An Accelerated-Limit-Crossing-Based Multilevel Algorithm for the p-Median Problem

Zhilei Ren, He Jiang, *Member, IEEE*, Jifeng Xuan, and Zhongxuan Luo

***Abstract*—In this paper, we investigate how to design an efficient heuristic algorithm under the guideline of the backbone and the fat, in the context of the p-median problem. Given a problem instance, the backbone variables are defined as the variables shared by all optimal solutions, and the fat variables are defined as the variables that are absent from every optimal solution. Identification of the backbone (fat) variables is essential for the heuristic algorithms exploiting such structures. Since the existing exact identification method, i.e., limit crossing (LC), is time consuming and sensitive to the upper bounds, it is hard to incorporate LC into heuristic algorithm design. In this paper, we develop the accelerated-LC (ALC)-based multilevel algorithm (ALCMA). In contrast to LC which repeatedly runs the time-consuming Lagrangian relaxation (LR) procedure, ALC is introduced in ALCMA such that LR is performed only once, and every backbone (fat) variable can be determined in $\mathcal{O}(1)$ time. Meanwhile, the upper bound sensitivity is eliminated by a dynamic pseudo upper bound mechanism. By combining ALC with the pseudo upper bound, ALCMA can efficiently find high-quality solutions within a series of reduced search spaces. Extensive empirical results demonstrate that ALCMA outperforms existing heuristic algorithms in terms of the average solution quality.**

***Index Terms*—Accelerated limit crossing (ALC), backbone, configuration landscape, fat, multilevel, p-median problem.**

I. Introduction

GIVEN a set of users and potential facilities, the goal of the p-median problem [27], [32] is to select a predetermined number of facilities as the medians so as to minimize the total distance that each user must traverse to reach its nearest median. Due to its numerous real-world applications (e.g., plant location allocation [8], network design [19], [40], [41], sensor deployment [13], and data mining [3]), the p-median problem has attracted much research attention in combinatorial optimization to date. Since it has been shown to be *NP-hard* [18], many heuristic algorithms have been proposed in the literature to achieve near optimal solutions in reasonable running time, including variable neighborhood search (VNS) [15], genetic algorithm (GA) [17], tabu search (TS) [35], scatter search (SS) [10], ant colony optimization (ACO) [23], simulated annealing (SA) [28], and population-based hybrid search (PBS) [30]. Among these published algorithms, PBS presents the state-of-the-art results [30].

In recent years, a lot of efforts have been focused on solving problems under the guideline of the backbone and the fat. Given a problem instance, the backbone (fat) consists of those variables that appear in all (none) of optimal solutions. According to their definitions, once the backbone and the fat are properly set, the problem solving process can be greatly improved over a reduced search space. Motivated by such an idea, many new heuristic algorithms have been presented for those traditional combinatorial optimization problems, including the traveling salesman problem (TSP) [52], the quadratic assignment problem (QAP) [25], and the Maximum SATisfiability problem (Max SAT) [51]. Since it is usually intractable to directly achieve the backbone (fat) variables, these new algorithms usually approximate them with the common (absent) parts of some local optima[1] based on the observation that many local optima cluster around optimal solutions to form a "big valley" structure [2], [24], [49], [50]. However, there are some potential drawbacks lying in these "big-valley"-based pseudo backbones (fats). First, the "big valley" structure varies sharply from problem to problem. For example, in the context of TSP, a local optimum to ATT532 (which is a well-known instance from TSPLIB) shares 70%–80% edges with an optimal tour [2]. In [25], the authors observe that, for QAP, a local optimum to selected instances (e.g., Chr25a from QAPLIB) has a smaller portion (around 20%) of common variables with an optimal solution. Furthermore, this structure may even vary over different instances of the same problem. In [26], the authors state that the "big valley" structure (measured by the fitness-distance correlation) varies greatly from instance to instance, in the context of QAP. Second, as stated in [52], the pseudo backbone (fat) must be constructed from "unbiased samples" of local optima, which is still a great challenge.

We address the backbone and the fat in the context of the p-median problem. In contrast to the "big valley" structure-based approaches, we extract the backbone and the fat with limit crossing (LC), proposed by Climer and Zhang [5] in solving TSP. The key idea of LC stems from the fact that the exclusion (inclusion) of a backbone (fat) variable must result

Manuscript received June 28, 2011; revised November 26, 2011; accepted January 22, 2012. This work was supported in part by the National Natural Science Foundation of China under Grants 61175062, 60805024, and 61033012, and in part by the "Software + X" funding of Dalian University of Technology. This paper was recommended by Associate Editor H. Takagi.

Z. Ren and Z. Luo are with the School of Mathematical Sciences, Dalian University of Technology, Dalian 116621, China (e-mail: ren@mail.dlut.edu.cn; zxluo@dlut.edu.cn).

H. Jiang and J. Xuan are with the School of Software, Dalian University of Technology, Dalian 116621, China (e-mail: hejiang@ieee.org; xuan@mail.dlut.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2188100

[1]Given the neighborhood definition, local optima refer to those solutions that have lower or equal (suppose that the problem to be solved is a minimization problem) objective cost than all their neighbors [16].

in an increase of the optimal cost (i.e., the objective cost of optimal solutions) for a minimization problem instance. Since it is usually intractable to retrieve optimal solutions, LC [5] introduces a relaxed method instead by employing an upper bound and a lower bound. If the lower bound of the modified instance exceeds the upper bound of the original instance, the variable can be detected as a backbone (fat) variable.

In this paper, we focus on the algorithm design that exploits the structural information in the form of backbone and fat. First, we propose LC in the context of the p-median problem. We show by some preliminary experiments that LC is time consuming and heavily depends upon the tightness of both the lower bound and the upper bound. Second, we develop an accelerated LC (ALC). Compared with LC which repeatedly runs the time-consuming procedure of Lagrangian relaxation (LR), ALC significantly saves the overall running time in such a way that LR only needs to be executed once. Third, we propose the ALC-based multilevel algorithm (ALCMA). To eliminate the dependence on both the lower bound and the upper bound, a pseudo upper bound is dynamically maintained in ALCMA. With ALC and the pseudo upper bound, ALCMA can easily retrieve some pseudo backbone (fat) variables and search efficiently within those search spaces restricted by fixing (excluding) these pseudo backbone (fat) variables. Extensive experiments over various benchmark instances (ORLIB, SL, GR, RW, and TSPLIB) demonstrate the performance of ALCMA. Over the total of 171 benchmark instances, ALCMA can achieve 52 new best solutions and locate 92 of the currently best known solutions. Moreover, statistical tests confirm that, over these test instances, the overall performance of ALCMA is better than that of the state-of-the-art algorithms in terms of the average solution quality. Finally, the configuration landscape analysis is employed to investigate both the strength and the weakness of the framework.

