A hybrid genetic algorithm for the Hamiltonian p-median problem

Pengfei He [a], Jin-Kao Hao [a,*], Qinghua Wu [b]

[a] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[b] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China*

Accepted to ***Networks, October 2023***

Abstract

The Hamiltonian p median problem consists of finding p (p is given) non-intersecting Hamiltonian cycles in a complete edge-weighted graph such that each cycle visits at least three vertices and each vertex belongs to exactly one cycle, while minimizing the total cost of p cycles. In this work, we present an effective and scalable hybrid genetic algorithm to solve this computationally challenging problem. The algorithm combines an edge-assembly crossover to generate promising offspring solutions from high-quality parents, and a multiple neighborhood local search to improve each offspring solution. To promote population diversity, the algorithm applies a mutation operator to the offspring solutions and a quality-and-distance update strategy to manage the population. We compare the method to the best reference algorithms in the literature based on three sets of 145 popular benchmark instances (with up to 318 vertices), and report improved best upper bounds for 8 instances. To evaluate the scalability of the method, we perform experiments on a new set of 70 large instances (with up to 1060 vertices). We examine the contributions of key components of the algorithm.

Keywords: p-median; Traveling salesman; Memetic search; Edge assembly crossover; Local search; Metaheuristic.

1 Introduction

The Hamiltonian p-median problem (HpMP) [3] is defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_0, v_1, \cdots, v_{n-1}\}$ is the vertex set and $\mathcal{E}$ is the edge

* Corresponding author.
Email addresses: `pengfeihe606@gmail.com` (Pengfei He), `jin-kao.hao@univ-angers.fr` (Jin-Kao Hao), `qinghuawu1005@gmail.com` (Qinghua Wu).

set. Let $\mathcal{C}$ be a non-negative cost matrix associated with $\mathcal{E}$. The HpMP is to find p (p is given) non-intersecting Hamiltonian cycles such that each cycle visits at least three vertices and each vertex appears on exactly one cycle with the objective of minimizing the total cost of the p cycles. A mathematical formulation of the problem is shown in Appendix A. The popular symmetric traveling salesman problem (TSP) is a particular case of HpMP when $p = 1$.

As a mixed routing location problem [16], the HpMP combines the p-median problem [20,25] and the TSP [1]. As such, the HpMP is a relevant model for a variety of practical problems related to school locations, depot locations, multi-depot vehicle routing, industrial process scheduling or leather cutting [7]. On the other hand, the HpMP is known to be $\mathcal{NP}$-hard for any $p \geq 1$ on Euclidean graphs [19] and is therefore computationally challenging.

Since the introduction of HpMP in 1990, a number of solution methods have been developed. Several formulations have been studied within the polyhedral approach [9,15,35]. Gollowitzer et al. [8] performed theoretical and computational comparisons of seven different formulations. Marzouk et al. [19] developed a branch-and-price (B&P) algorithm and presented results for three sets of 754 benchmark instances (21–318 vertices), including optimal solutions for 272 small and medium instances (with 21–127 vertices) and 10 optimal solutions for large instances (with 150–318 vertices). Independently, Erdoğan et al. [5] presented an effective branch-and-cut algorithm (HpMP2) and showed results for two sets of 110 instances with up to 100 vertices, including optimal solutions for all 55 small instances and 43 medium instances (with 58–100 vertices). In addition, Bektaş et al. [2] studied the related directed Hamiltonian p-median problem and proposed a dedicated branch-and-cut algorithm. According to the results in the literature, B&P [19] and HpMP2 [5] are the two state-of-the-art exact HpMP algorithms.

On the other hand, heuristics were investigated to obtain approximate solutions for large instances in acceptable runtimes. Glaab [6] studied some HpMP variants and presented fast heuristics and LP-relaxations to obtain upper and lower bounds. Üster and Kumar [31] studied a related balanced ring problem and presented a heuristic algorithm incorporating several GRASP-based randomized solution construction routines and an effective local search improvement procedure. Erdoğan et al. [5] introduced a heuristic algorithm that integrates a giant tour and a dynamic programming formulation as well as an iterated local search algorithm (ILS) using 2-exchange and 1-opt operators. Herrán et al. [14] proposed a general variable neighborhood search algorithm (PGVNS) for the HpMP. The algorithm consists of three neighborhoods based on classical moves for routing problems. Computational results on 145 benchmark instances showed that PGVNS outperformed other existing methods and is the state-of-the-art heuristic algorithm for the HpMP. However, large instances remain a challenge for all existing algorithms.

Our literature review shows that despite the relevance of HpMP in theory and practice, there are not many methods in the literature that effectively address the problem. This is in stark contrast to the related single-route TSP and multi-route vehicle routing problem (VRP), for which there are numerous solution methods that can handle large and even very large problem instances. On the other hand, population-based genetic algorithms are among the most powerful approaches for solving various routing and location problems. It is surprising that this approach has not yet been studied for solving the HpMP.

In this work, we conduct the first study on the application of the population-based hybrid search framework to the HpMP. In doing so, we take advantage of existing effective search operators and strategies for solving related TSP and VRPs to develop a highly effective heuristic algorithm for this challenging mixed routing location problem. The proposed population-based hybrid genetic search algorithm (HGA) incorporates an adapted popular edge assembly crossover, originally developed for TSP, and an effective local search procedure. The crossover generates promising offspring solutions by inheriting common edges from the parent solutions and assembling non-common edges, while the local search improves each offspring solution through an intensive neighborhood search. To further increase the search capacity of the algorithm, a mutation operator and an advanced population management are also incorporated, with the first operator introducing new edges into the descendant solutions and the second ensuring a high-quality and diverse population.

We evaluate the proposed algorithm on three sets of 145 benchmark instances (with up to 318 vertices) that are commonly tested in the literature, and compare the results with state-of-the-art algorithms. We also test the algorithm on a new set of 70 large instances (with 400 to 1060 vertices). In addition, we perform experiments to shed light on the role of key components of the algorithm. In particular, we show for the first time through experimental observations the relevance of the idea of edge assembly to the HpMP.

The rest of the paper is organized as follows. The proposed hybrid genetic algorithm is introduced in Section 2, including its search operators and detailed procedures. This is followed by a detailed computational comparison with the state-of-the-art methods in the literature in Section 3. Additional experiments are shown to analyze the main algorithmic ingredients and gain an understanding of their roles in Section 4 . We conclude with a summary of the main findings and future work in Section 5.

2 Hybrid genetic algorithm for HpMP

The proposed hybrid genetic algorithm (HGA) for the HpMP follows the general approach of memetic algorithms [21,26], which benefit from a synergistic combination of population-based search and neighborhood-based search. Indeed, this approach has been quite successful in solving several TSPs [11,24] and various routing problems [18,22,23,27,29,32,12,13]. We show in this paper that this approach is also very suitable for the HpMP.

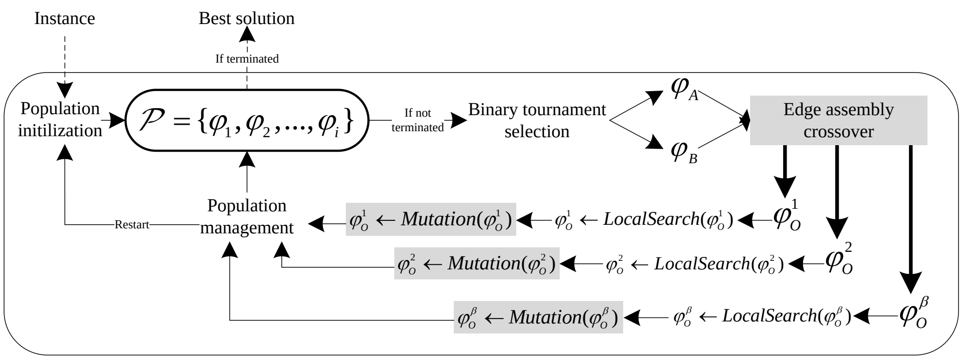


Fig. 1. Flow chart of the hybrid genetic algorithm

As illustrated in Fig. 1, the HGA algorithm starts with an initial population $\mathcal{P}$ in which each individual is constructed by a greedy heuristic (Section 2.1). The population is then evolved through multiple generations by applying three search operators, including crossover, local search, and mutation. For each generation, two parent solutions are selected and combined by the edge assembly crossover (EAX) [24] (Section 2.2), resulting in β offspring solutions (β is a parameter), that are first improved by local search (Section 2.3), and then diversified by the mutation (Section 2.4). Finally, each new solution is used to update the population based on a quality-and-distance strategy (Section 2.5). The algorithm terminates and returns the best solution φ^* if the predefined termination condition is satisfied (e.g., a maximum cutoff time or a maximum number of iterations).

Of particular interest is the edge-assembly crossover, which allows a descendant solution not only to inherit common edges (defined in Section 2.2) of the parents, but also to effectively assemble non-common edges. Since crossover can introduce relatively few edges that are not present in both parents, the mutation operator enhances the diversity of the descendant by introducing new edges. The quality-and-distance update strategy allows for desirable and continuous diversity of the population.

2.1 Population initialization

The population $\mathcal{P}$ is initialized as follows. An initial solution is constructed by a greedy heuristic and local search is then applied to improve the quality. If the solution is different from all other solutions in the population, it is inserted into $\mathcal{P}$. The quality and distance update strategy (2.5) is activated to keep μ solutions once the population reaches the maximum size $\mu + \lambda$. This process stops and returns the population when $4 \times \mu$ initial solutions are considered.

For each initial solution, the greedy heuristic operates according to the following steps. First, p vertices are randomly selected and each of them is used to initialize a cycle. To ensure that each cycle visits at least three vertices, we add two more vertices to the cycle in a greedy manner, chosen from the nearest neighbors (introduced in section 2.3) of the vertices in the cycle. Finally, the remaining vertices are added to arbitrary cycles in a greedy manner considering the nearest neighbor rule. Once all vertices are considered, a feasible initial solution is constructed. The time complexity is bounded by $\mathcal{O}(n \times \alpha)$, where α is a parameter of the nearest neighbor rule.

2.2 Edge assembly crossover

Before triggering the crossover to generate offspring solutions, the HGA selects two parent solutions φ_A and φ_B by a binary tournament strategy with respect to the objective value. In this work, we adopt the edge assembly crossover operator (EAX) to generate promising offspring solutions. EAX was originally introduced to solve the TSP [24] and has shown its effectiveness in vehicle routing problems [22,23]. The EAX operator has been further generalized to successfully solve the split delivery vehicle routing problem [12] and the min-max multiple traveling salesman problem [13]. Given that the HpMP includes routing as its subproblem, EAX is naturally suited to meet the requirements of the HpMP. However, since the HpMP is different from the TSP and routing problems, specific adaptations are needed, which concern the last step (Restore feasibility) of the crossover procedure as described below.

Given the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let φ_A and φ_B be two parent solutions. Let $\mathcal{G}_\mathcal{A} = (\mathcal{V}, \mathcal{E}_\mathcal{A})$ and $\mathcal{G}_\mathcal{B} = (\mathcal{V}, \mathcal{E}_\mathcal{B})$ be the corresponding partial graphs, where $\mathcal{E}_\mathcal{A}$ and $\mathcal{E}_\mathcal{B}$ are the sets of edges traversed by φ_A and φ_B, respectively. Note that the vertices in the corresponding partial graph of a solution have the same degree of two. EAX uses this property to naturally assemble the edges of the parents to produce offspring solutions. In what follows, an edge $e \in \mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}$ is qualified as a common edge of φ_A and φ_B if $e \in \mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}$, otherwise, it is a non-common edge.

Algorithm 1: The EAX procedure for the HpMP

Input: φ_A and φ_B parent solutions, β number of offspring to be created;
Output: β offspring solutions;

Step 1: Construct a joint graph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, (\mathcal{E}_\mathcal{A} \cup \mathcal{E}_\mathcal{B}) \backslash (\mathcal{E}_\mathcal{A} \cap \mathcal{E}_\mathcal{B}))$;
Step 2: Partition the joint graph $\mathcal{G}_{\mathcal{AB}}$ into *AB-cycles*.
Step 3: Generate β *E-sets* by combining *AB-cycles*.
Step 4: Construct β intermediate solutions according to *E-sets* and a basic solution.
Step 5: Reduce or add cycles in intermediate solutions if the number of cycles is not equal to p.

As shown in Algorithm 1, the EAX crossover generates β offspring solutions

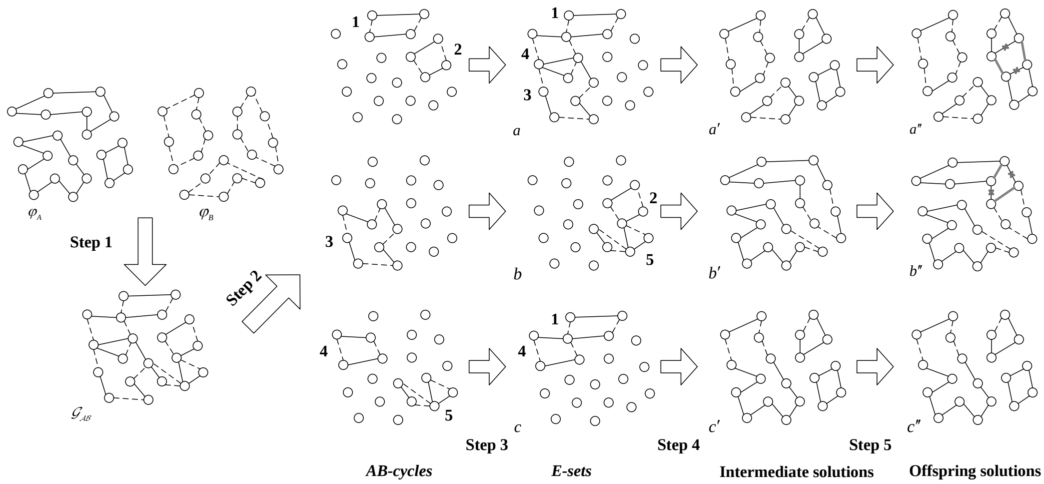


Fig. 2. Illustration of the EAX crossover for the HpMP

(β is a parameter) through the following steps.

