Frequent Pattern-Based Search: A Case Study on the Quadratic Assignment Problem

Yangming Zhou, *Member, IEEE*, Jin-Kao Hao, and Béatrice Duval

***Abstract*—We present frequent pattern-based search (FPBS) that combines data mining and optimization. FPBS is a general-purpose method that unifies data mining and optimization within the population-based search framework. The method emphasizes the relevance of a modular- and component-based approach, making it applicable to optimization problems by instantiating the underlying components. To illustrate its potential for solving difficult combinatorial optimization problems, we apply the method to the well-known and challenging quadratic assignment problem. We show the computational results and comparisons on the hardest QAPLIB benchmark instances. This work reinforces the recent trend toward closer cooperations between the optimization methods and machine learning or data mining techniques.**

***Index Terms*—Combinatorial optimization, heuristic design, learning-driven optimization, pattern-based optimization, quadratic assignment.**

I. Introduction

In RECENT years, the interplay of machine learning/data science and optimization has received increasing attention [8], [17], [21], [29], [30]. From the perspective of optimization, machine learning and data mining have been used to calibrate algorithm parameters [40], select heuristic algorithms [6], [20], improve the quality of computed solutions [19], [43], and ameliorate the search capacity [23], [39]. Specifically, data mining involves discovering useful rules and hidden patterns from data. By mining relevant information, such as frequent patterns, from specific solutions encountered during the search and exploring the mined information intelligently, search algorithms can hopefully make their search decisions more informed and thus improve their search performances. Indeed, as shown by the review of Section II, several successful examples have been reported in the literature that demonstrated the usefulness of data mining for helping an optimization method to better solve the optimization problems.

The current work is concerned with a general-purpose solution approach hybridizing data mining procedure and optimization for hard combinatorial optimization problems. Specifically, we introduce frequent pattern-based search (FPBS) that unifies data mining and optimization within the population-based search framework. As we discuss in Sections II and III-H, our work is motivated and inspired by the existing studies that reported excellent performances on several particular applications by mixing data mining techniques and a given metaheuristic. Meanwhile, our research targets a more general objective that goes beyond previous works in the sense that the proposed FPBS method intends to be problem independent and generally applicable to different optimization problems.

From the perspective of design methodology, FPBS is a modular- and component-based method where its composing parts are independent from one another with well-defined functionality. Basically, FPBS maintains a population of high-quality solutions discovered by a suitable optimization procedure and employs an appropriate data mining procedure to extract useful information (i.e., frequent patterns) from the population. The mined patterns are then used to create new starting solutions for the optimization procedure. Each improved solution is finally used to update the population according to a pool management procedure. By combining frequent pattern mining and effective optimization within the population-based search framework, the resulting FPBS algorithm is expected to be able to explore the given search space in an informed and focused manner and, consequently, to attain high-quality solutions effectively and efficiently.

To show the usefulness of the proposed FPBS method, we present a case study on the well-known and challenging quadratic assignment problem (QAP) in Section IV. Besides its popularity as one of the most studied NP-hard combinatorial optimization problems, QAP is also a relevant

Manuscript received July 29, 2019; revised March 24, 2020; accepted September 12, 2020. Date of publication October 27, 2020; date of current version February 17, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61903144; in part by the Shanghai Sailing Program under Grant 19YF1412400; in part by the Macao Young Scholars Program under Grant AM2020011; in part by the Key Project of Science and Technology Innovation 2030 supported by the Ministry of Science and Technology of China under Grant 2018AAA0101302; in part by the Fundamental Research Funds for the Central Universities of China under Grant 222201817006; and in part by the Funding from the Shenzhen Institute of Artificial Intelligence and Robotics for Society. This article was recommended by Associate Editor S. Xie. *(Corresponding author: Jin-Kao Hao.)*

Yangming Zhou is with the Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China, and also with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China (e-mail: ymzhou@ecust.edu.cn).

Jin-Kao Hao is with the Department of Computer Science, LERIA, Université d'Angers, 49045 Angers, France, and also with the Institut Universitaire de France, 75231 Paris, France (e-mail: jin-hao.hao@univ-angers.fr).

Béatrice Duval is with the Department of Computer Science, LERIA, Université d'Angers, 49045 Angers, France (e-mail: beatrice.duval@univ-angers.fr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2020.3027860.

Digital Object Identifier 10.1109/TSMC.2020.3027860

representative of the large class of permutation problems. To apply the FPBS method to solve QAP, we specify the design choices of the underlying components of the method. We then assess the resulting FPBS algorithm on the hardest QAP benchmark instances from the QAPLIB and show its competitiveness compared to the state-of-the-art algorithms. By this case study, we show how the general FPBS method can be conveniently adapted to create a very powerful search algorithm by properly instantiating its components (in the case of QAP, some components simply come from the existing algorithms).

The remainder of this article is organized as follows. Section II provides a review of related works. Section III is dedicated to the presentation of the proposed FPBS method. Section IV showcases the application of FPBS to solve the QAP. Section V investigates some key issues of the algorithm, followed by the conclusions and research perspectives in Section VI.

II. Related Works

In this section, we provide a literature review focusing on studies related to combinations of search methods with data mining techniques for solving combinatorial problems.

The Bayesian optimization algorithm (BOA) [26] is an early precursor of using ideas from machine learning to guide solution construction. In BOA, hidden structures of the optimization problem are discovered with learning techniques during optimization. Since 2004, several studies investigated combinations of particular metaheuristics [especially, greedy randomized adaptive search procedure (GRASP)] with data mining techniques to solve specific optimization problems. For instance, DM-GRASP [33], [35], a pioneer algorithm originally designed to solve the set packing problem, is composed of two phases where GRASP is run for the first half of the whole search to build an elite set of high-quality solutions. At the middle of the search, a data mining procedure is applied to the elite set to mine useful patterns. Then, the second phase uses the second half of the search to run GRASP again to improve each input solution created with a mined pattern (instead of using the usual greedy randomized construction procedure of GRASP). By the same token, DM-HH [23] combines a dedicated hybrid heuristic HH [mixing greedy search, local search, and path-relinking (PR)] and data mining to solve the particular p-median problem. MDM-GRASP [24], [27] extended DM-GRASP by performing data mining as soon as no change occurs in the elite set, instead of doing data mining only once at the midway of the search process like in DM-GRASP and DM-HH. The same idea was also explored by hybridizing data mining with GRASP enhanced by PR [7] to solve the particular 2-path network design problem or variable neighborhood descent [16] to solve the specific one-commodity pickup-and-delivery traveling salesman problem. The work of [14] used a memory of high-quality solutions to improve constructive multistart methods (e.g., GRASP). This early work does not explicitly call for a data mining procedure, instead, it extracts, from the memory, frequency-based information, which is used to improve the construction phase of GRASP.

In addition to GRASP, data mining has also been hybridized with other metaheuristics like evolutionary algorithms. To improve the performance of an evolutionary algorithm for solving an oil collecting vehicle routing problem, a hybrid algorithm (GADMLS) combining genetic algorithm, local search, and data mining was proposed in [34]. Another hybrid approach (GAAR) that uses a data mining module to guide an evolutionary algorithm was presented in [31] to solve the constraint satisfaction problem. Besides the standard components of a genetic algorithm, a data mining module is added to find association rules (between variables and values) from an archive of best individuals found in the previous generations. There are other related, but more distant works that showed the benefit of data mining procedure for heuristic search. For example, in the context of the set partitioning problem, data mining was applied to extract variable associations from previously solved instances for identifying promising pairs of flipping variables in a large neighborhood search method and thus reducing the explored search space [38]. Another example is the hybridization of neighborhood search with data mining techniques for solving the p-median problem [32].

One observes that the reviewed studies share the basic idea of using techniques from data science to improve the search process. It is worth noting, however, that previous approaches typically deal with *specific problems* (e.g., p-median and vehicle routing) with *particular optimization methods* (e.g., GRASP). As such, these approaches lacks generality and are not readily applicable to other problems. In this work, we aim to generalize the key ideas of these pioneer studies and propose a general-purpose approach that unifies data mining and optimization within the population-based search framework for solving combinatorial optimization problems.

III. Frequent Pattern-Based Search

In this section, we first show the general scheme of the FPBS method and then present its underlying components.

A. General Scheme

The basic idea of the proposed FPBS method is to hybridize data mining and optimization within the population-based framework with the purpose of achieving a suitable balance of exploration and exploitation of the search process. Data mining is responsible for useful patterns extraction from the population. Extracted patterns are then used as "building blocks" to create new promising solutions, which are further improved by optimization. As such, combining data mining and optimization provides the resulting algorithm with the capacity of continually exploring new promising search regions (with pattern-based new solutions) and exploiting particular regions in depth (with local optimization).