The rest of this paper is organized as follows. Section II presents the related work of both the backbone (fat) and the p-median problem. Then, in Section III, we present LC and ALC. In Section IV, the new algorithm ALCMA is proposed. Experimental results are given in Section V. Section VI briefly concludes this paper.

II. Related Work

A. Backbone and Fat

The concept of the backbone is first proposed by Parkes [29] when studying the hardness of problem solving for the SATisfiability problem (SAT). Parkes [29] observes that the number of backbone variables is highly correlated with the hardness of problem instances. In [5], Climer and Zhang propose the concept of the fat and develop the method of LC to extract some backbone (fat) variables.

In the literature, there has been some research about the computational complexity of searching for the backbone variables. Kilby *et al.* [21] prove that there is no polynomial time algorithm to obtain all the backbone variables of TSP under the assumption that $P \neq NP$. Similar results have been presented for SAT [20] and QAP [25] as well. Hence, many researchers achieve the pseudo backbone (fat) based on the "big valley" structure. This structure stems from the empirical observation by Boese [2] in TSP that a local optimal tour shares numerous common edges with an optimal tour. Afterward, similar characteristics have also been found in many other problems, e.g., QAP [25] and Max SAT [51].

With these "big-valley"-based pseudo backbone (fat) variables, a few heuristic algorithms have been presented. These algorithms intend to shrink the search space by fixing (excluding) the pseudo backbone (fat) variables and explore the reduced search space with some existing heuristic algorithms. For example, Cook and Seymour [7] develop a tour merging approach for TSP, in which a new search space is constructed by merging the variables that appear in a set of local optima. Jiang *et al.* [25] propose a pseudo backbone guided algorithm for QAP. Some other relevant algorithms include those in [42] and [43].

In contrast to the "big valley" structure, LC is a strategy proposed by Climer and Zhang [5] for detecting the backbone (fat) variables in TSP. Theoretically, an edge must belong to the backbone if removing it from a TSP instance leads to the increase of the optimal cost. Since it is usually intractable to retrieve optimal solutions, a relaxed method is introduced in LC by employing an upper bound and a lower bound. When the lower bound for the modified instance exceeds the upper bound for the original instance, the edge can be claimed to be a backbone edge. The fat can be determined in a similar way.

B. p-Median Problem

In this section, we first introduce the formal definition of the p-median problem, as well as the backbone and the fat to a given p-median instance. Then, existing heuristics for the p-median problem are briefly reviewed.

Definition 1: Given a set $F = \{1, 2, \ldots, m\}$ of m potential facilities, a set $U = \{1, 2, \ldots, n\}$ of n users, a matrix $D = (d_{ij})_{n\times m}$, where d_{ij} represents the distance between the user i and the facility j for all $i \in U$ and $j \in F$, as well as a predefined $p < m$, the p-median instance is denoted as $PMP(F, U, D, p)$. A solution to $PMP(F, U, D, p)$ is a subset $J \subset F$, $|J| = p$, whose cost is defined as $C_J(F, U, D, p) = \sum_{i\in U} \min_{j\in J} d_{ij}$. The p-median problem aims to find a solution J^* that minimizes the cost, i.e., $C_{J^*}(F, U, D, p) = \min_{J\in\Pi} C_J(F, U, D, p)$, where Π is the set of all the solutions.

Definition 2: Given a p-median instance $PMP(F, U, D, p)$, let $\Pi^* = \{J_1^*, J_2^*, \ldots, J_q^*\}$ be the set of all the optimal solutions to $PMP(F, U, D, p)$, where $q = |\Pi^*|$ represents the number of optimal solutions. The *backbone* of the p-median instance $PMP(F, U, D, p)$ is defined as $bone(F, U, D, p) = \bigcap_{1\leq i\leq q} J_i^*$, and the *fat* is similarly defined as $fat(F, U, D, p) = F \backslash \bigcup_{1\leq i\leq q} J_i^*$.

Many heuristic algorithms have been developed for the p-median problem. For example, there are VNS [15], hybrid heuristic [33], etc. Moreover, a lot of nature-inspired metaheuristics are also proposed, such as GA [17], TS [35], SS [10], ACO [23], SA [28], and PBS [30].

Although the concepts of the backbone and the fat have not been cited in the context of the p-median problem, Rosing *et al.* develop a series of heuristic algorithms (i.e., HC and its variants [36]–[39]) under a similar idea to the "big valley" structure. These algorithms consist of two stages. During the first stage (denoted as the HC-S1 reduction), a set of local optima is randomly sampled using the interchange operator [47], and a concentration set is constructed by merging all the medians appearing in these local optima. Then, during the second stage (denoted as the HC-S2 solving), the search is conducted over the concentration set either with some exact algorithms [37] or with a local search operator (namely, the 2-opt operator [36]). Furthermore, the medians that appear in all the sampled local optima are fixed and held unchanged during the second stage.

III. Extracting Backbone and Fat Variables

A. LC

In this section, we first validate the scheme of LC for the p-median problem and propose the framework of LC.

Definition 3: Given a p-median instance $PMP(F,U,D,p)$ and a facility set $K \subseteq F$, let $S_F(K) = \{J : J \in \Pi, K \subseteq J\}$ be the set of solutions, including K, and let ${}_{\neg}S_F(K) = \{J : J \in \Pi, K \cap J = \varnothing\}$ be the set of solutions containing no facility in K, where Π is the set of all the solutions to $PMP(F,U,D,p)$. Given a facility $f \in F$, let $C_{\text{opt}}(\neg f) = \min_{J \in {}_{\neg}S_F(\{f\})} C_J(F,U,D,p)$, $C_{\text{opt}}(f) = \min_{J \in S_F(\{f\})} C_J(F,U,D,p)$, and $C_{\text{opt}}(\Pi) = \min_{J \in \Pi} C_J(F,U,D,p)$.

