(1):1–30.

Di Summa M, Grosso A, Locatelli M (2012) Branch and cut algorithms for detecting critical nodes in undirected graphs. *Comput. Optim. Appl.* 53(3):649–680.

Doostmohammadian M, Rabiee HR, Khan UA (2020) Centrality-based epidemic control in complex social networks. *Soc. Netw. Anal. Min.* 10(1):32.

Fu ZH, Hao JK (2015) Dynamic programming driven memetic search for the steiner tree problem with revenues, budget, and hop constraints. *INFORMS J. Comput.* 27(2):221–237.

Glover FW (1997) A template for scatter search and path relinking. Hao JK, Lutton E, Ronald EMA, Schoenauer M, Snyers D, eds., *Proc. AE 1997*, 1–51 (Springer).

Grahne G, Zhu J (2005) Fast algorithms for frequent itemset mining using fp-trees. *IEEE Trans. Knowl. Data Eng.* 17(10):1347–1362.

Hao JK, Wu Q (2012) Improving the extraction and expansion method for large graph coloring. *Discret. Appl. Math.* 160(16-17):2397–2407.

Hopcroft J, Tarjan R (1973) Algorithm 447: Efficient algorithms for graph manipulation. *Commun. ACM* 16(6):372–378, ISSN 0001-0782.

Hosteins P, Scatamacchia R, Grosso A, Aringhieri R (2022) The connected critical node problem. *Theor. Comput. Sci.* 923:235–255, ISSN 0304-3975.

Kenny A, Li X, Ernst AT (2018) A merge search algorithm and its application to the constrained pit problem in mining. Aguirre HE, Takadama K, eds., *Proc. GECCO 2018*, 316–323 (ACM).

Kenny A, Li X, Ernst AT, Sun Y (2019) An improved merge search algorithm for the constrained pit problem in open-pit mining. *Proc. GECCO 2019*, 294–302 (ACM).

Le HL, Neri F, Triguero I (2022) SPMS-ALS: A single-point memetic structure with accelerated local search for instance reduction. *Swarm Evol. Comput.* 69:100991.

Luna JM, Fournier-Viger P, Ventura S (2019) Frequent itemset mining: A 25 years review. *WIREs Data Mining Knowl. Discov.* 9(6):e1329.

Mihic K, Ryan K, Wood A (2018) Randomized decomposition solver with the quadratic assignment problem as a case study. *INFORMS J. Comput.* 30(2):295–308.

Montiel O, Delgadillo FJD, Sepúlveda R (2013) Combinatorial complexity problem reduction by the use of artificial vaccines. *Expert Syst. Appl.* 40(5):1871–1879.

Mugisha S, Zhou HJ (2016) Identifying optimal targets of network attack by belief propagation. *Phys. Rev. E* 94:012305.

Nabli A, Carvalho M (2020) Curriculum learning for multilevel budgeted combinatorial problems. *Adv. Neural Inf. Process. Syst.* 33:7044–7056.

Naoum-Sawaya J, Buchheim C (2016) Robust critical node selection by benders decomposition. *INFORMS J. Comput.* 28(1):162–174.

Neri F, Cotta C (2012) Memetic algorithms and memetic computing optimization: A literature review. *Swarm Evol. Comput.* 2:1–14, ISSN 2210-6502.

Nguyen DT, Shen Y, Thai MT (2013) Detecting critical nodes in interdependent power networks for vulnerability assessment. *IEEE Trans. Smart Grid* 4(1):151–159.

Plastino A, Barbalho H, Santos LFM, Fuchshuber R, Martins SL (2014) Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic. *J. Heuristics* 20(1):39–74.

Pullan W (2015) Heuristic identification of critical nodes in sparse real-world graphs. *J. Heuristics* 21(5):577–598.

Purevsuren D, Cui G, Qu M, Win NH (2017) Hybridization of GRASP with exterior path relinking for identifying critical nodes in graphs. *IAENG Int. J. Comput. Sci.* 44(2).

Rezaei J, Zare-Mirakabad F, MirHassani SA, Marashi S (2021) EIA-CNDP: an exact iterative algorithm for critical node detection problem. *Comput. Oper. Res.* 127:105138.

Ribeiro MH, Plastino A, Martins SL (2006) Hybridization of GRASP metaheuristic with data mining techniques. *J. Math. Model. Algorithms* 5(1):23–41.

Salemi H, Buchanan A (2022) Solving the distance-based critical node problem. *INFORMS J. Comput.* 34(3):1309–1326.

Schaap H, Schiffer M, Schneider M, Walther G (2022) A large neighborhood search for the vehicle routing problem with multiple time windows. *Transp. Sci.* 56(5):1369–1392.

Subramanyam A, Gounaris CE (2018) A decomposition algorithm for the consistent traveling salesman problem with vehicle idling. *Transp. Sci.* 52(2):386–401.