From the perspective of system architecture, FPBS maintains a population of high-quality solutions for the purpose of pattern mining and optimization. Specifically, FPBS is composed of six independent operating components: 1) an initialization procedure (Section III-B); 2) a data mining procedure (Section III-C); 3) a pattern selection procedure (Section III-D); 4) a frequent pattern-based solution

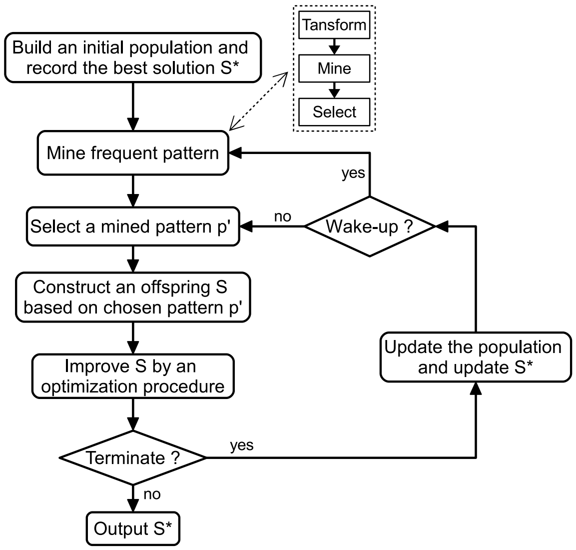

Fig. 1. Flow diagram of the proposed FPBS approach.

Algorithm 1: General Procedure of the FPBS Approach

Input: Problem instance I with a minimization objective f, population size k and number of patterns to be mined m
Output: The best found solution S^*
1 **begin**
2 // construct a population (POP)
3 $POP \leftarrow InitializePopulation()$;
4 // record the best solution S^*
5 $S^* \leftarrow \arg\min\{f(S_i): i = 1, 2, \ldots, k\}$;
6 // mine frequent patterns from POP
7 $\mathcal{P} \leftarrow MineFrequentPattern(POP, m)$;
8 **while** *a stopping condition is not reached* **do**
9 // select a mined frequent pattern p
10 $p \leftarrow SelectPattern(\mathcal{P})$;
11 // construct a new solution using p
12 $S \leftarrow ConstructSolutionBasedPattern(p)$;
13 // improve the constructed solution
14 $S' \leftarrow Optimize(S)$;
15 // update the best solution found so far
16 **if** $f(S') \leq f(S^*)$ **then**
17 $S^* \leftarrow S'$;
18 // update the population
19 $POP \leftarrow UpdatePopulation(POP, S')$;
20 // wake-up the mining procedure when the population stagnates
21 **if** *population does not evolve any more* **then**
22 $\mathcal{P} \leftarrow MineFrequentPattern(POP, m)$;
23 **return** The best found solution S^*;

construction procedure (Section III-E); 5) an optimization procedure (Section III-F); and 6) a pool management procedure (Section III-G).

The flow diagram and the pseudocode of the FPBS approach are shown in Fig. 1 and Algorithm 1, respectively. FPBS starts from a set of high-quality solutions that are obtained by the initialization procedure (line 3). From these high-quality solutions, a data mining procedure is invoked to mine frequent patterns (line 7). The algorithm enters the main "while" loop (lines 8–22) to perform a number of generations to evolve the solutions in the population. At each generation, a mined pattern is first selected (line 10) and used to create a new solution (line 12) that is further improved by the optimization procedure (line 14). The improved solution is finally inserted to the population according to the pool management policy (line 19). The process is repeated until a stopping condition (e.g., a time limit or a given maximum number of generations) is satisfied. In addition to its invocation just after the initialization procedure, the data mining procedure is also waked-up if no evolution of the population is observed during a predefined number of generations.

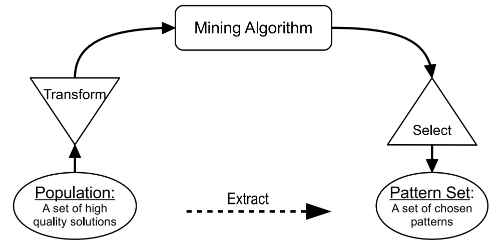

Fig. 2. Diagram displaying the process from population to pattern set.

B. Population Initialization

FPBS starts its search with a population (POP) composed of k distinct high-quality solutions. To build the population, we first generate, by any means (e.g., with a random or greedy construction method), an initial solution that is improved by an optimization procedure (see Section III-F). The improved solution is then inserted into the population according to the pool management strategy (see Section III-G). We repeat this process until k different solutions are built. Such an initialization process is very popular in memetic algorithms [25] as exemplified in [10] and [42].

C. Frequent Pattern Mining Procedure

The frequent pattern mining procedure is used to discover specific patterns that frequently occurs in high-quality solutions. One typical frequent pattern that can be mined is frequent itemset, which was originally introduced for mining transaction databases [3]. Given a transaction database defined over a set of items, a frequent itemset refers to a set of items that often appear together in the dataset. Frequent patterns are not limited to itemsets and can correspond to more complex entities such as subsequences or substructures [2]. To apply frequent pattern mining to combinatorial optimization problems, it is necessary to define a suitable pattern for the problem under consideration.

Fig. 2 shows the frequent pattern mining process composed of three steps: 1) a *transformation* step that transforms a set of high-quality solutions to a dataset recognizable by a given mining procedure; 2) a *mining* step that mines frequent patterns from the transformed dataset; and 3) a *selection* step that selects a predefined number of patterns from the mined patterns.

To handle a wide diversity of data types, numerous mining tasks and algorithms have been proposed in [2]. Within the FPBS framework, the mining algorithm will be selected based on the characteristics of the chosen pattern defined for the given problem. For instance, for our case study on QAP presented in Section IV, a pattern corresponds to a set of element-location pairs. Thus, frequent patterns can be conveniently represented as frequent itemsets (i.e., a set of identical element-location assignments). To identify frequent itemsets, any mining algorithm (like FPmax* [15]) can be applied. Other frequent pattern mining algorithms are available according to the types of intended patterns [2].

D. Pattern Selection

To construct new solutions based on the mined patterns, we use a pattern selection procedure to identify the next pattern that is used to create a new solution. The first strategy for pattern selection is the tournament selection that works as follows. Let λ be the size of the tournament pool. We randomly choose λ $(1 \leq \lambda \leq |\mathcal{P}|)$ individuals with replacement from the mined pattern set $\mathcal{P}$, and then pick the best one (i.e., with the largest size), where λ is a parameter. The computational complexity of this selection strategy is $O(|\mathcal{P}|)$. The advantage of the tournament selection strategy is that the selection pressure can be easily adjusted by changing the size of the tournament pool λ. The larger the tournament pool is, the smaller the chance for shorter patterns to be selected. The second selection strategy is to always pick the next longest pattern in the set of mined patterns [32]. One observes that the second selection strategy is a special case of the tournament selection when λ equals the size of the pattern set. For our study on QAP of Section IV, we adopt the tournament selection strategy.

E. Solution Construction Based on Mined Patterns

Since frequent patterns usually correspond to a set of common elements shared by the sampled high-quality solutions, each mined pattern directly defines a partial solution. To obtain a full solution, it is convenient to apply a random or greedy procedure to complete the partial solution. While a random construction procedure adds the missing elements at random, a greedy procedure completes each missing element according to a greedy criterion favoring objective optimization. In the latter case, the greedy criterion is a critical issue to consider with respect to the optimization objective under consideration.

Finally, the solution completion process can also be guided by high-quality solutions. In this case, the partial solution can be completed with elements from one or more specific high-quality solutions. In Section IV, we illustrate such an approach in the context of solving QAP.

F. Optimization Procedure

For solution improvement (to build the initial population and to improve each new solution built from a mined pattern), existing solution algorithms dedicated to the given problem can be applied in principle. On the other hand, since the optimization component ensures the key role of search intensification, it is desirable to call for a powerful search algorithm. Basically, the optimization procedure can be considered to be a black-box optimizer that is called to improve an input solution. For instance, for solving QAP (Section IV), we adopt the breakout local search (BLS) algorithm [9], which is also used as the underlying optimization procedure of the memetic algorithm BMA [10].

G. Population Management

For each new solution constructed using a mined frequent pattern, we use the optimization procedure to improve its quality. Then, we decide whether the improved solution should be inserted into the population *POP*. There are a number of pool updating strategies in [28] and [42] that can be applied within the FPBS approach. First, the classic quality-based replacement strategy simply inserts the new solution into the pool to replace the worst solution if the new solution is not worse than the worst solution in the pool [10]. Second, more elaborated updating strategies consider additional criteria. For instance, the quality-and-distance updating strategy considers not only the quality of the solution but also its distance to other solutions in the pool [28]. Finally, the rank-based quality-and-distance updating strategy proposed in [42] can be applied as well. For the QAP of Section IV, we adopt the second updating strategy.

H. Connections With Existing Studies

As evidenced by the review of Section II, FPBS relies on a combination of data science techniques and optimization methods. As such, it shares some basic principles and similarities with existing studies. Meanwhile, our work targets a broader objective of designing a general-purpose search framework that can be instantiated to solve various problems. This stands in sharp contrast to previous approaches that typically focus on particular problems and methods. In what follows, we show the connections between FPBS and the most related studies.

DM-HH [23] combines a dedicated HH and data mining to solve the specific p-median problem. The data mining procedure is used only one time at the middle of the whole HH search process. DM-GRASP and the multi-DM-GRASP (MDM-GRASP) [27] hybridize the GRAPS metaheuristic and data mining to solve the server replication for reliable multicast problem. DM-GRASP follows exactly the same approach as DM-HH [23] such that data mining is applied one time at the middle of the GRASP process. MDM-GRASP applies data mining multitimes: 1) as soon as no change occurs in the collected elite solutions throughout a given number of iterations and 2) when the elite set has been changed and again has become stable. The idea of using patterns is also related to exploiting the concepts of backbone in binary optimization (e.g., satisfiability [41] and unconstrained binary quadratic programming [45]) where stable value-to-variable assignments through a set of high-quality solutions form the backbone. The principle of backbone was used to define effective crossovers for problems, such as graph coloring [28], maximum diversity [42], and critical node detection [44], where common elements shared by two or multiple parent solutions serve as the partial offspring solution.