Proposition 1: Given a p-median instance $PMP(F,U,D,p)$ and a facility $f \in F$, the facility f must be a backbone variable if $LB(\neg f) > UB_\Pi$, where $LB(\neg f)$ is a lower bound of $C_{\text{opt}}(\neg f)$ and UB_Π is an upper bound of $C_{\text{opt}}(\Pi)$.

Proof: Otherwise, there must exist an optimal solution $J^* \in \Pi$ such that $f \notin J^*$. Hence, we have that $J^* \in {}_{\neg}S_F(\{f\})$ which implies that $C_{\text{opt}}(\neg f) = C_{J^*}(F,U,D,p)$. According to the assumption $LB(\neg f) \leq C_{\text{opt}}(\neg f)$, it can be inferred that $LB(\neg f) \leq C_{J^*}(F,U,D,p) = C_{\text{opt}}(\Pi) \leq UB_\Pi$ holds. It contradicts with the assumption that $LB(\neg f) > UB_\Pi$. This proposition is proved. ■

Similarly, we can easily verify the following proposition.

Proposition 2: Given a p-median instance $PMP(F,U,D,p)$ and a facility $f \in F$, the facility f must be a fat variable if $LB(f) > UB_\Pi$, where $LB(f)$ is a lower bound of $C_{\text{opt}}(f)$ and UB_Π is an upper bound of $C_{\text{opt}}(\Pi)$.

The framework of LC is presented in Algorithm 1. Given a p-median instance, every facility is checked whether it is a backbone (fat) variable with respect to Propositions 1 and 2.

Algorithm 1: Limit Crossing (LC)

```
Input: PMP(F, U, D, p), heuristic algorithm M
Output: bone and fat
begin
    bone = fat = ∅;
    Calculate UB_Π by calling M;
    for every facility f ∈ F do
        Calculate LB(¬f) ;
        if LB(¬f) > UB_Π then
            bone = bone ∪ {f};
        else
            Calculate LB(f);
            if LB(f) > UB_Π then
                fat = fat ∪ {f};
    return bone and fat;
end
```

B. Upper Bound and Lower Bound

As shown in Algorithm 1, both the upper bound and the lower bound are required to check whether a facility is a backbone (fat) variable. Obviously, the cost of any solution to the p-median instance can be taken as the upper bound. Therefore, we can simply use an existing heuristic algorithm to find an upper bound. On the other hand, the lower bound should be calculated for every facility. In this paper, we employ an LR method to calculate the lower bounds [i.e., $LB(f)$ and $LB(\neg f)$]. In this section, we first show the LR method to calculate a lower bound LB_Π of $C_{\text{opt}}(\Pi)$ and then demonstrate how to adjust it for computing $LB(f)$ and $LB(\neg f)$.

The p-median problem can be modeled as the following integer programming formulations [44]:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij} \tag{1}$$

s.t.

$$\sum_{j=1}^{m} x_{ij} = 1, \qquad i \in \{1,2,\ldots,n\} \tag{2}$$

$$\sum_{j=1}^{m} y_j = p \tag{3}$$

$$x_{ij} \leq y_j, \qquad i \in \{1,2,\ldots,n\},\ j \in \{1,2,\ldots,m\} \tag{4}$$

$$x_{ij}, y_j \in \{0,1\}, \qquad i \in \{1,2,\ldots,n\},\ j \in \{1,2,\ldots,m\}. \tag{5}$$

In the aforementioned formulations, a solution consists of an allocation matrix $X_{n\times m}$ and a vector Y_m indicating which facilities are chosen, where $x_{ij} = 1$ means that the user i is assigned to the facility j, and $x_{ij} = 0$ otherwise; $y_j = 1$ indicates that the facility j is selected as a median, and $y_j = 0$ otherwise. Constraints (2) and (4) ensure that each user is allocated to only one median. Constraint (3) guarantees that exactly p facilities are chosen as the medians, and constraint (5) presents the integer conditions.

For the aforementioned formulations, the lower bound $LB(\Pi)$ can be achieved by solving the relaxed version. In the literature, several LR methods [1], [6], [44] have been developed for relaxing a p-median instance. In this paper, we adopt the LR method proposed by Seene and Lorena [44] to calculate the lower bound. In [44], the formulations of the p-median problem are relaxed by removing constraint (2) and, meanwhile, introducing a penalty for violating the removed

constraint; thus, the problem can be transformed into a relaxed form as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij} + t\sum_{i=1}^{n}\lambda_i\left(1-\sum_{j=1}^{m}x_{ij}\right) = \min \sum_{i=1}^{n}\sum_{j=1}^{m}(d_{ij}-t\lambda_i)x_{ij} + t\sum_{i=1}^{n}\lambda_i \quad (6)$$

s.t. constraints (3)–(5).

Given t and λ, the second term of the function (6) is a constant, and the first term can be easily solved by decomposing it into n subproblems

$$\min \sum_{j=1}^{m}(d_{ij}-t\lambda_i)x_{ij}, \qquad i \in \{1,2,\ldots,n\} \quad (7)$$

s.t. constraints (3)–(5).

As shown in [44], each subproblem can be solved by a simple enumeration as follows. For each facility j, an auxiliary cost is defined as $\beta_j = \sum_{i=1}^{n}\min(0, d_{ij}-t\lambda_i)$. All these auxiliary costs are sorted in an ascending order: $\beta_{k_1} \le \beta_{k_2} \le \cdots \le \beta_{k_m}$. Let the index set $I = \{k_1, k_2, \ldots, k_p\}$. Then, the entries of Y are set with respect to the following equation, i.e., the facilities $k_1, k_2, \ldots, k_p$ are chosen as the medians:

$$y_j = \begin{cases} 1, & j \in I \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

Accordingly, the allocation variables of X are set to

$$x_{ij} = \begin{cases} 1, & y_j = 1, d_{ij} - t\lambda_i < 0 \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

For any t and λ, [44] shows that the lower bound LB_Π can be calculated by

$$LB_\Pi = \sum_{j=1}^{m}\beta_j y_j + t\sum_{i=1}^{n}\lambda_i = \sum_{j=1}^{p}\beta_{k_j} + t\sum_{i=1}^{n}\lambda_i. \quad (10)$$

The framework of LR is presented in Algorithm 2. Following [44], the parameters t and λ can be improved using the surrogate heuristic (see [44] for detail), and the stopping criterion is set to terminating the procedure if no better lower bound can be achieved within 100 iterations.