Tarjan R (1972) Depth-first search and linear graph algorithms. *SIAM J. Sci. Comput.* 1(2):146–160.

Thornton J, Pham DN, Bain S, Ferreira Jr V (2004) Additive versus multiplicative clause weighting for SAT. McGuinness DL, Ferguson G, eds., *Proc. AAAI 2004*, 191–196.

Ventresca M (2012) Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem. *Comput. Oper. Res.* 39(11):2763–2775.

Ventresca M, Aleman D (2014) A derandomized approximation algorithm for the critical node detection problem. *Comput. Oper. Res.* 43:261–270.

Ventresca M, Aleman D (2015) Efficiently identifying critical nodes in large complex networks. *Comput. Soc. Netw.* 2(1):1–16.

Veremyev A, Boginski V, Pasiliao EL (2014a) Exact identification of critical nodes in sparse networks via new compact formulations. *Optim. Lett.* 8(4):1245–1259.

Veremyev A, Prokopyev OA, Pasiliao EL (2014b) An integer programming framework for critical elements detection in graphs. *J. Comb. Optim.* 28(1):233–273.

Veremyev A, Prokopyev OA, Pasiliao EL (2019) Finding critical links for closeness centrality. *INFORMS J. Comput.* 31(2):367–389.

Vitoriano B, Ortuño MT, Tirado G, Montero J (2011) A multi-criteria optimization model for humanitarian aid distribution. *J. Global Optim.* 51(2):189–208.

Wang Z, Di Y (2022) Cluster expansion method for critical node problem based on contraction mechanism in sparse graphs. *IEICE Trans. Inf. Syst.* 105-D(6):1135–1149.

Wu Q, Hao JK (2012) Coloring large graphs based on independent set extraction. *Comput. Oper. Res.* 39(2):283–290.

Wu Q, Hao JK (2013) An extraction and expansion approach for graph coloring. *Asia Pac. J. Oper. Res.* 30(5):1350018.

Wu Q, Wang Y, Glover F (2020) Advanced tabu search algorithms for bipartite boolean quadratic programs guided by strategic oscillation and path relinking. *INFORMS J. Comput.* 32(1):74–89.

Zhang Y, Chen Q, Chen B, Liu J, Zheng H, Yao H, Zhang C (2020) Identifying hotspots of sectors and supply chain paths for electricity conservation in china. *J. Clean. Prod.* 251:119653.

Zhang Y, Mei Y, Zhang B, Jiang K (2021) Divide-and-conquer large scale capacitated arc routing problems with route cutting off decomposition. *Inf. Sci.* 553:208–224.

Zheng Y, Xue J (2010) A problem reduction based approach to discrete optimization algorithm design. *Computing* 88(1):31–54.

Zhou Y, Hao JK, Duval B (2022) Frequent pattern-based search: A case study on the quadratic assignment problem. *IEEE Trans. Syst. Man Cybern. Syst.* 52(3):1503–1515.

Zhou Y, Hao JK, Fu ZH, Wang Z, Lai X (2021a) Variable population memetic search: A case study on the critical node problem. *IEEE Trans. Evol. Comput.* 25(1):187–200.

Zhou Y, Hao JK, Glover F (2019) Memetic search for identifying critical nodes in sparse graphs. *IEEE Trans. Cybern.* 49(10):3699–3712.

Zhou Y, Kou Y, Zhou M (2023a) Bilevel memetic search approach to the soft-clustered vehicle routing problem. *Transp. Sci.* 57(3):701–716.

Zhou Y, Kundu T, Qin W, Goh M, Sheu JB (2021b) Vulnerability of the worldwide air transportation network to global catastrophes such as covid-19. *Transp. Res. Part E Logisti. Transp. Rev.* 154:102469.

Zhou Y, Li J, Hao JK, Glover F (2023b) Detecting critical nodes in sparse graphs via "reduce-solve-combine" memetic search. URL http://dx.doi.org/10.1287/ijoc.2022.0130.cd, https://github.com/INFORMSJoC/2022.0130.

Zhou Y, Wang G, Hao JK, Geng N, Jiang Z (2023c) A fast tri-individual memetic search approach for the distance-based critical node problem. *Eur. J. Oper. Res.* 308(2):540–554.

Zhou Y, Wang Z, Jin Y, Fu ZH (2021c) Late acceptance-based heuristic algorithms for identifying critical nodes of weighted graphs. *Knowl. Based Syst.* 211:106562.

Zhou Y, Zhang X, Geng N, Jiang Z, Wang S, Zhou M (2023d) Frequent itemset-driven search for finding minimal node separators and its application to air transportation network analysis. *IEEE Trans. Intell. Transp. Syst.* 1–13, URL http://dx.doi.org/10.1109/TITS.2023.3270334.