(1) Construct a joint graph $\mathcal{G}_{\mathcal{AB}}$. From the partial graphs $\mathcal{G}_{\mathcal{A}} = (\mathcal{V}, \mathcal{E}_{\mathcal{A}})$ and $\mathcal{G}_{\mathcal{B}} = (\mathcal{V}, \mathcal{E}_{\mathcal{B}})$ associated to the parent solutions φ_A and φ_B, the joint graph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, (\mathcal{E}_{\mathcal{A}} \cup \mathcal{E}_{\mathcal{B}}) \backslash (\mathcal{E}_{\mathcal{A}} \cap \mathcal{E}_{\mathcal{B}}))$ is built. One notices that all edges of $\mathcal{G}_{\mathcal{AB}}$ are non-common edges.

(2) Partition the joint graph into *AB-cycles*. An *AB-cycle* is defined as a cycle in $\mathcal{G}_{\mathcal{AB}}$. A random vertex associated with edges from $\mathcal{G}_{\mathcal{AB}}$ is selected to initialize an *AB-cycle*, which is extended by adjacent edges taken alternatively from $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$. When an added adjacent edge leads to a cycle and the number of edges is even, an *AB-cycle* is constructed and its edges are removed from $\mathcal{G}_{\mathcal{AB}}$. When $\mathcal{G}_{\mathcal{AB}} = \emptyset$, all edges are partitioned into *AB-cycles*. Since for each vertex in $\mathcal{G}_{\mathcal{AB}}$ the number of incident edges of $\mathcal{E}_{\mathcal{A}}$ is equal to that of $\mathcal{E}_{\mathcal{B}}$, $\mathcal{G}_{\mathcal{AB}}$ can always be completely and evenly partitioned into *AB-cycles*.

(3) Generate *E-sets*. An *E-set* is an union of *AB-cycles*. *AB-cycles* that share common vertices are combined to form *E-sets*. Then if the number of *E-sets* is greater than parameter β, some *E-sets* are randomly combined to retain β *E-sets*.

(4) Construct intermediate solutions. Given a basic solution (say φ_A) and an *E-set* (say $\mathcal{E}_s$), an intermediate solution $\varphi' = (\mathcal{E}_{\mathcal{A}} \backslash (\mathcal{E}_s \cap \mathcal{E}_{\mathcal{A}})) \cup (\mathcal{E}_s \cap \mathcal{E}_{\mathcal{B}})$ is created. We thus get β intermediate solutions.

(5) Restore feasibility. Given an intermediate solution φ', let p' be the number of its Hamiltonian cycles. There are three cases of the value of p', that is $p' > p$, $p' = p$ and $p' < p$. Infeasible solutions concern the first and third cases. For the first case ($p' > p$), $p' - p$ cycles are eliminated by the 2-opt* operator used in [22]. The process starts by randomly selecting a cycle, denoted as c_1. Next, two vertices, u from c_1 and v from another cycle c_2 are selected such that vertex v is among the α nearest neighbors of vertex

u. Subsequently, edges (u, x) and (v, y) are removed and replaced with new edges (u, v) and (x, y), where x and y are the successors of u and v, respectively. This results in the combination of cycles c_1 and c_2, with the objective of minimizing the total distance. The best acceptance strategy is used for this purpose. The iterative process continues until $p' = p$. For the third case ($p' < p$), $p - p'$ cycles are added via the 2-opt*. Similar to the first case, a random cycle, say c_1, is selected, and two vertices, u and v, from the cycle are chosen such that vertex v is among the α nearest neighbors of vertex u. Then, edges (u, x) and (v, y) are removed, and new edges (u, v) and (x, y) are added, resulting in the splitting of cycle c_1 into two cycles. This iterative process continues until $p' = p$.

Given an *E-set*, half of the edges come from $\mathcal{E}_{\mathcal{A}}$ and the other half from $\mathcal{E}_{\mathcal{B}}$. Since an intermediate solution is constructed based on an *E-set* and a basic solution, say φ_A, if the size of *E-set* is large, more non-common edges from φ_B are inherited by the intermediate solution. Nagata and Kobayashi [24] demonstrated that increasing the size of the *E-set* can help the algorithm escape local optima. However, excessively large *E-sets* may produce offspring solutions of low quality, as intermediate solutions with a high number of subtours can deviate too far from the initial solution. On the contrary, if *E-sets* are too small, offspring solutions tend to be similar to the basic solution since relatively few non-common edges coming from the other parent solution are involved. In this work, we experimentally set $\beta = 5$ (see Section 4.2 for a sensitivity analysis of β).

Fig. 2 illustrates an example of the EAX procedure with $p = 3$. There are four and two cycles in intermediate solutions a' and b', respectively. For solution a', two cycles are connected to restore feasibility. However, a cycle is divided to ensure the feasibility of solution b'. During this process, few common edges may be broken to re-connect two cycles. For example, as shown in Fig. 2, two common edges in solution b'' are broken. Indeed, in the first four steps, all common edges are inherited by intermediate solutions, while the last step may break few common edges to restore feasibility. Thus, the EAX crossover generates offspring by inheriting nearly all common edges of the parents, assembling non-common edges of the parents and occasionally introducing few new short edges.

A HpMP solution contains n edges. The space complexity of EAX is $\mathcal{O}(n)$. In the first four steps, $2 \times n$ edges are assembled, and the time complexity is bounded by $\mathcal{O}(n)$. In the last step, suppose that there are m cycles in an intermediate solution and the cycle with the largest number of edges includes $|\mathcal{E}_m|$ edges. The time complexity of step 5 is bounded by $\mathcal{O}(|\mathcal{E}_m| \times \alpha)$ when reducing or adding one cycle, where α is the number of the nearest neighbors introduced in Section 2.3.

2.3 Local search

In the hybrid genetic algorithm framework, local search is the key component for search intensification and offspring improvement [10]. To attain high-quality solutions within a limited time, local search typically integrates enriched neighborhood operators and speed-up techniques. For the HpMP, HGA adopts seven neighborhood operators that are popular for routing problems and explores them under the framework of variable neighborhood descent.

Although Erdoğan et al. [5] and Herrán et al. [14] presented local search procedures, they don't use any neighborhood reduction technique, making their algorithms less effective for large instances. In this work, we adopt the so-called α nearest neighbors rule where α ($\leq n$) is a granularity threshold [30] to restrict the neighborhood search to nearby vertices. The nearest neighbors rule aims to speed up the neighborhood search and avoid the examination of non-promising candidate solutions. This is the first time the nearest neighbors rule is adopted in the context of HpMP.

We define the following notations to introduce our neighborhood operators. Let vertex v be the nearest neighbor of u. Let $c(u)$ and $c(v)$ be two cycles which visit vertices u and v, respectively, and x and y are the successors of u in $c(u)$ and v in $c(v)$, respectively. Let (u, x) be the substring from vertex u to x and (v, y) be the substring from vertex v to y. Seven basic neighborhood operators (or moves) are defined as follows.

(1) M1: Vertex u is removed from $c(u)$ and inserted into $c(v)$ after vertex v.
(2) M2: Two consecutive vertices u and x are removed from $c(u)$ and inserted into $c(v)$ after vertex v.
(3) M3: Two consecutive vertices u and x are removed from $c(u)$ and place $(x,\ u)$ after vertex v.
(4) M4: Interchange the position of vertex u and vertex v.
(5) M5: Interchange $(u,\ x)$ and vertex v.
(6) M6: Interchange $(u,\ x)$ and $(v,\ y)$.
(7) M7: This is the 2-opt operator, which replaces (u, x) and (v, y) by (u, v) and (x, y) if $c(u) = c(v)$.

Given the nearest neighbors rule, the time complexity of all operators is bounded $\mathcal{O}(n \times \alpha)$.

The seven operators are explored under the framework of variable neighborhood descent according to the order in which they are presented, as illustrated in Algorithm 2, where $M_\theta(\varphi)$ $(\theta = 1, 2, ..., \theta_{max})$ is the current neighborhood and θ_{max} =7.

We mention that the iterated local search (ILS) of Erdoğan et al. [5] explores only M1 and M2. The PGVNS of Herrán et al. [14] adopts two parametric

Algorithm 2: The variable neighborhood descent with θ_{max} neighborhoods for the HpMP

Input: Solution φ, θ_{max} neighborhoods;
Output: The local optimum solution φ;

```
begin
    θ ← 1;
    while θ ≤ θ_max do
        (φ, Improve) ← M_θ(φ);
        if Improve = true then
            θ ← 1;
        else
            θ ← θ + 1;
        end
    end
    return φ;
end
```

operators ins_λ and $swap_\lambda$, which covers M1–M6 by varying λ. However, none of the previous studies employ the α nearest neighbors rule to explore the neighborhoods. Our experiments demonstrated that the α nearest neighbors rule is a highly effective strategy to improve the search efficiency of the local search considerably. Finally, PGVNS additionally applies M7 to improve each individual cycle.

2.4 Mutation

Preserving a healthy population diversity is among the core issues of a hybrid genetic algorithm [10], whose purpose is to prevent the algorithm from premature convergence. In HGA, since nearly all edges in an offspring solution come from its parent solutions and the subsequent local search introduces few new edges, the population $\mathcal{P}$ may face a tricky problem, i.e., the edges of offspring solutions are almost fully covered by parents and new edges are rarely present in the population. To cope with this problem, the HGA algorithm applies, with a probability ζ, a mutation operator to each offspring solution to introduce new edges. This is a simple and effective way to diversify the offspring and enhance population diversity.

Given a solution φ, the mutation changes φ in $\xi \times n$ steps, where ξ is the mutation length. During each step, the mutation randomly applies the move M1 or the move M4 to perturb the solution. Suppose that M1 is applied, two vertices (denoted by u and v) are randomly picked from distinct cycles, and vertex u is inserted into $r(v)$ after vertex v. Similarly, if M4 is applied, two vertices are randomly selected from distinct cycles and their places are swapped. As we show in Section 4.3, the mutation helps the algorithm to maintain a healthy population diversity all along the search process and prevents the search from premature convergence.

2.5 Population management

Algorithm 3: The quality-and-distance updating strategy

Input: Population $\mathcal{P}$ with size of $\mu + \lambda$ where μ is the minimal population size and λ is the generation size;
Output: Updated population $\mathcal{P}$ with size of μ;

begin
    The traveling distance of all solutions is saved in the matrix dis;
    for $i = 1$ *to* $|\mathcal{P}|$ **do**
        for $j = 1$ *to* i **do**
            $d[i,j] \leftarrow HammingDis(\varphi_i,\ \varphi_j)$;
        end
    end
    for $i = 1$ *to* $|\mathcal{P}|$ **do**
        Sort $d(i)$; `/* From smallest to largest */`
    end
    while $|\mathcal{P}| > \mu$ **do**
        for $i = 1$ *to* $|\mathcal{P}|$ **do**
            $dClost[i] \leftarrow \sum_{j=1}^{nbClost} d[i,j]$;
        end
        Sort $dClost$; `/* From largest to smallest */`
        Sort dis; `/* From smallest to largest */`
        for $i = 1$ *to* $|\mathcal{P}|$ **do**
            $biasedFit[i] \leftarrow \frac{dis_r^i}{|\mathcal{P}|} + (1 - \frac{nbElite}{|\mathcal{P}|}) \times \frac{dClost_r^i}{|\mathcal{P}|}$;
        end
        $w \leftarrow max_{i \in \{1,2,\cdots,|\mathcal{P}|\}} biasedFit[i]$;
        $\mathcal{P} \leftarrow \mathcal{P} \backslash \{\varphi_w\}$;
        for $i = 1$ *to* $|\mathcal{P}|$ **do**
            Update $d(i)$ by removing φ_w;
        end
        Update dis by removing φ_w;
    end
    return $\mathcal{P}$;
end

The main goal of population management is to maintain a healthy diversity of $\mathcal{P}$ all along the search process. HGA uses a population updating strategy similar to the technique described in [32]. Each new offspring solution is inserted into the population if it is not the same as any solution of the population. Once the number of solutions reaches the maximum size $\mu + \lambda$ where λ is the generation size, λ solutions are removed with respect to a biased fitness, and μ individuals go to the next generation. Now, we explain how the biased fitness for each individual is computed. Let d be a two dimensional matrix and $d[i,j]$ denote the Hamming distance between solution φ_i and φ_j. Let $d(i)$ be the row of d that stores the Hamming distances between solution φ_i and each other solution in $\mathcal{P}$.