We can incorporate LR into the framework of LC as follows. When calculating $LB(\neg f)$, $y_f = 0$ should be introduced as an additional constraint, i.e., at each iteration of LR, the p smallest entries of β, except β_f, are selected. Thus, LR can be employed to calculate $LB(\neg f)$ by replacing (10) and LB_Π in Algorithm 2 with (11) and $LB(\neg f)$, respectively.

$$LB(\neg f) = \begin{cases} \sum_{j=1}^{p}\beta_{k_j} + \beta_{k_{p+1}} - \beta_f + t\sum_{i=1}^{n}\lambda_i, & f \in I \\ \sum_{j=1}^{p}\beta_{k_j} + t\sum_{i=1}^{n}\lambda_i, & f \notin I. \end{cases} \quad (11)$$

Similarly, for $LB(f)$, $y_f = 1$ should be introduced as an additional constraint. Then, (10) and LB_Π in Algorithm 2 should be replaced with (12) and $LB(f)$, respectively, for calculating $LB(f)$.

$$LB(f) = \begin{cases} \sum_{j=1}^{p}\beta_{k_j} + t\sum_{i=1}^{n}\lambda_i, & f \in I \\ \sum_{j=1}^{p-1}\beta_{k_j} + \beta_f + t\sum_{i=1}^{n}\lambda_i, & f \notin I. \end{cases} \quad (12)$$

With the lower bounds calculated by LR, together with an upper bound provided by any solution, we can apply LC to identify the backbone (fat) variables in polynomial running time. However, one potential drawback of this approach is that LR has to be run at least once for every facility. Therefore, this approach might be very time consuming.

Algorithm 2: Lagrangian Relaxation (LR)

Input: $PMP(F, U, D, p)$
Output: LB_Π
begin
  $LB^*_\Pi = -\infty$;
  $t = 0$;
  for $i = 1$ *to* n **do**
    $\lambda_i = \min_{j \in F} d_{ij}$;
  while *Stopping criterion not met* **do**
    Update t and λ with respect to [44];
    for $j = 1$ *to* m **do**
      $\beta_j = \sum_{i=1}^{n}\min(0, d_{ij} - t\lambda_i)$;
    Sort the β vector such that $\beta_{k_1} \le \beta_{k_2} \le \ldots \le \beta_{k_m}$;
    $I = \{k_1, k_2, \ldots, k_p\}$;
    Calculate lower bound LB_Π with Equation (10);
    if $LB_\Pi > LB^*_\Pi$ **then**
      $LB^*_\Pi = LB_\Pi$;
      $t^* = t$, $\lambda^* = \lambda$, $\beta^* = \beta$, $I^* = I$;
  $LB_\Pi = LB^*_\Pi$;
  return LB_Π;
end

Algorithm 3: Simplified Lagrangian Relaxation (SLR)

Input: $PMP(F, U, D, p)$, t, and λ
Output: $LB(\neg f)$
begin
  for $j = 1$ *to* m **do**
    $\beta_j = \sum_{i=1}^{n}\min(0, d_{ij} - t\lambda_i)$;
  Sort the β vector such that $\beta_{k_1} \le \beta_{k_2} \le \ldots \le \beta_{k_m}$;
  $I = \{k_1, k_2, \ldots, k_p\}$;
  Calculate the lower bound $LB(\neg f)$ with Equation (11);
  return $LB(\neg f)$;
end

C. ALC

In the framework of LR, it can be observed that, given feasible values of t and λ, the lower bounds $LB(\neg f)$ and $LB(f)$ for each facility f can be calculated with (11) and (12) efficiently. Consequently, extensively much running time is spent on iteratively improving the parameters t and λ since LR has to be performed for each facility from scratch. Thus, given any facility $f \in F$, we can greatly speed up the procedure of computing $LB(\neg f)$ if t and λ required in (11) and (12) are precalculated. As a solution, instead of running LR for each facility to identify the backbone (fat) variables, we perform LR [with (10)] only once and pass the values of t^* and λ^* obtained by LR to the simplified LR algorithm (SLR).

When SLR takes in t^* and λ^*, which are achieved in LR for calculating LB_Π, the sorted entries of β in SLR must be identical to that of β^* achieved in LR. Therefore, SLR for calculating $LB(\neg f)$ can be further simplified to directly apply

Algorithm 4: Accelerated Limit Crossing (ALC)

Input: $PMP(F, U, D, p)$, heuristic algorithm M
Output: $bone$ and fat
begin
  $bone = fat = \varnothing$;
  Calculate UB_Π by calling M;
  Achieve LB_Π, t^*, λ^*, β^*, and I^* by calling LR;
  for *every facility* $f \in F$ **do**
    if $f \in I^*$ **then**
      $LB(\neg f) = LB_\Pi - \beta^*_f + \beta^*_{k_{p+1}}$;
      if $LB(\neg f) > UB_\Pi$ **then**
        $bone = bone \cup \{f\}$;
    else
      $LB(f) = LB_\Pi - \beta^*_{k_p} + \beta^*_f$;
      if $LB(f) > UB_\Pi$ **then**
        $fat = fat \cup \{f\}$;
  return $bone$ and fat;
end

(11) to those sorted entries of β^*. In a similar way, we can compute $LB(f)$ by applying (12) to β^*. Under the earlier discussion, we present the framework of ALC in Algorithm 4. First, an existing heuristic algorithm is called to retrieve an upper bound UB_Π. Then, we obtain those sorted entries of β^* by calling LR to find the lower bound LB_Π. In the remaining part of ALC, every facility is checked whether it is a backbone (fat) variable. In ALC, we use SLR instead of LR to calculate $LB(\neg f)$. As we have discussed, SLR will be further simplified to directly employ (11) to those sorted entries of β^* in LR. In a similar way, we can also apply (12) to those entries of β^* for computing $LB(f)$. In ALC, all facilities are divided into two categories such that a lot of running time can be further saved as follows.