Appendix. Online Supplement

A. Detailed Results of IRMS under the Time Limit $\hat{t} = 7200$ Seconds

Table 1 summarizes the detailed results of IRMS under a long time limit $\hat{t} = 3600$ seconds. In Table 1, columns 1-3 describe for each instance its name (Instance), K value, and best known value (BKV) reported in the literature, respectively. Columns 4-6 report the results of IRMS, including the best result (i.e., $\hat{f}$) found during 30 runs, the average result (i.e., $\bar{f}$), and average computation time (i.e., $\bar{t}$) at each run. From it, we observe that IRMS also performs excellent performance. In particular, it finds new upper bounds for nine instances, and matches previous best-known upper bounds on 23 instances.

Table 1 Results of IRMS on Synthetic and Real-world Benchmarks under $\hat{t} = 7200$ Seconds

			IRMS		
Instance	K	BKV	$\hat{f}$	$\bar{f}$	$\bar{t}$
BA500	50	195*	**195**	195.0	0.0
BA1000	75	558*	**558**	558.0	1.5
BA2500	100	3704*	**3704**	3704.0	4.5
BA5000	150	10196*	**10196**	10196.0	18.8
ER235	50	295	**295**	295.0	7.9
ER466	80	1524	**1524**	1524.0	79.2
ER941	140	5012	**5012**	5023.3	1817.8
ER2344	200	902498	918952	941170.2	5673.8
FF250	50	194*	**194**	194.0	0.0
FF500	110	257*	**257**	257.0	0.7
FF1000	150	1260*	**1260**	1260.0	18.5
FF2000	200	4545*	**4545**	4545.0	264.2
WS250	70	3083	**3083**	3132.7	4127.2
WS500	125	2072	**2072**	2078.1	813.2
WS1000	200	109677	137766	144237.3	3149.4
WS1500	265	13098	**13098**	13103.4	2461.6
Bovine	3	268	**268**	268.0	0.0
Circuit	25	2099	**2099**	2099.0	1.3
Ecoli	15	806	**806**	806.0	0.0
USAir97	33	4336	**4336**	4596.0	1452.8
HumanDi	52	1115	**1115**	1115.0	1.5
TreniR	26	918	**918**	918.0	1.3
EU_fli	119	348268	**348268**	348268.0	749.4
openfli	186	26783	26875	28363.8	4400.1
yeast1	202	1412	**1412**	1412.0	34.0
H1000	100	306349	**306349**	308345.0	3561.5
H2000	200	1242739	**1236887**⋆	1250761.8	5352.7
H3000a	300	2840690	**2799868**⋆	2840491.0	6055.2
H3000b	300	2837584	**2794262**⋆	2821455.9	6184.0
H3000c	300	2835369	**2783248**⋆	2825113.4	6682.2
H3000d	300	2828492	**2802615**⋆	2838392.3	5612.8
H3000e	300	2843000	**2798688**⋆	2841275.9	5836.7
H4000	400	5038611	**4977344**⋆	5074711.6	5894.7
H5000	500	7964765	**7956481**⋆	8058861.6	6062.2
powergr	494	15862	15863	15870.8	5764.8
Oclinks	190	611253	614467	614467.0	1494.1
faceboo	404	420334	705403	723233.5	5915.9
grqc	524	13591	**13590**⋆	13598.0	5945.4
hepth	988	106276	110352	114505.1	6367.1
hepph	1201	6155877	9309386	9543464.4	5009.6
astroph	1877	53963375	56849750	57688316.6	5149.1
condmat	2313	2298596	9359852	10415923.6	4155.8

Notes. "$*$" presents the optimal solution, and "$\star$" indicates the improved best upper bounds.

B. Detailed Results of CEMCNP under the Time Limit $\hat{t} = 3600$ Seconds

Since the source code of CEMCNP is not available to us, we have re-implemented it according its pseudo code. Detailed comparison between our implemented CEMCNP algorithm and reported CEMCNP algorithm are summarized in Table 2. From it, we observe that our implemented CEMCNP algorithm performs slightly worse performance than the reported results on most of instances. However, some reported results are obtained under a longer computation time than the time limit. For example, at least 4000 seconds are required to achieve the results of H3000e, hepth, hepph and condmat. It is worthy noting that for some instances (e.g, grqc, hepth and condmat), our implemented CEMCNP algorithm achieves better performance than the original version.