As shown in Algorithm 3, the Hamming distance between any pair of solutions equals the ratio between the number of non-common edges and n (lines 3 - 7). Then, given a solution φ_i, $|\mathcal{P}| - 1$ values of $d(i)$ are ranked from smallest to largest (lines 8 - 10), and the sum of the first $nbClost$ values ($nbClost$ is a parameter) are regarded as the diversity contribution of φ_i to $\mathcal{P}$, represented by $dClost[i]$ (lines 12 - 14). Then, the values of $dClost$ are arranged from largest to smallest and each solution φ_i is associated with a rank $dClost_r^i$ (line 15). Furthermore, we also rank solutions of $\mathcal{P}$ according to their objective values from the best to the worst, leading to a rank dis_r^i for each solution φ_i (line 16). Finally, the biased fitness of solution φ_i is defined as $biasedFit[i] = \frac{dis_r^i}{|\mathcal{P}|} + (1 - \frac{nbElite}{|\mathcal{P}|}) \times \frac{dClost_r^i}{|\mathcal{P}|}$ where $nbElite$ is a parameter and less than μ (lines 17 - 19). The solution associated with the largest biased fitness is removed from $\mathcal{P}$ and the biased fitness for each remaining solution of $\mathcal{P}$ is updated. The solution removal process is repeated until $|\mathcal{P}| = \mu$. Following [32], we set $nbClost = 5$ and $nbElite = 4$.

If the best solution found so far $\varphi*$ cannot be improved for γ consecutive iterations [1] (γ is a parameter called population rebuilding threshold), the algorithm restarts by generating a totally new population.

2.6 Discussions

As our literature review shows, the existing heuristic algorithms for the HpMP rely on single trajectory-based iterated local search [5] and variable neighborhood search [14], while ignoring the framework of population-based hybrid genetic search. Meanwhile, hybrid genetic search has been successfully applied to several related routing problems [10,22,33,34,12,13] and it is surprising to observe that this approach has never been studied in the context of the HpMP.

As the first algorithm of its kind, the proposed HGA algorithm fills this gap. In particular, we show that we are able to develop a competitive algorithm for the HpMP by leveraging the ideas of the successful EAX crossover originally developed for the TSP and the powerful neighborhood search for routing problems, as well as specific diversity preservation strategies. Indeed, extensive computational results show that HGA achieves remarkable results in terms of solution quality and runtime on various benchmark instances.

Given that the HpMP has a number of applications, the HGA algorithm can be used to better solve these practical problems. The code of the algorithm that we make publicly available will facilitate such applications.

[1] One iteration corresponds to one invocation of the local search procedure.

3 Experimental Evaluation and Comparisons

In this section, we experimentally evaluate the performance of the proposed algorithm and compare its results with the best existing algorithms.

3.1 Benchmark instances

Four sets of 215 HpMP instances are adopted for our experimental studies. The first three sets ($\mathbb{S}$, $\mathbb{M}$, $\mathbb{L}$) include 145 benchmark instances commonly tested in the literature while the last set ($\mathbb{N}$) includes 70 new large instances generated in this work. All of the instances are developed from graphs from the TSPLIB [2]. For sets $\mathbb{S}$, $\mathbb{M}$ and $\mathbb{L}$, given a TSPLIB graph, five instances are generated by using distinct values of $p \in \{\lfloor \frac{n}{10} \rfloor, \lfloor \frac{n}{7} \rfloor, \lfloor \frac{n}{5} \rfloor, \lfloor \frac{n}{4} \rfloor, \lfloor \frac{n}{3} \rfloor\}$. For set $\mathbb{N}$, seven instances per graph are obtained by setting $p \in \{\lfloor \frac{n}{30} \rfloor, \lfloor \frac{n}{20} \rfloor, \lfloor \frac{n}{10} \rfloor, \lfloor \frac{n}{7} \rfloor, \lfloor \frac{n}{5} \rfloor, \lfloor \frac{n}{4} \rfloor, \lfloor \frac{n}{3} \rfloor\}$.

- small set ($\mathbb{S}$): This set includes 55 instances from 11 TSPLIB graphs with 21 to 52 vertices.
- medium set ($\mathbb{M}$): This set includes 55 instances from 11 TSPLIB graphs with 58 to 100 vertices.
- large set ($\mathbb{L}$): The set includes 35 instances from 7 TSPLIB graphs with 150 to 318 vertices.
- new large set ($\mathbb{N}$): This new set includes 70 instances from 10 TSPLIB graphs (rd400, fl417, pcb442, d493, u574, rat575, p654, u724, rat783, u1060) with 400 to 1060 vertices.

It is worth mentioning that exact algorithms such as HpMP2 [5] and B&P [19] are able to obtain optimal solutions for all instances of set $\mathbb{S}$ (except two for B&P). Furthermore, most instances in set $\mathbb{M}$ are solved optimally by HpMP2 [5]. Thus, sets $\mathbb{S}$ and $\mathbb{M}$ are less challenging than sets $\mathbb{L}$ and $\mathbb{N}$ for the purpose of evaluating HpMP algorithms.

All these 215 instances are used in our experiments to extensively evaluate the performance of the proposed HGA algorithm. The instances and the best solutions obtained by HGA are available online [3].

3.2 Experimental protocol and reference algorithms

Parameter setting. The HGA algorithm has six parameters: the minimum population size μ, the generation size λ, the granularity threshold of nearest neighbors α, the mutation probability ζ, the mutation length ξ and the population rebuilding threshold γ. The automatic parameter tuning package Irace [17] is employed to calibrate these parameters. Given that HGA can ob-

[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
[3] https://github.com/pengfeihe-angers/HpMP.git

tain consistent results with different independent runs when solving small and medium instances, the instances used during tuning are selected from sets $\mathbb{L}$ and $\mathbb{N}$: pr299-42, lin318-31, rd400-80, d493-70, pcb442-44, d493-70, u574-82, p654-130, u724-72, rat783-195, u1060-151, where the values of p are selected randomly. Furthermore, the maximum number of experiments is 2000 and the stopping condition per experiment is 3600s or 300,000 iterations. The computer we used for parameter tuning is equipped with an Intel i7-6700HQ of 2.6GHz, where 7 cores are used. The candidate and final values are shown in Table 1. This setting can be considered as HGA's default setting and is consistently used for our experiments.

Table 1
Parameter tuning results.

Parameter	Section	Description	Considered values	Final values
μ	2.5	minimal size of population	$\{50, 100, 150, 200, 250\}$	100
λ	2.5	generation size	$\{25, 50, 75, 100, 125\}$	50
α	2.3	granularity threshold	$\{5, 8, 10, 12, 15, 20\}$	10
ζ	2.4	mutation probability	$\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$	0.15
ξ	2.4	mutation length	$\{0.05, 0.1, 0.15, 0.2, 0.25\}$	0.25
γ	2.5	population rebuilding threshold	$\{5000, 10000, 20000, 30000, 50000, 80000\}$	30000

Reference algorithms. We take the following best HpMP heuristic and exact algorithms, as well as the best known solutions BKS (best upper bounds), as the references for the comparative study.

- BKS. This indicates the best known solutions (upper bounds) that are summarized from all reference heuristic and exact approaches [5,19,14].
- HpMP2 [5]. The branch-and-cut algorithm was implemented in C++, running on a computer with an i7 2.5 GHz CPU. It solved optimally all small instances of set $\mathbb{S}$ and most medium instances of set $\mathbb{M}$ with a time limit of 3600s. No results were reported on set $\mathbb{L}$.
- B&P [19]. This branch-and-price algorithm was implemented in C++. In [14], the source code of B&P was used to solve the 215 instances of the sets $\mathbb{S}$, $\mathbb{M}$, and $\mathbb{L}$ on a computer with an Intel i7 6500U processor running at 2.5 GHz and 8 GB RAM. With a time limit of 3600s, B&P was able to obtain optimal solutions for all but two instances of $\mathbb{S}$ and more than half instances of set $\mathbb{M}$. The detailed results of B&P from [14] are used in our comparative study.
- PGVNS [14]. This algorithm was coded in C++ and experiments were conducted on a computer with an Intel i7 6500U processor running at 2.5 GHz and 8 GB RAM. The algorithm reported excellent results on the sets $\mathbb{S}$, $\mathbb{M}$, and $\mathbb{L}$. The source code was kindly provided by the authors. To make comparisons as fair as possible, we re-run the code on our computer and report its results under the heading 're-PGVNS'.

Given that B&P and HpMP2 are exact algorithms that aim to find optimal solutions, we consider the best heuristic algorithm PGVNS [14] as the most significant reference algorithm for our comparative study.

Experimental setting and stopping criterion. The HGA algorithm was coded in C++ and compiled using the g++ compiler with the -O3 option [4]. All experiments were run on an Intel Xeon E-2670 processor of 2.5 GHz and 2 GB RAM running Linux with a single thread. Both HGA and PGVNS were executed 20 times on each instance with distinct random seeds. The HGA algorithm terminates when it reaches a maximum of 500,000 iterations or the optimal solution. For PGVNS, we used its default parameter setting given in [14] with the stopping condition of a maximum of $0.3 \times p \times n$ iterations or a maximum of 3600s cutoff time.

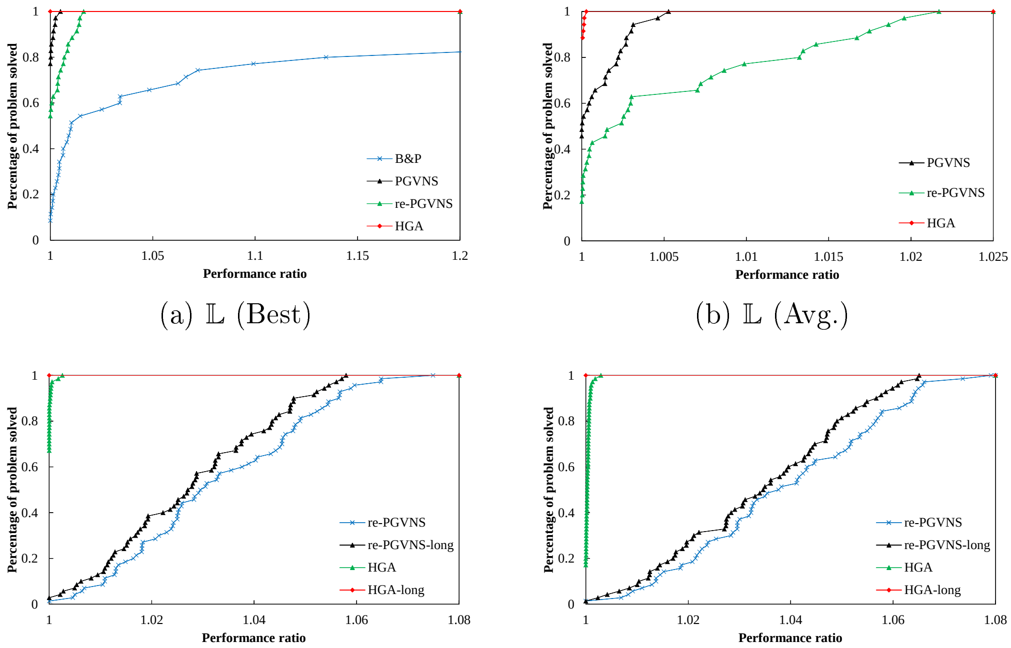
3.3 Computational results and comparisons

We report comparisons of the HGA algorithm with the reference algorithms on the four sets of benchmark instances. Detailed computational results on each instance are presented in Appendix B (Tables B.1–B.4), while a comparison summary is shown Table 2. To reveal the statistically significant difference between each pair of compared algorithms, the Wilcoxon signed-rank test with confidence level of 0.05 is used. Furthermore, a commonly used benchmarking tool, performance profile [4], is employed to compare distinct algorithms in a visual way. Given a set of algorithms $\mathcal{S}$ and a set of instances $\mathcal{I}$, the performance ratio $r_{q,a}$ of algorithm a on instance q with respect to the best approach for the minimization objective f is given by $r_{q,a} = \frac{f_{q,a}}{min\{f_{q,a:a\in\mathcal{S}}\}}$. The overall performance of approach a is determined by $Q_a(\tau) = \frac{|q\in\mathcal{I}:r_{q,a}\leq\tau|}{|\mathcal{I}|}$, which is the probability for algorithm a that its performance ratio $r_{q,a}$ is within a factor τ. $Q_a(\tau)$ represents the (cumulative) distribution function for the performance ratio. $Q_a(\tau = 1)$ is the percentage of instances on which algorithm a performs the best compared to all other algorithms.

Table 2
Summary of results between the HGA and reference algorithms on four sets of 215 instances.