For facility $f \in I^*$, we can easily infer from (11) that $LB(\neg f) = LB_\Pi - \beta^*_f + \beta^*_{k_{p+1}}$. Meanwhile, we have that $LB(f) = LB_\Pi$ as well. Note that $LB_\Pi < UB_\Pi$, so it makes no sense to compare $LB(f)$ with UB_Π for determining whether f belongs to the fat variable or not. Therefore, we only check whether this facility is a backbone variable in the **for** loop of ALC.

For facility $f \notin I^*$, we can draw from (12) that $LB(f) = LB_\Pi - \beta^*_{k_p} + \beta^*_f$. Meanwhile, we can have that $LB(\neg f) = LB_\Pi$. Hence, this facility cannot be detected as a backbone variable in ALC, and we only compare $LB(f)$ with UB_Π to check whether this facility belongs to the fat.

In contrast to LC, the advantage of ALC lies in that the time-consuming procedure of LR only needs to be executed once. For every facility, its lower bound can be easily returned in $\mathcal{O}(1)$ running time. However, a potential risk of ALC is that the lower bounds for those facilities may not be tight enough since t and λ have not been well tuned for every facility.

To evaluate the influences of the lower bounds (i.e., $LB(\neg f)$ and $LB(f)$ for each facility f) and the upper bound [i.e., $UB(\Pi)$], some experiments are conducted over several typical benchmark instances from ORLIB (pmed15 and pmed40), TSPLIB (PCB3038 with $p = 400$ and RL5934 with $p = 600$), and RW (RW1000 with $p = 100$). We conduct two sets of experiments. First, we intend to test how the acceleration strategy of ALC affects the number of returned backbone (fat) variables. Tables I and II present the results of LC and ALC.

TABLE I
LC/ALC WITH UPPER BOUNDS PROVIDED BY BEST KNOWN SOLUTIONS

Instance type	m	p	LC bone/p (%)	LC fat/(m-p) (%)	LC time (s)	ALC bone/p (%)	ALC fat/(m-p) (%)	ALC time (s)
pmed15	300	100	78.00	89.00	8.28	35.00	61.00	0.15
pmed40	900	90	94.44	99.26	77.58	5.56	79.88	0.63
PCB3038	3038	400	40.50	85.71	18412.65	0.00	21.08	4.49
RL5934	5934	600	4.50	71.73	278551.96	0.00	6.36	29.79
RW1000	1000	100	0.00	0.00	1730.85	0.00	0.00	0.80

TABLE II
LC/ALC WITH UPPER BOUNDS PROVIDED BY LOCAL OPTIMA

Instance type	m	p	LC bone/p (%)	LC fat/(m-p) (%)	LC time(s) (s)	ALC bone/p (%)	ALC fat/(m-p) (%)	ALC time (s)
pmed15	300	100	78.00	89.00	9.61	8.00	0.50	0.16
pmed40	900	90	15.56	90.12	132.07	0.00	28.77	0.62
PCB3038	3038	400	0.00	0.00	29323.08	0.00	0.00	4.50
RL5934	5934	600	0.00	0.00	408637.34	0.00	0.00	29.82
RW1000	1000	100	0.00	0.00	1859.53	0.00	0.00	0.90

In these tables, columns 1–3 specify the problem instances. Columns 4–6 present the results of LC, including the normalized size of the backbone (fat) obtained, as well as the time elapsed in seconds. The results of ALC are presented in columns 7–9. For all the results in Table I, the upper bound of each instance is provided by the best known solution achieved by the state-of-the-art algorithm (see Section V-D). In Table II, the upper bound of each instance is provided with a local optimum achieved by applying the interchange operator [47] over a random initial solution. The following observation can be found from Tables I and II.

On the one hand, comparing columns 4–6 with columns 7–9 in Tables I and II, respectively, we can find that ALC runs far faster than LC at the cost of missing a part of the backbone (fat) variables. For example, in Table I, 99.26% of the fat variables of pmed40 are found by LC in 77.58 s, whereas 79.88% of the fat variables can be returned by ALC in 0.63 s. The reason is that ALC relaxes the tightness of the lower bounds, which makes it less possible for the LC phenomenon to occur.

On the other hand, the upper bound also plays an important role. The comparison of the results of LC and ALC in Table I with those in Table II indicates that, with the upper bounds changed from the costs of the best known solutions to those of randomly sampled local optima, the number of the backbone (fat) variables obtained by both LC and ALC drops dramatically. With the upper bounds provided with local optima, e.g., for PCB3038 and RL5934, no backbone (fat) variable can be identified. Another interesting observation can be drawn over the RW instance. Over this instance, even with the best known objective cost as the upper bound, no backbone (fat) can be detected, which may imply that it is hard to extract structural information from RW instances, which are randomly generated (see [33] for the detailed description of RW instances).

Further insight can be gained by evaluating the influence of gradually increased upper bounds. Fig. 1 shows the behaviors of both LC and ALC with various upper bounds. The experiment is conducted as follows. Over each instance, the upper bound starts from the optimal cost and increases linearly; both LC and ALC are performed with such an upper bound. In Fig. 1,

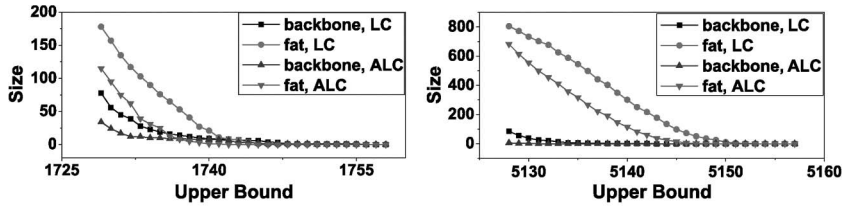

Fig. 1. Upper bound influence over ORLIB instances. (a) pmed15. (b) pmed40.

the number of the backbone (fat) variables is plotted against the upper bound. Due to the computation complexity, this experiment is only conducted over ORLIB instances. Some interesting phenomena can be concluded from Fig. 1. First, the curves of both LC and ALC follow a similar tendency that the number of the backbone (fat) variables obtained sharply decreases along with the growth of the upper bound. Second, LC can achieve more backbone (fat) variables than ALC when the same upper bound is employed. This phenomenon coincides with the conclusion drawn from Tables I and II that ALC may miss part of the backbone (fat).