FPBS distinguishes itself by the following features. First, it is a generic and unified framework applicable to various optimization problems. To apply FPBS to a new problem, it suffices to instantiate the underlying components. As showcased in Section IV, the instantiation can even benefit from existing algorithms and procedures. Second, FPBS follows the modular design principle, which eases its application to new problems by reusing the problem-independent components and adopting the most suitable problem-specific components. Third, FPBS unifies data mining and optimization within the general population-based framework, which makes the hybridization very flexible and favors the achievement of a search balance of exploration and exploitation.

Below, we show how FPBS can be conveniently adapted to the challenging QAP. This application also provides an example of using FPBS to solve the large class of permutation problems (quadratic assignment is a particular member).

IV. FPBS Applied to the Quadratic Assignment Problem

We now apply the general FPBS approach to QAP and compare its performance with the state-of-the-art algorithms.

A. Quadratic Assignment Problem

Given a set $N = \{1, \ldots, n\}$ of n facilities, a set $M = \{1, \ldots, n\}$ of n locations that can host the facilities, a flow a_{ij} from facility i to facility j for all $i, j \in N$, and a distance b_{uv} between locations u and v for all $u, v \in M$, QAP involves determining a minimal cost assignment of n facilities to n locations. Clearly, a facility-location assignment can be conveniently represented by a permutation $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ such that $\pi(i)$ represents the assigned location of facility i. Let Ω denote the set of all n-permutations, then the NP-hard QAP can be formulated as follows:

$$\min_{\pi \in \Omega} f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}. \tag{1}$$

The optimization objective of (1) is to find a permutation π^* in Ω that minimizes the sum of the products of the flows and distances, i.e., $f(\pi^*) \leq f(\pi)\ \forall \pi \in \Omega$.

In addition to the facility location problem, QAP finds applications in electrical circuit design, distributed computing, image processing, and so on [13], [22]. Moreover, a number of classic NP-hard problems, such as the traveling salesman problem, the maximum clique problem, the bin packing problem, and the graph partitioning problem, can also be recast as QAPs [22].

Due to its practical and theoretical significance, QAP has attracted much research effort [1], [9], [10], [18], [37]. In fact, QAP is one of the most studied combinatorial optimization problems. Since exact algorithms are impractical for instances with $n > 36$ [5], a large number of heuristic methods have been proposed to provide near-optimal approximate solutions in a reasonable computation time. Detailed reviews of heuristic and metaheuristic algorithms developed till 2007 for QAP are available in [13] and [22]. Brief reviews of more recent studies can be found in [4], [9], and [10].

Algorithm 2: FPBS Algorithm for QAP

Input: Instance G, population size k, the number of mined patterns m, time limit t_{max} and the maximum number of generations without updating *max_no_update*
Output: The best found solution π^*

begin
  $POP \leftarrow InitializePopulation()$;
  $\pi^* \leftarrow \arg\min\{f(\pi_i): i = 1, 2, \ldots, k\}$;
  $\mathcal{P} \leftarrow MineFrequentPattern(POP, m)$;
  $no\_update \leftarrow 0$;
  $t \leftarrow 0$;
  while $t < t_{max}$ **do**
    $p_i \leftarrow SelectPattern(\mathcal{P})$;
    // build a new solution based selected pattern
    $\pi \leftarrow ConstructSolutionBasedPattern(p_i)$;
    // improve the constructed solution
    $\pi' \leftarrow BreakoutLocalSearch(\pi)$;
    // update the best solution found so far
    if $f(\pi') < f(\pi^*)$ **then**
      $\pi^* \leftarrow \pi'$;
    // update the population
    if $UpdatePopulation(POP, \pi') = True$ **then**
      $no\_update \leftarrow 0$;
    else
      $no\_update \leftarrow no\_update + 1$;
    // wake-up the mining procedure when the population is steady
    if $no\_update > max\_no\_update$ **then**
      $\mathcal{P} \leftarrow MineFrequentPattern(POP, m)$;
      $no\_update \leftarrow 0$;
return The best found solution π^*;

B. FPBS for QAP

We use FPBS-QAP (Algorithm 2) to denote the resulting FPBS algorithm. Since FPBS-QAP inherits the main components of FPBS, we only present the specific features related to QAP: solution representation, optimization procedure, frequent pattern mining for QAP, solution construction using QAP patterns, and population update strategy.

1) Search Space and Neighborhood: Given a QAP instance with n facilities and n locations, the search space Ω is composed of all possible $n!$ permutations. For any solution $\pi \in \Omega$, its quality is given by (1).

To explore the search space, we adopt an effective neighborhood search algorithm called BLS (see Section IV-B2), which relies on the following swap neighborhood. Given a solution, i.e., a permutation π, its neighborhood $N(\pi)$ is defined as the set of all possible permutations that can be obtained by exchanging the values of any two locations $\pi(u)$ and $\pi(v)$ in π, i.e., $N(\pi) = \{\pi' \mid \pi'(u) = \pi(v), \pi'(v) = \pi(u), u \neq v \text{ and } \pi'(i) = \pi(i)\ \forall i \neq u, v\}$, which has a size of $n(n-1)/2$. Given a permutation π and its objective value $f(\pi)$, the objective value of a neighboring permutation π' can be effectively calculated according to an incremental evaluation technique [36].

2) Breakout Local Search: To ensure an effective examination of the search space, we adopt, like the memetic algorithm BMA [10], the BLS algorithm [9] as our black-box optimizer.

In addition to being a state-of-the-art QAP algorithm, its source code is publicly available, making it possible to perform meaningful comparative studies.

BLS follows the iterated local search scheme and repetitively alternates between a descent search phase (to find local optima) and a dedicated perturbation phase (to discover new promising regions). BLS starts from an initial random permutation, and then improves the initial solution to a local optimum by the best improvement descent search with the above swap neighborhood. Upon the discovery of a local optimum, BLS switches to the perturbation phase. The perturbation adaptively selects a directed perturbation or a random perturbation, which is applied L times (L is called the perturbation strength) [9].

The directed perturbation and random perturbation provide two complementary means for search diversification. The directed perturbation applies a selection rule that favors neighboring solutions with weak objective deterioration, under the constraint that the neighboring solutions have not been visited during the last γ iterations (where γ is the predefined tabu tenure). The random perturbation selects neighboring solutions in the neighborhood at random without considering objective deterioration. BLS alternates between these two types of perturbation in a probabilistic and adaptive way. The probability of selecting a particular perturbation is determined dynamically according to the current number of visited local optima without improving the best solution found, while the probability of applying the direct perturbation takes a value no smaller than a given minimal threshold Q. The perturbation strength L is determined based on a simple reactive strategy. L is increased if the search returns to the immediate previous local optimum, and reset to a given initial value L_0 otherwise. Finally, the selected perturbation with the strength L is applied to transform the current solution. The resulting solution is then used as the starting solution of the next round of the descent search procedure (see [9] for more details).

3) Mining Frequent Patterns for QAP: QAP is a typical permutation problem. For this problem, we define a frequent pattern to be a set of identical location-facility assignments shared by high-quality solutions and represent such a pattern by an itemset. To apply a frequent itemset mining algorithm, we need to transform a permutation into a set of items. In [16], a transformation is proposed for a generalized traveling salesman problem. For each pair of elements [$\pi(i)$ and $\pi(j)$] of a given permutation π, an arc $(\pi(i), \pi(j))$ is generated, thus mapping a permutation π to a set S' of $|\pi| - 1$ arcs. For example, consider a permutation $\pi = (5, 4, 7, 2, 1, 6, 3)$, π is transformed to the set of arcs $S' = \{(5, 4), (4, 7), (7, 2), (2, 1), (1, 6), (6, 3)\}$. This transformation conserves the order of elements. However, the information between the elements (facilities) and their locations is lost. In practice, we cannot identify the true location of an element when only a part of pairs is available.

To overcome this difficulty, we propose another transformation, which decomposes a permutation π of n elements into a set of ordered element-position pairs $\{(1, \pi(1)$, $(2, \pi(2), \ldots, (n, \pi(n))\}$. Thus, the permutation $\pi = (5, 4, 7, 2, 1, 6, 3)$ above is transformed into $\{(1, 5), (2, 4), (3, 7), (4, 2), (5, 1), (6, 6), (7, 3)\}$, where each element-position pair $(i, \pi(i))$ is considered as an item $(i - 1) * |\pi| + \pi(i)$. Fig. 3 shows three solutions (permutations) with the set $\{I_1, I_2, I_3\}$ of three resulting itemsets.

population

	1	2	3	4	5	6	7
π_1	5	4	7	2	1	6	3
π_2	7	4	5	3	1	6	2
π_3	7	4	5	2	3	6	1

→

item set

5	11	21	23	29	41	45	I_1
7	11	19	24	29	41	44	I_2
7	11	19	23	31	41	43	I_3

Fig. 3. Illustrative example of the transformation procedure, which transforms a set of three permutations $\{\pi_1, \pi_2, \pi_3\}$ to a set of three item sets $\{I_1, I_2, I_3\}$. For instance, the first element of solution π_1 is $\pi_1(1) = 5$, so the item will be $(1 - 1) * 7 + \pi_1(1) = 5$.