Instances	Pair algorithms	Best				Avg.			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
$\mathbb{S}$	HGA vs. HpMP2 [5]	0	55	0	0.00E+00	-	-	-	-
	HGA vs. B&P [19]	0	55	0	0.00E+00	-	-	-	-
	HGA vs. PGVNS [14]	0	55	0	0.00E+00	0	55	0	0.00E+00
$\mathbb{M}$	HGA vs. HpMP2 [5]	5	50	0	6.25E-02	-	-	-	-
	HGA vs. B&P [19]	23	32	0	2.70E-05	-	-	-	-
	HGA vs. PGVNS [14]	0	55	0	0.00E+00	-	-	-	-
	HGA vs. re-PGVNS	0	55	0	0.00E+00	6	48	1	2.64E-04
$\mathbb{L}$	HGA vs. B&P [19]	28	3	0	-	-	-	-	-
	HGA vs. PGVNS [14]	8	27	0	7.81E-03	19	12	4	6.31E-04
	HGA vs. re-PGVNS	16	19	0	4.38E-04	29	5	1	1.47E-06
$\mathbb{N}$	HGA vs. re-PGVNS	70	0	0	3.56E-13	70	0	0	3.56E-13
	HGA vs. re-PGVNS-long	68	2	0	7.64E-13	69	1	0	5.21E-13
	HGA vs. HGA-long	0	53	17	2.93E-04	0	13	57	3.51E-11

[4] The code of the HGA algorithm will be available at: https://github.com/pengfeihe-angers/HpMP.git



(a) $\mathbb{L}$ (Best) (b) $\mathbb{L}$ (Avg.)

(c) $\mathbb{N}$ (Best) (d) $\mathbb{N}$ (Avg.)

Fig. 3. Performance profiles of the compared algorithms on $\mathbb{L}$ and $\mathbb{N}$ sets

According to the summarized results of Table 2 and detailed results of Tables B.1–B.4, we make the following observations.

- Sets $\mathbb{S}$ and $\mathbb{M}$. For the small instances, the two heuristic algorithms HGA and PGVNS perform identically and are able to attain the optimal solutions proven by the exact algorithms HpMP2 and B&P generally in less than one second. Both HGA and PGVNS attain the optimal solutions proven by the exact algorithms. Between HGA and PGVNS, HGA has a better performance in terms of the average results and is significantly faster than PGVNS to report solutions of the same quality.

- Set $\mathbb{L}$. For the 35 large instances, our HGA algorithm updates 8 BKS (new upper bounds) (22.9%) and matches all BKS values for the remaining instances (see detailed results in Table B.3). The small p-values ($\ll 0.05$) demonstrate that our algorithm dominates all reference algorithms in terms of both solution quality and computation time. In particular, HGA is significantly better than PGVNS in terms of the best and average results. Moreover, HGA requires always roughly no more than one-third of the time required by PGVNS to find solutions of equal or better quality. This demonstrates a clear advantage over the exact algorithms HpMP2 and B&P and the best heuristic algorithm PGVNS for solving these large instances. The performance profiles shown in Fig. 3 further confirm the dominance of HGA.

- Set $\mathbb{N}$. For this new set of largest instances, it is only possible to compare

HGA against PGVNS. For this set of instances, in addition to the standard stopping condition (a maximum of 500,000 iterations), we also tested HGA and PGVNS under a relaxed condition, i.e., a maximum of 1,000,000 iterations for HGA and a maximum equivalent runtime of 10800s (3 hours) for PGVNS. The results of long runs are shown in Tables 2 and B.4 under the headings HGA-long and re-PGVNS-long. According to the reached results, HGA significantly outperforms PGVNS both under the standard and relaxed stopping conditions ($p \ll 0.05$). HGA holds 68 best solutions out of the 70 instances and 2 equal solutions compared to PGVNS. HGA also reports significantly better average results. The performance profiles shown in Fig. 3 also support these conclusions. Once again, HGA is much faster than its competitor to report better or equal results, as shown in Table B.4. It is also interesting to notice that HGA is able to improve its owe results when it is given a higher time budget. Indeed, HGA-long performs significantly better than HGA by obtaining 17 new upper bounds and equal results for the remaining instances. As shown in Fig. 3, HGA-long dominates all algorithms since $Q_a(\tau = 1)$ of HGA reaches 1 firstly, which indicates a high robustness.

To sum, exact algorithms HpMP2 [5] and B&P [19] are valuable for finding the optimal solutions for the small instances of sets $\mathbb{S}$ and some medium instances of set $\mathbb{M}$. For the large instances of $\mathbb{L}$ and $\mathbb{N}$, heuristic algorithms PGVNS and HGA are indispensable alternatives for finding high-quality approximate solutions, while they are also able to easily reach the proven optimal solutions for the instances of sets $\mathbb{S}$ and $\mathbb{M}$. Between HGA and PGVNS, HGA dominates PGVNS both in terms of the solution quality and computational efficiency. In the following, we show additional experiments to investigate the contributions of the key algorithmic components to the high performance of the HGA algorithm.

4 Additional experiments

We now present additional experiments to study the roles of the edge assembly crossover and the mutation. The experiments are based on the most challenging instances of sets $\mathbb{L}$ and $\mathbb{N}$.

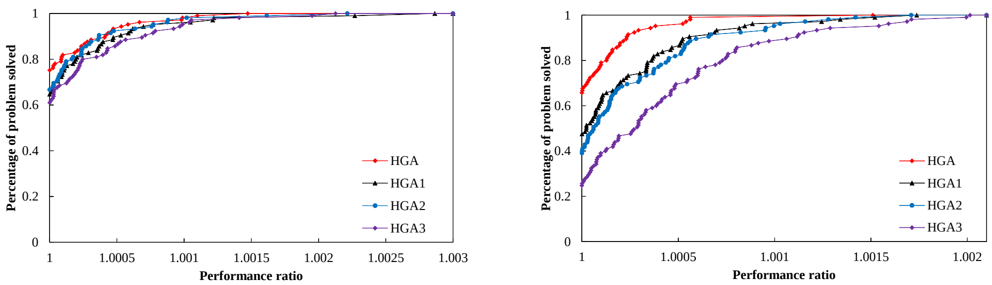
4.1 Significance of the crossover

The edge assembly crossover (EAX) produces offspring solutions by combining edges from parents and adding relatively few new short edges. Indeed, all common edges are inherited, while the size of *E-sets* determines how many non-common edges are involved in intermediate solutions. One notices that large *E-sets* may better promote diversity, but may result in low-quality offspring solutions due to the presence of too many cycles. Conversely, small

Table 3
Summary of comparative results between the HGA and five variants.

Pair algorithms	Best				Avg.			
	#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
HGA vs HGA1 ($\beta = 3$)	30	65	20	1.72E-01	52	39	24	2.79E-04
HGA vs HGA2 ($\beta = 10$)	26	73	16	2.09E-01	57	36	22	2.71E-06
HGA vs HGA3 ($\beta = 15$)	35	69	11	2.91E-03	76	34	5	7.69E-15
HGA vs HGA4 (Disable crossover)	105	10	0	5.84E-19	105	10	0	5.84E-19
HGA vs HGA5 (Disable mutation)	63	52	0	5.17E-12	87	27	1	4.00E-16

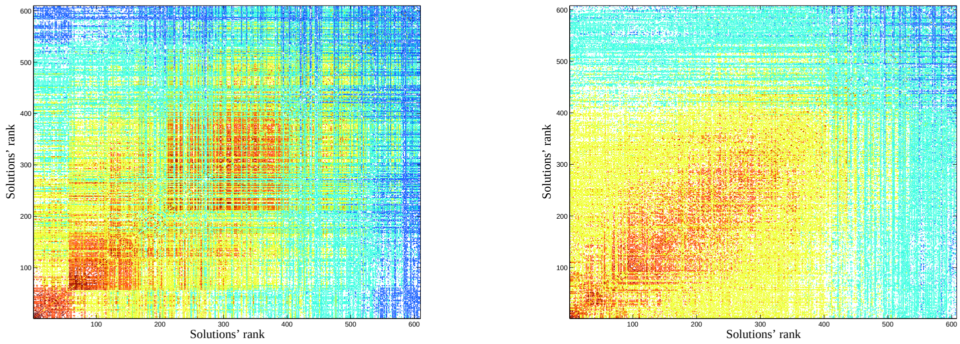
E-sets can produce offspring solutions that are very similar to their parents, potentially limiting diversity [24]. Thus, we need to know which size of *E-sets* is the best compromise for the quality and diversity. To gain insights into this issue, three HGA variants with distinct values of β, HGA1 ($\beta = 3$), HGA2 ($\beta = 10$), HGA3 ($\beta = 15$), are compared, along with the standard HGA with $\beta = 5$. An extra variant named HGA4 is also included where EAX is disabled. To ensure a fair comparison, the runtime budget of HGA provided by Tables B.3-B.4 was used to conduct the current experiment. We ran these algorithm variants on the same machine and report the comparative results in Table 3.



(a) Best (b) Avg.

Fig. 4. Performance profiles of the HGA and its variants.

The performance profiles, shown in Fig. 4, illustrate that the performance differences are more visible for the average results than for the best results. Still it is observed that HGA has a higher $Q_s(1)$, which reaches the value of 1 earlier than its variants. Indeed, the results summarized in Table 3 indicate that in terms of the best results, HGA is marginally better than HGA1 and HGA2, but significantly better than the other variants, while HGA significantly dominates all its variants in terms of the average results. It is worth observing that HGA1 (with a small $\beta = 3$) and HGA2-HGA3 (with large $\beta = 10, 15$) perform worse than HGA (with a moderate $\beta = 5$). This indicates that too large or too small β is harmful for HGA's performance. Finally, one observes that HGA4 (without the crossover) has the worst results, indicating that the EAX crossover is a key driving search operator of the HGA algorithm.



(a) pr299, $p = 29$ (b) lin318, $p = 31$

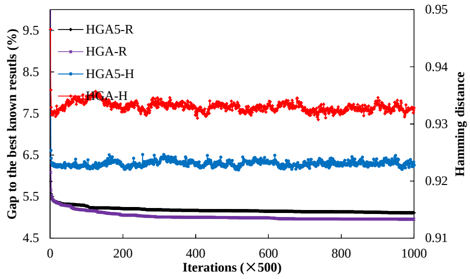
Fig. 5. Hamming distance between each pair of local optimal solutions. Brighter colors correspond to smaller Hamming distances, indicating pairs of similar or closely related solutions. The brightest colors indicate that more than 95% of the edges are shared by two solutions, while the darkest blue colors indicate that less than 70% of the edges are shared by two solutions.

4.2 Rationale behind the crossover

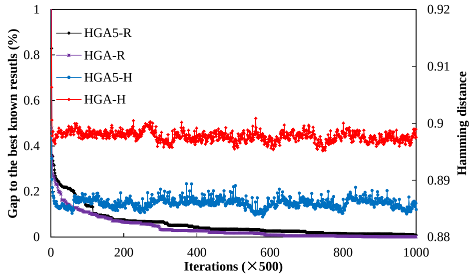
To shed insights on why the EAX crossover is a meaningful operator for the HpMP, we investigate the relationship between high-quality local optimal solutions in terms of the Hamming distance. Intuitively, if two high-quality local optimal solutions have a small distance, that means that they share many common edges. This is then a favorable feature for the EAX crossover, because EAX allows offspring solutions to inherit the common edges that form the backbone of a high-quality solution.

For this experiment, we use both HGA and PGVNS to sample various local optimal solutions, which are both of high-quality and diverse. Specifically, we adopt two representative instances (pr299, $p = 29$ and lin318, $p = 31$) with their best known results from Table B.3. We run HGA and PGVNS on these instances and record the local optimal solutions whose objective value is within 5% of the best known value. For each instance, we yield 600 distinct solutions. The Hamming distance between each pair of these solutions is calculated and the results are shown in Fig. 5 as two dimensional heat map. The abscissa and ordinate axes represent the rank of solutions from smallest to largest with respect to the objective value. The colored pixels represent the Hamming distance between each pair of solutions. Brighter colors correspond to small Hamming distances, indicating pairs of similar (or close) solutions. From Fig. 5, one notices that brighter colors center around the bottom left corner of both figures. This means that higher quality solutions share more common edges than less good solutions. Given that EAX transmits the common edges from parents to offspring, the backbone of high-quality solutions is systematically preserved. This also explains why the EAX crossover needs to use relatively large *E-sets* when recombining high-quality parents to preserve sufficient diversity in offspring solutions. It is worth noting that these findings are fully consistent with the conclusions of Nagata and Kobayashi [24] in the context of applying EAX to the TSP.

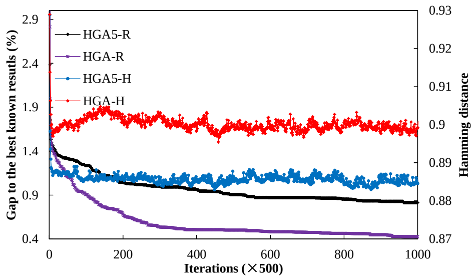
4.3 Benefits of the mutation



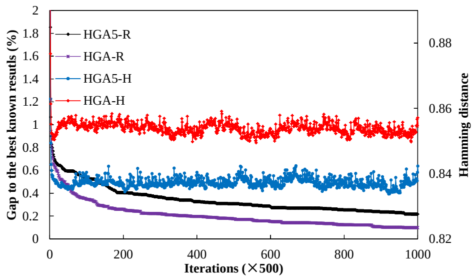
(a) lin318, p=31



(b) lin318, p=45

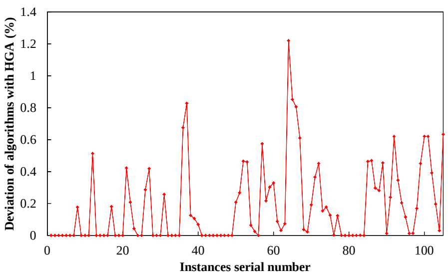


(c) pcb442, p=63

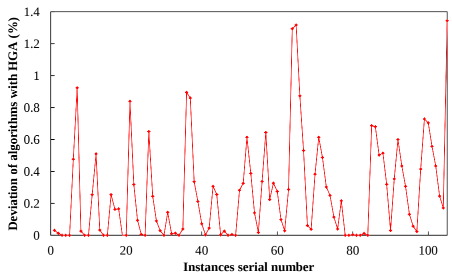


(d) pcb442, p=88

Fig. 6. Convergence charts of HGA and HGA5 for solving four representative instances



(a) Best



(b) Avg.

