Exact and heuristic solution approaches for the Generalized Independent Set Problem

Mingming Zheng [a], Jin-Kao Hao [b], Qinghua Wu [a,*]

[a] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China, email: qinghuawu1005@gmail.com*

[b] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France, email: jin-kao.hao@univ-angers.fr*

Accepted to Computers and Operations Research, Jan. 2024.

Abstract

The generalized independent set problem (GIS) is a generalization of the classical maximum independent set problem and has various practical applications, such as forest harvesting and image/video processing. In this work, we present highly effective exact and heuristic algorithms for the GIS. In the proposed exact algorithm, a new upper bound on the maximum net benefit of an independent set in a subgraph is derived using a Lagrangian relaxation of a linear integer programming formulation of the GIS problem. This bound is then employed in a combinatorial branch-and-bound (B&B) algorithm. To solve larger instances, we propose an adaptive local search procedure which jointly considers several neighborhoods and selects a neighborhood to explore in an adaptive manner at each iteration. Our proposed exact and heuristic algorithms are evaluated on a set of 216 GIS benchmark instances and compared with several state-of-the-art algorithms. Computational results indicate that our proposed algorithm competes favorably with the best existing approaches for the GIS. In particular, the exact algorithm is able to attain all known optimal solutions and to solve 26 more instances to optimality for the first time.

Keywords: Heuristic; branch and bound; adaptive local search; the generalized independent set problem.

* Corresponding author.

1 Introduction

Given an undirected graph $G = (V, E, E')$ with two disjoint sets of edges E and E' (i.e., $E \cap E' = \emptyset$), each vertex $v \in V$ is associated with a positive revenue w_v, and each removable edge $u, v \in E'$ is associated with a positive cost c_{uv}. The objective of the generalized independent set (GIS) problem is to find an independent set $I \subseteq V$ such that no two vertices in I are connected by an edge in E, while maximizing the net benefit of I, which is defined as the difference between the revenues of the vertices in I and the costs of the removable edges with both endpoints in I. An example of the GIS is given in Fig. 1, where the dashed edges indicate the removable edges E' with their costs, and the set of red vertices (with the revenues shown next to the vertices) represents a candidate solution of the instance. The net benefit of the candidate solution is computed as $w_b + w_c + w_e - c_{ce} = 3 + 2 + 6 - 2 = 9$. In Section 3.2, a mathematical model of the GIS derived from Colombi et al. (2017) is presented.

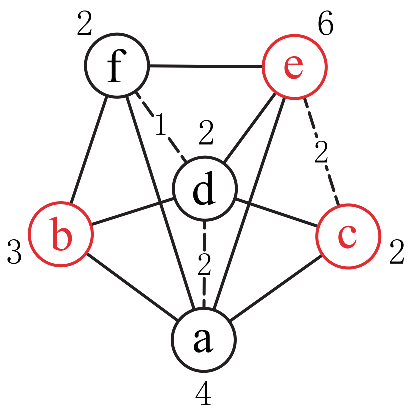


Fig. 1. An illustration of the generalized independent set problem

When E' is an empty set and each vertex has a unit weight, the GIS problem degenerates to the NP-hard maximum independent set (MIS) problem which involves finding an independent set of maximum cardinality. Therefore, the GIS problem is at least as difficult as the MIS problem. The GIS is equivalent to the NP-hard maximum weight independent set problem (Wu et al. 2012) when E' is an empty set. Moreover, the GIS is closely related to several other combinatorial optimization problems, such as the maximum edge weighted clique problem (Pullan 2008), the minimum weighted vertex cover (Singh & Gupta 2006), and the knapsack problem with conflicts (Coniglio et al. 2021). In addition to its theoretical significance, the GIS is a useful model for many applications. A typical application concerns the problem of forest management and harvesting (Hochbaum & Pathria 1997), where a forest is partitioned into a number of cells and one needs to determine which cells to harvest. More specifically, harvesting a cell can result in a revenue brought by the timber harvested in that cell, while harvesting two adjacent cells can incur a penalty due to the consideration of wildlife habitat protection. Furthermore, two adjacent cells with a combined area exceeding a stipulated threshold are considered incompatible. The objective of the problem is decide which cells to harvest so as to maximize the net profits. The problem can be formulated by constructing a GIS instance where a vertex corresponds to a cell, a permanent (non-removable) edge corresponds to a pair of incompatible cells, and a removable edge associated with a penalty is created for two compatible adjacent cells. Other applications of GIS can be found in facility location (Hochbaum 2004), cartographic label placement (Mauri et al. 2010), and handling geographic uncertainty in spatial information (Wei & Murray 2012).

While numerous methods are available for the classic MIS problem and its equivalent maximum clique problem (see review of Wu & Hao (2015)), only a few exact and heuristic approaches have been proposed in the literature for the GIS as reviewed in Section 2. Given the wide application and NP-hard nature of the problem, powerful exact and heuristic algorithms are needed to increase our capability of solving this challenging problem. The contributions of this work can be summarized as follows.

- We develop an effective exact algorithm based on the branch-and-bound (B&B) framework, which has several novelties. First, to get a tight upper bound on the maximum net benefit of an independent set in the subgraph, we employ a Lagrangian relaxation method inspired by the basic idea of Hosseinian et al. (2020), which explores a linear integer programming formulation of the problem. Second, we devise an adaptive local search to generate a tight initial lower bound on the optimal objective value of this problem, which helps the exact algorithm to prune more effectively during its search. Third, to efficiently prune the search tree, we employ an effective branching rule by presenting the vertices to the algorithm in a descending revenue order. By incorporating the Lagrangian relaxation upper bound, the tight initial lower bound and the descending revenue order branching rule into the B&B framework, we obtain an effective exact algorithm for the GIS.
- To obtain high-quality approximate solutions for large instances, we devise an adaptive local search procedure. Our proposed local search is characterized by its neighborhood exploring strategy, which jointly explores several neighborhoods induced by different types of moves and adaptively selects the most promising neighborhood capable of generating high quality solutions.
- To verify the effectiveness of our proposed exact and heuristic algorithms, we compare them with the currently best-performing algorithms by carrying out experiments on a set of 216 well-known GIS benchmark instances. Computational results exhibit that our algorithms compete well with the best existing exact and heuristic algorithms. In particular, our exact

algorithm is able to attain all known optimal solutions reported in the literature, and solve 26 more instances to optimality for the first time. Besides, we generate a new set of 47 larger and more challenging instances to further validate the performance of our proposed heuristic approach.

The rest of the paper is organized as follows. Section 2 presents a literature review on exact and heuristic solution approaches for the GIS problem. Section 3 presents in detail the main components of the proposed branch-and-bound algorithm for the GIS problem, whereas Section 4 provides the main components of the proposed heuristic approach. Computational results and comparisons with the currently best-performing algorithms are given in Section 5. Section 6 investigates some key ingredients of the proposed algorithms, followed by concluding remarks directions in Section 7.

2 Literature review

Due to the relevance of the GIS, several attempts have been devoted to solving this problem. Table 1 summarizes the exact and heuristic solution approaches for the GIS discussed in this section.

To find optimal solutions for the GIS, several exact solution approaches have been proposed in literature. For instance, Hochbaum & Pathria (1997) developed an integer programming formulation for the GIS in the context of solving forest harvesting optimization problems arising in forest management. For special cases of graphs such as bipartite graphs, they also proposed polynomial-time algorithms to find optimal solutions to the GIS. Colombi et al. (2017) provided the first polyhedral analysis of the problem and studied several classes of valid inequalities. By introducing some of the derived valid inequalities into a 0-1 linear programming formulation, they developed a branch-and-cut algorithm to exactly solve the GIS. Along with the proposed

Table 1

Representative exact and heuristic algorithms for the GIS

Literature	Framework
Exact algorithms	
Hochbaum & Pathria (1997)	Integer programming formulation
Colombi et al. (2017)	Branch and cut
Hosseinian & Butenko (2019)	Branch and bound
Heuristic approaches	
Kochenberger et al. (2007)	Tabu search
Colombi et al. (2017)	Tabu search
Nogueira et al. (2021)	Local search

exact algorithm, they also proposed a large set of 216 randomly generated instances by extending the classic DIMACS instances, which is widely adopted nowadays to test different GIS approaches in the literature. Their proposed branch-and-cut algorithm solved 94 out of the 216 instances to optimality. Hosseinian & Butenko (2019) proposed an exact algorithm for the GIS by taking advantage of a quadratic formulation of the GIS. An upper bound was obtained by solving a quadratic relaxation of this formulation. By incorporating the upper bound into the branch-and-bound (B&B) framework, an exact method denoted by CB&B for the GIS was developed. The algorithm was tested on the same set of 216 instances proposed by Colombi et al. (2017), and was able to solve 118 out of the 216 instances to optimality.

Though exact algorithms are valuable for finding the optimal solutions, their computation time becomes prohibitive when the size of the instance is large. For large sized instances, heuristic approaches are indispensable alternatives for obtaining high-quality near-optimal solutions. Several effective heuristic approaches have been proposed in the literature for the GIS. Kochenberger et al. (2007) developed an unconstrained binary quadratic programming formulation for the problem, which has the advantage of requiring only variables associated with the vertices (no variables associated with the removable edges), leading thus to a nonlinear formulation typically much smaller than its linear counterpart in terms of the number of variables. Based on the nonlinear model, they also developed a tabu search approach and shown its effectiveness by computational experiments. Colombi et al. (2017) developed linear programming (LP) based heuristics for the GIS by introducing some of their derived valid inequalities into a 0-1 linear programming formulation. In addition, inspired by the probabilistic GRASP-tabu search algorithm proposed by Wang et al. (2013), they also proposed a meta-heuristic method exploiting the binary quadratic programming formulation of the problem proposed by Kochenberger et al. (2007). Recently, Nogueira et al. (2021) proposed a highly effective variable neighborhood descent based iterated local search heuristic (ILS-VND) to solve the GIS by extending their previous iterated local search (ILS) heuristic originally proposed for the maximum weight independent set (MWIS) problem. The ILS-VND heuristic jointly relies on two new neighborhoods, which are explored using a variable neighborhood descent procedure. They reported experiments on the set of 216 benchmark instances proposed by Colombi et al. (2017), indicating that the ILS-VND algorithm is very competitive with the best existing heuristics for the problem.

Our review indicates that the CB&B algorithm proposed by Hosseinian & Butenko (2019) and the ILS-VND algorithm proposed by Nogueira et al. (2021) showed an overall best performance when exactly and approximately solving the GIS. Thus, these two algorithms are used as the reference approaches for our comparative study.

3 The branch-and-bound algorithm for GIS

3.1 The basic procedure

Our exact algorithm for the GIS (called LA-B&B) relies on the standard B&B framework, which proves to be one of the most successful paradigms for devising exact algorithms for the independent set problem, its equivalent maximum clique problem, and its weighted case (i.e., the maximum weight independent set problem). The success of a B&B method for these problems mainly depends on the refined techniques used to derive lower and upper bounds on the size (or weight) of the independent set (clique), and the proper pruning strategies.

LA-B&B implicitly enumerates the independent sets in the graph by excluding from consideration through a pruning strategy any independent set which can never lead to the maximum net benefit. Basically, the enumeration of independent sets in our B&B method relies on two global vertex sets: the current independent set I to be expended (also called solution) and a set P of candidate vertices to expand the incumbent independent set. By reference to I, set P, also called candidate list, is a subset of $V \setminus I$ such that any vertex $v \in P$ can be inserted to I to reach a larger independent set $I = I \cup \{v\}$. To maintain the feasibility of I, each vertex of P cannot be connected to any vertex in I by any permanent edge in E. This property constitutes a key foundation to our B&B method.

The general scheme of our LA-B&B algorithm is summarized in Algorithm 1. At the beginning of the search, we use an adaptive local search procedure to find a feasible solution I_{max} for the GIS (see Section 4), whose net benefit W_{max} serves as an initial lower bound for the problem. Then our B&B starts with an empty independent set $I = \emptyset$ and a candidate list $P = V$ (see lines 2 and 4 in Alg. 1), and continues by examining all independent sets in V until an independent set with the maximum net benefit is found. To achieve this, the algorithm calls the function $MaxGIS$ in a recursive manner, and at each recursion of $MaxGIS$, a vertex $v \in P$ is selected to expand the independent set I. On backtracking, v is removed from P and I, and a new vertex in P is selected to append to I by calling again $MaxGIS$ (see lines 24-26 in Alg. 1).

Without considering the pruning strategy (line 18 in Alg. 1), the algorithm will traverse each independent set in the graph in such a way that it first finds the independent set I_1 with the maximum net benefit containing the first vertex appearing in P. Then it finds the independent set I_2 with the maximum net benefit in $G \setminus \{v_1\}$ that contains the second vertex appearing in P and so on. During the search process, each time an independent set I

Algorithm 1 The branch-and-bound algorithm for GISP

Require: A graph $G = (V, E, E')$
Ensure: The maximum generalized independent set I_{max} and its weight W_{max}

function $Main$
    $I_{max} \leftarrow \phi$
    $W \leftarrow 0$
    $P \leftarrow V$ /*P is the candidate set containing the vertices that can be added to I*/
    $I_{max}, W_{max} \leftarrow ALS(G)$ /* Get an initial lower bound by ALS */
    $VertexSort(P)$ /* Sort vertices by revenue in descending order */
    $I_{max}, W_{max} \leftarrow MaxGIS(P)$
    return I_{max}, W_{max}
end function

function $MaxGIS(P)$
    if $P = \emptyset$ and $W > W_{max}$ **then**
        $I_{max} \leftarrow I$
        $W_{max} \leftarrow W$
    end if
    while $P \neq \emptyset$ **do**
        Compute the upper bound $UB(G_P)$ for G_P (Section 3.2)
        if $W + UB(G_P) > W_{max}$
            Select the first vertex v in P /* branching rule (Section 3.3) */
            $I \leftarrow I \cup \{v\}$
            $W \leftarrow W + w_v - \sum_{u \in I, \{u,v\} \in E'} c_{uv}$
            $P' \leftarrow P \setminus N(v)$ /* $N(v)$ is the subset of vertices in P that are connected to v by a permanent edge in E*/
            $MaxGISP(P')$/* go to the next level of recursion */
            $I \leftarrow I \setminus \{v\}$ /* end of branching step for vertex v */
            $W \leftarrow W - w_v + \sum_{u \in I, \{u,v\} \in E'} c_{uv}$
            $P \leftarrow P \setminus \{v\}$ /* continue to try the next vertex in P */
        end if
    end while
    return I_{max}, W_{max}
end function

with its net benefit W larger than that of I_{max} is found, I_{max} and W_{max} are updated by I and W, which can serve as a tight lower bound employed in the pruning strategy to better prune some branches of the search tree during the subsequent search. The pruning strategy operates as follows: given the current independent set I and its corresponding candidate list P, an upper bound on the maximum net benefit of the independent set in G_P is calculated using the Lagrangian relaxation method (see Section 3.2), where G_P is the subgraph induced by P. If this upper bound is not larger than the current lower bound W_{max} found so far, I cannot lead to an independent set with a net benefit larger than W_{max}, and as a result, the associated node in the search tree can be safely pruned by excluding from further consideration of the corresponding subgraph. Otherwise, the node in the search tree rooted at I and P needs to be further explored. In this case, a branching rule is applied to determine the next vertex v to be selected from P to append to the current independent I (line 19 in Alg. 1). In our B&B algorithm, we employ a simple branching rule where the vertices in P are sorted in the decreasing order with respect to their revenue and the vertices in P are selected in that order (Section 3.3). After each branching step, we update P by removing from P all vertices connected to v by a permanent edge in E in order to ensure the required property of P.

3.2 A Lagrangian relaxation upper-bounding method

Given an undirected graph $G = (V, E, E')$ with E and E' respectively denoting the two disjoint sets of permanent (non-removable) edges and removable edges. For the convenience of presenting the Lagrangian relaxation upper-bounding method, we introduce a removable edge with a cost $c_{ij} = 0$ for each pair of vertices $i, j \in V$ which are neither connected by a permanent edge nor by a removable edge (i.e., $\{i, j\} \notin E \cup E'$). The optimal solution of the given graph remains unchanged despite the introduction of these removable edges with a cost of 0. Then the problem can be formulated as the following integer programming model (Colombi et al. 2017):

$$W^* = \max \sum_{i \in V} w_i x_i - \sum_{\{i,j\} \notin E} c_{ij} y_{ij} \tag{1}$$

$$\text{s.t. } x_i + x_j \le 1 \qquad \{i, j\} \in E \tag{2}$$

$$x_i + x_j - y_{ij} \le 1 \qquad \{i, j\} \notin E \tag{3}$$

$$x_i \in \{0, 1\} \qquad i \in V \tag{4}$$

$$y_{ij} \in \{0, 1\} \qquad \{i, j\} \notin E. \tag{5}$$

In the above formulation, each vertex $i \in V$ is associated with a binary variable x_i indicating whether vertex i is selected to be a member of the independent set, and each removable edge $\{i, j\} \in E'$ (equivalent to $\{i, j\} \notin E$) is associated with a binary variable y_{ij} indicating whether a removable edge $\{i, j\} \in E'$ is in the independent set. The objective (1) is to maximize the net benefit, defined by the difference between the sum of the revenues of the selected vertices and the costs of those removable edges with both endpoints in the independent set. Constraints (2) ensure that two vertices connected by a permanent edge cannot appear in the independent set together. Constraints (3) ensure that a removable edge $\{i, j\} \in E'$ is selected if both of the vertices corresponding to its endpoints are included in the independent set.