With the help of our transformation procedure, mining frequent patterns from multiple permutations becomes the task of mining frequent itemsets. The main drawback of mining all frequent itemsets is that if there are many frequent items, then a high number of subsets of the frequent items need to be examined. However, it usually suffices to find only the maximal frequent itemsets (a maximal frequent itemset is that it has no superset that is frequent). Thus mining frequent itemsets can be reduced to mine only maximal frequent itemsets. For this purpose, we adopt the popular FPmax*[1] algorithm [15].

FPmax* is one of the most efficient implementations for computing the maximal frequent itemsets of a database. It relies on the representation of the database by a prefix tree, called frequent pattern tree (FP-tree), that contains condensed information about the frequent patterns. Then recursive traversals of this FP-tree enable to obtain the frequent itemsets or maximal frequent itemsets. Efficient array-based representations are used to streamline the computation needed to traverse the FP-tree and obtain the maximal frequent itemsets. In FPmax*, a user-specified minimum support θ is necessary to find all frequent itemsets in a database, where θ can be any integer between 2 and the size of database. We set $\theta = 2$ according to preliminary experiments (see Table I).

4) Solution Construction Based on Mined Pattern: From the mined frequent items, we apply the tournament selection strategy (see Section III-D) to select the next pattern that is used to construct a new solution. Algorithm 3 describes the main steps of the construction procedure. Initially, we remap the chosen pattern into a partial solution π (line 3). If the partial solution π contains a number $|\pi|$ of elements fewer than a given threshold (i.e., $\beta * n$), we use a high-quality solution to guide the construction (lines 4–8). Specifically, we first randomly select a high-quality solution π^0 (called guiding solution) from the population (line 6), and then we complete π based on the guiding solution π^0, by copying the element of each unassigned position of π^0 to π under the condition that the element is unassigned in π (line 8). Finally, if π is still an incomplete solution, we randomly assign the remaining elements to the unassigned positions until a full solution is obtained (line 10). As explained in Section III, the constructed new solution is then improved by the optimization procedure (i.e., BLS) and used to update the population.

[1]The source code of the FPmax* algorithm is publicly available at http://fimi.ua.ac.be/src/.

Algorithm 3: Solution Construction Based on Mined Pattern

Input: A selected pattern p and a population *POP* of size k
Output: A new solution π
```
begin
    // re-map the selected pattern as a partial solution
    π ← Re-map(p);
    if |π| < β * n then
        // select a guiding solution
        π^0 ← SelectGuidedSolution(POP);
        // construct based on guiding solution
        π ← GuidedConstruct(π, π^0);
    //complete at random
    π ← RandomComplete(π);
return A new solution π;
```

5) Population Updating: The last step of FPBS-QAP uses the improved solution (call it π') from the BLS procedure to update the population *POP*, according to the following strategy. π' is inserted into *POP* if two conditions are satisfied simultaneously: 1) π' is different from any solution in *POP* and 2) π' is no worse that the worst solution in *POP*, i.e., $f(\pi') \leq f(\pi^w)$, where $\pi^w \leftarrow \arg\max_{\pi \in POP}\{f(\pi)\}$ is the worst solution in the population.

C. Computational Studies of FPBS for QAP

To evaluate the FPBS-QAP algorithm, we first perform a detailed comparison between FPBS-QAP and two state-of-the-art algorithms, i.e., BLS [9] and BMA [10], whose source codes are available. Then, we compare FPBS-QAP with four additional recent state-of-the-art algorithms.

1) Benchmark Instances: Experimental evaluations of QAP algorithms are usually performed on 135 popular benchmark instances from QAPLIB[2]. The instance size n ranges from 12 to 150, and is indicated in the instance name. These instances can be classified into four categories.

1) *Type I:* 114 *real-life instances* are obtained from practical QAP applications.
2) *Type II:* 5 *unstructured, randomly generated instances* whose distance and flow matrices are randomly generated based on a uniform distribution.
3) *Type III:* 5 *real-like-life instances* are generated instances that are similar to the real-life QAP instances.
4) *Type IV:* 11 *instances with grid-based distances* in which the distances are the Manhattan distance between points on a grid.

Like [9] and [10], we ignore the 114 easy instances from Type I because the known optimal solutions can be found easily within a short time, often less than one second by our method and other modern methods. Our experiments focus on the 21 hard instances with unknown optima from Types II–IV. It is worth mentioning that for these 21 most challenging instances, no single algorithm including the most recent algorithms can attain the best-known results for all the instances. Indeed, even the currently best performing algorithms miss at least two best-known results. This is understandable given that these instances have been studied for a long time and some best-known objective values given on the QAPLIB page have been achieved under specific and relaxed conditions.

[2]https://www.opt.math.tugraz.at/qaplib/

TABLE I
PARAMETER SETTINGS OF THE FPBS-QAP ALGORITHM

Parameter	description	value
t_{max}	time limit (hours)	0.5 or 2.0
k	population size	15
max_no_update	number of times without updating	15
θ	minimum support	2
m	number of mined patterns in pattern set	11
λ	tournament pool size	3
β	length threshold	0.75
max_iter	number of iterations for BLS*	10000

* We used BLS as the local optimization procedure in our algorithm. Other six parameters of BLS adopt the default values provided in [9].

2) Experimental Settings: Our FPBS-QAP algorithm[3] was implemented in the C++ programming language and complied with the gcc 4.1.2 and the flag "−O3." All the experiments were carried out on a computer equipped with an Intel E5-2670 processor (2.5 GHz and 2-GB RAM) running Linux. With the -O3 flag, running the well-known DIMACS machine benchmark procedure dfmax.c[4] on our machine requires 0.19, 1.17, and 4.54 s to solve the benchmark graphs r300.5, r400.5, and r500.5, respectively. Our computational results were obtained by running FPBS-QAP with the parameter setting provided in Table I. To identify an appropriate value for a given parameter, we compared the performance of the algorithm with different parameter values, while fixing other parameter values. Appendix shows an example to select the number of the mined patterns m (i.e., the number of patterns in pattern set). We mention that the setting of Table I was obtained without using a fine-tuning procedure and was consistently used to solve all 21 QAP instances. Fine-tuning some parameters for a specific instance would lead to improved results. However, doing this will deviate from the main goal of the work. On the other hand, when applying FPBS-QAP to new problems, it would be advantageous to adjust some parameters to achieve the best possible results.

Following the QAP literature (e.g., [1], [9], [10], [12], [18], and [37]) and to ensure fair comparisons, the proposed FPBS-QAP algorithm independently ran 30 times on each test instance. Our assessment is based on the percentage deviation (PD) metrics that are widely used in the previous studies [1], [4], [9], [10], [12], [18], [37]. The PD metrics measures the PD from the best-known value (BKV). For example, the best PD (BPD), the average PD (APD), and the worst PD (WPD) are, respectively, calculated according to

$$XPD = 100 * \frac{X - BKV}{BKV}[\%] \tag{2}$$

where $X \in \{B, A, W\}$ corresponds to the best, average, and worst objective value achieved by an algorithm. A smaller XPD value indicates a better performance.

[3]We will make the program of the FPBS-QAP algorithm available at http://www.info.univ-angers.fr/pub/hao/fpbs.html
[4]dfmax: ftp://dimacs.rutgers.edu/pub/dsj/clique

TABLE II
PERFORMANCE COMPARISON OF THE PROPOSED FPBS-QAP ALGORITHM WITH BLS AND BMA ON 21 HARD INSTANCES UNDER $t_{\max} = 30$ MIN. FOR EACH ALGORITHM, WE CONDUCT 30 INDEPENDENT RUNS ON EACH INSTANCE

		BPD			APD			WPD		
Instance	BKV	BLS	BMA	**FPBS-QAP**	BLS	BMA	**FPBS-QAP**	BLS	BMA	**FPBS-QAP**
tai40a	3139370	0.000	0.000	0.000	0.067	0.067	0.067	0.074	0.074	0.074
tai50a	4938796	0.000	0.039	0.000	0.199	0.203	0.288	0.364	0.372	0.445
tai60a	7205962	0.204	0.165	0.165	0.375	0.362	0.351	0.472	0.384	0.471
tai80a	13499184	0.375	0.379	0.320	0.599	0.517	0.509	0.730	0.681	0.682
tai100a	21052466	0.323	0.290	0.275	0.568	0.452	0.430	0.632	0.623	0.613
tai50b	458821517	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
tai60b	608215054	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
tai80b	818415043	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
tai100b	1185996137	0.000	0.000	0.000	0.001	0.033	0.033	0.045	0.198	0.142
tai150b	498896643	0.003	0.000	0.000	0.205	0.183	0.193	0.397	0.318	0.321
sko72	66256	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.063	0.000
sko81	90998	0.000	0.000	0.000	0.007	0.000	0.000	0.011	0.000	0.000
sko90	115534	0.000	0.000	0.000	0.023	0.009	0.007	0.095	0.038	0.038
sko100a	152002	0.000	0.000	0.000	0.006	0.008	0.000	0.024	0.043	0.000
sko100b	153890	0.000	0.000	0.000	0.001	0.000	0.000	0.004	0.000	0.000
sko100c	147862	0.000	0.000	0.000	0.001	0.000	0.000	0.004	0.000	0.000
sko100d	149576	0.000	0.000	0.000	0.003	0.000	0.000	0.011	0.000	0.000
sko100e	149150	0.000	0.000	0.000	0.002	0.000	0.001	0.006	0.000	0.004
sko100f	149036	0.000	0.000	0.000	0.003	0.002	0.001	0.032	0.025	0.021
wil100	273038	0.000	0.000	0.000	0.001	0.000	0.000	0.003	0.000	0.000
tho150	8133398	0.013	0.002	0.000	0.069	0.031	0.041	0.135	0.129	0.130
avg.value		0.044	0.042	**0.036**	0.101	**0.089**	0.091	0.145	**0.140**	**0.140**
avg.rank		2.167	2.048	**1.785**	2.500	1.857	**1.643**	2.548	1.810	**1.642**