Fig. 7. The differences between HGA and HGA without mutation for solving sets $\mathbb{L}$ and $\mathbb{N}$.

HGA uses the mutation operator to diversify offspring solution and promote population diversity. To assess its usefulness, a new variant (HGA5) is constructed by disabling the mutation operator in HGA. HGA is then compared with HGA5 in terms of population diversity by using the following diversity measure [28]. Let $|\mathcal{P}|$ be the number of solutions in the population $\mathcal{P}$. Let h_{ij} be the Hamming distance between two solutions φ_i and φ_j. During each iteration, Equation (1) is used to measure the population diversity. We draw the convergence charts of HGA and HGA5 together with the population diversity, based on four instances (lin318, p = 31, lin318, p = 45, pcb442, p = 63, and pcb442, p=88). The results are visualized in Fig. 6, where HGA-R and HGA5-R indicate the best results found while HGA-H and HGA5-H are the average Hamming distance η of the population. HGA has a better convergence and dominates HGA5. HGA always keeps a higher value of η along its evolution compared to HGA5, which indicates that the mutation contributes to preserve diversity without sacrificing quality.

$$\eta = \frac{2}{|\mathcal{P}|(|\mathcal{P}| - 1)} \sum_{i=1}^{|\mathcal{P}|} \sum_{j=i+1}^{|\mathcal{P}|} h_{ij} \qquad (1)$$

Furthermore, Fig. 7 shows the comparative results of HGA and HGA5 in terms of both the best and average results on the 105 instances of sets $\mathbb{L}$ and $\mathbb{N}$. The results are presented as the percentage deviation of the results of HGA5 compared to the results of HGA. Together with the summarized results reported in Table 3, it is clear that the performance of HGA will degrade significantly if the mutation operator is disabled. These evidences confirm that the mutation operator plays a positive role in our algorithm.

5 Conclusions

In this paper, we presented a hybrid genetic algorithm (HGA) for the Hamiltonian p-median problem. The method includes a versatile edge assembly crossover allowing a diversified search and a neighborhood-based search ensuring aggressive solutions improvement. Furthermore, a diversification-oriented mutation operator and a quality-and-distance population updating strategy are integrated into the algorithm to manage the population.

Computational experiments on three sets of 145 commonly used benchmark instances show that the algorithm can effectively solve a wide range of instances within a short time by either improving or matching the optimal or best known results reported in the literature. In particular, HGA outperformed all reference algorithms and provides 8 new best upper bounds. We also assessed the algorithm on a new set of 70 large instances and compared with the best heuristic algorithm and provided the first upper bounds for these challenging instances. These bounds and the 8 new bounds for the conventional benchmark instances can be useful for future research on the HpMP. Additional experiments were conducted to get insights into the roles and rationale of the edge assembly crossover for the HpMP and the impacts of the mutation operator.

Given that the HpMP is a relevant model for a number of real-world problems, our algorithm whose code will be publicly available can be used to better solve some of these practical applications.

This work demonstrates that the hybrid genetic approach is highly effective for this computationally challenging problem, thank to a fruitful synergy between a meaningful crossover, a powerful local search and suitable diversity preserving strategies. Finally, we highlight that the general idea of assembling promising edges of high-quality solutions is much relevant for the HpMP and this idea can be advantageously adopted to deal with other routing problems.

Acknowledgments

We are grateful to the associate editor and two reviewers for their insightful and constructive comments, which helped us to significantly improve the paper. We also would like to thank the following colleagues: Prof. A. Herrán and Prof. A. Duarte for kindly sharing their source code of the PGVNS algorithm [14], Prof. G. Erdoğan and Prof. E. Moreno-Centeno for answering our questions about their HPMP2 algorithm [5] and B&P algorithm [19]. This work was partially supported by the National Natural Science Foundation Program of China (Grant No. 72122006). Support from the China Scholarship Council (CSC, No. 201906850087) for the first author is also acknowledged.

References

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2007.

[2] T. Bektaş, L. Gouveia, D. Santos, Revisiting the Hamiltonian p-median problem: a new formulation on directed graphs and a branch-and-cut algorithm, European Journal of Operational Research 276 (1) (2019) 40–64.

[3] I. Branco, J. Coelho, The Hamiltonian p-median problem, European Journal of Operational Research 47 (1) (1990) 86–95.

[4] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2) (2002) 201–213.

[5] G. Erdoğan, G. Laporte, A. M. R. Chía, Exact and heuristic algorithms for the Hamiltonian p-median problem, European Journal of Operational Research 253 (2) (2016) 280–289.

[6] H. Glaab, A new variant of a vehicle routing problem: Lower and upper bounds, European Journal of Operational Research 139 (3) (2002) 557–577.

[7] H. Glaab, A. Pott, The Hamiltonian p–median problem, The Electronic Journal of Combinatorics 7 (2000) R42–R42.

[8] S. Gollowitzer, L. Gouveia, G. Laporte, D. L. Pereira, A. Wojciechowski, A comparison of several models for the Hamiltonian p-median problem, Networks 63 (4) (2014) 350–363.

[9] S. Gollowitzer, D. L. Pereira, A. Wojciechowski, New models for and numerical tests of the Hamiltonian p-median problem, in: International Conference on Network Optimization, pp. 385–394, Springer, 2011.

[10] J.-K. Hao, Memetic algorithms in discrete optimization, in: Handbook of memetic algorithms, Springer, 2012, pp. 73–94.

[11] P. He, J. Hao, Q. Wu, Grouping memetic search for the colored traveling salesmen problem, Information Sciences 570 (2021) 689–707.

[12] P. He, J.-K. Hao, General edge assembly crossover-driven memetic search for split delivery vehicle routing, Transportation Science, https://doi.org/10.1287/trsc.2022.1180.

[13] P. He, J.-K. Hao, Memetic search for the minmax multiple traveling salesman problem with single and multiple depots, European Journal of Operational Research 307 (3) (2023) 1055–1070.

[14] A. Herrán, J. M. Colmenar, A. Duarte, A variable neighborhood search approach for the Hamiltonian p-median problem, Applied Soft Computing 80 (2019) 603–616.

[15] L. Hupp, F. Liers, A polyhedral study of the Hamiltonian p-median problem, Electronic Notes in Discrete Mathematics 41 (2013) 213–220.

[16] G. Laporte, Y. Nobert, P. Pelletier, Hamiltonian location problems, European Journal of Operational Research 12 (1) (1983) 82–89.

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

[18] Y. Lu, J. Hao, Q. Wu, Hybrid evolutionary search for the traveling repairman problem with profits, Information Sciences 502 (2019) 91–108.

[19] A. M. Marzouk, E. Moreno-Centeno, H. Üster, A branch-and-price algorithm for solving the Hamiltonian p-median problem, INFORMS Journal on Computing 28 (4) (2016) 674–686.

[20] N. Mladenović, J. Brimberg, P. Hansen, J. A. Moreno-Pérez, The p-median problem: A survey of metaheuristic approaches, European Journal of Operational Research 179 (3) (2007) 927–939.

[21] P. Moscato, Memetic algorithms: A short introduction, New Ideas in Optimization (1999) 219–234.

[22] Y. Nagata, O. Bräysy, Edge assembly-based memetic algorithm for the capacitated vehicle routing problem, Networks: An International Journal 54 (4) (2009) 205–215.

[23] Y. Nagata, O. Bräysy, W. Dullaert, A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows, Computers & Operations Research 37 (4) (2010) 724–737.

[24] Y. Nagata, S. Kobayashi, A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem, INFORMS Journal on Computing 25 (2) (2013) 346–363.

[25] S. C. Narula, U. I. Ogbu, H. M. Samuelsson, An algorithm for the p-median problem, Operations Research 25 (4) (1977) 709–713.

[26] F. Neri, C. Cotta, P. Moscato (eds.), Handbook of Memetic Algorithms, vol. 379 of Studies in Computational Intelligence, Springer, 2012.

[27] C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, Computers & Operations Research 31 (12) (2004) 1985–2002.

[28] J. Ren, J.-K. Hao, E. Rodriguez-Tello, A study of recombination operators for the cyclic bandwidth problem, in: Artificial Evolution: 14th International Conference on Artificial Evolution, Mulhouse, France, October 29–30, 2019, Lecture Notes in Computer Science 12052, pp. 177–191, Springer, 2020.

[29] J. Ren, J.-K. Hao, F. Wu, Z.-H. Fu, An effective hybrid search algorithm for the multiple traveling repairman problem with profits, European Journal of Operational Research 304 (2) (2023) 381–394.

[30] P. Toth, D. Vigo, The granular tabu search and its application to the vehicle-routing problem, INFORMS Journal on Computing 15 (4) (2003) 333–346.

[31] H. Üster, S. K. Kumar, Algorithms for the design of network topologies with balanced disjoint rings, Journal of Heuristics 16 (1) (2010) 37–63.

[32] T. Vidal, Hybrid genetic search for the CVRP: Open-source implementation and swap* neighborhood, Computers & Operations Research 140 (2022) 105643.

[33] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Operations Research 60 (3) (2012) 611–624.

[34] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, A unified solution framework for multi-attribute vehicle routing problems, European Journal of Operational Research 234 (3) (2014) 658–673.

[35] M. Zohrehbandian, A new formulation of the Hamiltonian p-median problem, Applied Mathematical Sciences 1 (8) (2007) 355–361.

Appendix

A Mathematical model

The HpMP can be formulated as a set partition problem with additional constraints [19,14] to ensure that a feasible solution contains p cycles and each cycle visits at least three vertices. Let Ω be the set of cycles, each cycle being given by a sequence of edges. The travel cost c_k of a cycle $k \in \Omega$ is given by the sum of the cost of the edges in its cycle. Let a_{ik} denote the number of times vertex i is visited by cycle k. Let x_k be a binary variable such that $x_k = 1$ if the cycle k is in the optimal solution, $x_k = 0$ otherwise. The set partition formulation of HpMP is as follows.

$$\text{minimize} \quad \sum_{k \in \Omega} c_k x_k \tag{A.1}$$

$$\text{subject to :} \quad \sum_{k \in \Omega} a_{ik} x_k = 1, \quad \forall i \in \mathcal{V} \tag{A.2}$$

$$\sum_{i \in \mathcal{V}} a_{ik} x_k \geq 3, \quad \forall k \in \Omega \tag{A.3}$$

$$\sum_{k \in \Omega} x_k = p \tag{A.4}$$

$$x_k \in \{0, 1\}, \quad \forall k \in \Omega \tag{A.5}$$

Objective function A.1 minimizes the overall of costs associated to each cycle. Constraints A.2 guarantee that each vertex is visited by exactly one cycle. Constraints A.3 state that each cycle needs to visit at least three vertices. Constraint A.4 guarantees that the number of cycles should equal p.

B Computational results

This section presents the detailed computational results of the proposed HGA algorithm together with the results of the reference algorithms: exact algorithms HPMP2 [5] and B&P [19] as well as heuristic algorithm PGVNS [14]. For HPMP2, its results are extracted from [5], while for B&P and PGVNS, their results are compiled from [14].

In the tables presented hereafter, column Instance indicates the name of each instance and corresponding value of p; column BKS is the optimal values (indicated by the '*' symbol) or best-known values (best upper bounds) summarized from the literature; Best and Avg. are the best and average results over 20 independent runs obtained by the corresponding algorithm in the column header, respectively; MRT(s) in each column represents the time of each corresponding exact algorithm to find the optimal solution or the total runtime if no optimal solution is found; Time(s) in each column means the average runtime in seconds of the corresponding algorithm. In Tables B.1-B.4, *Gap* in the last column is calculated as $Gap = 100 \times (f_{best} - BR)/BR$, where f_{best} is the best objective value of HGA and BR is the best results of all other algorithms including BKS. The *Average* row is the average value of a performance indicator over the instances of a benchmark set. Improved best results (new bounds) are indicated by negative *Gap* values highlighted in boldface. In Table B.4, the dark gray color indicates that the corresponding algorithm obtains the best result among the compared algorithms on the corresponding instance; the medium gray color displays the second best results, and so on.

Table B.1
Results for the HpMP on the instances of set $\mathbb{S}$. The timing information for the reference algorithms has the following meanings. For PGVNS, STMB(s) is the shortest run time to attain the best solution among 10 runs (extracted from Table 9 of [14]). The average time of PGVNS for set $\mathbb{S}$ is unavailable. For HGA, Time(s) is the average runtime over 20 runs.