Inspired by the work of Hosseinian et al. (2020), we can obtain the following Lagrangian relaxation of the integer programming formulation defined by

Equations (1)-(3), with the exception of the integrality of the variables,

$$\begin{aligned} LR(\lambda^1, \lambda^2) = \max_{x,y} & \sum_{i \in V} w_i x_i - \sum_{\{i,j\} \notin E} c_{ij} y_{ij} + \sum_{\{i,j\} \in E} \lambda^1_{ij}(1 - x_i - x_j) \\ & + \sum_{\{i,j\} \notin E} \lambda^2_{ij}(1 - x_i - x_j + y_{ij}) \qquad (6) \\ \text{s.t. } & x_i \in \{0,1\} \qquad i \in V \qquad (7) \\ & y_{ij} \in \{0,1\} \qquad \{i,j\} \notin E. \qquad (8) \end{aligned}$$

where $\lambda^1, \lambda^2 \geqslant 0$ denote the Lagrange multiplier vectors corresponding to constraints (2) and (3). Let the Lagrange multipliers in Equation (6) take the same values, we obtain a simpler problem as follows,

$$\begin{aligned} LR(\lambda) = \max_{x,y} & \sum_{i \in V} w_i x_i - \sum_{\{i,j\} \notin E} c_{ij} y_{ij} + \sum_{\{i,j\} \in E} \lambda(1 - x_i - x_j) \\ & + \sum_{\{i,j\} \notin E} \lambda(1 - x_i - x_j + y_{ij}) \\ \text{s.t. } & x_i \in \{0,1\} \qquad i \in V \\ & y_{ij} \in \{0,1\} \qquad \{i,j\} \notin E. \end{aligned} \qquad (9)$$

Let n be the number of vertices in the considered graph, $|E|$ be the number of permanent edges, and $|E'|$ be the number of removable edges. Then we write $LR(\lambda)$ as follows:

$$\begin{aligned} LR(\lambda) &= \max_{x,y} \sum_{i \in V} w_i x_i - \sum_{\{i,j\} \notin E} c_{ij} y_{ij} + \sum_{\{i,j\} \in E} \lambda(1 - x_i - x_j) \\ &\quad + \sum_{\{i,j\} \notin E} \lambda(1 - x_i - x_j + y_{ij}) \\ &= \max_{x,y} \sum_{i \in V} w_i x_i + \sum_{\{i,j\} \notin E} (\lambda - c_{ij}) y_{ij} + \lambda[\sum_{\{i,j\} \in E} (1 - x_i - x_j) \\ &\quad + \sum_{\{i,j\} \notin E} (1 - x_i - x_j)] \\ &= \max_{x,y} \sum_{i \in V} w_i x_i + \sum_{\{i,j\} \notin E} (\lambda - c_{ij}) y_{ij} + \lambda[|E| + |E'| \\ &\quad - \sum_{\{i,j\} \in E} (x_i + x_j) - \sum_{\{i,j\} \notin E} (x_i + x_j)] \\ &= \max_{x,y} \sum_{i \in V} w_i x_i + \sum_{\{i,j\} \notin E} (\lambda - c_{ij}) y_{ij} + \lambda[|E| + |E'| \\ &\quad - \sum_{i \in V} d_i x_i - \sum_{i \in V} (n - 1 - d_i) x_i] \\ &= \max_{x,y} \sum_{i \in V} [w_i - \lambda(n-1)] x_i + \sum_{\{i,j\} \notin E} (\lambda - c_{ij}) y_{ij} + \lambda\tbinom{n}{2} \qquad (10) \end{aligned}$$

where d_i denotes the degree of the vertex $i \in V$ in the graph considering only permanent edges, and $\binom{n}{2}$ equals the number of edges in the complete graph with n vertices, i.e., $\binom{n}{2} = \frac{n(n-1)}{2}$. With Equation (10), the restricted Lagrangian relaxation problem of Equation (9) can be written as,

$$\begin{aligned} LR(\lambda) = \max_{x,y} \sum_{i \in V} [w_i - \lambda(n-1)]x_i + \sum_{\{i,j\} \notin E} (\lambda - c_{ij}) y_{ij} + \lambda \tbinom{n}{2} \\ \text{s.t. } x_i \in \{0,1\} \qquad i \in V \\ y_{ij} \in \{0,1\} \qquad \{i,j\} \notin E. \end{aligned} \tag{11}$$

Obviously, the optimal solution to Equation (11) only depends on the signs of the coefficients of the binary decision variables x_i and y_{ij}. To achieve the optimal solution, we only need to set the binary decision variables x_i and y_{ij} with non-negative coefficient to 1 while setting those with negative coefficient to 0. Therefore, for each $\lambda > 0$, $LR(\lambda)$ can be computed according to Equation (12),

$$LR(\lambda) = \sum_{i \in V^+} [w_i - \lambda(n-1)] + \sum_{\{i,j\} \in E^+} (\lambda - c_{ij}) + \lambda \tbinom{n}{2} \tag{12}$$

where $V^+ \subseteq V$ and $E^+ \subseteq E$ respectively denote the set of vertices with non-negative coefficient and the set of removable edges with non-negative coefficient,

$$V^+ = \{v \in V | w_i \geqslant \lambda(n-1)\} \tag{13}$$
$$E^+ = \{(i,j) \notin E | c_{ij} \leqslant \lambda\} \tag{14}$$

Both V^+ and E^+ depend on their revenues (costs) and the Lagrangian multiplier λ. Then for a given value $\lambda > 0$, $LR(\lambda)$ in Equation (12) provides an upper bound to the optimal value of GIS. To obtain an upper bound as tight as possible, we need to identify a value for λ that minimizes $LR(\lambda)$ as much as possible.

In Equation (12), its left part $\sum_{i \in V^+} [w_i - \lambda(n-1)]$ decreases with λ whereas its right part $\sum_{\{i,j\} \in E^+} (\lambda - c_{ij}) + \lambda \binom{n}{2}$ increases with λ, and it is difficult to determine the best value for λ that minimizes Equation (12). Then we try to identify an appropriate value for λ which could lead to as smaller value for $LR(\lambda)$ as possible. Let w_{max} be the maximum revenue of the vertices in the considered graph, i.e., $w_{max} = max\{w_v : v \in V\}$. For $\lambda \geqslant \frac{w_{max}}{n-1}$, V^+ becomes empty and $\sum_{i \in V^+} [w_i - \lambda(n-1)] = 0$, then $LR(\lambda)$ in Equation (12) increases for $\lambda \in (\frac{w_{max}}{n-1}, +\infty)$. As a result, we can restrict the optimization interval for λ to $[0, \frac{w_{max}}{n-1}]$. When $0 \leqslant \lambda \leqslant \frac{w_{max}}{n-1}$, the monotonicity of $LR(\lambda)$ is not clear as it depends on both the revenues of the vertices and the cost of the edges in the considered graph. Then we test different values for $\lambda \in [0, \frac{w_{max}}{n-1}]$, and experimental results indicate that $\lambda = \frac{w_{max}}{n-1}$ is a good choice which can lead to a smaller value for $LR(\lambda)$. With the analysis above, we can obtain an upper bound for the GIS defined by $LR(\lambda)$ with $\lambda = \frac{w_{max}}{n-1}$, that is,

$$W^* \leq LR(\frac{w_{max}}{n-1}) = \sum_{\{i,j\} \in E^+} (\frac{w_{max}}{n-1} - c_{ij}) + \frac{n}{2} w_{max} \tag{15}$$

where $E^+ = \{(i,j) \notin E | c_{ij} \leq \frac{w_{max}}{n-1}\}$.

Then given the current independent set I and its corresponding expending candidate list P, the following pruning strategy is employed in our B&B algorithm. Precisely, let G_P be the subgraph induced by P, and $UB(G_P)$ be the Lagrangian relaxation upper bound calculated by Equation (15) on G_P. In the case that $UB(G_P) \geq \sum_{i \in P} w_i$, $UB(G_P)$ is simply set to be $\sum_{i \in P} w_i$. For any independent set I^0 in G, let $W(I^0) = \sum_{i \in I^0} w_i - \sum_{i,j \in I^0, \{i,j\} \in E'} c_{ij}$ be the net benefit of I^0. Then given an arbitrary independent set I' in G_P, we have $W(I \cup I') = W(I) + W(I') - \sum_{i \in I, j \in I', \{i,j\} \in E'} c_{ij} \leq W(I) + W(I') \leq W(I) + UB(G_P)$. Therefore, the search subtree rooted at I and P can be safely pruned if the following pruning condition is satisfied,

$$W(I) + UB(G_P) \leq W_{max} \tag{16}$$

where W_{max} is the maximum net benefit of the independent set found so far.

3.3 The branching strategy

At each branching step, a branching rule is applied to determine the next vertex v to be selected from P to append to the current independent I. A simple branching strategy is employed in our B&B algorithm. Initially, vertices in the original graph G are sorted in a descending order with respect to their revenues, and then copied back to P according to that sorting order. Then at each level of recursion, the vertices in the candidate set P are always kept in the same order as they initially appear in P, and the first vertex in P is always selected with priority to be added to the current independent set I.

4 An adaptive local search for GIS

To solve larger instances for GIS, we propose an adaptive local search for GIS, which is also used in our proposed branch and bound algorithm to produce a tight initial lower bound. Our proposed adaptive local search (ALS) is based on the tabu search framework which has been successfully applied in a wide range of combinatorial optimization problems (Glover & Laguna 1998). As shown in Algorithm 2, ALS starts with an initial solution S (line 1 in Alg.

2) generated by the randomized procedure presented in Section 4.1. Then it repeats the main 'while' loop after initializing the best solution S_{best} found so far with S (line 2 in Alg. 2). For each iteration of the main loop, ALS jointly explores four neighborhoods and selects a best admissible neighboring solution from the neighborhood that is chosen in an adaptive manner (lines 5–6 in Alg. 2). Whenever an improved solution S is found during the search, S_{best} is updated with S (lines 8-10 in Alg. 2). The ALS algorithm continues this process until the given stopping condition (typically a cutoff time limit) is verified.

Algorithm 2 The adaptive local search for GIS

Require: An initial solution $S_{initial}$, time limit t_{max}
Ensure: The best solution found S_{best}
1: $S \leftarrow S_{initial}$ /* Apply a randomized procedure to generate an initial solution */
2: $S_{best} \leftarrow S_{initial}$
3: $Iter \leftarrow 0$
4: **while** $Time() < t_{max}$ **do**
5: Construct neighborhoods N_1, N_2, N_3 and $N_1 \cup N_2 \cup N_3$ from S
6: Choose a candidate solution S' according to the neighborhood exploration rule as described in Section 4.2
7: $S \leftarrow S'$
8: **if** $f(S) > f(S_{best})$ **then**
9: $S_{best} \leftarrow S$
10: **end if**
11: $Iter \leftarrow Iter + 1$
12: **end while**
13: **return** S_{best}

4.1 Randomized procedure for initial solutions

Our ALS procedure begins with an initial solution I and then improves I by maximizing its net benefit $W(I)$. The initial solution I is constructed using the following randomized procedure. A seeding vertex i is first randomly selected from V and the current independent set I is set to be composed of only this single vertex. Then at each step, among the candidate set of vertices $P = \{u : u \in V \backslash I, \{u, i\} \notin E, \forall i \in I\}$, (i.e., a vertex $u \in P$ is never connected to any vertex in I by a permanent edge), a vertex v is selected randomly and put in I. The above process is repeated until the candidate set P becomes empty.

4.2 Neighborhood structures

The ALS procedure jointly explores three neighborhoods induced by three basic move operators. The definition of these three move operators are based on two vertex subsets: PA and OM relative to the current independent set I.

PA consists of all vertices which are excluded from I and connected to none of the vertices in I by a permanent edge, i.e., $PA = \{v : v \in V \setminus I, \{v, i\} \notin E, \forall i \in I\}$. We can append any vertex $v \in PA$ to I such that the resulting solution is still a feasible independent set.

OM is composed of all vertices which are excluded from I and connected to only one vertex in I by a permanent edge, i.e., $OM = \{v : v \in V \setminus I, | N(v) \cap I |= 1\}$ where $N(v) = \{u : \{v, u\} \in E\}$ denotes the set of vertices connected to v by permanent edges. We can swap a vertex $u \in OM$ with the only vertex $v \in I$ connected to u such that the resulting solution remains to be a feasible independent set.

Based on the subsets *PA* and *OM*, the three move operators employed in our ALS procedure are defined as follows.

-$ADD(i)$: This move operator is applied only when PA is not empty and consists in appending a vertex $i \in PA$ to the current solution I. The neighborhood defined by the ADD move operator is denoted by N_1. One key concept related to a move is the move gain, which measures how much the net benefit of the current solution I is changed when a move is applied to I. For a fast calculation of the move gain, a n-dimensional ($n = |V|$) vector B **then** is used where $B_i = w_i - \sum_{\{u,i\} \in E', u \in I} c_{ui}$ denotes the potential contribution of a vertex i to the net benefit of the current independent set. With the vector B, the move gain of appending a vertex $u \in PA$ can be fast computed by the following expression:

$$\Delta_i = B_i \tag{17}$$

Obviously, the calculation of the move gain value for an add move can be achieved with a complexity of $O(1)$. After an $ADD(i)$ move is performed, the vector B can be fast updated by the following expression:

$$B_u = B_u - c_{ui}, \forall \{u, i\} \in E' \tag{18}$$

Therefore, for each performed add move, the vector B is updated in $O(n)$.

-$DROP(i)$: This move operator consists in dropping a vertex $i \in I$ from the current solution I. The neighborhood defined by the $DROP$ move operator is denoted by N_2. With the vector B, the move gain value of dropping a vertex $i \in I$ (i.e., denoted by Δ_i) can be quickly calculated using the following equation:

$$\Delta_i = -B_i \tag{19}$$

After performing a drop move denoted by $DROP(i)$, the vector B can be quickly updated in the following manner:

$$B_u = B_u + c_{ui}, \forall \{u, i\} \in E' \tag{20}$$

Therefore, the vector B after a drop move is also updated in $O(n)$.

-$SWAP(i, j)$: This move operator is applied only when the subset OM is not empty and consists in exchanging a vertex $i \in OM$ with the only vertex $j \in I$ which is connected to i in I. The neighborhood defined by the SWAP move operator is denoted by N_3. With the vector B, the move gain of a swap move, denoted by Δ_{ij}, can be fast calculated as follows.

$$\Delta_{ij} = B_i - B_j \tag{21}$$

It is noted that a swap move $SWAP(i, j)$ can be decomposed into a drop move $DROP(j)$ followed by an add move $ADD(i)$. Thus in order to update the vector B after a swap move $SWAP(i, j)$, we could first apply Equation (20) to update the change induced by $DROP(j)$, then use Equation (18) to update the change induced by $ADD(i)$. It is clear that the update of vector B after a swap move can also be implemented in $O(n)$.

When several neighborhoods are available, the method to combine these neighborhoods so as to enhance the search ability of the algorithm becomes essential. There are several methods to effectively explore the neighborhoods in the literature, such as neighborhood union, probabilistic neighborhood union, and token-ring search (Hao 2012). For instance in Wu et al. (2012), the union of the basic neighborhoods induced by the ADD, $SWAP$, and $DROP$ moves is explored for solving the maximum weight clique problem. The motivation for combining multiple neighborhoods is to allow the algorithm to examine candidate solutions with different structures and characteristics, increasing its chance to discover high-quality optima. After testing different methods for combining the basic neighborhoods induced by the ADD, $SWAP$, and $DROP$ moves, the following adaptive rule is adopted in our ALS procedure.

The ALS algorithm jointly considers four neighborhoods denoted by N_1, N_2, N_3 and $N_1 \cup N_2 \cup N_3$ (i.e., the union of N_1, N_2 and N_3), and selects one of these four neighborhoods to explore in a probabilistic way at each iteration. Specifically, we employ four counters σ_1, σ_2, σ_3, and σ_4 to respectively record the number of times each neighborhood improves the recorded best solution. At the start of the search, we set the probability of choosing the four neighborhoods to be $\frac{\sigma_i}{\sum_{j=1}^{4} \sigma_j}$ with $\sigma_i = 1$ $(i = 1, 2, 3, 4)$. At each iteration of the local search, one of the four neighborhoods is selected according to the given probability, and the best admissible solution is selected from this chosen neighborhood to replace the current solution. Ties are broken randomly when multiple moves have the same gain. During the search process, each time the selected neighborhood produces an updated best solution, its probability is updated by increasing the corresponding σ value by 1. In such a manner, the chance to apply the neighborhood generating high-quality solutions is increased. We mention that compared with the neighborhood union strategy proposed by Wu et al. (2012), which only explores the union of the three neighborhoods, and ensures an intensified and aggressive examination of the search space, our adaptive selection strategy offers more search diversification, and favors a better search balance between intensification and diversification.