To analyze these results, we resort a two-step statistical test procedure [11]. First, we conduct a *Friedman* test which makes the null hypothesis that all compared algorithms are equivalent. Once the null hypothesis is rejected, we then proceed with the two-tailed *Nemenyi* post-hoc test. Both tests are based on the average ranks. We order the algorithms for each instance separately, the best performing algorithm obtaining the rank of 1, the second best rank of 2, and so on. In case of ties, average ranks are assigned. Finally, we obtain the average rank of each algorithm by averaging the ranks of all 21 instances.

3) Comparison of FPBS-QAP With BLS and BMA: To evaluate FPBS-QAP, we first show a detailed comparison with the two main reference algorithms: 1) BLS [9] and 2) BMA (population-based memetic algorithm) [10]. This experiment was conducted based on three considerations. First, BLS and BMA are among the best performing QAP algorithms currently available in the literature. Second, the source codes of BLS and BMA are available, making it possible to make a fair comparison (using the same computing platform and stopping conditions). Third, since both FPBS-QAP and BMA use BLS as their underlying optimization procedure, this comparison allows us to assess the added value of the data mining component of FPBS-QAP compared to the population-based approach BMA. For this experiment, we ran, like in [9] and [10], FPBS-QAP and the two reference algorithms under two stopping conditions, i.e., a cutoff time of $t_{\max} = 30$ min (0.5 h) and a cutoff time of $t_{\max} = 120$ min (2 h). This allows us to study the behavior of the compared algorithms under short and long time limits.

The comparative performances of FPBS-QAP with BLS and BMA under the above conditions are presented in Tables II and III. We report the BPD, APD, and WPD values of each algorithm. At the last two rows of each table, we also show the average value and the average rank of each indicator. The smaller the value, the better the performance of an algorithm.

Table II shows that FPBS-QAP achieves the best performance compared to BLS and BMA under $t_{\max} = 30$ min. First, FPBS-QAP finds all BKVs except three cases (tai60a, tai80a, tai100a) while BLS and BMA fail to do so for five instances. Second, FPBS-QAP is able to reach the BKVs of the two largest instances (tai150b and tho150) within the given time limit. BLS fails to find the BKVs for these two instances within the limit of 30 min (it can find these values only under a very long time limit of $t_{\max} = 10$ h). BMA performs better than BLS by attaining the BKV of tai150b, but still fails on tho150. When we check the average performance of each algorithm over the 21 instances given in the last two rows of Table II, we observe that the average BPD value of FPBS-QAP is only 0.036%, which is better than 0.044% of BLS, and 0.042% of BMA. Similar observations can be made for the average WPD indicator. For the average APD indicator, FPBS-QAP is slightly worse than BMA, but it is better than BLS. It is worth noting that FPBS-QAP achieves the smallest average ranks for all three performance indicators.

With three algorithms and 21 instances, the critical value of F(2,40) for the significant level 0.05 is 3.232. At a significant level of 0.05, we reject the null hypothesis for the APD indicator ($F_F = 4.967 > 3.232$) and the WPD indicator ($F_F = 6.051 > 3.232$), but we accept the null hypothesis for the BPD indicator ($F_F = 0.795 < 3.232$) according to the Friedman test. For both APD and WPD indicators, we additionally conduct a Nemenyi test. At a significant level of 0.05, the critical value is $2.343 \times \sqrt{[(3 \times 4)/(6 \times 21)]} = 0.723$. We observe that FPBS-QAP significantly outperforms BLS both in terms of the APD indicator (i.e., $2.500 - 1.643 > 0.723$) and the WPD indicator (i.e., $2.548 - 1.642 > 0.723$). Compared

TABLE III
PERFORMANCE COMPARISON OF THE PROPOSED FPBS-QAP ALGORITHM WITH BLS AND BMA ON 21 HARD INSTANCES UNDER $t_{\max} = 120$ MIN. FOR EACH ALGORITHM, WE CONDUCT 30 INDEPENDENT RUNS ON EACH INSTANCE

		BPD			APD			WPD		
Instance	BKV	BLS	BMA	**FPBS-QAP**	BLS	BMA	**FPBS-QAP**	BLS	BMA	**FPBS-QAP**
tai40a	3139370	0.000	0.000	0.000	0.012	0.047	0.037	0.074	0.074	0.074
tai50a	4938796	0.000	0.000	0.000	0.077	0.091	0.106	0.251	0.289	0.231
tai60a	7205962	0.036	0.161	0.000	0.241	0.195	0.189	0.353	0.352	0.311
tai80a	13499184	0.397	0.303	0.288	0.510	0.434	0.467	0.637	0.564	0.618
tai100a	21052466	0.281	0.223	0.250	0.455	0.378	0.380	0.574	0.513	0.466
tai50b	458821517	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
tai60b	608215054	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
tai80b	818415043	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
tai100b	1185996137	0.000	0.000	0.000	0.000	0.010	0.000	0.000	0.100	0.000
tai150b	498896643	0.001	0.000	0.000	0.109	0.200	0.092	0.243	0.427	0.313
sko72	66256	0.000	0.000	0.000	0.000	0.004	0.000	0.000	0.063	0.000
sko81	90998	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
sko90	115534	0.000	0.000	0.000	0.000	0.009	0.010	0.000	0.038	0.038
sko100a	152002	0.000	0.000	0.000	0.001	0.002	0.000	0.008	0.043	0.000
sko100b	153890	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
sko100c	147862	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
sko100d	149576	0.000	0.000	0.000	0.001	0.000	0.000	0.005	0.000	0.000
sko100e	149150	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
sko100f	149036	0.000	0.000	0.000	0.001	0.001	0.003	0.005	0.005	0.005
wil100	273038	0.000	0.000	0.000	0.001	0.000	0.000	0.002	0.000	0.000
tho150	8133398	0.007	0.000	0.000	0.049	0.026	0.006	0.126	0.104	0.064
avg.value		0.034	0.033	**0.026**	0.069	0.067	**0.061**	0.108	0.123	**0.101**
avg.rank		2.190	1.953	**1.857**	2.072	2.071	**1.857**	2.143	2.167	**1.690**

to BMA, FPBS-QAP achieves smaller average ranks but there is no significant differences between them.

Under the long time limit of $t_{\max} = 120$ min, our FPBS-QAP algorithm is able to achieve even better results as well as BLS and BMA. As we see from Table III, the BKVs are attained more often than under the limit of $t_{\max} = 30$ min. Interestingly, FPBS-QAP successfully finds the BKV for one more instance (tai60a), missing only 2 BKV against the unchanged 5 cases for BLS and 3 cases for BMA. For the average performance, the average BPD value of FPBS-QAP is 0.026%, which is the best compared to 0.033% of BMA and 0.034% of BLS. FPBS-QAP also achieves the smallest average APD value and average WPD value. Furthermore, FPBS-QAP has the best average rank on all three indicators compared to BLS and BMA. Finally, FPBS-QAP achieves a marginally better average performance, there is no significant difference among the compared algorithm at a significance level of 0.05.

In summary, the FPBS-QAP algorithm competes favorably with the two best-performing (sequential) QAP algorithms (i.e., BLS and BMA) according to the different indicators used. The computational results demonstrate the effectiveness of FPBS-QAP and further show the usefulness of using frequent patterns mined from high-quality solutions to guide the search for an effective exploration of the search space.

4) Comparison With More State-of-the-Art Algorithms: We now extend our experimental study by comparing FPBS-QAP with four other recent state-of-the-art QAP algorithms.

1) Parallel hybrid algorithm (PHA) [37] using the MPI libraries was run on a high-performance cluster with 46 nodes (each node with 2 CPUs, 4 cores per CPU, and 16 GB of RAM), a total of 736-GB RAM and a high capacity disk of 6.5 TB configured in a high-performance RAID.
2) Two-stage memory powered great deluge algorithm (TMSGD) [1] was run on a computer (2.1 GHz and 8-GB RAM). The algorithm was run until the number of fitness evaluations reaches $20000*n$ (n is the instance size).
3) Parallel multistart hyper-heuristic algorithm (MSH) [12] was run on the same high performance cluster as the above PHA algorithm.
4) Parallel BLS using OpenMP (BLS-OpenMP) [4], implemented on OpenMP (an API for shared-memory parallel computations that runs on multicore computers), was executed on a personal computer with an Intel Core i7-6700 CPU 3.4 GHZ with 4 cores and 16-GB RAM.