Instance		BKS	HPMP2 [5]		B&P [19]		PGVNS [14]			HGA		
Name	p		Best	MRT(s)	Best	MRT(s)	Best	Avg.	STMB(s)	Best	Avg.	Time(s)
gr21	2	**2773.00***	**2773.00**	0.49	**2773.00**	251.00	**2773.00**	2773.00	0.01	**2773.00**	2773.00	0.09
	3	**2774.00***	**2774.00**	0.34	**2774.00**	41.00	**2774.00**	2774.00	0.03	**2774.00**	2774.00	0.01
	4	**2757.00***	**2757.00**	0.19	**2757.00**	8.00	**2757.00**	2757.00	0.03	**2757.00**	2757.00	0.01
	5	**2832.00***	**2832.00**	0.46	**2832.00**	35.00	**2832.00**	2832.00	0.03	**2832.00**	2832.00	0.01
	7	**3043.00***	**3043.00**	0.45	**3043.00**	16.00	**3043.00**	3043.00	0.02	**3043.00**	3043.00	0.01
ulysses22	2	**68.33***	**68.33**	0.39	**68.33**	3601.00	**68.33**	68.33	0.05	**68.33**	68.33	0.01
	3	**66.43***	**66.43**	0.38	**67.18**	3612.00	**66.43**	66.43	0.04	**66.43**	66.43	0.01
	4	**64.23***	**64.23**	0.19	**64.23**	3618.00	**64.23**	64.23	0.05	**64.23**	64.23	0.01
	5	**63.08***	**63.08**	0.16	**63.08**	7.00	**63.08**	63.08	0.03	**63.08**	63.08	0.01
	7	**65.08***	**65.08**	0.18	**65.08**	25.00	**65.08**	65.08	0.03	**65.08**	65.08	0.01
gr24	2	**1238.00***	**1238.00**	0.31	**1238.00**	32.00	**1238.00**	1238.00	0.03	**1238.00**	1238.00	0.01
	3	**1227.00***	**1227.00**	0.25	**1227.00**	3601.00	**1227.00**	1227.00	0.03	**1227.00**	1227.00	0.12
	4	**1227.00***	**1227.00**	0.27	**1227.00**	16.00	**1227.00**	1227.00	0.04	**1227.00**	1227.00	0.05
	6	**1266.00***	**1266.00**	0.51	**1266.00**	102.00	**1266.00**	1266.00	0.05	**1266.00**	1266.00	0.08
	8	**1317.00***	**1317.00**	0.24	**1317.00**	22.00	**1317.00**	1317.00	0.02	**1317.00**	1317.00	0.17
fri26	2	**911.00***	**911.00**	0.41	**911.00**	52.00	**911.00**	911.00	0.02	**911.00**	911.00	0.06
	3	**903.00***	**903.00**	0.31	**903.00**	38.00	**903.00**	903.00	0.03	**903.00**	903.00	0.06
	5	**893.00***	**893.00**	0.44	**893.00**	33.00	**893.00**	893.00	0.05	**893.00**	893.00	0.02
	6	**886.00***	**886.00**	0.37	**886.00**	12.00	**886.00**	886.00	0.07	**886.00**	886.00	0.02
	8	**885.00***	**885.00**	0.21	**885.00**	10.00	**885.00**	885.00	0.05	**885.00**	885.00	0.02
bayg29	2	**1562.00***	**1562.00**	0.56	**1562.00**	291.00	**1562.00**	1562.00	0.02	**1562.00**	1562.00	0.02
	4	**1549.00***	**1549.00**	0.50	**1549.00**	29.00	**1549.00**	1549.00	0.08	**1549.00**	1549.00	0.07
	5	**1555.00***	**1555.00**	0.53	**1555.00**	17.00	**1555.00**	1555.00	0.07	**1555.00**	1555.00	0.07
	7	**1618.00***	**1618.00**	2.15	**1618.00**	75.00	**1618.00**	1618.00	0.11	**1618.00**	1618.00	0.03
	9	**1676.00***	**1676.00**	1.73	**1676.00**	52.00	**1676.00**	1676.00	0.06	**1676.00**	1676.00	0.11
swiss42	4	**1232.00***	**1232.00**	1.37	**1232.00**	1195.00	**1232.00**	1232.00	0.17	**1232.00**	1232.00	0.27
	6	**1231.00***	**1231.00**	1.70	**1231.00**	693.00	**1231.00**	1231.00	0.27	**1231.00**	1231.00	0.56
	8	**1231.00***	**1231.00**	1.56	**1231.00**	110.00	**1231.00**	1231.00	0.37	**1231.00**	1231.00	0.20
	10	**1238.00***	**1238.00**	2.02	**1238.00**	20.00	**1238.00**	1238.00	0.36	**1238.00**	1238.00	0.16
	14	**1292.00***	**1292.00**	1.12	**1292.00**	69.00	**1292.00**	1292.00	0.16	**1292.00**	1292.00	0.10
att48	4	**31903.30***	**31903.30**	3.73	**31903.30**	510.00	**31903.30**	31903.30	0.41	**31903.30**	31903.30	0.16
	6	**31836.12***	**31836.12**	3.41	**31836.12**	73.00	**31836.12**	31836.12	0.66	**31836.12**	31836.12	0.09
	9	**32195.53***	**32195.53**	3.99	**32195.53**	117.00	**32195.53**	32195.53	0.74	**32195.53**	32195.53	0.18
	12	**32742.91***	**32742.91**	3.99	**32742.91**	64.00	**32742.91**	32742.91	0.68	**32742.91**	32742.91	0.20
	16	**37068.82***	**37068.82**	285.90	38113.80	3632.00	**37068.82**	37068.82	0.27	**37068.82**	37068.82	0.17
gr48	4	**4841.00***	**4841.00**	2.82	**4961.00**	3613.00	**4841.00**	4841.00	0.35	**4841.00**	4841.00	0.20
	6	**4805.00***	**4805.00**	1.76	**4805.00**	284.00	**4805.00**	4805.00	0.54	**4805.00**	4805.00	0.24
	9	**4926.00***	**4926.00**	13.70	**4926.00**	816.00	**4926.00**	4926.00	0.63	**4926.00**	4926.00	0.54
	12	**5011.00***	**5011.00**	4.91	**5011.00**	69.00	**5011.00**	5011.00	0.63	**5011.00**	5011.00	0.18
	16	**5445.00***	**5445.00**	24.25	**5445.00**	914.00	**5445.00**	5445.00	0.27	**5445.00**	5445.00	0.19
hk48	4	**11271.00***	**11271.00**	3.48	**11271.00**	1388.00	**11271.00**	11271.00	0.34	**11271.00**	11271.00	0.30
	6	**11197.00***	**11197.00**	2.88	**11197.00**	37.00	**11197.00**	11197.00	0.55	**11197.00**	11197.00	0.65
	9	**11292.00***	**11292.00**	3.05	**11292.00**	218.00	**11292.00**	11292.00	0.69	**11292.00**	11292.00	0.23
	12	**11450.00***	**11450.00**	3.41	**11450.00**	242.00	**11450.00**	11450.00	0.63	**11450.00**	11450.00	0.21
	16	**12215.00***	**12215.00**	10.04	**12215.00**	236.00	**12215.00**	12215.00	0.27	**12215.00**	12215.00	0.15
eil51	5	**422.32***	**422.32**	4.58	**422.32**	921.00	**422.32**	422.32	0.50	**422.32**	422.32	1.13
	7	**424.36***	**424.36**	6.88	**424.36**	401.00	**424.36**	424.36	0.68	**424.36**	424.36	1.14
	10	**432.49***	**432.49**	41.32	**432.49**	1771.00	**432.49**	432.49	0.81	**432.49**	432.49	0.55
	12	**436.59***	**436.59**	14.41	**436.59**	189.00	**436.59**	436.59	0.79	**436.59**	436.59	0.98
	17	**473.98***	**473.98**	50.96	**473.98**	1136.00	**473.98**	473.98	0.34	**473.98**	473.98	0.85
berlin52	5	**7182.23***	**7182.23**	3.66	7194.76	3662.00	**7182.23**	7182.23	0.53	**7182.23**	7182.23	0.64
	7	**7167.20***	**7167.20**	2.57	**7167.20**	49.00	**7167.20**	7167.20	0.82	**7167.20**	7167.20	2.59
	10	**7206.70***	**7206.70**	4.43	**7206.70**	159.00	**7206.70**	7206.70	1.00	**7206.70**	7206.70	0.25
	13	**7298.63***	**7298.63**	4.68	**7298.63**	169.00	**7298.63**	7298.63	0.90	**7298.63**	7298.63	0.20
	17	**7800.77***	**7800.77**	48.81	**7800.77**	1352.00	**7800.77**	7800.77	0.45	**7800.77**	7800.77	0.88
Average	-	5936.15	5935.15	10.43	5957.57	722.00	5936.15	5936.15	-	5936.15	5936.15	-
p-value	-	0.00E+00	0.00E+00	-	1.25E-01	-	0.00E+00	-	-	-	-	-

Table B.2. Results for the HpMP on the instances of set $\mathbb{M}$. The timing information for the reference algorithms has the following meanings. For PGVNS, STMB(s) is the shortest run time to attain its best solution among 10 runs (extracted from Table 10 of [14]). The average time of PGVNS for set $\mathbb{M}$ is unavailable. For re-PGVNS and HGA, Time(s) is the average time over 20 runs.