Finally, a simple tabu mechanism is adopted to prevent the search from short-term cycles, which forbids a vertex moved by the ADD, $SWAP$, and $DROP$ operators to be moved again for the next tl iterations, where tl is a parameter called tabu tenure (Glover & Laguna 1998).

5 Computational experiments

This section is dedicated to an extensive evaluation of the proposed exact and heuristic approaches. For this purpose, we present experimental results achieved by our exact and heuristic approaches on a large set of benchmark instances and compare them with other state-of-the-art exact and heuristic methods for the GIS proposed in the literature.

5.1 Test instances and parameter settings

The proposed LA-B&B and ALS approaches are both tested on a set of 216 benchmark instances which were first introduced by Colombi et al. (2017). The generation of these instances is based on 12 DIMACS graphs (Johnson & Trick 1996) with 125 to 400 vertices and 6963 to 71820 edges. The DIMACS graph set includes randomly generated graphs, graphs where the optimal solution has been hidden by incorporating low-degree vertices, as well as graphs constructed from various applications, such as coding theory, fault diagnosis problems, Keller's conjecture on tilings using hypercubes, and the Steiner triple problem. Each of these 12 graphs is associated with three different sets of removable edges and six different values for the revenue and cost, leading thus to a total of 216 instances [1]. In these benchmark instances, each edge of the graph is

[1] The benchmark instances are available at https://or-dii.unibs.it/index.php?page=gisp.

randomly marked as a removable edge with a probability p_r, such that three classes of instances were produced by considering $p_r = 0.25, 0.50$ and 0.75. For each instance class generated with a fixed p_r value, the following two sets of instances were generated by imposing different values for the revenue and cost.

- SET1: for each vertex i, its revenue value is set to an integer randomly taken in $\{1, ..., 100\}$, while for each removable edge $\{i, j\} \in E'$, its associated cost is set to three different values $c_{ij} = [\frac{1}{100}(w_i + w_j)]$, $[\frac{1}{50}(w_i + w_j)]$, and $[\frac{1}{25}(w_i + w_j)]$, where [.] denotes the closest rounded integer.
- SET2: for each vertex i, its revenue value is set to three different values $w_i =$ 100, 50, and 25, while for each removable edge $\{i, j\} \in E'$, its associated cost is set to $c_{ij} = 1$.

The instances generated with the three different p_r values in both sets are marked with A, B, and C in their names, respectively.

In addition to these 216 standard instances, and to further evaluate the performance of our ALS approach on larger instances, we randomly selected 15 graphs from the OpenStreetMap [2] files of North America and 15 graphs from the PACE 2019 Challenge (Dzulfikar et al. 2019). The OpenStreetMap graphs are vertex-weighted, generated by associating map labels with vertices and assigning each vertex a weight based on the importance of the corresponding map label. The graphs from the PACE 2019 Challenge are from various domains and have been gathered from eight different sources. [3] These 30 selected graphs have a larger number of vertices, ranging from 786 to 17,903, and a greater number of edges, ranging from 1,948 to 604,867. These instances have been previously used in the literature to test the minimum vertex cover problem and the maximum weight independent set problem (Cai et al. 2018; Lamm et al. 2019). Since the density of the graphs from the PACE 2019 Challenge and OpenStreetMap is very low (generally below 0.1), we additionally include all the 17 graphs with at least 700 (up to 4000) vertices and 6480 to 3997732 edges from the DIMACS benchmark, and use the inverted versions of these original graphs by transforming them into their complementary graphs. In total, we obtain 47 diverse large GIS instances. In these 47 large graphs, each edge of the graph is randomly marked as a removable edge with a random probability $p_r = 0.25, 0.50$ and 0.75, and the cost of each removable edge is set in the same way as for the SET1-C instances. The revenue value of each vertex in the graphs from the DIMACS benchmark and PACE 2019 Challenge is set in the same way as for the SET1-C instances, while for the OpenStreetMap graphs, we retain the original vertex weights as the vertex revenues, given that these graphs are already vertex-weighted. As

[2] https://www.openstreetmap.org.

[3] https://github.com/daajoe/pace2019_vc_instances.

a result, these graphs produce a total of 47 instances for the GIS problem [4].

We set the parameter required by our ALS algorithm as follows: the tabu tenure $tl = 7$. The setting of this parameter is tuned using the general IRACE automatic parameter configuration tool (López-Ibáñez et al. 2016). For this parameter tuning task, we used a different set of 20 instances with 100 to 400 vertices [5], which are randomly generated as follows. Each pair of vertices is randomly marked as a permanent edge with a random probability p_1 from the range $[0.1, 0.9]$. For each pair of vertices not connected by a permanent edge, it is randomly marked as a removable edge with a random probability p_2 from the range $[0.3, 0.7]$. The revenue value of each vertex is set to a random integer value from $[10, 100]$, while for each removable edge, its cost is set to a random integer value from $[1, 5]$.

5.2 Experimental protocol and reference algorithms

To show the effectiveness of our proposed LA-B&B and ALS algorithms, we compare them respectively with the currently best-performing exact and heuristic approaches for the GIS. As shown in the most recent study (Hosseinian & Butenko 2019; Nogueira et al. 2021), the following two reference algorithms are among the best exact and heuristic methods for the GIS, and thus constitute the reference approaches to evaluate the performance of our LA-B&B and ALS algorithms.

- CB&B: A combinatorial B&B (CB&B) method (Hosseinian & Butenko 2019), which also relies on the branch and bound framework similar to our LA-B&B algorithm, but uses different bounding and branching strategies. Specifically, CB&B takes advantage of a nonlinear formulation of the GIS problem and employs a spherical relaxation of a quadratic function over a hypersphere in its bounding subroutine. Additionally, CB&B employs a branching strategy where the vertices in the graph $G = (V, E)$ (E denotes the set of permanent edges) are sorted in a descending order of their degrees, and the vertices are selected in that order at each branching step.
- ILS-VND: A variable neighborhood descent based iterated local search heuristic approach (Nogueira et al. 2021), which relies on two neighborhoods, one involving the addition of a single vertex to the solution, and the other involving the addition of two vertices. These two neighborhoods are explored using a variable neighborhood descent procedure. ILS-VND differs

[4] These benchmark instances are available at https://github.com/m2-Zheng/GISP/tree/main/large instance.

[5] The instances for parameter setting are available at https://github.com/m2-Zheng/GISP/tree/main/instance for parameter setting.

from our ALS algorithm mainly in its neighborhoods and the manner in which it explores the neighborhoods.

Moreover, the above reference approaches are tested very recently by Nogueira et al. (2021) under the same platform (an Intel i7 processor with 3.6 GHz and 16 GB of memory). The source codes of CB&B and ILS-VND were made available by the authors. To make the comparison as fair as possible, we run the source codes of these reference algorithms on our computing platform under the same time limit as adopted by Hosseinian & Butenko (2019) and Nogueira et al. (2021), which is set to be 3 hours for CB&B and LA-B&B, and 30 seconds for ILS-VND and ALS on the SET1 and SET2 instances. In addition, for the 47 larger instances generated in this work, we set the time limit to be 5 minutes.

Our LA-B&B and ALS approaches [6] are programmed in Java and compiled on an Intel i5 processor with 2.8 GHz CPU and 16G RAM. Our platform requires respectively 0.31, 1.93 and 7.35 CPU seconds for the graphs r300.5, r400.5 and r500.5 when running the DIMACS MC Machine Benchmark program (available at http://archive.dimacs.rutgers.edu/pub/dsj/clique/).

5.3 Computational results of LA-B&B

We assess the performance of the LA-B&B algorithm by comparing it with the reference algorithm CB&B introduced in Section 5.2. In this comparison, the reference CB&B algorithm did not include a procedure to generate a tight initial lower bound for its pruning subroutine. Therefore, our LA-B&B algorithm excludes the ALS procedure from its branch and bound framework to ensure a fair comparison with the reference method. Table 2 summarizes the comparative results between LA-B&B and CB&B on the whole set of the 216 instances while Tables 5-7 in the Appendix present the detailed results for each instance. Columns 1-2 in Table 2 respectively give the name of each instance set and the number of instances in each set. Columns 3-8 summarize the results obtained by the two compared algorithms on each instance set, including the number of instances solved to optimality, the required computation time averaged on the instances solved to optimality, and the number of better results in terms of lower bounds on instances where both algorithms fail to obtain the optimal solution. Finally, the summarized results for each column are presented in the last row of the table.

When comparing CB&B with our LA-B&B algorithm, one observes that LA-B&B is able to solve 143 instances (66%) to optimality within the given

[6] The source code of the algorithms are available at https://github.com/m2-Zheng/GISP.

Table 2
Summary of comparative results on the 216 standard GIS benchmark instances between LA-B&B and CB&B

Set	Total	LA-B&B			CB&B		
		#Optimality	Avg. time(s)	#Better lower bound	#Optimality	Avg. time(s)	#Better lower bound
$p_r = 0.25$							
SET1	36	33	54.68	3	33	586.50	0
SET2	36	33	843.52	3	33	979.79	0
$p_r = 0.5$							
SET1	36	30	64.60	6	26	4092.37	0
SET2	36	30	207.29	3	20	5314.38	3
$p_r = 0.75$							
SET1	36	10	2508.35	26	3	7698.81	0
SET2	36	7	3503.61	28	3	6515.30	0
Summary	216	143	1197.01	69	118	4197.86	3

time limit of 3 hours, while CB&B can only solve 118 instances (54%) to optimality. For the 73 instances where both exact algorithms fail to reach the optimal solutions within the given time limit, our LA-B&B algorithm is able to achieve better lower bounds on 69 instances but worse lower bounds on 3 instances. In terms of the computational efficiency, LA-B&B requires a significantly shorter average time on the instances solved to optimality. Especially, as shown by the detailed results in Tables 5-7, our LA-B&B algorithm is 10 times faster than the CB&B method to solve 112 out of the 118 instances solved by both algorithms. The advantage of our LA-B&B algorithm in terms of the computational efficiency becomes even more evident when the removable-edge density of the graphs decreases. For 96 out of the 112 instances with the removable-edge density $\rho_2 \leq 0.5$, our LA-B&B algorithm is 100 times faster than the CB&B method to reach the optimal solution. For 12 out of the 67 instances with the removable-edge density $\rho_2 \leq 0.25$, our LA-B&B algorithm is even 1000 times faster than the CB&B method. These observations demonstrate that our proposed LA-B&B algorithm is highly efficient for the GIS compared to the currently best-performing exact approach.

By incorporating the tight initial lower bound produced by the adaptive local search procedure ALS, the performance of the LA-B&B algorithm can be further improved. Columns 9-10 in Tables 5-7 provide the detailed results of LA-B&B with the initial lower bound (denoted by LA-B&B+ALS) on the 216 instances. As shown by columns 9-10 in Tables 5-7, LA-B&B+ALS is able to solve one more instance to optimality within the given time limit. In terms of the average computation time for the instances solved to optimality by both algorithms, LA-B&B+ALS requires less time than LA-B&B. Finally, for these instances where both algorithms fail to attain the optimal solution, LA-B&B+ALS is able to produce better lower bounds on much more instances than LA-B&B. These comparative results clearly demonstrate the improved performance of LA-B&B when incorporating the tight initial lower bound produced by ALS.

Table 3
Summary of comparative results on the 216 standard GIS benchmark instances between ALS and ILS-VND

Set	Total	Indicator: f_{best}				Indicator: f_{avg}				Indicator:time(s)		
		#Wins	#Ties	#Losses	Avg. Imp	#Wins	#Ties	#Losses	Avg. Imp	#Wins	#Ties	#Losses
$p_r = 0.25$												
SET1	36	0	36	0	0	0	36	0	0	35	0	1
SET2	36	0	36	0	0	0	36	0	0	36	0	0
$p_r = 0.5$												
SET1	36	0	36	0	0	0	36	0	0	36	0	0
SET2	36	0	36	0	0	6	30	0	0.10	36	0	0
$p_r = 0.75$												
SET1	36	0	36	0	0	0	36	0	0	36	0	0
SET2	36	1	35	0	<0.01	10	26	0	0.44	36	0	0
Summary	216	1	215	0		16	200	0		215	0	1

5.4 Computational results of ALS

To demonstrate the effectiveness of our heuristic approach ALS in terms of producing highly competitive results (lower bound), we compare it with the currently best-performing heuristic approach ILS-VND. Given the stochastic nature of the two compared algorithms, each instance is solved 10 times independently with different random seeds by each algorithm under the same experimental protocol as described in Section 5.2. Table 3 summarizes the comparative results between ALS and ILS-VND on the whole set of the 216 instances while Tables 8-10 in the Appendix present the detailed results per instance. Columns 1-2 in Table 3 respectively show the name of each instance set and the number of instances in each set. The remaining columns give the number of instances where our ALS algorithm achieves better (#Wins), equal (#Ties) or worse (#Losses) results compared to ILS-VNS in terms of the best objective value (f_{best}), the average objective value (f_{avg}), and the average running time in seconds ($time(s)$) to reach the best result across the 10 independent runs. We also present the average percentage improvement (Avg. Imp) of ALS over ILS-VND in terms of the best or average objective value across the 36 instances in each set. For each instance set, the average percentage improvement of the best and the average objective value are computed as $\frac{\sum_{i=1}^{36}(f^i_{ALS}-f^i_{ILS-VND})/f^i_{ALS}}{36} \times 100$, where f^i_{ALS} ($f^i_{ILS-VND}$) is the best or the average objective value obtained by the given algorithm on the i-th instance in the set. The summarized results for each column are presented in the last row of Table 3.

From Table 3, one can observe in terms of the best objective value f_{best}, our ALS is able to obtain respectively 1 better, 215 equal, and 0 worse results compared to ILS-VND, while in terms of the average objective value f_{avg}, our ALS is capable of achieving respectively 16 better, 200 equal, and 0 worse results. In terms of the computation time, our ALS algorithm requires a shorter time than ILS-VND to find equal or better solutions for 215 out of the 216 instances. Furthermore, as shown by the detailed results in Tables 8-10 in Appendix, our ALS approach is able to improve the previous best known result (new lower bound) for one instance (gen400_p0.9_75_B_75), and it reaches its best solutions with a success ratio of 100% for all the 216 tested instances, while this is done by ILS-VND only for 200 cases within the given time limit, further confirming the robustness of the proposed ALS. We can conclude that ALS is highly efficient for the GIS compared to the currently best-performing heuristic approach proposed in the literature.

5.5 *Computational results of ALS on large instances*

To further evaluate the performance of the proposed ALS algorithm, this section experimentally compares ALS with the reference algorithm ILS-VND on the 47 large instances. The comparison is carried out under the same experimental protocol described in Section 5.2. Table 4 summarizes the comparative results. The first six columns in Table 4 respectively indicate the instance name (Instance), the number of vertices ($|V|$), the number of permanent edges ($|E|$), the number of removable edges ($|E'|$), the permanent-edge density ρ_1 (computed as $\frac{2|E|}{|V|(|V|-1)}$), and the removable-edge density ρ_2 (computed as $\frac{2|E'|}{|V|(|V|-1)}$). Columns 7-14 report the results obtained by the two compared heuristic approaches, including the best solution (f_{best}), the average result (f_{avg}), the success rate ($success$) to achieve the best result, and the average running time ($time(s)$) in seconds needed to reach the best result. For each instance where ALS and ILS-VND reach different best or average values, we also indicate in parentheses the percentage improvement of ALS over ILS-VND, which is calculated as $(f_{ALS} - f_{ILS-VND})/f_{ALS} \times 100$, where f_{ALS} and $f_{ILS-VND}$ respectively represent the best or average result obtained by the two compared methods. Additionally, in the last four rows, we present the summarized results between the two compared algorithms, including the number of instances where each algorithm performed better in terms of the best and average results, the average running time to reach the best result, and the *p-value* from the Wilcoxon signed-rank test.

Table 4 shows that our ALS algorithm achieves highly competitive results compared to ILS-VND. In terms of the best objective value, ALS is able to find a better, equal and worse result on 22, 21, 4 instances respectively compared to ILS-VND, while in terms of the average objective value, ALS is able to find a better, equal and worse result on 30, 11, 6 instances respectively. Regarding the computation time required by both algorithms to reach their best objective value, ALS is faster than ILS-VND on 16 out of the 21 instances where both algorithms reach the same best result. Finally, the p-values (< 0.05) of the Wilcoxon signed-rank test indicates a significant difference between ALS and ILS-VND in terms of the best result, average result and computation time.