One notices that three of these four recent QAP algorithms are implemented and run on parallel machines and their results have been obtained on different computing platforms, with different stopping conditions. Thus, the comparison shown in this section was provided mainly for indicative purposes. On the other hand, the availability of our FPBS-QAP program makes it possible for researchers to make fair comparisons with FPBS-QAP.

Table IV presents the comparative results between our FPBS-QAP algorithm and the four reference algorithms. Following [1], [4], [12], and [37], we focus on the APD indicator (defined in Section IV-C2) for this study and include, only for indicative purposes, the computation times ($T(m)$), which should be interpreted with caution for the reasons raised above. For completeness, we also include the results of BLS and BMA from Table IV. In the last row of the table, we again indicate the average value of each indicator. Since the results of MSH and BLS-OpenMP, for instances tai150b, wil100, and tho150, are not available, it is not meaningful to include average ranking information of the compared algorithms like in Tables II and III.

TABLE IV
COMPARATIVE PERFORMANCE BETWEEN THE FPBS-QAP ALGORITHM AND THE STATE-OF-THE-ART ALGORITHMS ON HARD INSTANCES IN TERMS OF THE APD VALUE. COMPUTATIONAL TIME IS GIVEN IN MINUTES FOR INDICATIVE PURPOSES

		APD							T(m)						
Instance	BKV	BLS★	BMA★	PHA○	MSH○	BLS-OpenMP○	TMSGD	**FPBS-QAP**	BLS★	BMA★	PHA○	MSH○	BLS-OpenMP○	TMSGD	**FPBS-QAP**
tai40a	3139370	0.012	0.047	0.000	0.261	0.000	0.261	0.037	40.7	28.9	10.6	30.0	32.2	27.8	52.5
tai50a	4938796	0.077	0.091	0.000	0.165	0.000	0.276	0.106	47.0	38.8	12.7	37.5	68.2	41.1	67.8
tai60a	7205962	0.241	0.195	0.000	0.270	0.000	0.448	0.189	73.9	34.7	19.6	45.0	107.9	78.9	60.0
tai80a	13499184	0.510	0.434	0.644	0.530	0.504	0.832	0.467	58.0	69.9	40.0	60.0	236.0	111.3	55.2
tai100a	21052466	0.455	0.378	0.537	0.338	0.617	0.874	0.380	58.4	59.9	71.9	75.0	448.5	138.3	36.1
tai50b	458821517	0.000	0.000	0.000	0.000	0.000	0.005	0.000	0.2	0.1	5.8	3.0	0.7	10.2	0.2
tai60b	608215054	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.3	0.4	9.5	3.2	18.6	33.6	0.4
tai80b	818415043	0.000	0.000	0.000	0.000	0.000	0.025	0.000	3.9	2.0	27.7	4.0	218.1	0.0	1.4
tai100b	1185996137	0.000	0.010	0.000	0.000	0.000	0.028	0.000	6.1	8.1	42.5	5.0	160.8	72.6	2.9
tai150b	498896643	0.109	0.200	0.026	*	*	0.051	0.092	42.7	56.0	177.4	*	*	258.0	46.4
sko72	66256	0.000	0.004	0.000	0.000	0.000	0.007	0.000	3.0	0.7	33.6	3.6	1.8	38.0	4.8
sko81	90998	0.000	0.000	0.000	0.000	0.000	0.019	0.000	10.3	3.2	39.9	4.1	2.4	57.1	3.3
sko90	115534	0.000	0.009	0.000	0.000	0.000	0.031	0.010	19.6	3.6	40.5	4.5	3.3	93.8	2.4
sko100a	152002	0.001	0.002	0.000	0.003	0.000	0.029	0.000	51.0	28.6	41.7	75.0	29.8	153.2	8.5
sko100b	153890	0.000	0.000	0.000	0.004	0.000	0.015	0.000	21.6	11.0	42.3	75.0	8.5	164.3	5.8
sko100c	147862	0.000	0.000	0.000	0.003	0.000	0.013	0.000	22.4	7.1	42.2	75.0	4.3	154.5	8.7
sko100d	149576	0.001	0.000	0.000	0.004	0.000	0.017	0.000	38.5	12.6	41.9	75.0	12.9	148.9	16.2
sko100e	149150	0.000	0.000	0.000	0.000	0.000	0.016	0.000	44.2	5.3	42.5	75.0	4.3	146.1	12.2
sko100f	149036	0.001	0.001	0.000	0.000	0.000	0.013	0.003	40.2	23.7	42.0	75.0	17.1	153.4	4.0
wil100	273038	0.001	0.000	0.000	*	*	0.008	0.000	28.9	6.9	42.0	*	*	155.1	16.4
tho150	8133398	0.049	0.026	0.009	*	*	0.039	0.006	64.2	83.9	177.4	*	*	512.8	57.4
avg.value		0.067	0.069	**0.058**	0.088	0.062	0.143	0.061	32.2	23.1	47.8	40.3	76.4	121.4	**22.0**

★ The results of BLS and BMA were obtained by running the programs on our computer with $t_{max} = 120$ minutes. These results are slightly different from the results reported in [9], [10].
○ PHA, MSH and BLS-OpenMP are parallel algorithms that were run on high-performance platforms under various stopping conditions.

Table IV shows that the average APD value of FPBS-QAP is 0.061%, which is only slightly higher than 0.058% of the parallel PHA algorithm and better than all remaining reference algorithms. If we check the average computation times of the compared algorithms, we see that FPBS-QAP requires the least time to achieve its results, even if three reference algorithms were run on parallel computers. This comparison thus provides additional supporting evidences about the competitiveness of FPBS-QAP compared to the state-of-the-art QAP algorithms in terms of solution quality and computation efficiency.

V. ANALYSIS AND DISCUSSION

In this section, we perform additional experiments to gain understandings of the FPBS algorithm including the rationale and the effectiveness of the pattern-based solution construction.

A. Rationale Behind Pattern-Based Solution Construction

To explain the rationale behind the solution construction technique based on mined frequent patterns, we analyze the structural similarity between high-quality solutions in the population (POP), and the length distribution of the frequent patterns mined from POP. Given two high-quality solutions π^s and π^t, we define their similarity by $sim(\pi^s, \pi^t) = [(|\pi^s \cap \pi^t|)/n]$, where n ($n = |\pi|$) is the length of a feasible solution, and $\pi^s \cap \pi^t$ is the set of common elements shared by π^s and π^t. The larger the similarity between two solutions, the more common elements they share.

As mentioned above, a mined frequent pattern corresponds to a set of identical elements shared by two or more solutions under a given minimum support θ. A frequent pattern can be directly converted to a partial solution, thus we define the length of a pattern p by $\text{len}(p) = (|p|/n)$, where the length of a pattern is the proportion of the number of identical elements over the total number of elements. A longer pattern length indicates thus more shared elements. The solution similarity can be shown to be a special case of the pattern length as follows. Given a population (POP) of high-quality solutions, we suppose p is a frequent pattern mined from POP. When the minimum support $\theta = 2$, the pattern p is simplified as the set of common elements shared by two solutions π^i and π^j, i.e., $p \leftarrow \pi^i \cap \pi^j$. Therefore, the length of pattern p can be computed as $[(|\pi^i \cap \pi^j|)/n]$. The pattern length $len(p)$ is thus reduced to the solution similarity between π^i and π^j, i.e., $\text{len}(p) = [(|\pi^i \cap \pi^j|)/n] = \text{sim}(\pi^i, \pi^j)$. This observation is further confirmed according to the results reported in Fig. 4, where the curve of the maximum solution similarity (left subfigure) is exactly the same as the curve of the maximum length (right subfigure).

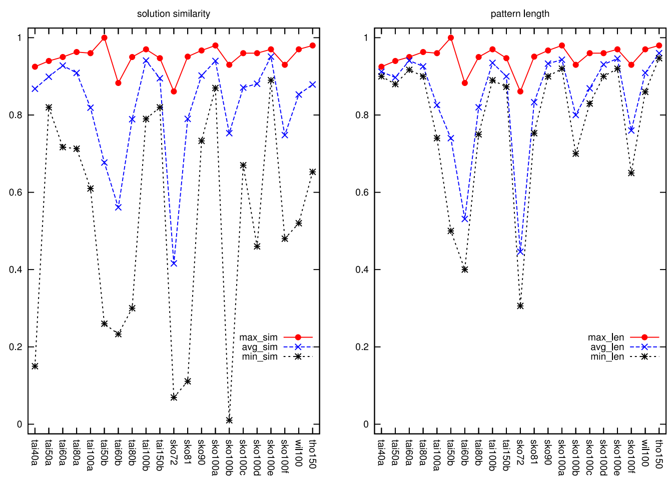

Fig. 4. Solution similarity between high-quality solutions (left subfigure) and length distribution of the mined patterns (right subfigure).