In summary, in this section, we propose LC and ALC in search of the backbone and the fat variables. However, empirical results demonstrate that the extraction of these variables is difficult, which is sensitive to both the lower bound and the upper bound. Thus, in the following section, we intend to exploit the idea of LC to guide the search procedure and, meanwhile, prevent the search from being sensitive to the lower bound and the upper bound.

IV. ALCMA

In this section, we propose the new algorithm ALCMA. The key idea of ALCMA is to employ ALC to achieve some (pseudo) backbone (fat) variables and transform the search space into a series of reduced ones so as to efficiently improve the solutions. It should be noted that, since ALC is called in ALCMA for multiple times and some steps in ALC need to be executed only once, ALC is implicitly embedded in ALCMA.

As discussed in Section III, LR has to be performed at least once in LC for each facility so as to identify the backbone (fat) variables. The high complexity of LR makes LC impractical to be incorporated into a heuristic algorithm for problem solving. In contrast to LC, ALC can speed the procedure up, although some backbone (fat) variables may be missed. Moreover, ALC is still sensitive to the tightness of the upper bound. In ALCMA, we intend to retain the fast characteristic of ALC and, meanwhile, prevent the algorithm from being sensitive to the upper bound. Therefore, a pseudo upper bound is introduced and dynamically adjusted in ALCMA for returning a part of the backbone (fat) variables in short time.

Given a lower bound LB_Π and an upper bound UB_Π, there always exists a factor α^* such that $C_{\text{opt}}(\Pi) = LB_\Pi + \alpha^*(UB_\Pi - LB_\Pi)$. Therefore, in ALC, every facility $f \in I^*$ satisfying $\beta_f^* < \beta_{k_{p+1}}^* - \alpha^*(UB_\Pi - LB_\Pi)$ can be identified as a backbone variable, and facility $f \notin I^*$ for which $\beta_f^* > \beta_{k_p}^* + \alpha^*(UB_\Pi - LB_\Pi)$ holds can be identified as a fat variable. However, α^* is instance dependent and cannot be obtained in advance. As a tradeoff, with α^* replaced with a pseudo factor α, the pseudo upper bound in ALCMA is defined as $pseudo\_UB_\Pi = LB_\Pi + \alpha(UB_\Pi - LB_\Pi)$. Furthermore, we define the pseudo backbone (fat) as follows:

$$pseudo\_bone(\alpha) = \left\{ f : \beta_f^* < \beta_{k_{p+1}}^* - \alpha(UB_\Pi - LB_\Pi), f \in I^* \right\} \quad (13)$$

$$pseudo\_fat(\alpha) = \left\{ f : \beta_f^* > \beta_{k_p}^* + \alpha(UB_\Pi - LB_\Pi), f \notin I^* \right\}. \quad (14)$$

By replacing the backbone (fat) identification criteria in Algorithm 4 with criteria (13) and (14), ALC can be easily adjusted to detect the pseudo backbone (fat) variables. However, some nonbackbone (nonfat) variables may be falsely detected as the backbone (fat) variables if the factor α is too small (e.g., $\alpha \ll \alpha^*$). Meanwhile, some backbone (fat) variables cannot be included in $pseudo\_bone(\alpha)$ ($pseudo\_fat(\alpha)$) if $\alpha > \alpha^*$. Since α^* is usually unknown before the given p-median instance is exactly solved, it is extremely hard to set a static value for α. Therefore, we dynamically adjust the value of α in ALCMA. At the beginning of ALCMA, α is initialized with a very small positive value such that a large scale of $pseudo\_bone(\alpha)$ ($pseudo\_fat(\alpha)$) can be easily determined. By fixing (trimming out) those pseudo backbone (fat) variables, some high-quality solutions can be searched in the reduced search space. Then, the upper bound can be updated accordingly with the newly found solutions. Along with the progress of ALCMA, the factor α gradually grows until $pseudo\_bone(\alpha)$ ($pseudo\_fat(\alpha)$) does not contain any nonbackbone (nonfat) variable.

Now, we shall describe ALCMA in more detail. From Algorithm 5, we can observe that ALCMA consists of two phases. In the **Initialization** phase, the factor α is first initialized with a sufficiently small value (line 3). In addition, the pseudo backbone and the pseudo fat are set to be empty (line 4). Then, a solution is achieved by calling an existing heuristic algorithm (line 5). For simplicity, we use the same heuristic M that will be applied in the second phase. When the heuristic M terminates, the returned solution J is recorded as the currently best solution J', and its cost $C_J(F, U, D, p)$ is taken as the upper bound UB_Π (lines 6 and 7). As the final step of the **Initialization** phase, we apply LR in Algorithm 2 to achieve the data structures required in the following of the algorithm, including LB_Π, t^*, λ^*, β^*, and I^*.

After the **Initialization** phase, the **Multilevel Solving** phase is conducted, which is a **while** loop composed of a few iterations. Within each iteration, the pseudo backbone (fat) variables

Algorithm 5: Accelerated Limit Crossing based Multilevel Algorithm (ALCMA)

```
   Input: PMP(F, U, D, p), heuristic algorithm M
   Output: J'
1  begin
2      // Phase 1: Initialization
3      Initialize α with a sufficiently small positive value;
4      pseudo_bone(α) = pseudo_fat(α) = ∅;
5      Initialize a feasible solution J by calling M;
6      J' = J;
7      UB_Π = C_J(F, U, D, p);
8      Achieve LB_Π, t*, λ*, β*, and I* by calling LR;
9      // Phase 2: Multilevel Solving
10     t = 0
11     while α < 1 do
12         t = t + 1;
13         Identify pseudo_bone(α) and pseudo_fat(α) with Criteria (13)
           and (14);
14         if both pseudo_bone(α) and pseudo_fat(α) are empty then
15             return J';
16         Fix pseudo_bone(α) and trim out pseudo_fat(α);
17         Search the reduced search space with algorithm M, with J' as the
           initial solution;
18         if better solution reached then
19             Let J' be the currently best solution;
20             UB_Π = C_J'(F, U, D, p);
21         α = α × 2;
22     return J';
23 end
```

are first determined by criteria (13) and (14) (line 13). Since the entries of β^* have been obtained from the **Initialization** phase, every facility can be checked in only $\mathcal{O}(1)$ running time. Note that if none of the pseudo backbone (fat) variables can be detected, the **while** loop stops and the best solution J' is returned, because at this time, the search space has degraded into the original search space (lines 14 and 15). Otherwise, these pseudo backbone (fat) variables are fixed (trimmed out) from the p-median instance so as to derive a reduced search space (line 16). Then, within the reduced search space, the embedded heuristic algorithm M is invoked, with the best solution J' as the initial solution (line 17). Since the search space has been dramatically reduced, the search procedure of the embedded heuristic algorithm can be conducted efficiently and effectively. Once a better solution is achieved, both the best solution J' and the upper bound $UB(\Pi)$ are updated (lines 18–20). Finally, the factor α gets doubled (line 21). Obviously, the pseudo upper bound will be greater than the upper bound when α exceeds one. Therefore, the **while** loop repeats until the value of α exceeds one.