Table 4: Summary of comparative results between ALS and ILS-VND on the 47 large PACE, OpenStreetMap and DIMACS instances

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	ILS-VND				ALS			
						f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
brock800_2_C_75	800	54669	56765	0.17	0.18	**2125**	2125	10	7.86	**2125**	2125	10	0.08
brock800_4_C_75	800	26831	85126	0.08	0.27	**2501**	2499.7	9	17.18	**2501**	2501(5.2E-04)	10	1.06
C1000.9_C_75	1000	36630	12791	0.07	0.03	**5284**	5280.6	8	67.97	**5284**	5284(0.06)	10	0.14
C2000.5_C_50	2000	239602	759562	0.12	0.38	**2069**	2054.5	3	109.17	**2069**	2069(0.70)	10	39.28
C2000.9_C_25	2000	147548	51920	0.07	0.03	**6396**	6275.5	2	88.22	**6396**	6396(1.88)	10	5.27
C4000.5_C_50	4000	1957884	2039848	0.24	0.26	1962	1902.9	1	153.87	**1965(0.15)**	1956.5(2.74)	1	71.18
DSJC1000_5_C_75	1000	59694	189980	0.12	0.38	**1861**	1861	10	53.05	**1861**	1861	10	0.36
hamming10-4_C_25	1024	66369	23231	0.13	0.04	**3660**	3610	6	44.61	**3660**	3660(1.37)	10	0.75
keller5_C_75	776	36867	37843	0.12	0.13	**3133**	3133	10	8.29	**3133**	3133	10	0.06
keller6_C_25	3361	759816	266766	0.13	0.05	**5236**	5189.4	4	217.51	**5236**	5236(0.89)	10	37.36
MANN_a81_C_50	3321	1520	4960	2.76E-04	9.00E-04	**131298**	131294	1	245.84	131251(-0.04)	131242.5(-0.04)	1	299.36
p_hat1500-1_C_50	1500	620920	218407	0.55	0.19	**1020**	1020	10	8.16	**1020**	1020	10	0.33
p_hat1500-2_C_25	1500	133431	421859	0.12	0.38	**6035**	6035	10	17.51	**6035**	6035	10	24.16
p_hat1500-3_C_75	1500	136008	140998	0.12	0.13	**6693**	6693	10	64.42	**6693**	6693	10	4.42
p_hat700-1_C_25	700	44183	139468	0.18	0.57	**1289**	1289	10	10.23	**1289**	1289	10	0.07
p_hat700-2_C_25	700	60206	62716	0.25	0.26	**2956**	2956	10	0.82	**2956**	2956	10	0.07
p_hat700-3_C_50	700	45706	15934	0.19	0.07	**4038**	4038	10	0.08	**4038**	4038	10	0.26
bio-dmela_C_25	7393	18897	6672	6.92E-04	2.44E-04	301705	301009	1	299.92	**302692(0.33)**	302560.7(0.51)	1	229.42
bio-yeast_C_50	1458	980	968	9.23E-04	9.11E-04	**68574**	68571.3	9	36.74	**68574**	68573(2.48E-03)	5	178.63
ca-AstroPh_C_75	17903	47135	149837	2.94E-04	9.35E-04	600744	600301	1	299.91	**603111(0.39)**	602835.4(0.42)	1	113.23
ca-GrQc_C_75	4158	3179	10243	3.68E-04	1.19E-03	**174782**	174768	1	202.22	174756(-0.01)	174709.7(-0.03)	1	93.50
ca-HepPh_C_50	11204	57721	59898	9.20E-04	9.54E-04	368966	368104	1	299.87	**371351(0.64)**	371104.7(0.81)	1	134.49
soc-wiki-Vote_C_75	889	727	2187	1.84E-03	5.54E-03	150129	149575	1	299.93	**151207(0.71)**	151120.4(1.02)	1	266.45
socfb-Duke14_C_25	9885	374731	131706	7.67E-03	2.70E-03	129751	129591	1	299.20	**129872(0.09)**	129798.4(0.16)	1	45.02
socfb-MIT_C_50	6402	123064	128166	6.01E-03	6.26E-03	250191	249466	1	423.48	**251613(0.57)**	251448.3(0.79)	1	49.71

Continued on next page

Continued

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	ILS-VND				ALS			
						f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
socfb-Stanford3_C_75	11586	136823	431486	2.04E-03	6.43E-03	267175	265962	1	299.98	**275010(2.85)**	274851.8(3.23)	1	91.10
socfb-UConn_C_25	17206	448038	156829	3.03E-03	1.06E-03	248580	247864	1	299.96	**255848(2.84)**	255447.9(2.97)	1	219.10
socfb-UCSB37_C_25	14917	356918	125297	3.21E-03	1.13E-03	**37358**	37358	10	1.37	**37358**	37358	10	4.08
tech-routers-rf_C_75	2113	1539	5093	6.90E-04	2.28E-03	**97799**	97799	10	24.71	**97799**	97796.7(-2.35E-03)	5	134.02
tech-WHOIS_C_25	7476	42181	14762	1.51E-03	5.28E-04	324746	324398	1	298.04	**325467(0.22)**	325405.3(0.31)	1	175.86
web-edu_C_25	3031	4813	1661	1.05E-03	3.62E-04	**114140**	114125	2	95.84	113929(-0.19)	113847.6(-0.24)	1	137.92
web-spam_C_75	4767	8989	28386	7.91E-04	2.50E-03	**190503**	190446	1	283.77	190418(-0.04)	190372.7(-0.04)	1	118.55
vc-exact_001_C_50	6160	19655	20552	1.04E-03	1.08E-03	215669	215577	1	280.81	**215711(0.02)**	215662.8(0.04)	1	214.58
vc-exact_008_C_50	7537	35678	37155	1.26E-03	1.31E-03	246445	246254	1	293.42	**246770(0.13)**	246703.6(0.18)	1	276.36
vc-exact_011_C_25	9877	19102	6871	3.92E-04	1.41E-04	301218	300909	1	296.52	**301710(0.16)**	301616.6(0.23)	1	248.51
vc-exact_024_C_75	7620	11367	35926	3.92E-04	1.24E-03	235446	235217	1	291.62	**235581(0.06)**	235413.6(0.08)	1	237.99
vc-exact_026_C_25	6140	27159	9608	1.44E-03	5.10E-04	201135	201039	1	288.21	**201316(0.09)**	201278.8(0.12)	1	86.82
vc-exact_038_C_25	786	10321	3703	0.03	0.01	**12015**	12013.8	4	160.96	**12015**	12014.7(0.01)	7	83.56
vc-exact_039_C_75	6795	2574	8046	1.12E-04	3.49E-04	**259455**	259369	1	273.71	258779(-0.26)	258692.4(-0.26)	1	49.51
vc-exact_078_C_75	11349	4289	13450	3.66E-01	1.30E-01	437948	437851	1	294.19	**437982(0.01)**	437850.4(-1.37E-04)	1	254.08
vc-exact_087_C_75	13590	5126	16114	5.55E-05	1.75E-04	**517473**	517208	1	297.25	517392(-0.02)	515537.9(-0.32)	1	108.32
vc-exact_107_C_25	13590	15646	5594	1.69E-04	6.06E-05	403831	403194	1	299.15	**405158(0.33)**	404045.2(0.21)	1	68.81
vc-exact_131_C_75	2980	1191	4169	2.68E-04	9.39E-04	**109630**	109609	1	283.42	109392(-0.22)	109320.5(-0.26)	1	110.93
vc-exact_151_C_50	15783	12144	12519	3.66E-01	3.82E-01	**527400**	526994	1	297.37	527310(-0.02)	523850.6(-0.60)	1	280.26
vc-exact_167_C_25	15783	18235	6428	3.66E-01	1.08E-03	460107	459516	1	299.14	**461409(0.28)**	457487.7(-0.44)	1	37.15
vc-exact_194_C_50	1150	39520	41331	0.06	0.06	**5419**	5419	10	61.48	**5419**	5419	10	0.11
vc-exact_196_C_25	1534	93273	32809	0.08	0.03	**5321**	5305.4	6	95.76	**5321**	5321(0.29)	10	0.25
# of best						29				39			
# of best Mean							17				41		
Avg. time(s)									178.57				96.44
p-value						0.01	0.04		1.16E-07				

6 Analysis

In this section, we focus our attention on the analysis of the main components of our proposed LA-B&B algorithm in order to show their important roles to the performance of LA-B&B.

6.1 Impact of the Lagrange multiplier in the upper bound

In Section 3.2, we derived an upper bound on the Lagrange multiplier λ. Our analysis revealed that the optimal value for λ falls within the interval $[0, \alpha]$, where $\alpha = \frac{w_{max}}{n-1}$. To determine a suitable value for λ in the interval $[0, \alpha]$, we evaluated the LA-B&B algorithm using four different values for the Lagrange multiplier. These four values were chosen as $\lambda_i = 0.25 \times i \times \alpha$ ($i = 1, 2, 3, 4$). To conduct this comparison, we executed the LA-B&B algorithm with the four chosen values for λ. We then compared the computational time required by the LA-B&B algorithm with the four chosen λ values to reach the optimal solution. We excluded the initial lower bound and followed the same experimental protocol described in Section 5.2. The evaluation was performed on 36 instances which were randomly selected from the 216 instances.

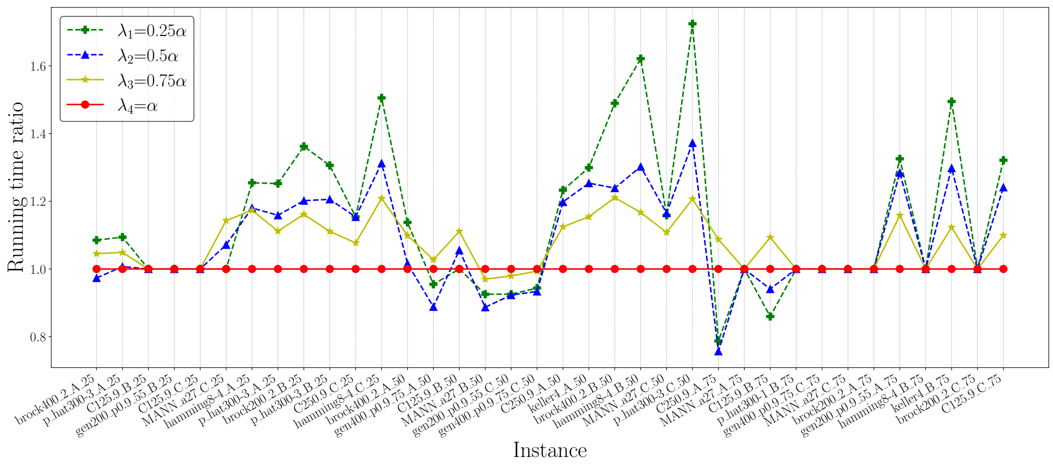
Fig. 2(a) summarizes the normalized running time required by the LA-B&B algorithm with the four chosen values for λ. The horizontal axis in Fig. 2(a) indicates the names of the instances, while the vertical axis indicates the normalized running time required by the LA-B&B with the four chosen values for λ, respectively. Due to significant variations in running time across different instances, we normalize the running time required by the LA-B&B with the chosen λ values using the running time required by LA-B&B with λ_4 as a baseline. Specifically, for a given instance, the running time required by LA-B&B with λ_i ($i = 1, 2, 3, 4$) is normalized by $\frac{t_i}{t_4}$, where t_i is the runtime required by LA-B&B with λ_i to reach the optimal solution on that instance. If the instance can not be solved to optimality with the given time of 10800 seconds by LA-B&B with λ_i, t_i is set to be 10800 seconds. As shown by Fig. 2(a), LA-B&B with λ_4 shows an overall best performance among the four compared variants. For the 29 instances where LA-B&B with all λ_i can reach the optimal solution, LA-B&B with λ_4 is able to reach the optimal solution within the shortest running time on 22 cases. Additionally, Fig. 2(a) reveals that the running time required by the four variants typically falls within the range of $[0.8t_4, 1.4t_4]$ across most of the 36 instances, indicating the differences in running time between the four compared variants are relatively small.

To further investigate the influence of Lagrange multipliers on the tightness of the bound, we present in Fig. 2(b) the ratio of the best upper bound achieved by employing the Lagrangian relaxation method with the four selected values for the Lagrange multiplier. Specifically, for a given instance, we calculate the Lagrangian relaxation upper bound $UB(G_P)$ at each bounding step using the four chosen values for the Lagrange multiplier, where G_P represents the subgraph induced by P. We then track the number of times each chosen value for the Lagrange multiplier produces the best Lagrangian relaxation upper bound during the LA-B&B search, denoted as n_1, n_2, n_3, and n_4 respectively. Afterward, we compute the ratio of the best upper bound achieved by each chosen value λ_i as $\frac{n_i}{\sum_{j=1}^{4} n_j} \times 100$. From Fig. 2(b), we can observe that $\lambda_i = \alpha$ shows a slightly overall better performance by achieving the largest ratio on 18 out of the 36 instances. Especially, for several instances where the revenues of the vertices are very evenly distributed, $\lambda_4 = \alpha$ is able to produce the best Lagrangian relaxation upper bound with a ratio of nearly 100%. Intuitively, when the revenues of the vertices are evenly distributed, the revenues of all vertices are close to w_{max}. This results in the first part of Equation 12 (i.e., $\sum\limits_{i \in V^+} [w_i - \lambda(n-1)]$), used for calculating the Lagrangian relaxation upper bound, approaching 0 when λ is set to $\alpha = \frac{w_{max}}{n-1}$. This can potentially lead to a significantly better upper bound. However, on instances where the revenues of the vertices exhibit a wider range of variation, other values for λ enable the LA-B&B algorithm to perform much better compared to $\lambda = \alpha$. We conclude that no single λ dominates the others on all these 36 instances, and the performance of the Lagrangian relaxation method with each Lagrange multiplier varies with the specific instance, making it difficult to determine the best value for λ.

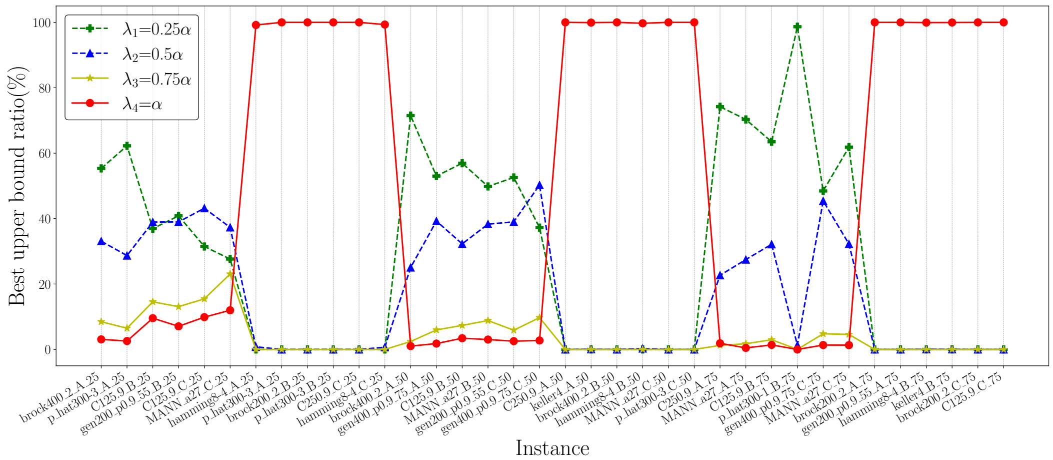
6.2 Effectiveness of the Lagrangian relaxation upper bound

As shown in Section 3.2, the LA-B&B algorithm employs two upper bounding strategies for its pruning subroutine. The first strategy is the Lagrangian relaxation upper bounding strategy (LUB), while the second strategy is a simple upper bounding strategy (SUB) that relies on the revenues of the vertices in P (i.e., $\sum_{i \in P} w_i$) to compute an upper bound on the net benefit of the subgraph G_P. In this section, we analyze the frequency that the two bounding strategies are used for pruning to demonstrate their contributions to the LA-B&B algorithm. The evaluation was performed on the same 36 instances used in Section 6.1.

Fig. 3 summarizes the frequency that the two bounding strategies are used for pruning on the 36 instances. The horizontal axis in Fig. 3 indicates the the names of the instances, while the vertical axis represents the frequency of application for the two strategies, which is defined as $\frac{n_1}{n_1+n_2}$ (or $\frac{n_2}{n_1+n_2}$) where n_1 and n_2 respectively denotes the number of times that LUB and SUB are



(a) The normalized running time required by the LA-B&B algorithm using four different values for the Lagrange multiplier



(b) The ratio of the best upper bound achieved by employing the Lagrangian relaxation method with the four selected values for the Lagrange multiplier

Fig. 2. Comparisons of LA-B&B algorithms with four different values of the Lagrange multiplier.

used for bounding during the search.

From Fig. 3, we can observe that both bounding strategies positively contribute to the performance of the LA-B&B algorithm. However, LUB is generally applied with a higher frequency on many more instances. Especially, for several instances where the revenues of the vertices are very evenly distributed, LUB is applied with a frequency of 100%, indicating that LUB can always produce better upper bound than SUB throughout the search process on these instances. As discussed in Section 6.1, when the revenues of the vertices are evenly distributed, the revenues of all vertices are very close to w_{max}, allowing the LUB produce very tight upper bound on these instance. Further, LUB is more effective than SUB on instances with higher cost of the removable edges, since on these instances, SUB does not take the cost of the removable edges into account. On the other hand, SUB works better than LUB on instances where the revenues of the vertices exhibit a wider range of variation, since on these instances, LUB considers the revenues of all vertices to be w_{max} (the maximum revenue) when bounding, leading to a less tighter bound on the revenues of all vertices. The above observation further confirms that LUB plays an important role to the performance of LA-B&B.