In this experiment, we solved each benchmark instance with $t_{\max} = 30$ min. After each run, we obtain a population of high-quality solutions. To analyze the solution similarity of these high-quality solutions, we calculate the maximum similarity (denoted as max_sim) and minimum similarity (denoted as min_sim) between any two solutions by $max\_sim = \max_{1 \le i < j \le |POP|}\{sim(S^i, S^j)\}$ and $min\_sim = \min_{1 \le i < j \le |\text{POP}|}\{\text{sim}(S^i, S^j)\}$, respectively. We also

TABLE V
COMPARISONS BETWEEN FPBS-QAP$_0$ AND FPBS-QAP UNDER THE TIME LIMIT OF $t_{\max} = 30$ MIN. THE NUMBER OF TIMES THE BKV HAS BEEN REACHED AFTER 10 RUNS IS INDICATED IN PARENTHESES

	FPBS-QAP$_0$ *			**FPBS-QAP**		
Instance	BPD	APD	WPD	BPD	APD	WPD
tai40a	0.000(2)	0.059	0.074	0.000(1)	0.067	0.074
tai50a	0.241(0)	0.318	0.392	0.000(1)	0.279	0.415
tai60a	0.164(0)	0.334	0.486	0.165(0)	0.377	0.469
tai80a	0.446(0)	0.533	0.622	0.430(0)	0.516	0.614
tai100a	0.316(0)	0.466	0.615	0.311(0)	0.402	0.553
tai50b	0.000(10)	0.000	0.000	0.000(10)	0.000	0.000
tai60b	0.000(10)	0.000	0.000	0.000(10)	0.000	0.000
tai80b	0.000(10)	0.000	0.000	0.000(10)	0.000	0.000
tai100b	0.000(8)	0.018	0.100	0.000(6)	0.040	0.100
tai150b	0.000(1)	0.204	0.358	0.000(1)	0.191	0.321
sko72	0.000(9)	0.006	0.063	0.000(10)	0.000	0.000
sko81	0.000(10)	0.000	0.000	0.000(10)	0.000	0.000
sko90	0.000(9)	0.004	0.038	0.000(6)	0.015	0.038
sko100a	0.000(9)	0.002	0.016	0.000(10)	0.000	0.000
sko100b	0.000(8)	0.001	0.004	0.000(10)	0.000	0.000
sko100c	0.000(10)	0.000	0.000	0.000(10)	0.000	0.000
sko100d	0.000(10)	0.000	0.000	0.000(10)	0.000	0.000
sko100e	0.000(10)	0.000	0.000	0.000(7)	0.001	0.004
sko100f	0.000(7)	0.002	0.005	0.000(7)	0.003	0.021
wil100	0.000(9)	0.000	0.002	0.000(10)	0.000	0.000
tho150	0.002(0)	0.022	0.080	0.000(1)	0.051	0.123
avg.value	0.056	0.094	0.136	**0.043**	**0.092**	**0.130**
avg.rank	1.571	**1.500**	1.595	**1.429**	**1.500**	**1.405**

* FPBS-QAP$_0$ can also be treated as a variant of BMA [10] by removing the mutation procedure.

calculate the average similarity (denoted as *avg_sim*) between any two solutions by $avg\_sim = (2/[|POP| * (|POP| - 1)]) \sum_{1 \leq i < j \leq |POP|} \text{sim}(S^i, S^j)$. Then, we calculate the length distribution (i.e., *max_len*, *min_len* and *avg_len*) of a set of |POP| longest frequent patterns mined from POP. The results of the similarity between high-quality solutions and the length distribution of the mined frequent patterns are presented in Fig. 4.

Fig. 4 clearly shows a high similarity between high-quality solutions. Specifically, for all instances, the maximum solution similarity is larger than 0.9. Also, the average solution similarities between any two high-quality solutions are larger than 0.5 except for sko72, for which the average solution similarity is about 0.4. A more significant observation can be derived based on the lengths of the mined patterns showed in the right subfigure. The high structural similarities between the high-quality solutions provide the rationale behind our solution construction based on mined patterns.

B. Effectiveness of Pattern-Based Solution Construction

The frequent pattern-based solution construction method is a good alternative to the general crossover operator in evolutionary algorithms. To demonstrate the effectiveness of the solution construction using frequent patterns, we compared FPBS-QAP with its alternative version FPBS-QAP$_0$ where the frequent pattern-based solution construction is replaced by the standard uniform crossover operator used in the memetic algorithm BMA [10]. Since BMA also uses BLS as its key local optimization procedure and the uniform crossover, FPBS-QAP$_0$ can be treated as a BMA variant. For this experiment, we ran both FPBS-QAP and FPBS-QAP$_0$ on each benchmark instance 10 times with a time limit of $t_{\max} = 30$ min. The comparative results are summarized in Table V.

Table V indicates that FPBS-QAP performs better than FPBS-QAP$_0$. FPBS-QAP achieves a better or equal BPD value on all instances except tai60a. For tai60a, the BPD value of FPBS-QAP is 0.165%, which is only marginally worse than 0.164% for FPBS-QAP$_0$. The average BPD value of FPBS-QAP is also better than that of FPBS-QAP$_0$, i.e., 0.043% < 0.056%. FPBS-QAP also achieves better results in terms of the average APD value and the average WPD value. For the average rank, FPBS-QAP always achieves a better or equal average rank on all three indicators.

Table V shows that both algorithms achieve the same BPD value on 16 out of the 21 instances, and FPBS-QAP outperforms FPBS-QAP$_0$ on 4 out of the 5 remaining instances. Importantly, while FPBS-QAP$_0$ misses the BKVs of two of the hardest instances (tai50a and tho150) (indeed, these BKV can only be reached by few state-of-the-art algorithms), FPBS-QAP manages to hit these values. Given the small number of different BPD values, it is not surprising that there is no significant difference between these two algorithms in terms of the BPD indicator. These observations confirm that the data mining and pattern-based solution construction procedures contribute to the effectiveness of the FPBS-QAP algorithm, in particular for solving difficult instances.

VI. CONCLUSION

We presented the FPBS method, which aims to unify data mining and optimization within the population-based search approach. The method relies on a modular- and component-based approach to enable a wide range of applications, including in particular subset selection and permutation problems. We demonstrated the viability of the proposed FPBS method on the well-known QAP. Extensive computational results on popular QAPLIB benchmarks showed that the resulting FPBS-QAP algorithm performs remarkably well compared to very recent state-of-the-art algorithms.

For future work, two directions can be followed. First, this study focused on exploring maximal frequent itemsets. It is worth studying alternative patterns like sequential patterns and graph patterns. Second, the proposed method intends to be of general purpose, it would be interesting to investigate its application to more combinatorial problems, particularly other permutation problems (e.g., linear ordering and traveling salesman problem) and subset selection problems (e.g., diversity and critical node problems).

APPENDIX
PARAMETER TUNING

To investigate the impact of number of mined patterns m ($m \geq 1$) on the new solutions constructed by the solution construction method, we varied the values of m within a reasonable range and compared their performances. The box and whisker plots shown in Fig. 5 are obtained by considering ten different values $m \in \{1, 3, \ldots, 21\}$. The experiments were conducted on four representative instances selected from different families (tai100a, tai150b, sko100f and tho150). For

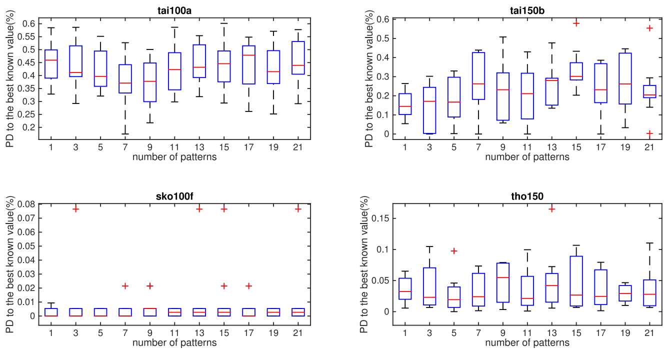

Fig. 5. Impact of the number of patterns m in pattern set. Box and whisker plots corresponding to ten different values of $m \in \{1, 3, \ldots, 21\}$ in terms of the PD from the BKV.

each m value and each instance, we ran the algorithm 10 times with $t_{\max} = 30$ min.

In Fig. 5, X-axis indicates the values for m and Y-axis shows the performance (i.e., the PD from the BKV). We observe small PD fluctuations ranging from 0 to 0.6% and depending on the instances. Indeed, although there are some outliers, the first quartile is very close to the third quartile for each m value. This indicates that even if m influences the performance of the algorithm, this influence is limited and in particular depends on the structure of the problem instances. Generally, our experiments on other parameters of FPBS-QAP led to similar conclusions. That is, they do influence the performance of the algorithm according to the tested instances, yet their impacts are rather limited. As a result, the parameter values given in Table I can be used as the default values of FPBS-QAP.

Acknowledgment

The authors would like to thank the reviewers for their useful comments and suggestions which helped them to considerably improve the work.