Instance		BKS	HPMP2 [5]		B&P [19]		PGVNS [14]		re-PGVNS			HGA			
Name	*p*		Best	MRT(s)	Best	MRT(s)	Best	STMB(s)	Best	Avg.	Time(s)	Best	Avg.	Time(s)	*Gap(%)*
brazil58	5	**21744.00***	**21744.00**	78.90	22502.00	3657.00	**21744.00**	1.04	**21744.00**	21744.00	22.92	**21744.00**	21744.00	0.35	0.00
brazil58	8	**21289.00***	**21289.00**	36.95	**21289.00**	2547.00	**21289.00**	1.81	**21289.00**	21289.00	39.30	**21289.00**	21289.00	0.32	0.00
brazil58	11	**21080.00***	**21080.00**	5.14	**21080.00**	3601.00	**21080.00**	1.86	**21080.00**	21080.00	38.33	**21080.00**	21080.00	0.26	0.00
brazil58	14	**21221.00***	**21221.00**	4.72	**21221.00**	54.00	**21221.00**	1.65	**21221.00**	21221.00	32.14	**21221.00**	21221.00	0.22	0.00
brazil58	19	**22635.00***	**22635.00**	31.13	**22635.00**	637.00	**22635.00**	0.78	**22635.00**	22635.00	14.72	**22635.00**	22635.00	0.15	0.00
st70	7	**638.22***	**638.22**	18.11	**638.22**	2678.00	**638.22**	2.83	**638.22**	638.22	65.43	**638.22**	638.22	0.35	0.00
st70	10	**632.54***	**632.54**	12.56	**632.54**	2334.00	**632.54**	3.57	**632.54**	632.54	79.19	**632.54**	632.54	9.45	0.00
st70	14	**630.90***	**630.90**	8.66	**630.90**	88.00	**630.90**	3.43	**630.90**	630.90	73.48	**630.90**	630.90	11.60	0.00
st70	17	**636.19***	**636.19**	11.16	**636.19**	148.00	**636.19**	3.38	**636.19**	636.19	69.98	**636.19**	636.19	1.72	0.00
st70	23	**694.49***	**694.49**	1137.77	713.36	3616.00	**694.49**	1.66	**694.49**	694.49	32.15	**694.49**	694.49	0.33	0.00
eil76	7	**542.95***	**542.95**	20.97	**542.95**	1060.00	**542.95**	4.21	**542.95**	543.11	99.45	**542.95**	542.95	0.53	0.00
eil76	10	**545.02***	**545.02**	18.60	**545.02**	580.00	**545.02**	5.34	**545.02**	545.89	122.13	**545.02**	545.02	0.84	0.00
eil76	15	**552.15***	**552.15**	207.04	**552.15**	629.00	**552.15**	5.42	**552.15**	552.15	117.51	**552.15**	552.15	7.60	0.00
eil76	19	**563.95***	**563.95**	371.35	**563.95**	1429.00	**563.95**	5.39	**563.95**	563.95	99.93	**563.95**	563.95	0.57	0.00
eil76	25	**601.71***	**601.71**	1025.73	**601.71**	3610.00	**601.71**	2.57	**601.71**	601.71	45.75	**601.71**	601.71	8.96	0.00
pr76	7	**101401.33***	**101401.33**	25.29	101644.57	3602.00	**101401.33**	2.83	**101401.33**	101401.33	62.84	**101401.33**	101401.33	0.84	0.00
pr76	10	**101779.42***	**101779.42**	224.40	**101779.42**	3625.00	**101779.42**	5.56	**101779.42**	101779.42	98.15	**101779.42**	101779.42	6.52	0.00
pr76	15	**103663.31**	103822.35	3608.81	103718.47	3607.00	**103663.31**	7.25	**103663.31**	103663.31	123.44	**103663.31**	103663.31	1.56	0.00
pr76	19	**104481.75***	**104481.75**	45.62	**104481.75**	523.00	**104481.75**	5.85	**104481.75**	104481.75	106.71	**104481.75**	104481.75	0.27	0.00
pr76	25	**110073.94***	**110073.94**	867.49	**110073.94**	2553.00	**110073.94**	2.64	**110073.94**	110073.94	46.31	**110073.94**	110073.94	1.33	0.00
rat99	9	**1209.09***	1209.14	90.16	**1209.09**	3613.00	**1209.09**	18.95	**1209.09**	1209.21	405.08	**1209.09**	1209.09	28.17	0.00
rat99	14	**1224.10***	1249.35	3622.70	**1224.10**	2865.00	**1224.10**	20.73	**1224.10**	1224.22	461.26	**1224.10**	1224.10	69.39	0.00
rat99	19	**1245.16***	1264.52	3618.81	**1245.16**	472.00	**1245.16**	20.37	**1245.16**	1245.16	425.38	**1245.16**	1245.16	17.84	0.00
rat99	24	**1273.23***	1276.13	3621.86	**1273.23**	1029.00	**1273.23**	18.21	**1273.23**	1273.23	353.42	**1273.23**	1273.23	5.98	0.00
rat99	33	**1373.37**	**1373.37**	3609.14	1378.93	3612.00	**1373.37**	6.61	**1373.37**	1373.37	125.91	**1373.37**	1373.37	295.09	0.00
kroA100	10	**19900.87***	**19900.87**	2993.41	21785.20	3616.00	**19900.87**	22.81	**19900.87**	19900.87	509.91	**19900.87**	19900.87	0.75	0.00
kroA100	14	**19637.52***	**19637.52**	40.47	19743.30	3608.00	**19637.52**	27.52	**19637.52**	19637.52	569.37	**19637.52**	19637.52	0.73	0.00
kroA100	20	**19868.64**	**19868.64**	57.24	**19868.64**	664.00	**19868.64**	24.09	**19868.64**	19868.64	481.94	**19868.64**	19868.64	4.88	0.00
kroA100	25	**20279.51***	**20279.51**	77.87	**20279.51**	533.00	**20279.51**	19.23	**20279.51**	20279.51	398.47	**20279.51**	20279.51	1.13	0.00
kroA100	33	**22303.23**	**22303.23**	3609.77	23230.10	3607.00	**22303.23**	9.00	**22303.23**	22303.23	176.39	**22303.23**	22303.23	250.90	0.00
kroB100	10	**20823.12***	**20823.12**	1575.86	20865.20	3618.00	**20823.12**	18.39	**20823.12**	20823.12	434.13	**20823.12**	20823.12	11.26	0.00
kroB100	14	**20762.88***	**20762.88**	1292.72	20801.70	3614.00	**20762.88**	22.76	**20762.88**	20762.88	504.08	**20762.88**	20762.88	13.18	0.00
kroB100	20	**20660.05***	**20660.05**	114.70	**20660.05**	2624.00	**20660.05**	22.61	**20660.05**	20660.05	494.14	**20660.05**	20663.10	156.91	0.00
kroB100	25	**20786.92***	**20786.92**	34.89	**20786.92**	604.00	**20786.92**	19.97	**20786.92**	20786.92	408.67	**20786.92**	20786.92	7.46	0.00
kroB100	33	**22923.42**	**22923.42**	3610.08	24254.70	3608.00	**22923.42**	8.73	**22923.42**	22923.42	169.97	**22923.42**	22923.42	279.33	0.00
kroC100	10	**19923.30***	**19923.30**	93.61	20158.20	3607.00	**19923.30**	15.19	**19923.30**	19923.30	355.38	**19923.30**	19923.30	4.88	0.00
kroC100	14	**19938.84***	**19938.84**	77.78	19963.80	3607.00	**19938.84**	20.59	**19938.84**	19938.84	473.53	**19938.84**	19938.84	8.52	0.00
kroC100	20	**20135.00***	**20135.00**	229.62	20148.90	3606.00	**20135.00**	22.34	**20135.00**	20135.00	488.53	**20135.00**	20135.00	11.98	0.00
kroC100	25	**20427.96***	**20427.96**	197.60	20430.40	3614.00	**20427.96**	19.08	**20427.96**	20427.96	396.96	**20427.96**	20427.96	0.81	0.00
kroC100	33	**22465.73**	**22465.73**	3609.81	23157.00	3632.00	**22465.73**	8.88	**22465.73**	22465.73	172.57	**22465.73**	22465.73	265.32	0.00
kroD100	10	**20270.57***	**20270.57**	50.50	20270.60	3643.00	**20270.57**	18.84	**20270.57**	20270.57	450.94	**20270.57**	20270.57	4.25	0.00
kroD100	14	**20267.23***	**20267.23**	46.87	**20267.23**	164.00	**20267.23**	24.12	**20267.23**	20267.23	558.66	**20267.23**	20267.23	0.96	0.00
kroD100	20	**20457.00***	**20457.00**	254.33	**20457.00**	2403.00	**20457.00**	22.59	**20457.00**	20457.00	501.43	**20457.00**	20457.00	120.77	0.00
kroD100	25	**20671.19***	**20671.19**	154.50	20671.20	271.00	**20671.19**	19.22	**20671.19**	20671.19	402.39	**20671.19**	20671.19	34.12	0.00
kroD100	33	**22238.56**	**22238.56**	3609.46	22401.90	3624.00	**22238.56**	8.81	**22238.56**	22238.56	171.73	**22238.56**	22238.56	255.53	0.00
kroE100	10	**20766.43***	**20766.43**	28.92	22461.70	3615.00	**20766.43**	19.17	**20766.43**	20766.43	457.03	**20766.43**	20766.43	1.24	0.00
kroE100	14	**20777.69***	**20777.69**	28.45	**20777.69**	387.00	**20777.69**	22.56	**20777.69**	20777.69	528.86	**20777.69**	20777.69	5.26	0.00
kroE100	20	**20937.39***	**20937.39**	51.43	**20937.39**	174.00	**20937.39**	22.16	**20937.39**	20937.39	493.63	**20937.39**	20937.39	3.88	0.00
kroE100	25	**21174.94***	**21174.94**	62.60	**21174.94**	857.00	**21174.94**	19.61	**21174.94**	21174.94	411.75	**21174.94**	21174.94	2.24	0.00
kroE100	33	**22782.98***	**22782.98**	3054.13	22900.00	3640.00	**22782.98**	9.00	**22782.98**	22782.98	175.61	**22782.98**	22782.98	1.59	0.00
rd100	10	**7524.08***	**7524.08**	177.19	7580.83	3621.00	**7524.08**	14.00	**7524.08**	7527.23	330.50	**7524.08**	7524.08	19.79	0.00
rd100	14	**7500.44***	**7500.44**	42.96	**7500.44**	479.00	**7500.44**	18.92	**7500.44**	7500.73	418.89	**7500.44**	7500.44	36.89	0.00
rd100	20	**7537.98***	**7537.98**	149.61	**7537.98**	1234.00	**7537.98**	24.25	**7537.98**	7537.98	500.35	**7537.98**	7537.98	7.96	0.00
rd100	25	**7555.83***	**7555.83**	51.30	**7555.83**	180.00	**7555.83**	19.71	**7555.83**	7555.83	394.34	**7555.83**	7555.83	10.12	0.00
rd100	33	**8131.25**	**8131.25**	3609.83	8202.99	3645.00	**8131.25**	8.68	**8131.25**	8131.25	172.55	**8131.25**	8131.25	273.23	0.00
Average	-	21839.32	21843.07	-	21993.53		21839.32	-	21839.32	21839.40	-	21839.32	21839.37	-	-
p-value	-	-	6.25E-02	-	2.70E-05	-	0.00E+00	-	0.00E+00	2.64E-04	-	-	-	-	-

Table B.3. Results for the HpMP on the instances of set $\mathbb{L}$. The timing information for the reference algorithms has the following meanings. For PGVNS, Time(s) is the average execution time of 10 independent runs (extracted from Table 11 of [14]). For re-PGVNS and HGA, Time(s) is the average time over 20 runs. The '-' symbol indicates that the result is unavailable.

Instance		BKS	B&P [19]		PGVNS [14]			re-PGVNS			HGA			
Name	p		Best	MRT(s)	Best	Avg.	Time(s)	Best	Avg.	Time(s)	Best	Avg.	Time(s)	*Gap(%)*
	15	**25217.42**	25374.25	3618.00	**25217.42**	25237.84	197.00	**25217.42**	25233.46	3598.01	**25217.42**	25217.42	495.57	0.00
	21	**25153.61**	25188.74	3635.00	**25153.61**	25153.61	214.00	**25153.61**	25159.21	3591.58	**25153.61**	25153.61	495.76	0.00
kroA150	30	**25333.1**	25444.28	3688.00	**25333.10**	25341.37	175.00	**25333.10**	25340.84	3596.10	**25333.10**	25333.10	490.38	0.00
	37	**25737.15**	25992.02	3741.00	**25737.15**	25737.15	145.00	**25737.15**	25737.98	3113.91	**25737.15**	25737.15	489.98	0.00
	50	**28540.82**	29510.71	3629.00	**28540.82**	28540.82	50.00	**28540.82**	28540.82	994.12	**28540.82**	28540.82	349.35	0.00
	15	**41238.46**	41610.26	3826.00	**41238.46**	41305.83	223.00	**41238.46**	41296.74	3595.83	**41238.46**	41238.46	550.48	0.00
	22	**41208.78**	**41208.78**	735.00	**41208.78**	41208.78	240.00	**41208.78**	41313.39	3596.76	**41208.78**	41208.78	536.73	0.00
u159	31	**41805.27**	41971.89	3632.00	**41805.27**	41856.86	217.00	**41805.27**	41876.56	3597.81	**41805.27**	41805.27	539.03	0.00
	39	**42362.95**	42373.42	3617.00	**42362.95**	42362.95	184.00	**42362.95**	42362.92	3589.69	**42362.95**	42362.95	467.63	0.00
	53	**47320.58**	50450.30	3708.00	**47320.58**	47320.58	63.00	**47320.58**	47320.58	1250.37	**47320.58**	47320.58	364.92	0.00
	20	27726.83	28654.95	3635.00	27726.83	27795.06	868.00	27813.84	27951.34	3604.23	**27711.71**	27711.71	696.84	**-0.05**
	28	**27429.12**	27712.71	3616.00	**27429.12**	27442.18	959.00	27449.31	27511.41	3602.96	**27429.12**	27431.11	672.54	0.00
kroA200	40	**27555.39**	**27555.39**	3952.00	**27555.39**	27555.39	772.00	**27555.39**	27597.44	3592.52	**27555.39**	27556.91	676.97	0.00
	50	**27943.7**	28068.38	3809.00	**27943.70**	27943.70	625.00	**27943.70**	27943.70	3599.45	**27943.70**	27943.70	631.96	0.00
	66	**30937.66**	32867.54	3645.00	**30937.66**	30937.66	311.00	**30937.66**	30937.66	3589.31	**30937.66**	30937.66	471.48	0.00
	20	**27924.13**	-	3604.00	**27924.13**	28006.89	882.00	28063.93	28132.87	3601.34	**27924.13**	27931.26	679.56	0.00
	28	**27771.8**	27946.80	3697.00	**27771.80**	27780.05	972.00	27810.91	27862.99	3600.26	**27771.80**	27771.80	670.53	0.00
kroB200	40	**27885.56**	27920.67	3791.00	**27885.56**	27885.56	820.00	**27885.56**	27963.78	3601.41	**27885.56**	27888.13	661.00	0.00
	50	**28247.44***	**28247.44**	3751.00	**28247.44**	28247.44	629.00	**28247.44**	28249.11	3590.16	**28247.44**	28247.44	21.88	0.00
	66	**30661.42**	32867.51	3645.00	**30661.42**	30661.42	324.00	**30661.42**	30661.42	3595.67	**30661.42**	30661.42	479.16	0.00
	26	**2260.32**	-	3602.00	**2260.32**	2267.93	3464.00	2292.86	2307.25	3612.93	**2260.32**	2261.70	923.13	0.00
	37	**2263.31**	2568.12	3602.00	**2263.31**	2268.81	3601.00	2283.03	2296.08	3613.31	**2263.31**	2265.26	932.65	0.00
gil262	52	**2279.55**	2285.15	3614.00	**2279.55**	2280.68	3498.00	2288.49	2298.57	3598.70	**2279.55**	2281.17	917.00	0.00
	65	**2312.86**	2346.55	3789.00	**2312.86**	2313.28	2553.00	2313.53	2318.64	3601.01	**2312.86**	2313.05	819.29	0.00
	87	**2530.86**	3071.54	3601.00	**2530.86**	2530.86	1119.00	**2530.86**	2531.07	3589.76	**2530.86**	2530.86	643.08	0.00
	29	45742.07	-	3604.00	45742.07	45811.87	3602.00	46166.32	46488.20	3622.67	**45679.92**	45706.43	995.51	**-0.14**
	42	45894.64	58641.29	3603.00	45894.64	46011.59	3601.00	46370.34	46635.14	3610.84	**45813.60**	45893.95	1010.22	**-0.18**
pr299	59	46204.85	47354.15	3609.00	46204.85	46288.08	3601.00	46505.54	46802.33	3606.55	**46191.46**	46195.29	978.38	**-0.03**
	74	46882.03	49135.93	3680.00	46882.03	46945.34	3600.00	47034.83	47207.57	3596.11	**46869.27**	46879.96	912.54	**-0.03**
	99	**51202.12**	56283.44	3608.00	**51202.12**	51202.12	2222.00	**51202.12**	51204.68	3589.72	**51202.12**	51202.12	667.09	0.00
	31	39898.69	-	3611.00	39898.69	39991.24	3602.00	40347.60	40644.61	3631.56	**39700.78**	39777.52	1167.56	**-0.50**
	45	39449.62	39676.24	3660.00	39449.62	39540.37	3602.00	39909.59	40092.21	3627.15	**39358.36**	39359.28	1111.89	**-0.23**
lin318	63	39361.28	39428.09	3841.00	39361.28	39407.06	3601.00	39629.30	39874.19	3605.25	**39300.56**	39317.39	1161.83	**-0.15**
	79	**39515.49**	39549.60	3734.00	**39515.49**	39539.40	3601.00	39767.90	39905.51	3604.62	**39515.49**	39515.49	934.84	0.00
	106	**45744.13**	94781.81	3605.00	**45744.13**	45744.13	2325.00	**45744.13**	45764.35	3591.02	**45744.13**	45748.16	735.67	0.00
Average	-	30844.09	-	-	30844.09	30870.40	-	30933.54	31010.42	-	30828.82	30835.63	-	-
p-value	-	7.81E-03	-	-	7.81E-03	1.12E-04	-	4.38E-04	7.78E-07		-	-	-	-

Table B.4. Results for the HpMP on the instances of set ℕ. Time(s) in each column is the average runtime over 20 runs.