In summary, in this section, we propose the ALCMA. By combining ALC and the multilevel mechanism in a unified framework, ALCMA might benefit from the effective reduction of the search space provided by ALC and, meanwhile, avoid the upper bound sensitivity. On the one hand, with the pseudo upper bound, ALC is hopeful to significantly alter the search space structure so as to improve the search efficiency. On the other hand, with the help of the multilevel strategy, the upper bound dependence of ALC is partially eliminated. Moreover, we shall note that ALCMA is not restricted to certain algorithms. In Algorithm 5, ALCMA does not make any assumption about the embedded heuristic M, which, to some extent, demonstrates the generality of the framework. In the following section, extensive experiments will be conducted so as to evaluate the performance of the framework.

TABLE III
RESULTS OF ALCMA WITH VARYING PARAMETER k

	PCB3038 (p = 400)		RL5934 (p = 600)		RW1000 (p=100)	
k	%err	time	%err	time	%err	time
10	0.003	8.71	0.001	38.71	2.401	1.83
20	0.002	12.75	<0.001	53.32	2.020	3.09
50	0.002	28.17	<0.001	110.19	1.917	6.61
100	0.001	60.95	<0.001	222.7	1.797	13.42
200	0.001	121.4	<0.001	544.35	1.322	24.49
500	< 0.001	309.71	<0.001	1265.22	1.175	59.28

V. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, extensive experiments and analysis are conducted. First, we describe the experiment environment and some implementation issues in Section V-A. Then, in Sections V-B and V-C, we conduct the preliminary parameter tuning tasks. In Section V-D, we examine the performance of our framework by comparing ALCMA with the state-of-the-art algorithm PBS and other relevant algorithms. Furthermore, to demonstrate the effectiveness of the framework with statistical evidence, in Section V-E, hypothesis tests are introduced to compare ALCMA and the other algorithms. Finally, in Section V-F, we investigate why ALCMA works through the configuration landscape analysis.

A. Experiment Environment and Implementation Issues

All the experiments are performed on a Pentium-IV 3.2-GHz PC with 4-GB memory running GNU/Linux with kernel 2.6.25. The algorithms are implemented in C++, compiled using g++ 4.30 with flag -O2. The running time is measured in seconds.

Theoretically, any existing heuristic algorithm for the p-median problem can be incorporated into ALCMA. In this paper, we embed VNS into ALCMA, which combines local search with systematic changes of the neighborhood. In this paper, we adopt the same version of VNS as the one proposed in [15], except for the stopping criterion. In our approach, the stopping criterion is changed to terminating VNS if no improvement can be achieved within k iterations of the interchange operator, where k is a parameter [for the rest of this paper, this variant version of VNS is denoted as VNS(k)]. The reason that we change the stopping criterion is that, within ALCMA, VNS(k) converges much faster than VNS.

From the implementation perspective, there are two local search operators used in the experiments, i.e., the interchange operator and the 2-opt operator. For the interchange operator in VNS(k) and HC, there are two widely used implementations with the same time complexity, as proposed in [48] and [34], respectively. However, with a time–space tradeoff, the implementation in [34] significantly accelerates the local search procedure. Therefore, in this paper, the implementation in [34] is employed. Meanwhile, for the 2-opt operator in HC, we implement a variant based on the same tradeoff [34].

B. Parameter Tuning

Before the numerical results and other discussions are presented, we first conduct some preliminary experiments for tuning the value of the parameter k within VNS(k). The value of k

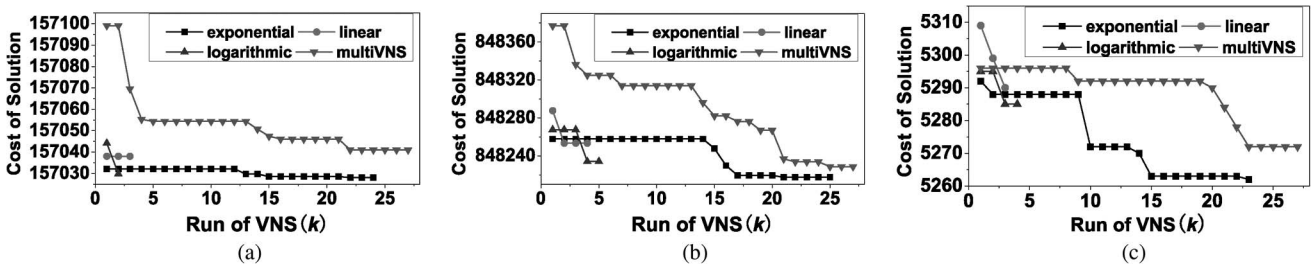

Fig. 2. Anytime performance of ALCMA with different factor adjustment strategies. (a) PCB3038 ($p = 400$). (b) RL5934 ($p = 600$). (c) RW1000 ($p = 100$).

is selected from {10, 20, 50, 100, 200, 500}. As the benchmark instances, we choose PCB3038 with $p = 400$, RL5934 with $p = 600$ from TSPLIB, and RW1000 with $p = 100$ from RW. We do not employ the ORLIB instances because they are too easy to tell differences among various parameter setups. The $\%err$ columns in the tables of this paper are defined according to the method in [15] as follows, where C_{opt} indicates the optimal cost or the best known upper bound (which can be found in Section V-D) and C represents the cost of the current solution:

$$\%err = \frac{C - C_{\text{opt}}}{C_{\text{opt}}} \times 100. \tag{15}$$

From Table III, ALCMA is not very sensitive to the parameter k as long as the value of k is not too small. On the other hand, the running time will increase accordingly as k grows. Thus, in this paper, the value of k is set to 100.