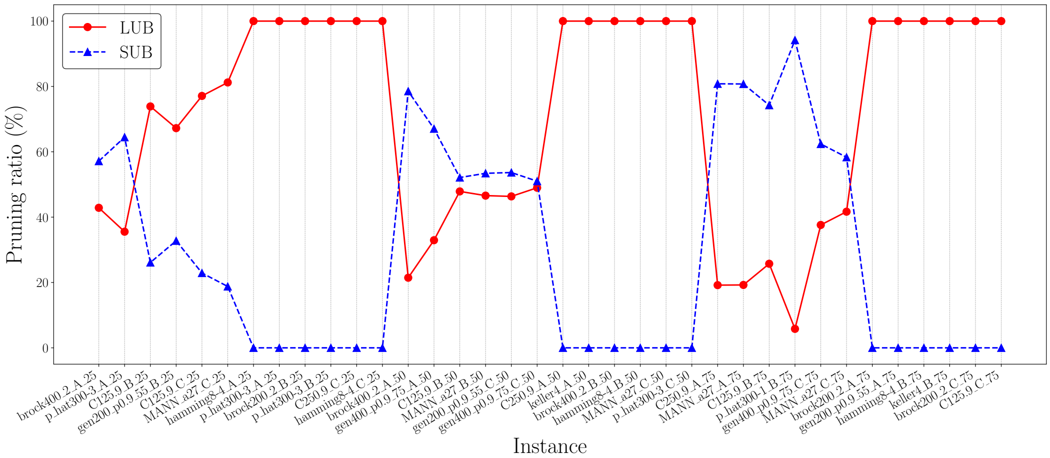
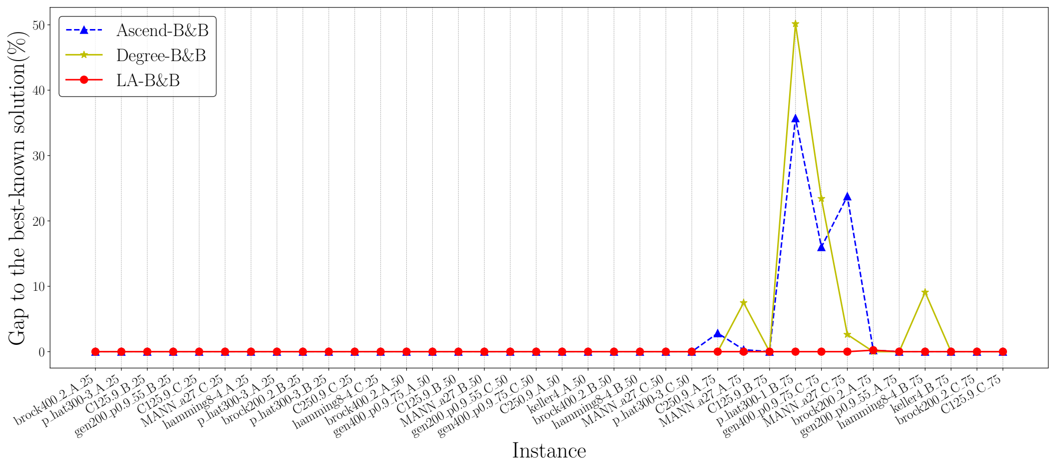


Fig. 3. The frequency of LUB and SUB used for bounding

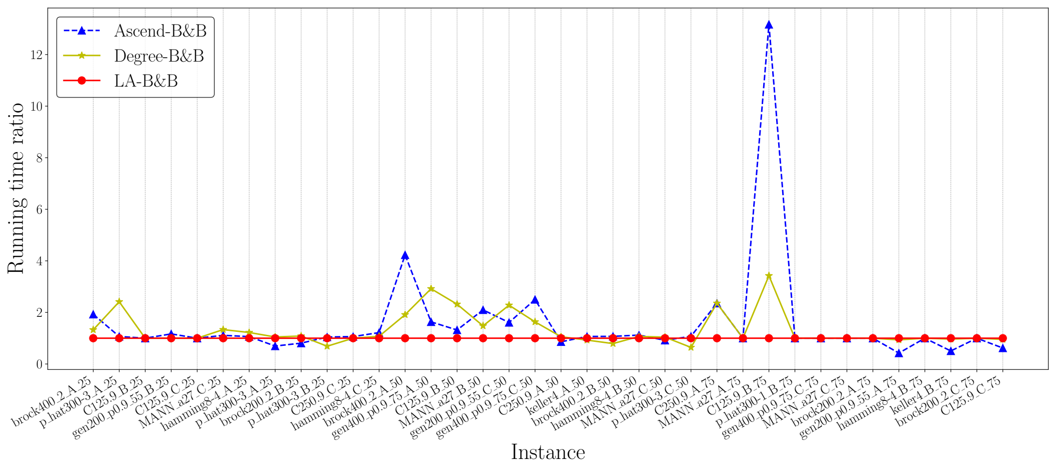
6.3 Impact of the branching strategy

As shown in Section 3.3, the LA-B&B algorithm employs a branching strategy where the vertices in the candidate list P are sorted in decreasing order with respect to their revenue, and the vertices in P are selected in that order. To verify the effectiveness of this branching strategy, we compare it with two other branching strategies widely used in the context of solving the independent set and clique problems. The first branching strategy selects the vertices in P in an ascending order with respect to their revenue, while the second branching strategy selects the vertices in P in an descending order with respective to the degree of the vertices, where the degree of a vertex i in P is defined by the number of permanent edges connected to i in P. By replacing our adopted branching strategy with these two branching strategies, we obtain two LA-B&B variants, which are respectively denoted by Ascend-B&B and Degree-B&B. To make a fair comparison, we run all three approaches without a initial lower bound under the same experimental protocol as described in Section 5.2 on the same 36 instances which are used in Section 6.1.

Fig. 4(a) and 4(b) illustrate the comparative performance of LA-B&B, Ascend-B&B, and Degree-B&B on 36 instances. In Fig. 4(a), the horizontal axis represents the names of the instances, while the vertical axis indicates the percentage gap of the solution obtained by each method compared to the best



(a) Comparative results between LA-B&B and its two variants in terms of best results



(b) Comparative results between LA-B&B and its two variants in terms of normalized running time

Fig. 4. Comparisons of LA-B&B with its two variants.

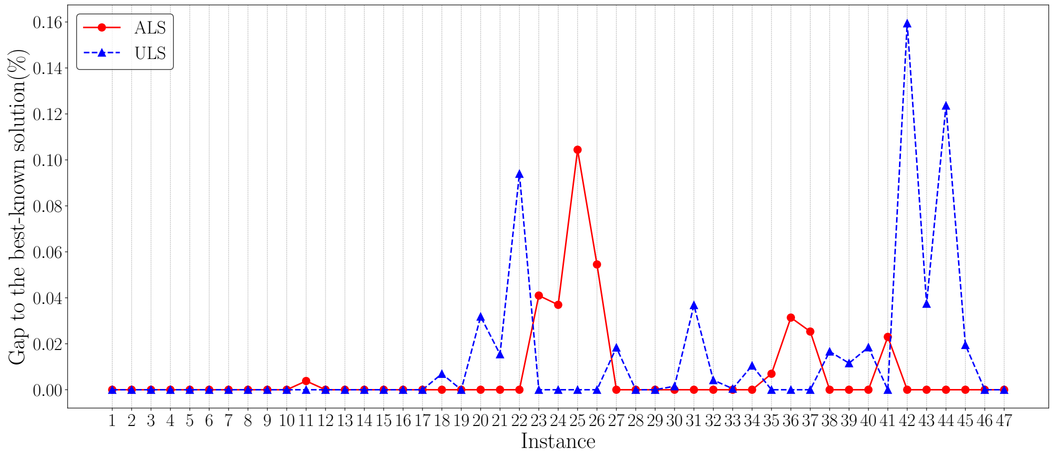
solution achieved with all three compared methods. For each instance, the percentage gaps of the results are calculated as $(f - f_b)/f_b \times 100$, where f represents the result obtained by each respective method, and f_b represents the best solution achieved with all three methods. In Fig. 4(a), we present the normalized running time required by each compared algorithm using the running time required by LA-B&B as a baseline.

From Fig. 4(a), it can be observed that within the given time limit, LA-B&B achieves the best results for 35 instances, compared to 32 and 31 for Ascend-B&B and Degree-B&B, respectively. Moreover, in terms of computational time, Fig. 4(b) discloses that LA-B&B is able to reach the optimal solution in a shorter average computational time compared to Ascend-B&B and Degree-B&B. Overall, the p-values from the Wilcoxon signed-rank test demonstrate statistically significant difference between both LA-B&B and the two compared methods in terms of solution quality (4.31E-02 for LA-B&B vs Ascend-B&B and 4.64E-02 for LA-B&B vs Degree-B&B). In terms of computational time, there is a statistically significant difference between LA-B&B and Ascend-B&B (2.45E-02 for LA-B&B vs Ascend-B&B), while no statistically significant difference is observed between LA-B&B and Degree-B&B (6.13E-02 for LA-B&B vs Degree-B&B).

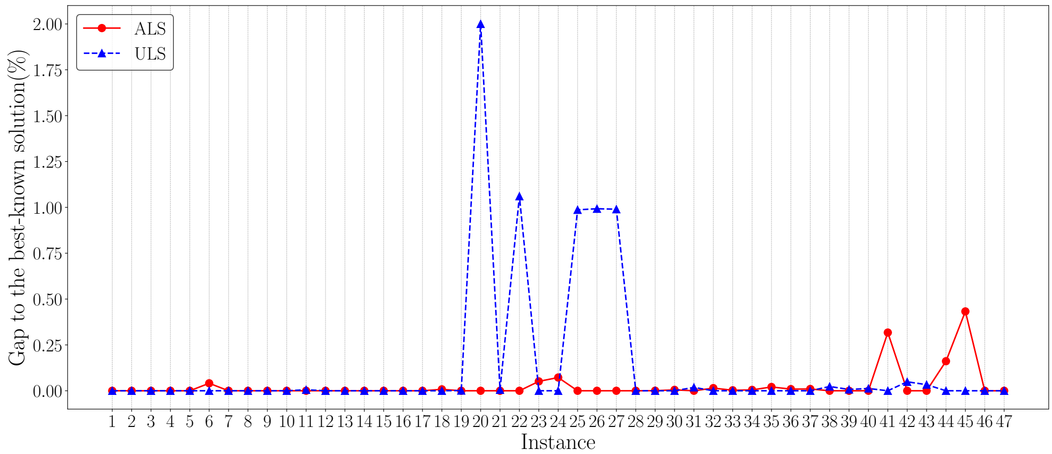
6.4 Effectiveness of the adaptive neighborhood exploration strategy

When multiple neighborhoods are available, a crucial issue arises regarding how to effectively combine these neighborhoods to explore the search space efficiently. In our work, we proposed an adaptive neighborhood exploration strategy that dynamically selects the most promising neighborhood capable of generating high-quality solutions. To demonstrate the effectiveness of our proposed adaptive neighborhood exploration strategy, we compared it with a widely used method from the literature called the neighborhood union method, which jointly considers all neighborhoods and selects the best non-tabu neighboring solution from all considered neighborhoods. The neighborhood union method allows for an aggressive exploration of the search space, and has been proven to be effective in the context of local search for the maximum weight clique problem (Wu et al. 2012).

By keeping other ingredients unchanged, we conducted experiments on both strategies within our local search method under the same experimental protocol described in Section 5.2 on the 47 large instances. We refer to the variant using the neighborhood union strategy as ULS. In Fig. 5, the horizontal axis represents the serial number of the instances, while the vertical axis indicates the percentage gap of the solution obtained by each method compared to the best solution achieved by both methods. For each instance, the percentage gaps of the best and average results are calculated as $|f - f_b|/f_b \times 100$, where f represents the best or average result obtained by the respective method, and f_b represents the best solution achieved by both methods. From Fig. 5, we can observe that in terms of best results, our ALS algorithm achieves better, equal, and worse results respectively for 17, 21 and 9 instances. In terms of average results, ALS achieves better, equal, and worse results for 14, 18, 15 instances. Overall, ALS shows a slightly better stability in achieving high quality solutions compared to ULS and also shows superiority in finding improved solutions, validating the effectiveness of our adaptive neighborhood exploration strategy for exploring the solution space. These findings highlight the potential of ALS in enhancing the performance of local searches.



(a) Best results on large instances



(b) Average results on large instances

Fig. 5. Comparisons of ALS with its variant ULS.

7 Conclusion

In this work, we studied the generalized independent set problem, which is an important generalization of the classical maximum independent set problem with various practical applications. To effectively solve the problem, we proposed highly effective exact and heuristic solution approaches. The exact method derives a new upper bound for the problem using a Lagrangian relaxation method, and a tight lower bound employing our proposed adaptive local search heuristic. By incorporating these lower and upper bound techniques into the general B&B framework, we obtained an effective exact method for this challenging problem.

We assessed the performance of the proposed methods on a set of 216 benchmark instances in the literature and compared our results with those from the currently best-performing exact and heuristic approaches. The comparative studies showed that our algorithms are highly competitive with the best reference algorithms. In particular, our exact algorithm was able to solve 26 more instances to optimality for the first time. We also carried out additional experiments to confirm the effectiveness of the proposed heuristic approach ALS on 47 new generated large instances, which is shown to be competitive with the currently best heuristic approach for generating high-quality lower bounds for our studied problem.

Acknowledgments

We are grateful to the anonymous referees for their valuable suggestions and comments, which helped us to improve the paper significantly. We would like to thank the authors of Hosseinian & Butenko (2019) and Nogueira et al. (2021) for sharing the source codes of their algorithms. This work was partially supported by the National Natural Science Foundation Program of China (Grant No. 72122006).

References

Cai, S., Hou, W., Lin, J., & Li, Y. (2018). Improving local search for minimum weight vertex cover by dynamic strategies. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence* (p. 1412-1418).

Colombi, M., Mansini, R., & Savelsbergh, M. (2017). The generalized independent set problem: Polyhedral analysis and solution approaches. *European Journal of Operational Research*, *260*(1), 41–55.

Coniglio, S., Furini, F., & San Segundo, P. (2021). A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research*, *289*(2), 435–455.

Dzulfikar, M. A., Fichte, J. K., & Hecher, M. (2019). The PACE 2019 Parameterized Algorithms and Computational Experiments Challenge: the Fourth Iteration. In *14th International Symposium on Parameterized and Exact Computation (IPEC 2019).*

Glover, F., & Laguna, M. (1998). *Tabu search*. Springer.

Hao, J.-K. (2012). Memetic algorithms in discrete optimization. In *Handbook of memetic algorithms* (pp. 73–94). Springer.

Hochbaum, D. S. (2004). Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today. *Management Science*, *50*(6), 709–723.

Hochbaum, D. S., & Pathria, A. (1997). Forest harvesting and minimum cuts: a new approach to handling spatial constraints. *Forest Science*, *43*(4), 544–554.

Hosseinian, S., & Butenko, S. (2019). Algorithms for the generalized independent set problem based on a quadratic optimization approach. *Optimization Letters*, *13*(6), 1211–1222.

Hosseinian, S., Fontes, D. B., & Butenko, S. (2020). A Lagrangian bound on the clique number and an exact algorithm for the maximum edge weight clique problem. *INFORMS Journal on Computing*, *32*(3), 747–762.

Johnson, D. S., & Trick, M. A. (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge* (Vol. 26). American Mathematical Soc.

Kochenberger, G., Alidaee, B., Glover, F., & Wang, H. (2007). An effective modeling and solution approach for the generalized independent set problem. *Optimization Letters*, *1*, 111–117.

Lamm, S., Schulz, C., Strash, D., Williger, R., & Zhang, H. (2019). Exactly solving the maximum weight independent set problem on large real-world graphs. In *2019 proceedings of the twenty-first workshop on algorithm engineering and experiments (alenex)* (pp. 144–158).

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

Mauri, G. R., Ribeiro, G. M., & Lorena, L. A. (2010). A new mathematical model and a Lagrangean decomposition for the point-feature cartographic label placement problem. *Computers & Operations Research*, *37*(12), 2164–2172.

Nogueira, B., Pinheiro, R. G., & Tavares, E. (2021). Iterated local search for the generalized independent set problem. *Optimization Letters*, *15*, 1345–1369.

Pullan, W. (2008). Approximating the maximum vertex/edge weighted clique using local search. *Journal of Heuristics*, *14*, 117–134.

Singh, A., & Gupta, A. K. (2006). A hybrid heuristic for the minimum weight vertex cover problem. *Asia-Pacific Journal of Operational Research*, *23*(02), 273–285.

Wang, Y., Lü, Z., Glover, F., & Hao, J.-K. (2013). Probabilistic GRASP-tabu search algorithms for the UBQP problem. *Computers & Operations Research*, *40*(12), 3100–3107.

Wei, R., & Murray, A. T. (2012). An integrated approach for addressing geographic uncertainty in spatial optimization. *International Journal of Geographical Information Science*, *26*(7), 1231–1249.