Instance		re-PGVNS			re-PGVNS-long			HGA			HGA-long		
Name	p	Best	Avg.	Time(s)	Best	Avg.	Time(s)	Best	Avg.	Time(s)	Best	Avg.	Time(s)
	13	15377.70	15469.60	3653.81	15298.00	15429.50	10786.60	14884.31	14920.82	1562.73	**14883.41**	14910.56	3102.63
	20	15249.50	15361.80	3660.25	15245.40	15330.60	10781.30	**14766.68**	14808.40	1533.25	**14766.68**	14795.81	3106.35
	40	15058.80	15167.20	3670.63	15045.40	15149.70	10784.00	**14687.60**	14696.96	1510.27	**14687.60**	14694.13	3037.35
rd400	57	15051.40	15132.90	3646.84	15030.80	15099.80	10782.50	**14684.15**	14697.44	1543.40	**14684.15**	14694.28	3117.91
	80	15101.40	15148.50	3621.71	15012.10	15098.20	10245.30	**14835.51**	14841.96	1443.44	**14835.51**	14841.19	2928.11
	100	15227.30	15264.30	3607.03	15150.30	15197.80	10788.80	**15066.52**	15069.51	1367.49	**15066.52**	15069.47	2835.72
	133	16448.90	16528.30	3599.43	**16373.36**	16417.00	10776.10	**16373.36**	16379.19	1021.13	**16373.36**	16375.91	2169.29
	13	10213.90	10384.20	3678.91	10239.50	10378.80	10251.00	**9955.88**	9955.88	1376.56	**9955.88**	9955.88	2756.02
	20	10048.60	10155.60	3697.59	10067.40	10201.60	10782.90	**9615.69**	9615.69	1345.25	**9615.69**	9615.69	2669.20
	41	9477.67	9631.73	3662.66	9477.67	9625.27	10778.80	**9246.58**	9246.58	1429.64	**9246.58**	9246.58	2845.00
fl417	59	9392.39	9433.18	3650.86	9346.47	9421.36	10781.40	**9169.85**	9169.85	1362.11	**9169.85**	9169.85	2713.50
	83	9318.43	9344.68	3627.53	9305.46	9333.66	10781.30	**9218.31**	9218.31	1382.71	**9218.31**	9218.31	2773.83
	104	9326.31	9343.62	3610.47	9312.95	9327.21	10788.50	**9266.97**	9267.57	1271.06	**9266.97**	9266.97	2543.23
	139	10926.45	10926.50	3597.63	**10926.41**	10926.41	10786.30	**10926.41**	10926.41	1009.62	**10926.41**	10926.41	2006.71
	14	51596.40	51837.90	3685.34	51230.00	51792.30	10787.00	**50380.99**	50412.91	1598.01	**50380.99**	50406.39	3158.65
	22	51647.80	51885.90	3702.13	51319.90	51757.80	10783.00	50387.46	50408.30	1604.89	**50380.99**	50397.36	3186.96
	44	51926.10	52279.90	3687.70	51725.40	52065.40	10784.90	**50442.61**	50518.13	1577.89	**50442.61**	50504.37	3132.55
pcb442	63	52503.60	52798.80	3668.46	52440.60	52725.30	10787.10	**50804.12**	51009.56	1555.21	**50804.12**	50996.68	3104.36
	88	53295.30	53546.00	3654.71	52976.10	53330.30	10787.10	51836.61	51878.40	1497.48	**51827.11**	51857.91	2983.56
	110	53977.40	54296.70	3616.31	53810.10	54074.50	10786.60	53017.91	53036.49	1362.34	**53008.73**	53029.38	2750.14
	147	57748.70	58268.60	3601.46	57136.00	57620.90	10778.70	**56979.26**	57143.17	1024.13	**56979.26**	57081.26	2085.71
	16	35171.70	35389.70	3744.66	34791.20	35238.10	10786.80	33552.19	33601.77	1858.08	**33549.99**	33587.17	3725.19
	24	34795.60	35121.90	3717.95	34669.60	34965.60	10789.10	**33360.49**	33400.02	1797.46	**33360.49**	33390.28	3614.26
	49	34043.50	34227.00	3681.39	33745.90	34104.70	10784.40	**33107.63**	33178.65	1809.64	**33107.63**	33168.50	3595.51
d493	70	33823.00	33945.40	3685.30	33703.10	33860.80	10785.10	33126.36	33169.47	1795.64	**33114.20**	33154.71	3575.87
	98	33753.90	33896.90	3624.82	33743.60	33861.30	10791.10	**33258.69**	33281.10	1734.18	**33258.69**	33273.42	3458.70
	123	33978.10	34047.80	3632.32	33930.30	33970.40	10778.10	**33550.20**	33566.12	1621.29	**33550.20**	33558.52	3272.95
	164	36899.50	37145.20	3595.44	36529.10	36714.60	10789.70	**36308.08**	36395.84	1176.92	**36308.08**	36372.68	2423.99
	19	37282.90	37672.80	3839.55	37183.20	37465.20	10785.90	**35300.88**	35372.83	2116.51	**35300.88**	35349.32	4246.62
	28	36960.40	37329.90	3887.82	36861.90	37085.20	10778.60	**35019.18**	35135.99	2081.63	**35019.18**	35103.98	4172.94
	57	36246.20	36467.00	3767.43	36089.40	36311.00	10782.70	34626.17	34718.77	2068.87	**34620.58**	34681.52	4111.88
u574	82	35915.40	36128.00	3710.58	35665.30	35954.70	10774.30	**34526.30**	34629.26	2057.38	**34526.30**	34592.72	4126.35
	114	35810.70	35957.10	3662.61	35697.40	35944.70	10788.70	34787.94	34805.63	1949.52	**34786.12**	34798.61	3933.31
	143	35828.70	36015.50	3656.37	35788.40	35945.30	10793.10	35242.25	35276.38	1791.66	**35229.66**	35254.03	3605.12
	191	38963.00	39563.70	3601.69	38676.00	38954.90	10786.40	**38270.00**	38365.25	1315.39	**38270.00**	38343.88	2672.68
	19	7044.60	7087.02	4039.39	6997.81	7043.92	10785.60	6742.09	6749.21	1921.29	**6737.00**	6746.05	4060.14
	28	7028.39	7058.84	3941.85	7004.34	7032.00	10781.90	**6722.70**	6734.38	1948.46	**6722.70**	6730.89	4043.61
	57	7000.49	7034.23	3792.23	6979.45	7020.09	10250.70	**6726.09**	6739.07	2051.13	**6726.09**	6735.42	4160.82
rat575	82	7031.85	7057.39	3699.51	6991.99	7042.98	10788.70	6771.03	6780.46	2002.89	**6768.41**	6778.53	4049.62
	115	7080.82	7098.76	3662.54	7057.17	7087.36	10788.10	6875.50	6881.05	1929.68	**6872.72**	6878.63	3883.96
	143	7128.70	7162.08	3646.50	7123.17	7158.07	10785.40	**7002.74**	7005.12	1782.79	**7002.74**	7004.00	3567.02
	191	7676.38	7741.48	3608.14	7598.15	7647.01	10780.80	**7505.99**	7527.64	1321.16	**7505.99**	7518.67	2648.29
	21	29957.20	30643.90	5266.74	29988.30	30728.10	10789.90	**29203.54**	29203.54	2549.03	**29203.54**	29203.54	5120.75
	32	29674.60	30033.10	5133.45	29597.30	30040.90	10784.40	**28789.92**	28789.92	2536.36	**28789.92**	28789.92	5085.95
	65	29044.00	29201.10	3755.09	29038.60	29211.50	10786.20	**28670.81**	28670.81	2718.30	**28670.81**	28670.81	5428.71
p654	93	28866.50	28985.80	3721.82	28905.30	29027.80	10779.50	**28670.81**	28670.81	2680.17	**28670.81**	28670.81	5328.00
	130	28981.80	29063.70	3671.12	29001.90	29072.20	10787.50	**28670.81**	28670.81	2275.56	**28670.81**	28670.81	4526.92
	163	29357.60	29428.50	3656.52	29295.60	29405.00	10789.20	**28981.80**	28989.63	1911.36	**28981.80**	28988.13	3819.07
	218	36566.90	36636.00	3625.61	36462.80	36537.80	10784.30	**36385.02**	36385.02	1521.54	**36385.02**	36385.02	3086.58
	24	43645.30	43916.30	3716.62	43369.90	43641.80	10777.20	**41216.64**	41300.21	2848.39	**41216.64**	41263.06	5876.78
	36	43281.50	43674.10	3670.67	43007.80	43349.60	10783.90	**41053.14**	41120.00	2797.58	**41053.14**	41103.27	5769.21
	72	43062.50	43288.00	3672.83	42811.10	42973.90	10789.50	40915.37	40970.97	2869.02	**40904.96**	40949.25	5775.82
u724	103	42898.70	43190.50	3642.09	42725.90	42930.70	10772.80	**40932.41**	41034.50	2749.21	**40932.41**	41003.37	5555.61
	144	42873.20	43129.60	3626.73	42824.50	43024.00	10782.20	**41311.12**	41449.78	2669.00	**41311.12**	41428.60	5396.46
	181	43048.40	43298.00	3607.34	42833.50	43205.90	10785.80	42050.20	42064.26	2480.29	**42048.91**	42059.38	5037.35
	241	48019.70	48686.50	3596.42	47223.60	47722.80	10781.40	46013.12	46162.43	1826.88	**45933.29**	46103.54	3643.58
	26	9272.39	9305.08	3837.89	9212.88	9293.30	10788.60	8728.00	8745.52	3090.63	**8725.77**	8739.79	6602.91
	39	9175.54	9255.61	3778.01	9178.91	9234.04	10770.60	**8681.94**	8703.88	3078.73	**8681.94**	8699.07	6480.44
	78	9168.18	9213.14	3600.14	9065.32	9137.62	10247.40	**8650.22**	8664.47	3173.83	**8650.22**	8662.34	6398.22
rat783	111	9107.87	9168.66	5658.60	9071.19	9108.78	10780.50	8678.00	8689.73	3064.11	**8676.40**	8685.92	6229.26
	156	9092.42	9155.42	4989.09	9032.89	9087.08	10789.80	**8777.96**	8784.42	2927.48	**8777.96**	8783.65	5902.70
	195	9108.34	9153.10	4591.37	9059.83	9111.41	10786.60	8923.15	8928.36	2673.91	**8923.14**	8925.57	5470.59
	261	10399.61	10500.91	4212.07	9986.41	10156.98	10780.00	9697.29	9756.80	1944.80	**9682.53**	9732.97	3958.87
	35	225134.96	227986.45	3777.89	223394.65	225963.13	10793.20	**215349.22**	215752.09	3812.70	**215349.22**	215680.26	8274.27
	53	224693.16	226981.65	3785.05	223564.65	225163.12	10780.60	**213635.07**	214084.55	3786.22	**213635.07**	214008.07	8226.76
	106	223489.43	225697.95	3689.33	223034.00	224354.00	10788.90	211508.48	212143.01	4106.70	**211476.25**	211983.56	8362.29
u1060	151	221948.00	223480.00	3685.01	220495.00	222330.00	10774.40	211256.84	211758.02	3982.87	**211134.54**	211604.66	8066.05
	212	222046.00	223717.00	3640.27	220793.00	221992.00	10238.70	211833.89	212092.69	3697.17	**211833.89**	212068.69	7660.95
	265	223974.00	225573.00	3627.94	221439.00	222668.00	10783.50	214473.67	214647.45	3445.79	**214453.60**	214595.63	7018.27
	353	256443.00	259554.00	3595.42	254342.00	257489.00	10788.30	241271.85	242884.90	2595.44	**240747.48**	242170.01	5215.90
Average	-	46867.27	47246.72	-	46600.74	46952.45	-	45041.42	45121.65	-	45031.85	45094.61	-
p-value	-	3.56E-13	3.56E-13	-	7.64E-13	5.21E-13	-	-	-	-	2.93E-04	3.51E-11	-