Table 2 Detailed Results of CEMCNP on Synthetic and Real-world Benchmarks under $\hat{t} = 3600$ Seconds

			Reported Results		Implemented Results		
Instance	K	BKV	$\hat{f}$	$\bar{t}$	$\hat{f}$	$\overline{f}$	$\bar{t}$
BA500	50	195	**195**	0.0	**195**	195.0	0.0
BA1000	75	558	**558**	0.2	**558**	558.0	54.0
BA2500	100	3704	**3704**	0.2	**3704**	3704.0	347.3
BA5000	150	10196	**10196**	1.8	**10196**	10196.0	9.6
ER235	50	295	**295**	19.8	297	302.8	0.0
ER466	80	1524	**1524**	98.4	1569	1630.6	1.7
ER941	140	5012	**5012**	694.6	5363	5635.3	2.7
ER2344	200	902498	**912346**	3069.0	1012527	1060618.6	847.9
FF250	50	194	**194**	0.3	**194**	194.0	178.2
FF500	110	257	**257**	0.2	**257**	258.6	0.3
FF1000	150	1260	**1260**	120.6	**1260**	1260.0	1080.9
FF2000	200	4545	**4545**	486.4	4546	4552.5	1458.1
WS250	70	3083	**3083**	623.1	4203°	5447.9	0.5
WS500	125	2072	**2072**	105.8	2085	2193.0	153.7
WS1000	200	109677	**109935**	1256.2	154899	169877.2	19.3
WS1500	265	13098	**13098**	658.2	13664	27810.6	108.0
Bovine	3	268	**268**	0.0	**268**	268.0	0.0
Circuit	25	2099	**2099**	0.1	2101	2188.7	0.1
Ecoli	15	806	**806**	0.0	**806**	808.8	0.1
USAir97	33	4336	**4336**	856.4	**4336**	5149.5	0.1
HumanDi	52	1115	**1115**	0.5	**1115**	1115.0	319.9
TreniR	26	918	**918**	0.2	**918**	918.0	10.9
EU_fli	119	348268	**348325**	431.6	350762	357502.7	89.5
openfli	186	26783	**26796**	3165.8	29481	31377.7	86.4
yeast1	202	1412	**1412**	9.8	**1412**	1414.3	1085.8
H1000	100	306349	**307113**	1068.4	330493	337217.0	211.4
H2000	200	1242739	**1245637**	1527.2	1324988	1344914.2	126.4
H3000a	300	2840690	**2842695**	1204.3	2962661	3042695.4	165.2
H3000b	300	2837584	**2840867**	2040.0	2968601	3035272.7	438.0
H3000c	300	2835369	**2831643**	2159.3	2956916	3008793.3	903.6
H3000d	300	2828492	**2830284**	3895.2°	2967747	3030236.3	51.6
H3000e	300	2843000	**2846536**	4035.6°	3012046	3043889.5	25.9
H4000	400	5038611	**5096437**	1563.7	5261850	5383301.3	347.5
H5000	500	7964765	**8007638**	2024.1	8206499	8348589.5	696.5
powergr	494	15862	**15906**	856.7	15965	16084.9	3393.5
Oclinks	190	611253	**615467**	267.8	622237	626701.5	836.1
faceboo	404	420334	**589763**	3526.9	794938	857245.1	685.4
grqc	524	13591	13743	1025.0	**13666**	13701.7	3391.1
hepth	988	106276	115309	4521.4°	**109504**	111495.6	1602.3
hepph	1201	6155877	**7556094**	4025.1°	8464199	10530358.9	3513.9
astroph	1877	53963375	**57895042**	3105.2	59476077	60449655.3	3512.9
condmat	2313	2298596	7658643	4203.5°	**3756761**	4632078.6	3555.9

Note. "○" indicates a longer computation time than 3600 seconds.

C. TarjanInComponent Procedure

Algorithm 1 realizes the TarjanInComponent procedure used in the articulation point impact strategy. It starts the search from a root node, and recursively builds a depth first search (DFS) tree. During DFS phase, all articulation points are identified. Once all nodes are visited, the evaluation of all nodes is finished.

Algorithm 1 Pseudo Code of TarjanInComponent Procedure

Input: A large connected component $\mathcal{C}$, root node x, time stamp $Count$, γ and η

$dfn[x] \leftarrow ++Count$
$low[x] \leftarrow dfn[x]$
for each neighbour w of x in $\mathcal{C}$ **do**
  if w has not been visited **then**
    TarjanInComponent$(\mathcal{C}, w, Count, \gamma, \eta)$
    $low[x] \leftarrow \min\{low[x], low[w]\}$
    // w is not the parent of v
    if $dfn[x] < dfn[w]$ **then**
      $\gamma[x] += \gamma[w]$
    end if
    if $dfn[x] < low[w]$ **then**
      The number of x's subtrees increases one
      if x is not the root node **then**
        x is marked as an articulation point
        $\eta[x] += \gamma[w]$
        $\psi[x] += \frac{\gamma[w](\gamma[w]-1)}{2}$
      else
        if x is the root $\wedge$ x has more than one subtree **then**
          x is marked as an articulation point
        end if
      end if
    end if
  else
    $low[x] \leftarrow \min\{low[x], dfn[w]\}$
  end if
end for