Wu, Q., & Hao, J.-K. (2015). A review on algorithms for maximum clique problems. *European Journal of Operational Research*, *242*(3), 693–709.

Wu, Q., Hao, J.-K., & Glover, F. (2012). Multi-neighborhood tabu search for the maximum weight clique problem. *Annals of Operations Research*, *196*, 611–634.

Appendix: Detailed computational results

Detailed comparative results between our exact LA-B&B algorithm and the reference exact algorithm CB&B on the whole set of the 216 benchmark instances are provided in Tables 5-7. The first six columns in Tables 5-7 respectively indicate the instance name (Instance), the number of vertices ($|V|$), the number of permanent edges ($|E|$), the number of removable edges ($|E'|$), the permanent-edge density ρ_1 (computed as $\frac{2|E|}{|V|(|V|-1)}$), and the removable-edge density ρ_2 (computed as $\frac{2|E'|}{|V|(|V|-1)}$). Columns 7-10 present the results obtained by the two compared exact approaches under a time limit of 3 hours, including the best solution f_{best} achieved by the algorithms (marked with an asterisk '$*$' if the instance is solved to optimality by the corresponding algorithm), and the time required by the algorithm to solve the instance to optimality (denoted by 10800 seconds if the algorithm fails to attain the optimal solutions within the given time limit).

Detailed comparative results between the proposed heuristic algorithm ALS and the reference heuristic algorithm ILS-VND are presented in Tables 8-10. Column 2 reports the previously best-known results reported in the literature (marked with an asterisk '$*$' if the optimal solution for the instance is known). Columns 3-8 report for each algorithm the best result (f_{best}), the average result (f_{avg}), the success rate ($success$) to achieve the best result, and the average running time ($time(s)$) in seconds to reach the best result across the 10 independent runs.

The best results found by the proposed algorithm ALS and the reference algorithm ILS-VND are marked in bold text.

Table 5: Computational results of the exact algorithms for the instances with $p_r = 0.25$

Instance	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E'\rvert$	$\lvert\rho_1\rvert$	$\lvert\rho_2\rvert$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
SET1											
brock200_2_A_25	200	7416	2460	0.37	0.12	986*	280.47	986*	1.38	986*	1.43
brock400_2_A_25	400	44847	14939	0.56	0.19	765*	441.41	765*	1.03	765*	1.05
C125.9_A_25	125	5250	1713	0.68	0.22	454*	0.78	454*	<0.01	454*	<0.01
C250.9_A_25	250	20940	7044	0.67	0.23	581*	10.42	581*	0.01	581*	0.01
gen200_p0.9_55_A_25	200	13434	4476	0.68	0.22	535*	4.11	535*	<0.01	535*	<0.01
gen400_p0.9_75_A_25	400	53671	18149	0.67	0.23	628*	82.19	628*	0.08	628*	0.09
hamming8-4_A_25	256	15580	5284	0.48	0.16	1094*	240.42	1094*	0.75	1094*	0.78
keller4_A_25	171	7057	2378	0.49	0.16	941*	15.13	941*	0.05	941*	0.06
MANN_a27_A_25	378	53050	17501	0.74	0.25	533*	28.30	533*	0.02	533*	0.02
p_hat300-1_A_25	300	8239	2694	0.18	0.06	2308	10800	2744	10800	2680	10800
p_hat300-2_A_25	300	16403	5525	0.37	0.12	2076*	5181.81	2076*	536.27	2076*	541.14
p_hat300-3_A_25	300	25132	8258	0.56	0.18	739*	99.72	739*	0.34	739*	0.34
brock200_2_B_25	200	7416	2460	0.37	0.12	962*	276.31	962*	1.43	962*	1.42
brock400_2_B_25	400	44847	14939	0.56	0.19	741*	445.22	741*	1.11	741*	1.12
C125.9_B_25	125	5250	1713	0.68	0.22	437*	1.02	437*	<0.01	437*	<0.01
C250.9_B_25	250	20940	7044	0.67	0.23	549*	11.27	549*	0.01	549*	0.01
gen200_p0.9_55_B_25	200	13434	4476	0.68	0.22	510*	4.56	510*	<0.01	510*	<0.01
gen400_p0.9_75_B_25	400	53671	18149	0.67	0.23	595*	80.30	595*	0.10	595*	0.11
hamming8-4_B_25	256	15580	5284	0.48	0.16	1094*	233.59	1094*	0.81	1094*	0.84
keller4_B_25	171	7057	2378	0.49	0.16	941*	14.50	941*	0.06	941*	0.06
MANN_a27_B_25	378	53050	17501	0.74	0.25	503*	31.02	503*	0.02	503*	0.02
p_hat300-1_B_25	300	8239	2694	0.18	0.06	2259	10800	2712	10800	2643	10800
p_hat300-2_B_25	300	16403	5525	0.37	0.12	2062*	5194.12	2062*	573.86	2062*	582.77
p_hat300-3_B_25	300	25132	8258	0.56	0.18	713*	101.70	713*	0.33	713*	0.34

Continued on next page

Continued

Instance	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert E' \rvert$	$\lvert \rho_1 \rvert$	$\lvert \rho_2 \rvert$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
brock200_2_C_25	200	7416	2460	0.37	0.12	932*	279.53	932*	1.45	932*	1.50
brock400_2_C_25	400	44847	14939	0.56	0.19	698*	455.22	698*	1.11	698*	1.13
C125.9_C_25	125	5250	1713	0.68	0.22	403*	1.03	403*	<0.01	403*	<0.01
C250.9_C_25	250	20940	7044	0.67	0.23	502*	11.67	502*	0.01	502*	0.01
gen200_p0.9_55_C_25	200	13434	4476	0.68	0.22	467*	4.28	467*	<0.01	467*	<0.01
gen400_p0.9_75_C_25	400	53671	18149	0.67	0.23	533*	80.80	533*	0.10	533*	0.10
hamming8-4_C_25	256	15580	5284	0.48	0.16	1094*	229.03	1094*	0.80	1094*	0.82
keller4_C_25	171	7057	2378	0.49	0.16	941*	13.98	941*	0.06	941*	0.06
MANN_a27_C_25	378	53050	17501	0.74	0.25	443*	29.52	443*	0.02	443*	0.02
p_hat300-1_C_25	300	8239	2694	0.18	0.06	2205	10800	2649	10800	2566	10800
p_hat300-2_C_25	300	16403	5525	0.37	0.12	2033*	5359.95	2033*	659.99	2033*	668.75
p_hat300-3_C_25	300	25132	8258	0.56	0.18	688*	111.02	688*	0.36	688*	0.37
SET2											
brock200_2_A_25	200	7416	2460	0.37	0.12	1489*	452.25	1489*	3.92	1489*	4.10
brock400_2_A_25	400	44847	14939	0.56	0.19	1084*	593.22	1084*	1.62	1084*	1.64
C125.9_A_25	125	5250	1713	0.68	0.22	685*	1.08	685*	<0.01	685*	<0.01
C250.9_A_25	250	20940	7044	0.67	0.23	785*	12.05	785*	0.02	785*	0.02
gen200_p0.9_55_A_25	200	13434	4476	0.68	0.22	778*	4.91	778*	<0.01	778*	<0.01
gen400_p0.9_75_A_25	400	53671	18149	0.67	0.23	882*	94.33	882*	0.12	882*	0.12
hamming8-4_A_25	256	15580	5284	0.48	0.16	1790*	331.34	1790*	1.40	1790*	1.54
keller4_A_25	171	7057	2378	0.49	0.16	1500*	22.55	1500*	0.08	1500*	0.08
MANN_a27_A_25	378	53050	17501	0.74	0.25	683*	32.16	683*	0.02	683*	0.02
p_hat300-1_A_25	300	8239	2694	0.18	0.06	3981	10800	4674	10800	4182	10800
p_hat300-2_A_25	300	16403	5525	0.37	0.12	2994*	8743.06	2994*	9118.22	2994*	9402.32
p_hat300-3_A_25	300	25132	8258	0.56	0.18	1180*	136.75	1180*	0.62	1180*	0.65
brock200_2_B_25	200	7416	2460	0.37	0.12	739*	455.52	739*	3.91	739*	4.10

Continued on next page

Continued

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
brock400_2_B_25	400	44847	14939	0.56	0.19	534*	611.47	534*	1.59	534*	1.66
C125.9_B_25	125	5250	1713	0.68	0.22	335*	1.09	335*	<0.01	335*	<0.01
C250.9_B_25	250	20940	7044	0.67	0.23	385*	13.36	385*	0.02	385*	0.02
gen200_p0.9_55_B_25	200	13434	4476	0.68	0.22	378*	5.38	378*	<0.01	378*	<0.01
gen400_p0.9_75_B_25	400	53671	18149	0.67	0.23	432*	95.73	432*	0.12	432*	0.12
hamming8-4_B_25	256	15580	5284	0.48	0.16	890*	333.17	890*	1.40	890*	1.53
keller4_B_25	171	7057	2378	0.49	0.16	750*	22.20	750*	0.08	750*	0.08
MANN_a27_B_25	378	53050	17501	0.74	0.25	333*	33.55	333*	0.02	333*	0.02
p_hat300-1_B_25	300	8239	2694	0.18	0.06	1981	10800	2324	10800	2082	10800
p_hat300-2_B_25	300	16403	5525	0.37	0.12	1494*	8947.42	1494*	9327.29	1494*	9322.18
p_hat300-3_B_25	300	25132	8258	0.56	0.18	580*	134.41	580*	0.64	580*	0.67
brock200_2_C_25	200	7416	2460	0.37	0.12	364*	460.38	364*	4.05	364*	4.17
brock400_2_C_25	400	44847	14939	0.56	0.19	259*	646.13	259*	1.66	259*	1.72
C125.9_C_25	125	5250	1713	0.68	0.22	160*	0.95	160*	<0.01	160*	<0.01
C250.9_C_25	250	20940	7044	0.67	0.23	185*	13.20	185*	0.02	185*	0.02
gen200_p0.9_55_C_25	200	13434	4476	0.68	0.22	178*	5.45	178*	<0.01	178*	<0.01
gen400_p0.9_75_C_25	400	53671	18149	0.67	0.23	207*	98.13	207*	0.13	207*	0.13
hamming8-4_C_25	256	15580	5284	0.48	0.16	440*	355.00	440*	1.40	440*	1.52
keller4_C_25	171	7057	2378	0.49	0.16	375*	22.80	375*	0.07	375*	0.08
MANN_a27_C_25	378	53050	17501	0.74	0.25	158*	32.58	158*	0.02	158*	0.02
p_hat300-1_C_25	300	8239	2694	0.18	0.06	981	10800	1149	10800	1032	10800
p_hat300-2_C_25	300	16403	5525	0.37	0.12	744*	9479.23	744*	9095.56	744*	9087.09
p_hat300-3_C_25	300	25132	8258	0.56	0.18	280*	142.30	280*	0.67	280*	0.69
# of best						66		66		66	
Avg. time(s)							1617.88		1307.59		1311.68
p-value						2.77E-02	1.65E-10			2.77E-02	2.27E-05

Table 6: Computational results of the exact algorithms for the instances with $p_r = 0.5$

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
SET1											
brock200_2_A_50	200	4916	4960	0.25	0.25	1298*	8389.52	1298*	76.68	1298*	79.71
brock400_2_A_50	400	29711	30075	0.37	0.38	1103	10800	1123*	273.61	1123*	299.29
C125.9_A_50	125	3500	3463	0.45	0.45	627*	4.69	627*	0.02	627*	0.02
C250.9_A_50	250	14017	13967	0.45	0.45	817*	266.20	817*	1.01	817*	1.12
gen200_p0.9_55_A_50	200	8908	9002	0.45	0.45	785*	68.00	785*	0.27	785*	0.28
gen400_p0.9_75_A_50	400	35823	35997	0.45	0.45	895*	5698.50	895*	22.62	895*	23.36
hamming8-4_A_50	256	10329	10535	0.32	0.32	1301*	7607.67	1301*	41.63	1301*	43.49
keller4_A_50	171	4738	4697	0.33	0.32	1118*	153.22	1118*	1.41	1118*	1.51
MANN_a27_A_50	378	35345	35206	0.50	0.49	812*	1217.00	812*	4.31	812*	4.47
p_hat300-1_A_50	300	5505	5428	0.12	0.12	2568	10800	3129	10800	3026	10800
p_hat300-2_A_50	300	11051	10877	0.25	0.24	2063	10800	2477	10800	2477	10800
p_hat300-3_A_50	300	16820	16570	0.38	0.37	1029*	3747.61	1029*	28.60	1029*	30.49
brock200_2_B_50	200	4916	4960	0.25	0.25	1224*	9588.61	1224*	97.03	1224*	102.83
brock400_2_B_50	400	29711	30075	0.37	0.38	1010	10800	1035*	343.30	1035*	360.79
C125.9_B_50	125	3500	3463	0.45	0.45	582*	5.41	582*	0.02	582*	0.02
C250.9_B_50	250	14017	13967	0.45	0.45	744*	292.36	744*	1.18	744*	1.24
gen200_p0.9_55_B_50	200	8908	9002	0.45	0.45	716*	72.05	716*	0.32	716*	0.31
gen400_p0.9_75_B_50	400	35823	35997	0.45	0.45	805*	6602.48	805*	25.84	805*	26.94
hamming8-4_B_50	256	10329	10535	0.32	0.32	1255*	7779.77	1255*	47.64	1255*	50.52
keller4_B_50	171	4738	4697	0.33	0.32	1094*	155.55	1094*	1.64	1094*	1.64
MANN_a27_B_50	378	35345	35206	0.50	0.49	707*	1417.42	707*	5.02	707*	5.15
p_hat300-1_B_50	300	5505	5428	0.12	0.12	2492	10800	3023	10800	2950	10800
p_hat300-2_B_50	300	11051	10877	0.25	0.24	1979	10800	2405	10800	2405	10800
p_hat300-3_B_50	300	16820	16570	0.38	0.37	967*	4066.88	967*	34.41	967*	36.66

Continued on next page

Continued

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
brock200_2_C_50	200	4916	4960	0.25	0.25	1101	10800	1101*	161.31	1101*	169.54
brock400_2_C_50	400	29711	30075	0.37	0.38	861	10800	892*	501.59	892*	525.79
C125.9_C_50	125	3500	3463	0.45	0.45	506*	5.80	506*	0.03	506*	0.03
C250.9_C_50	250	14017	13967	0.45	0.45	623*	333.94	623*	1.59	623*	1.61
gen200_p0.9_55_C_50	200	8908	9002	0.45	0.45	597*	90.66	597*	0.40	597*	0.41
gen400_p0.9_75_C_50	400	35823	35997	0.45	0.45	651*	7858.08	651*	36.60	651*	37.46
hamming8-4_C_50	256	10329	10535	0.32	0.32	1184*	7447.34	1184*	65.22	1184*	69.99
keller4_C_50	171	4738	4697	0.33	0.32	1049*	144.64	1049*	1.88	1049*	2.06
MANN_a27_C_50	378	35345	35206	0.50	0.49	552*	1848.84	552*	6.79	552*	6.95
p_hat300-1_C_50	300	5505	5428	0.12	0.12	2303	10800	2897	10800	2818	10800
p_hat300-2_C_50	300	11051	10877	0.25	0.24	1852	10800	2263	10800	2263	10800
p_hat300-3_C_50	300	16820	16570	0.38	0.37	851*	4708.78	851*	51.53	851*	54.27
SET2											
brock200_2_A_50	200	4916	4960	0.25	0.25	2034	10800	2034*	537.13	2034*	547.48
brock400_2_A_50	400	29711	30075	0.37	0.38	1628	10800	1630*	1002.85	1630*	1020.85
C125.9_A_50	125	3500	3463	0.45	0.45	1152*	7.19	1152*	0.03	1152*	0.03
C250.9_A_50	250	14017	13967	0.45	0.45	1236*	343.06	1236*	1.96	1236*	1.98
gen200_p0.9_55_A_50	200	8908	9002	0.45	0.45	1151*	113.13	1151*	0.57	1151*	0.59
gen400_p0.9_75_A_50	400	35823	35997	0.45	0.45	1335*	8466.62	1335*	50.17	1335*	51.26
hamming8-4_A_50	256	10329	10535	0.32	0.32	2155	10800	2155*	127.59	2155*	132.52
keller4_A_50	171	4738	4697	0.33	0.32	1759*	352.73	1759*	3.45	1759*	3.77
MANN_a27_A_50	378	35345	35206	0.50	0.49	1226*	1950.19	1226*	7.25	1226*	7.50
p_hat300-1_A_50	300	5505	5428	0.12	0.12	4740	10800	5637	10800	4725	10800
p_hat300-2_A_50	300	11051	10877	0.25	0.24	3342	10800	3943	10800	3845	10800
p_hat300-3_A_50	300	16820	16570	0.38	0.37	1658*	7242.83	1658*	119.22	1658*	123.13
brock200_2_B_50	200	4916	4960	0.25	0.25	984	10800	984*	569.10	984*	580.08