References

[1] A. Acan and A. Ünveren, "A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem," *Appl. Soft Comput.*, vol. 36, pp. 185–203, Jul. 2015.
[2] C. C. Aggarwal, M. A. Bhuiyan, and M. A. Hasan, "Frequent pattern mining algorithm algorithms: A survey," in *Frequent Pattern Mining*. Heidelberg, Germany: Springer Int., 2014, pp. 19–64.
[3] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1993, pp. 207–216.
[4] Y. Aksan, T. Dokeroglu, and A. Cosar, "A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem," *Comput. Ind. Eng.*, vol. 103, pp. 105–115, Jul. 2017.
[5] K. Anstreicher, N. Brixius, J.P. Goux, and J. Linderoth, "Solving large quadratic assignment problems on computational grids," *Math. Program.*, vol. 91, no. 3, pp. 563–588, 2002.
[6] S. Asta and E. Özcan, "A tensor-based selection hyper-heuristic for cross-domain heuristic search," *Inf. Sci.*, vol. 299, pp. 412–432, Jun. 2015.
[7] H. Barbalho, I. Rosseti, S. L. Martins, and A. Plastino, "A hybrid data mining GRASP with path-relinking," *Comput. Oper. Res.*, vol. 40, no. 12, pp. 3159–3173, 2013.
[8] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, to be published, doi: 10.1016/j.ejor.2020.07.063.
[9] U. Benlic and J. K. Hao, "Breakout local search for the quadratic assignment problem," *Appl. Math. Comput.*, vol. 219, no. 9, pp. 4800–4815, 2013.
[10] U. Benlic and J. K. Hao, "Memetic search for the quadratic assignment problem," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 584–595, 2015.
[11] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[12] T. Dokeroglu and A. Cosar, "A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem," *Eng. Appl. Artif. Intell.*, vol. 52, pp. 10–25, Jun. 2016.
[13] Z. Drezner, P. M. Hahn, and É. D. Taillard, "Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for metaheuristic methods," *Ann. Oper. Res.*, vol. 139, no. 1, pp. 65–94, 2005.
[14] C. Fleurent and F. Glover, "Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory," *INFORMS J. Comput.*, vol. 11, no. 2, pp. 198–204, 1999.
[15] G. Grahne and J. Zhu, "Efficiently using prefix-trees in mining frequent itemsets," in *Proc. ICDM Workshop Frequent Itemset Min. Implement.*, Dec. 2003, p. 7. [Online]. Available: http://ceur-ws.org/Vol-90/grahne.pdf
[16] M. Guerine, I. Rosseti, and A. Plastino, "Extending the hybridization of metaheuristics with data mining: Dealing with sequences," *Intell. Data Anal.*, vol. 20, no. 5, pp. 1133–1156, 2016.
[17] A. Hottung, S. Tanaka, and K. Tierney, "Deep learning assisted heuristic tree search for the container pre-marshalling problem," *Comput. Oper. Res.*, vol. 113, Sep. 2020, Art. no. 104781.
[18] T. James, C. Rego, and F. Glover, "Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 3, pp. 579–596, May 2009.
[19] Y. Jin and J. K. Hao, "Effective learning-based hybrid search for bandwidth coloring," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 4, pp. 624–635, Oct. 2015.
[20] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evol. Comput.*, vol. 27, no. 1, pp. 3–45, 2019.
[21] J. Lauri, S. Dutta, M. Grassia, and D. Ajwani, "Learning fine-grained search space pruning and heuristics for combinatorial optimization," 2020. [Online]. Available: arXiv:2001.01230.
[22] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 176, no. 2, pp. 657–690, 2007.
[23] D. Martins, G. M. Vianna, I. Rosseti, S. L. Martins, and A. Plastino, "Making a state-of-the-art heuristic faster with data mining," *Ann. Oper. Res.*, vol. 263, no. 1, pp. 141–162, 2018.
[24] S. L. Martins, I. Rosseti, and A. Plastino, "Data mining in stochastic local search," in *Handbook of Heuristics*. Cham, Switzerland: Springer Int., 2016, pp. 1–49.
[25] P. Moscato, "Memetic algorithms: A short introduction," in *New Ideas in Optimization*, D. Corne *et al.*, Eds. London, U.K.: McGraw-Hill, 1999, pp. 219–234.
[26] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Linkage problem, distribution estimation, and Bayesian networks," *Evol. Comput.*, vol. 8, no. 3, pp. 311–340, 2000.
[27] A. Plastino, H. Barbalho, L. F. M. Santos, R. Fuchshuber, and S. L. Martins, "Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic," *J. Heurist.*, vol. 20, no. 1, pp. 39–74, 2014.
[28] D. C. Porumbel, J. K. Hao, and P. Kuntz, "An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring," *Comput. Oper. Res.*, vol. 37, no. 10, pp. 1822–1832, 2010.
[29] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.
[30] P. Rakshit *et al.*, "Realization of an adaptive memetic algorithm using differential evolution and *Q*-learning: A case study in multirobot path planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 814–831, Jul. 2013.
[31] M. Raschip, C. Croitoru, and K. Stoffel, "Using association rules to guide evolutionary search in solving constraint satisfaction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2015, pp. 744–750.
[32] D.S. Reddy, A. Govardhan, and S. Sarma, "Hybridization of neighbourhood search metaheuristic with data mining technique to solve *p*-median problem," *Int. J. Comput. Eng. Res.*, vol. 2, no. 7, pp. 159–163, 2012.

[33] M. H. Ribeiro, V. A. Trindade, A. Plastino, and S. L. Martins, "Hybridization of GRASP metaheuristics with data mining techniques," in *Proc. 1st Int. Workshop Hybrid Metaheurist.*, Valencia, Spain, Aug. 2004, pp. 69–78.

[34] H. G. Santos, L. S. Ochi, E. H. Marinho, and L. M. Drummond, "Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem," *Neurocomputing*, vol. 70, no. 1, pp. 70–77, 2006.

[35] L. F. Santos, S. L. Martins, and A. Plastino, "Applications of the DM-GRASP heuristic: A survey," *Int. Trans. Oper. Res.*, vol. 15, no. 4, pp. 387–416, 2008.

[36] É. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel Computing*, vol. 17, nos. 4–5, pp. 443–455, 1991.

[37] U. Tosun, "On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem," *Eng. Appl. Artif. Intell.*, vol. 39, pp. 267–278, Mar. 2015.

[38] S. Umetani, "Exploiting variable associations to configure efficient local search in large-scale set partitioning problems," in *Proc. 29th Conf. Artif. Intell. (AAAI)*, 2015, pp. 1226–1232.

[39] R. Václavík, A. Novak, P. Sucha, and Z. Hanzálek, "Accelerating the branch-and-price algorithm using machine learning," *Eur. J. Oper. Res.*, vol. 271, no. 3, pp. 1055–1069, 2018.

[40] T. Wauters, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "Boosting metaheuristic search using reinforcement learning," in *Hybrid Metaheuristics*, Berlin, Germany: Springer, 2013, pp. 433–452.

[41] W. Zhang, "Configuration landscape analysis and backbone guided local search part 1: Satisfiability and maximum satisfiability," *Artif. Intell.*, vol. 158, pp. 1–26, Jan. 2004.

[42] Y. Zhou, J. K. Hao, and B. Duval, "Opposition-based memetic search for the maximum diversity problem," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 731–745, Sep. 2017.

[43] Y. Zhou, B. Duval, and J. K. Hao, "Improving probability learning based local search for graph coloring," *Appl. Soft Comput.*, vol. 65, pp. 542–553, Aug. 2018.

[44] Y. Zhou, J. K. Hao, and F. Glover, "Memetic search for identifying critical nodes in sparse graphs," *IEEE Trans. Cybern.*, vol. 49, no. 10, pp. 3699–3712, Apr. 2019.

[45] Y. Wang, Z. Lü, F. Glover, and J. K. Hao, "Backbone guided tabu search for solving the UBQP problem," *J. Heurist.*, vol. 19, no. 4, pp. 679–695, 2013.

Yangming Zhou (Member, IEEE) received the B.Sc. degree in automation from Donghua University, Shanghai, China, in 2011, the M.Sc. degree in control engineering from Zhejiang University, Hangzhou, China, in 2014, and the Ph.D. degree in computer science from the University of Angers, Angers, France, in 2018.

He is currently a Lecturer with the School of Information Science and Engineering, East China University of Science and Technology, Shanghai. His research interests include machine learning and metaheuristic approaches for solving large-scale combinatorial optimization problems.

Jin-Kao Hao received the B.Sc. degree from the School of Computer Science, National University of Defense Technology, Changsha, China, in 1982, the M.Sc. degree in applied informatics from the National Institute of Applied Sciences, Lyon, France, in 1987, the Ph.D. degree in computer science from the University of Franche-Comté, Besançon, France, in 1991, and the Professorship Diploma HDR (Habilitation Diriger des Recherches) degree from the University of Science and Technology of Montpellier, Montpellier, France, in 1998.

He is a Full Professor of Computer Science with the University of Angers, Angers, France, and a Senior Fellow of the Institut Universitaire de France, Paris, France. He has authored or coauthored over 250 peer-reviewed publications and coedited nine books in Springer's LNCS series. His research interests include design of effective algorithms and intelligent computational methods for solving large-scale combinatorial search problems, bioinformatics, data science, telecommunication networks, and transportation.

Béatrice Duval received the B.Sc. degree in mathematics and the M.Sc. degree in computer science from the Ecole Normale Supérieur de Fontenay-aux-Roses, Paris, France, in 1984, and the Ph.D. degree in computer science from the University of Paris-Sud, Paris, in 1991.

She is currently a Professor with the Department of Computer Science, University of Angers, Angers, France. Her main research field concerns data mining and machine learning. For some years, she has been working on applications in bioinformatics with hybrid methods combining machine learning techniques and metaheuristics.