Continued on next page

Continued

Instance	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert E'\rvert$	$\lvert\rho_1\rvert$	$\lvert\rho_2\rvert$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
brock400_2_B_50	400	29711	30075	0.37	0.38	765	10800	780*	1051.65	780*	1082.37
C125.9_B_50	125	3500	3463	0.45	0.45	552*	7.30	552*	0.03	552*	0.03
C250.9_B_50	250	14017	13967	0.45	0.45	586*	390.59	586*	2.01	586*	2.10
gen200_p0.9_55_B_50	200	8908	9002	0.45	0.45	551*	120.63	551*	0.61	551*	0.62
gen400_p0.9_75_B_50	400	35823	35997	0.45	0.45	635*	9394.86	635*	53.07	635*	54.34
hamming8-4_B_50	256	10329	10535	0.32	0.32	1055	10800	1055*	126.85	1055*	132.97
keller4_B_50	171	4738	4697	0.33	0.32	859*	367.13	859*	3.44	859*	3.78
MANN_a27_B_50	378	35345	35206	0.50	0.49	576*	2160.95	576*	7.66	576*	7.97
p_hat300-1_B_50	300	5505	5428	0.12	0.12	2340	10800	2787	10800	2325	10800
p_hat300-2_B_50	300	11051	10877	0.25	0.24	1642	10800	1943	10800	1895	10800
p_hat300-3_B_50	300	16820	16570	0.38	0.37	808*	7790.11	808*	120.73	808*	124.22
brock200_2_C_50	200	4916	4960	0.25	0.25	459	10800	459*	693.21	459*	707.89
brock400_2_C_50	400	29711	30075	0.37	0.38	341	10800	355*	1233.26	355*	1281.71
C125.9_C_50	125	3500	3463	0.45	0.45	252*	8.00	252*	0.04	252*	0.04
C250.9_C_50	250	14017	13967	0.45	0.45	261*	500.92	261*	2.37	261*	2.45
gen200_p0.9_55_C_50	200	8908	9002	0.45	0.45	251*	146.20	251*	0.67	251*	0.70
gen400_p0.9_75_C_50	400	35823	35997	0.45	0.45	285	10800	285*	61.22	285*	63.20
hamming8-4_C_50	256	10329	10535	0.32	0.32	505	10800	505*	132.53	505*	139.74
keller4_C_50	171	4738	4697	0.33	0.32	409*	388.67	409*	3.83	409*	3.91
MANN_a27_C_50	378	35345	35206	0.50	0.49	251*	2666.12	251*	9.02	251*	9.37
p_hat300-1_C_50	300	5505	5428	0.12	0.12	1138	10800	1362	10800	1125	10800
p_hat300-2_C_50	300	11051	10877	0.25	0.24	788	10800	943	10800	900	10800
p_hat300-3_C_50	300	16820	16570	0.38	0.37	383*	9014.17	383*	125.63	383*	131.99
# of best						46		60		60	
Avg. time(s)							5719.48		1909.45		1913.28
p-value						1.36E-03	1.63E-11			7.69E-03	9.21E-11

Table 7: Computational results of the exact algorithms for the instances with $p_r = 0.75$

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
SET1											
brock200_2_A_75	200	2438	7438	0.12	0.37	1615	10800	1885	10800	1851	10800
brock400_2_A_75	400	14751	45035	0.18	0.56	1403	10800	1728	10800	1667	10800
C125.9_A_75	125	1733	5230	0.22	0.67	1023*	247.27	1023*	3.42	1023*	3.45
C250.9_A_75	250	7073	20911	0.23	0.67	1200	10800	1236*	4485.02	1236*	4584.71
gen200_p0.9_55_A_75	200	4425	13485	0.22	0.68	1186	10800	1206*	518.07	1206*	550.74
gen400_p0.9_75_A_75	400	18063	53757	0.23	0.67	1155	10800	1490	10800	1490	10800
hamming8-4_A_75	256	5173	15691	0.16	0.48	1549	10800	1759	10800	1723	10800
keller4_A_75	171	2400	7035	0.17	0.48	1400	10800	1434*	1355.84	1434*	1451.69
MANN_a27_A_75	378	17580	52971	0.25	0.74	1313	10800	1323	10800	1323	10800
p_hat300-1_A_75	300	2734	8199	0.06	0.18	2504	10800	4164	10800	3932	10800
p_hat300-2_A_75	300	5603	16325	0.12	0.36	2120	10800	2990	10800	2902	10800
p_hat300-3_A_75	300	8388	25002	0.19	0.56	1350	10800	1564	10800	1561	10800
brock200_2_B_75	200	2438	7438	0.12	0.37	1416	10800	1641	10800	1602	10800
brock400_2_B_75	400	14751	45035	0.18	0.56	1129	10800	1386	10800	1341	10800
C125.9_B_75	125	1733	5230	0.22	0.67	856*	395.23	856*	6.01	856*	6.37
C250.9_B_75	250	7073	20911	0.23	0.67	960	10800	1001*	10629.31	1001	10800
gen200_p0.9_55_B_75	200	4425	13485	0.22	0.68	961	10800	983*	1301.00	983*	1382.12
gen400_p0.9_75_B_75	400	18063	53757	0.23	0.67	920	10800	1120	10800	1105	10800
hamming8-4_B_75	256	5173	15691	0.16	0.48	1356	10800	1579	10800	1486	10800
keller4_B_75	171	2400	7035	0.17	0.48	1230	10800	1268*	3598.64	1268*	3855.16
MANN_a27_B_75	378	17580	52971	0.25	0.74	922	10800	1021	10800	1011	10800
p_hat300-1_B_75	300	2734	8199	0.06	0.18	2306	10800	3886	10800	3547	10800
p_hat300-2_B_75	300	5603	16325	0.12	0.36	1762	10800	2782	10800	2588	10800
p_hat300-3_B_75	300	8388	25002	0.19	0.56	1144	10800	1299	10800	1209	10800

Continued on next page

Continued

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
brock200_2_C_75	200	2438	7438	0.12	0.37	1158	10800	1321	10800	1303	10800
brock400_2_C_75	400	14751	45035	0.18	0.56	793	10800	1033	10800	961	10800
C125.9_C_75	125	1733	5230	0.22	0.67	644*	745.61	644*	15.59	644*	16.18
C250.9_C_75	250	7073	20911	0.23	0.67	688	10800	734	10800	734	10800
gen200_p0.9_55_C_75	200	4425	13485	0.22	0.68	707	10800	727*	3880.52	727*	4055.63
gen400_p0.9_75_C_75	400	18063	53757	0.23	0.67	613	10800	772	10800	756	10800
hamming8-4_C_75	256	5173	15691	0.16	0.48	1120	10800	1378	10800	1248	10800
keller4_C_75	171	2400	7035	0.17	0.48	1060	10800	1109*	8463.12	1109*	9177.50
MANN_a27_C_75	378	17580	52971	0.25	0.74	583	10800	651	10800	648	10800
p_hat300-1_C_75	300	2734	8199	0.06	0.18	1866	10800	3480	10800	3191	10800
p_hat300-2_C_75	300	5603	16325	0.12	0.36	1511	10800	2473	10800	2249	10800
p_hat300-3_C_75	300	8388	25002	0.19	0.56	886	10800	1004	10800	919	10800
SET2											
brock200_2_A_75	200	2438	7438	0.12	0.37	2999	10800	3326	10800	3078	10800
brock400_2_A_75	400	14751	45035	0.18	0.56	2603	10800	2941	10800	2787	10800
C125.9_A_75	125	1733	5230	0.22	0.67	1837*	365.22	1837*	19.42	1837*	22.59
C250.9_A_75	250	7073	20911	0.23	0.67	2011	10800	2171	10800	2171	10800
gen200_p0.9_55_A_75	200	4425	13485	0.22	0.68	2094	10800	2096*	2844.80	2096*	3062.39
gen400_p0.9_75_A_75	400	18063	53757	0.23	0.67	2164	10800	2404	10800	2310	10800
hamming8-4_A_75	256	5173	15691	0.16	0.48	2813	10800	3124	10800	3124	10800
keller4_A_75	171	2400	7035	0.17	0.48	2532	10800	2690*	5738.66	2690*	7062.94
MANN_a27_A_75	378	17580	52971	0.25	0.74	1976	10800	2208	10800	2206	10800
p_hat300-1_A_75	300	2734	8199	0.06	0.18	5566	10800	7899	10800	6666	10800
p_hat300-2_A_75	300	5603	16325	0.12	0.36	4102	10800	5343	10800	4714	10800
p_hat300-3_A_75	300	8388	25002	0.19	0.56	2477	10800	2838	10800	2658	10800
brock200_2_B_75	200	2438	7438	0.12	0.37	1398	10800	1533	10800	1427	10800

Continued on next page

Continued

Instance	$\|V\|$	$\|E\|$	$\|E'\|$	$\|\rho_1\|$	$\|\rho_2\|$	CB&B		LA-B&B+ALS		LA-B&B	
						f_{best}	time(s)	f_{best}	time(s)	f_{best}	time(s)
brock400_2_B_75	400	14751	45035	0.18	0.56	1161	10800	1291	10800	1210	10800
C125.9_B_75	125	1733	5230	0.22	0.67	837*	523.06	837*	25.25	837*	29.63
C250.9_B_75	250	7073	20911	0.23	0.67	911	10800	971	10800	969	10800
gen200_p0.9_55_B_75	200	4425	13485	0.22	0.68	944	10800	946*	3883.01	946*	4332.73
gen400_p0.9_75_B_75	400	18063	53757	0.23	0.67	936	10800	1054	10800	1010	10800
hamming8-4_B_75	256	5173	15691	0.16	0.48	1309	10800	1474	10800	1474	10800
keller4_B_75	171	2400	7035	0.17	0.48	1190	10800	1254*	8269.17	1254*	9950.06
MANN_a27_B_75	378	17580	52971	0.25	0.74	875	10800	958	10800	956	10800
p_hat300-1_B_75	300	2734	8199	0.06	0.18	2666	10800	3818	10800	3216	10800
p_hat300-2_B_75	300	5603	16325	0.12	0.36	1952	10800	2543	10800	2177	10800
p_hat300-3_B_75	300	8388	25002	0.19	0.56	1101	10800	1288	10800	1208	10800
brock200_2_C_75	200	2438	7438	0.12	0.37	607	10800	658	10800	607	10800
brock400_2_C_75	400	14751	45035	0.18	0.56	461	10800	500	10800	465	10800
C125.9_C_75	125	1733	5230	0.22	0.67	337*	1518.80	337*	62.68	337*	64.96
C250.9_C_75	250	7073	20911	0.23	0.67	359	10800	372	10800	370	10800
gen200_p0.9_55_C_75	200	4425	13485	0.22	0.68	352	10800	371	10800	371	10800
gen400_p0.9_75_C_75	400	18063	53757	0.23	0.67	356	10800	395	10800	379	10800
hamming8-4_C_75	256	5173	15691	0.16	0.48	556	10800	652	10800	649	10800
keller4_C_75	171	2400	7035	0.17	0.48	510	10800	558	10800	554	10800
MANN_a27_C_75	378	17580	52971	0.25	0.74	325	10800	335	10800	335	10800
p_hat300-1_C_75	300	2734	8199	0.06	0.18	1205	10800	1793	10800	1491	10800
p_hat300-2_C_75	300	5603	16325	0.12	0.36	854	10800	1159	10800	981	10800
p_hat300-3_C_75	300	8388	25002	0.19	0.56	453	10800	529	10800	494	10800
# of best						6		18		18	
Avg. time(s)							9952.71		8865.27		8938.24
p-value						2.40E-12	1.96E-04			3.51E-09	1.96E-04

Table 8: Computational results of the heuristic algorithms for the instances with $p_r = 0.25$

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
SET1									
brock200_2_A_25	986*	**986**	986	10	<0.01	**986**	986	10	<0.01
brock400_2_A_25	765*	**765**	765	10	0.07	**765**	765	10	<0.01
C125.9_A_25	454*	**454**	454	10	<0.01	**454**	454	10	<0.01
C250.9_A_25	581*	**581**	581	10	0.03	**581**	581	10	<0.01
gen200_p0.9_55_A_25	535*	**535**	535	10	<0.01	**535**	535	10	<0.01
gen400_p0.9_75_A_25	628*	**628**	628	10	3.10	**628**	628	10	<0.01
hamming8-4_A_25	1094*	**1094**	1094	10	0.05	**1094**	1094	10	<0.01
keller4_A_25	941*	**941**	941	10	0.01	**941**	941	10	<0.01
MANN_a27_A_25	533*	**533**	533	10	0.26	**533**	533	10	<0.01
p_hat300-1_A_25	2744	**2744**	2744	10	0.20	**2744**	2744	10	<0.01
p_hat300-2_A_25	2076*	**2076**	2076	10	<0.01	**2076**	2076	10	<0.01
p_hat300-3_A_25	739*	**739**	739	10	0.07	**739**	739	10	<0.01
brock200_2_B_25	962*	**962**	962	10	0.03	**962**	962	10	<0.01
brock400_2_B_25	741*	**741**	741	10	0.09	**741**	741	10	<0.01
C125.9_B_25	437*	**437**	437	10	<0.01	**437**	437	10	<0.01
C250.9_B_25	549*	**549**	549	10	0.03	**549**	549	10	<0.01
gen200_p0.9_55_B_25	510*	**510**	510	10	0.01	**510**	510	10	<0.01
gen400_p0.9_75_B_25	595*	**595**	595	10	0.71	**595**	595	10	<0.01
hamming8-4_B_25	1094*	**1094**	1094	10	0.14	**1094**	1094	10	<0.01
keller4_B_25	941*	**941**	941	10	0.03	**941**	941	10	<0.01
MANN_a27_B_25	503*	**503**	503	10	0.05	**503**	503	10	0.08
p_hat300-1_B_25	2712	**2712**	2712	10	0.44	**2712**	2712	10	<0.01
p_hat300-2_B_25	2062*	**2062**	2062	10	<0.01	**2062**	2062	10	<0.01
p_hat300-3_B_25	713*	**713**	713	10	0.04	**713**	713	10	<0.01
brock200_2_C_25	932*	**932**	932	10	0.04	**932**	932	10	<0.01
brock400_2_C_25	698*	**698**	698	10	0.06	**698**	698	10	<0.01
C125.9_C_25	403*	**403**	403	10	<0.01	**403**	403	10	<0.01
C250.9_C_25	502*	**502**	502	10	0.08	**502**	502	10	<0.01
gen200_p0.9_55_C_25	467*	**467**	467	10	0.02	**467**	467	10	<0.01
gen400_p0.9_75_C_25	533*	**533**	533	10	0.59	**533**	533	10	<0.01
hamming8-4_C_25	1094*	**1094**	1094	10	0.06	**1094**	1094	10	<0.01
keller4_C_25	941*	**941**	941	10	0.03	**941**	941	10	<0.01
MANN_a27_C_25	443*	**443**	443	10	0.15	**443**	443	10	<0.01
p_hat300-1_C_25	2649	**2649**	2649	10	0.28	**2649**	2649	10	<0.01
p_hat300-2_C_25	2033*	**2033**	2033	10	<0.01	**2033**	2033	10	<0.01
p_hat300-3_C_25	688*	**688**	688	10	0.04	**688**	688	10	<0.01
SET 2									
brock200_2_A_25	1489*	**1489**	1489	10	0.12	**1489**	1489	10	<0.01
brock400_2_A_25	1084*	**1084**	1084	10	1.79	**1084**	1084	10	0.02
C125.9_A_25	685*	**685**	685	10	0.21	**685**	685	10	<0.01
C250.9_A_25	785*	**785**	785	10	0.12	**785**	785	10	<0.01
gen200_p0.9_55_A_25	778*	**778**	778	10	0.15	**778**	778	10	<0.01
gen400_p0.9_75_A_25	882*	**882**	882	10	0.83	**882**	882	10	<0.01
hamming8-4_A_25	1790*	**1790**	1790	10	0.62	**1790**	1790	10	0.02
keller4_A_25	1500*	**1500**	1500	10	<0.01	**1500**	1500	10	<0.01
MANN_a27_A_25	683*	**683**	683	10	0.04	**683**	683	10	<0.01
p_hat300-1_A_25	4674	**4674**	4674	10	1.21	**4674**	4674	10	<0.01
p_hat300-2_A_25	2994*	**2994**	2994	10	0.03	**2994**	2994	10	<0.01

Continued on next page

Continued

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
p_hat300-3_A_25	1180*	**1180**	1180	10	0.78	**1180**	1180	10	<0.01
brock200_2_B_25	739*	**739**	739	10	0.07	**739**	739	10	<0.01
brock400_2_B_25	534*	**534**	534	10	2.10	**534**	534	10	0.02
C125.9_B_25	335*	**335**	335	10	0.18	**335**	335	10	<0.01
C250.9_B_25	385*	**385**	385	10	0.18	**385**	385	10	<0.01
gen200_p0.9_55_B_25	378*	**378**	378	10	0.11	**378**	378	10	<0.01
gen400_p0.9_75_B_25	432*	**432**	432	10	0.89	**432**	432	10	<0.01
hamming8-4_B_25	890*	**890**	890	10	0.65	**890**	890	10	<0.01
keller4_B_25	750*	**750**	750	10	<0.01	**750**	750	10	<0.01
MANN_a27_B_25	333*	**333**	333	10	0.05	**333**	333	10	<0.01
p_hat300-1_B_25	2324	**2324**	2324	10	1.70	**2324**	2324	10	<0.01
p_hat300-2_B_25	1494*	**1494**	1494	10	0.04	**1494**	1494	10	<0.01
p_hat300-3_B_25	580*	**580**	580	10	1.30	**580**	580	10	<0.01
brock200_2_C_25	364*	**364**	364	10	0.11	**364**	364	10	<0.01
brock400_2_C_25	259*	**259**	259	10	1.31	**259**	259	10	0.02
C125.9_C_25	160*	**160**	160	10	0.17	**160**	160	10	<0.01
C250.9_C_25	185*	**185**	185	10	0.14	**185**	185	10	<0.01
gen200_p0.9_55_C_25	178*	**178**	178	10	0.15	**178**	178	10	<0.01
gen400_p0.9_75_C_25	207*	**207**	207	10	0.74	**207**	207	10	<0.01
hamming8-4_C_25	440*	**440**	440	10	0.63	**440**	440	10	<0.01
keller4_C_25	375*	**375**	375	10	<0.01	**375**	375	10	<0.01
MANN_a27_C_25	158*	**158**	158	10	0.05	**158**	158	10	<0.01
p_hat300-1_C_25	1149	**1149**	1149	10	1.27	**1149**	1149	10	<0.01
p_hat300-2_C_25	744*	**744**	744	10	0.03	**744**	744	10	<0.01
p_hat300-3_C_25	280*	**280**	280	10	0.83	**280**	280	10	<0.01
# of best		72				72			
# of best Mean			72				72		
Avg. time(s)					0.353				<0.01
p-value		1.00	1.00		4.99E-11				

Table 9: Computational results of the heuristic algorithms for the instances with $p_r = 0.5$

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
SET1									
brock200_2_A_50	1298*	**1298**	1298	10	0.04	**1298**	1298	10	<0.01
brock400_2_A_50	1123	**1123**	1123	10	0.25	**1123**	1123	10	<0.01
C125.9_A_50	627*	**627**	627	10	0.01	**627**	627	10	<0.01
C250.9_A_50	817*	**817**	817	10	0.45	**817**	817	10	<0.01
gen200_p0.9_55_A_50	785*	**785**	785	10	0.04	**785**	785	10	<0.01
gen400_p0.9_75_A_50	895*	**895**	895	10	0.66	**895**	895	10	<0.01
hamming8-4_A_50	1301*	**1301**	1301	10	0.04	**1301**	1301	10	<0.01
keller4_A_50	1118*	**1118**	1118	10	0.02	**1118**	1118	10	<0.01
MANN_a27_A_50	812*	**812**	812	10	0.24	**812**	812	10	<0.01
p_hat300-1_A_50	3129	**3129**	3129	10	0.86	**3129**	3129	10	0.20
p_hat300-2_A_50	2477	**2477**	2477	10	0.01	**2477**	2477	10	<0.01
p_hat300-3_A_50	1029*	**1029**	1029	10	0.26	**1029**	1029	10	<0.01

Continued on next page

Continued

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
brock200_2_B_50	1224*	**1224**	1224	10	0.04	**1224**	1224	10	<0.01
brock400_2_B_50	1035	**1035**	1035	10	0.13	**1035**	1035	10	<0.01
C125.9_B_50	582*	**582**	582	10	<0.01	**582**	582	10	<0.01
C250.9_B_50	744*	**744**	744	10	0.32	**744**	744	10	<0.01
gen200_p0.9_55_B_50	716*	**716**	716	10	0.08	**716**	716	10	<0.01
gen400_p0.9_75_B_50	805*	**805**	805	10	0.49	**805**	805	10	<0.01
hamming8-4_B_50	1255*	**1255**	1255	10	0.03	**1255**	1255	10	<0.01
keller4_B_50	1094*	**1094**	1094	10	0.03	**1094**	1094	10	<0.01
MANN_a27_B_50	707*	**707**	707	10	1.32	**707**	707	10	<0.01
p_hat300-1_B_50	3023	**3023**	3023	10	0.89	**3023**	3023	10	0.05
p_hat300-2_B_50	2405	**2405**	2405	10	0.02	**2405**	2405	10	<0.01
p_hat300-3_B_50	967*	**967**	967	10	0.17	**967**	967	10	<0.01
brock200_2_C_50	1101*	**1101**	1101	10	0.02	**1101**	1101	10	<0.01
brock400_2_C_50	892	**892**	892	10	0.12	**892**	892	10	<0.01
C125.9_C_50	506*	**506**	506	10	<0.01	**506**	506	10	<0.01
C250.9_C_50	623*	**623**	623	10	0.15	**623**	623	10	<0.01
gen200_p0.9_55_C_50	597*	**597**	597	10	0.04	**597**	597	10	<0.01
gen400_p0.9_75_C_50	651*	**651**	651	10	0.19	**651**	651	10	<0.01
hamming8-4_C_50	1184*	**1184**	1184	10	0.09	**1184**	1184	10	<0.01
keller4_C_50	1049*	**1049**	1049	10	0.04	**1049**	1049	10	<0.01
MANN_a27_C_50	552*	**552**	552	10	1.44	**552**	552	10	<0.01
p_hat300-1_C_50	2897	**2897**	2897	10	0.05	**2897**	2897	10	<0.01
p_hat300-2_C_50	2263	**2263**	2263	10	0.03	**2263**	2263	10	<0.01
p_hat300-3_C_50	851*	**851**	851	10	0.16	**851**	851	10	<0.01
SET 2									
brock200_2_A_50	2034*	**2034**	2034	10	0.80	**2034**	2034	10	<0.01
brock400_2_A_50	1630	**1630**	1629.6	9	11.24	**1630**	1630	10	0.07
C125.9_A_50	1152*	**1152**	1152	10	0.03	**1152**	1152	10	<0.01
C250.9_A_50	1236*	**1236**	1236	10	3.17	**1236**	1236	10	0.01
gen200_p0.9_55_A_50	1151*	**1151**	1151	10	2.76	**1151**	1151	10	<0.01
gen400_p0.9_75_A_50	1335*	**1335**	1335	10	2.10	**1335**	1335	10	0.02
hamming8-4_A_50	2155*	**2155**	2155	10	0.94	**2155**	2155	10	<0.01
keller4_A_50	1759*	**1759**	1759	10	0.27	**1759**	1759	10	<0.01
MANN_a27_A_50	1226*	**1226**	1208.8	8	11.64	**1226**	1226	10	0.02
p_hat300-1_A_50	5637	**5637**	5637	10	0.64	**5637**	5637	10	<0.01
p_hat300-2_A_50	3943	**3943**	3943	10	0.33	**3943**	3943	10	<0.01
p_hat300-3_A_50	1658*	**1658**	1658	10	0.14	**1658**	1658	10	<0.01
brock200_2_B_50	984*	**984**	984	10	0.71	**984**	984	10	<0.01
brock400_2_B_50	780	**780**	779.2	8	9.88	**780**	780	10	0.07
C125.9_B_50	552*	**552**	552	10	0.02	**552**	552	10	<0.01
C250.9_B_50	586*	**586**	586	10	4.39	**586**	586	10	<0.01
gen200_p0.9_55_B_50	551*	**551**	551	10	1.38	**551**	551	10	<0.01
gen400_p0.9_75_B_50	635*	**635**	635	10	0.69	**635**	635	10	0.01
hamming8-4_B_50	1055*	**1055**	1055	10	0.86	**1055**	1055	10	<0.01
keller4_B_50	859*	**859**	859	10	0.41	**859**	859	10	<0.01
MANN_a27_B_50	576*	**576**	568.8	8	13.49	**576**	576	10	0.02
p_hat300-1_B_50	2787	**2787**	2787	10	0.78	**2787**	2787	10	<0.01
p_hat300-2_B_50	1943	**1943**	1943	10	0.47	**1943**	1943	10	<0.01
p_hat300-3_B_50	808*	**808**	808	10	0.20	**808**	808	10	<0.01
brock200_2_C_50	459	**459**	459	10	0.83	**459**	459	10	<0.01

Continued on next page

Continued

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
brock400_2_C_50	355	**355**	354.8	9	11.97	**355**	355	10	0.04
C125.9_C_50	252*	**252**	252	10	0.03	**252**	252	10	<0.01
C250.9_C_50	261*	**261**	261	10	3.32	**261**	261	10	<0.01
gen200_p0.9_55_C_50	251*	**251**	251	10	2.23	**251**	251	10	<0.01
gen400_p0.9_75_C_50	285*	**285**	285	10	1.79	**285**	285	10	<0.01
hamming8-4_C_50	505*	**505**	505	10	0.40	**505**	505	10	<0.01
keller4_C_50	409*	**409**	409	10	0.10	**409**	409	10	<0.01
MANN_a27_C_50	251*	**251**	248.8	8	9.99	**251**	251	10	0.02
p_hat300-1_C_50	1362	**1362**	1362	10	0.58	**1362**	1362	10	<0.01
p_hat300-2_C_50	943	**943**	943	10	0.21	**943**	943	10	<0.01
p_hat300-3_C_50	383*	**383**	383	10	0.13	**383**	383	10	<0.01
# of best		72				72			
# of best Mean			66				72		
Avg. time(s)					1.49				<0.01
p-value		1	2.77E-02		1.96E-13				

Table 10: Computational results of the heuristic algorithms for the instances with $p_r = 0.75$

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
SET1									
brock200_2_A_75	1885	**1885**	1885	10	0.47	**1885**	1885	10	<0.01
brock400_2_A_75	1728	**1728**	1728	10	4.74	**1728**	1728	10	0.04
C125.9_A_75	1023*	**1023**	1023	10	0.02	**1023**	1023	10	<0.01
C250.9_A_75	1236	**1236**	1236	10	1.74	**1236**	1236	10	<0.01
gen200_p0.9_55_A_75	1206	**1206**	1206	10	0.26	**1206**	1206	10	<0.01
gen400_p0.9_75_A_75	1490	**1490**	1490	10	0.57	**1490**	1490	10	<0.01
hamming8-4_A_75	1759	**1759**	1759	10	0.45	**1759**	1759	10	<0.01
keller4_A_75	1434	**1434**	1434	10	0.03	**1434**	1434	10	<0.01
MANN_a27_A_75	1323	**1323**	1323	10	0.86	**1323**	1323	10	<0.01
p_hat300-1_A_75	4164	**4164**	4164	10	0.20	**4164**	4164	10	<0.01
p_hat300-2_A_75	2990	**2990**	2990	10	1.01	**2990**	2990	10	<0.01
p_hat300-3_A_75	1564	**1564**	1564	10	0.45	**1564**	1564	10	<0.01
brock200_2_B_75	1641	**1641**	1641	10	0.48	**1641**	1641	10	<0.01
brock400_2_B_75	1386	**1386**	1386	10	2.40	**1386**	1386	10	0.03
C125.9_B_75	856*	**856**	856	10	0.02	**856**	856	10	<0.01
C250.9_B_75	1001	**1001**	1001	10	1.10	**1001**	1001	10	<0.01
gen200_p0.9_55_B_75	983	**983**	983	10	0.23	**983**	983	10	<0.01
gen400_p0.9_75_B_75	1120	**1120**	1120	10	0.49	**1120**	1120	10	<0.01
hamming8-4_B_75	1579	**1579**	1579	10	0.12	**1579**	1579	10	<0.01
keller4_B_75	1268	**1268**	1268	10	0.04	**1268**	1268	10	<0.01
MANN_a27_B_75	1021	**1021**	1021	10	0.58	**1021**	1021	10	<0.01
p_hat300-1_B_75	3886	**3886**	3886	10	0.42	**3886**	3886	10	<0.01
p_hat300-2_B_75	2782	**2782**	2782	10	0.30	**2782**	2782	10	<0.01
p_hat300-3_B_75	1299	**1299**	1299	10	0.55	**1299**	1299	10	<0.01
brock200_2_C_75	1321	**1321**	1321	10	0.31	**1321**	1321	10	<0.01
brock400_2_C_75	1033	**1033**	1033	10	0.87	**1033**	1033	10	<0.01

Continued on next page

Continued

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
C125.9_C_75	644*	**644**	644	10	0.23	**644**	644	10	<0.01
C250.9_C_75	734	**734**	734	10	1.06	**734**	734	10	<0.01
gen200_p0.9_55_C_75	727	**727**	727	10	0.05	**727**	727	10	<0.01
gen400_p0.9_75_C_75	772	**772**	772	10	0.95	**772**	772	10	0.01
hamming8-4_C_75	1378	**1378**	1378	10	0.14	**1378**	1378	10	<0.01
keller4_C_75	1109	**1109**	1109	10	0.03	**1109**	1109	10	<0.01
MANN_a27_C_75	651	**651**	651	10	2.04	**651**	651	10	<0.01
p_hat300-1_C_75	3480	**3480**	3480	10	0.51	**3480**	3480	10	<0.01
p_hat300-2_C_75	2473	**2473**	2473	10	0.11	**2473**	2473	10	<0.01
p_hat300-3_C_75	1004	**1004**	1004	10	0.84	**1004**	1004	10	<0.01
SET 2									
brock200_2_A_75	3326	**3326**	3300.2	4	7.54	**3326**	3326	10	0.07
brock400_2_A_75	2941	**2941**	2862.7	1	12.25	**2941**	2941	10	0.19
C125.9_A_75	1837*	**1837**	1837	10	0.29	**1837**	1837	10	<0.01
C250.9_A_75	2171	**2171**	2171	10	4.72	**2171**	2171	10	<0.01
gen200_p0.9_55_A_75	2096	**2096**	2096	10	0.14	**2096**	2096	10	<0.01
gen400_p0.9_75_A_75	2404	**2404**	2367.6	1	18.56	**2404**	2404	10	0.11
hamming8-4_A_75	3124	**3124**	3124	10	1.00	**3124**	3124	10	<0.01
keller4_A_75	2690	**2690**	2690	10	0.12	**2690**	2690	10	<0.01
MANN_a27_A_75	2208	**2208**	2110.1	1	0.45	**2208**	2208	10	0.02
p_hat300-1_A_75	7899	**7899**	7871.1	1	24.72	**7899**	7899	10	<0.01
p_hat300-2_A_75	5343	**5343**	5343	10	0.61	**5343**	5343	10	<0.01
p_hat300-3_A_75	2838	**2838**	2838	10	2.00	**2838**	2838	10	<0.01
brock200_2_B_75	1533	**1533**	1533	10	0.63	**1533**	1533	10	<0.01
brock400_2_B_75	1291	**1291**	1279.1	3	15.55	**1291**	1291	10	0.09
C125.9_B_75	837*	**837**	837	10	0.25	**837**	837	10	<0.01
C250.9_B_75	971	**971**	971	10	2.71	**971**	971	10	0.01
gen200_p0.9_55_B_75	946	**946**	946	10	0.08	**946**	946	10	<0.01
gen400_p0.9_75_B_75	1052	1052	1036	1	11.16	**1054**	1054	10	0.13
hamming8-4_B_75	1474	**1474**	1474	10	0.91	**1474**	1474	10	<0.01
keller4_B_75	1254	**1254**	1254	10	0.10	**1254**	1254	10	<0.01
MANN_a27_B_75	958	**958**	932.7	1	10.89	**958**	958	10	0.02
p_hat300-1_B_75	3818	**3818**	3818	10	1.88	**3818**	3818	10	<0.01
p_hat300-2_B_75	2543	**2543**	2543	10	0.95	**2543**	2543	10	<0.01
p_hat300-3_B_75	1288	**1288**	1288	10	1.83	**1288**	1288	10	<0.01
brock200_2_C_75	658	**658**	658	10	0.32	**658**	658	10	<0.01
brock400_2_C_75	500	**500**	500	10	3.22	**500**	500	10	0.01
C125.9_C_75	337*	**337**	337	10	0.13	**337**	337	10	<0.01
C250.9_C_75	372	**372**	372	10	4.84	**372**	372	10	0.02
gen200_p0.9_55_C_75	371	**371**	371	10	0.07	**371**	371	10	<0.01
gen400_p0.9_75_C_75	395	**395**	393.2	4	16.31	**395**	395	10	0.10
hamming8-4_C_75	652	**652**	652	10	0.45	**652**	652	10	0.01
keller4_C_75	558	**558**	558	10	0.12	**558**	558	10	<0.01
MANN_a27_C_75	335	**335**	334.1	4	10.50	**335**	335	10	<0.01
p_hat300-1_C_75	1793	**1793**	1793	10	1.38	**1793**	1793	10	<0.01
p_hat300-2_C_75	1159	**1159**	1159	10	0.36	**1159**	1159	10	<0.01
p_hat300-3_C_75	529	**529**	529	10	1.04	**529**	529	10	<0.01
# of best		71				72			
# of best Mean			62				72		

Continued on next page

Continued

Instance	Best known	ILS-VND				ALS			
		f_{best}	f_{avg}	success	time(s)	f_{best}	f_{avg}	success	time(s)
Avg. time(s)					2.54				0.01
p-value		3.17E-01	5.06E-03		1.66E-13				