C. Sensitivity Analysis of Factor Adjustment Strategy

In ALCMA, the factor α is exponentially adjusted in the **Multilevel Solving** phase. A straightforward extension is to introduce other factor adjustment strategies, such as linear and logarithmic. In this section, some experiments are carried out to analyze the sensitivity of ALCMA to different factor adjusting strategies. The selected benchmark instances are the same as those in Section V-B, including PCB3038 with $p = 400$, RL5934 with $p = 600$, and RW1000 with $p = 100$.

Fig. 2 shows the anytime performance of ALCMA with different factor adjustment strategies. In every strategy, α is initialized with a sufficiently small positive value (1.00×10^{-8} in this study). For the exponential strategy, α gets doubled at each iteration, i.e., ALCMA runs at most $\lfloor \log_2 10^8 \rfloor + 1 = 27$ times of the embedded VNS(k). For comparison, the value of α in the other two strategies is updated such that ALCMA runs at most 27 times of VNS(k), i.e., $\alpha = \alpha + (1 - 10^{-8})/27$ in the linear strategy and $\alpha = 10^{-8} + (1 - 10^{-8}) \times \log_2 t / \log_2 27$ in the logarithmic strategy. Fig. 2 implies that both the linear strategy and the logarithmic strategy may suffer from early convergence. Over all the three instances, ALCMA with either the linear strategy or the logarithmic strategy exits the **Multilevel Solving** phase after only a few runs of VNS(k) since no backbone (fat) variable can be returned. In contrast, the exponential strategy is more robust and performs well over all the four instances. Furthermore, we compare ALCMA with the multistart version of VNS(k) (denoted as multiVNS). In multiVNS, VNS(k) is totally executed for 27 times, and the initial solution of each run of VNS(k) is provided by the best solution achieved up to the previous run. It can be observed from Fig. 2(a) and (b) that, over the TSPLIB instances, ALCMA with the exponential strategy outperforms multiVNS in terms of solution quality. Furthermore, note that, over PCB3038 with $p = 400$ and RL5934 with $p = 600$, all the three variants of ALCMA can obtain better solutions than multiVNS within the first several runs of VNS(k). This observation over these instances partially confirms the prediction in Section IV that, with a small upper bound factor, ALCMA can restrict the search process in a small region of the search space that contains high-quality solutions. However, an exception can be observed in Fig. 2(c). Over RW1000 with $p = 100$, the solution quality of ALCMA is similar with that obtained by multiVNS within the first several runs of VNS(k), which conforms with the observation in Section III-C.

D. Numerical Results

In order to evaluate the performance of ALCMA, we conduct the experiments over three broad classes of benchmark instances: 1) graph-based instances, including 40 instances from ORLIB, 3 instances from SL, and 16 instances from GR; 2) random instances, which consist of 28 instances from RW; and 3) Euclidean instances, which consist of 84 instances from TSPLIB. All the details about these benchmark instances can be found in [30]. For comparison, the experimental results of three other algorithms are also presented, including multiVNS, HC, and PBS [30]. For HC, the Gamma version [38] is chosen.

For the parameter configurations, the value of k in ALCMA and multiVNS is set to 100. The parameters of HC are set with the same values as in [38]. The numerical results are measured in the same way as other existing approaches [15], [30], [33], i.e., over each instance, every algorithm is independently performed for nine independent times. In the tables, the *best* column indicates the best cost obtained among the nine runs. The *MED* column presents the median cost over the nine runs. The $\%err$ column is calculated using (15), in which C represents the average cost in this case. Finally, the *time* column shows the average time elapsed for each algorithm.

Before comparing the results of the algorithms, there are several points that are worth noting.

1) Due to the computational reason, two exceptions are as follows. The first is about the RL11849 instances. Because of the extremely large number of facilities (as there are 11 849 facilities for each instance), we only perform ALCMA once and compare the results with PBS. The

TABLE IV
NUMERICAL RESULTS FOR FL1400 INSTANCES

	PBS				multiVNS				HC				ALCMA			
p	best	MED	%err	time	best	MED	%err	time	best	MED	%err	time	best	MED	%err	time
10	**101249.47**	101249.55	0.000	$< \epsilon$	101249.55	101249.55	0.000	50.25	101249.54	101249.55	0.000	5.06	101249.55	101249.55	0.000	16.58
20	**57857.55**	57857.94	0.001	0.44	57857.94	57857.94	0.001	31.46	57857.94	57857.94	0.001	3.55	57857.94	57857.94	0.001	18.58
30	**44013.02**	44013.48	0.001	4.22	44013.48	44013.48	0.001	29.81	44013.48	44013.48	0.001	43.84	44013.48	44013.48	0.001	23.23
40	**35002.02**	35002.52	0.002	0.44	35002.52	35002.52	0.001	31.77	35007.48	35015.41	0.042	32.19	35002.52	35002.52	0.001	21.30
50	**29089.71**	29090.23	0.002	0.78	29090.22	29090.22	0.002	30.97	29090.22	29090.22	0.002	73.87	29090.22	29090.22	0.002	19.15
60	**25160.40**	25161.12	0.003	10.22	25161.12	25161.12	0.008	31.41	25166.78	25168.53	0.065	133.81	25161.12	25161.12	0.003	19.85
70	**22125.46**	22126.03	0.003	4.67	22126.03	22126.03	0.003	34.53	22126.03	22126.03	0.042	202.87	22126.03	22126.03	0.003	15.83
80	**19870.28**	19870.85	0.003	8.67	19870.85	19870.85	0.005	33.71	19878.45	19897.17	0.118	263.32	19870.85	19870.85	0.003	18.83
90	**17987.91**	17988.60	0.004	4.56	17988.60	17988.60	0.005	34.37	17988.60	17990.47	0.047	358.50	17988.60	17988.60	0.004	16.70
100	**16551.20**	16552.22	0.006	10.22	16554.70	16562.39	0.059	40.54	16565.83	16576.42	0.147	228.09	16552.22	16553.83	0.015	19.51
150	**12026.41**	12034.98	0.071	16.56	12026.45	12029.03	0.012	40.57	12034.53	12043.10	0.137	914.55	12026.41	12026.45	0.016	30.43
200	9356.66	9360.18	0.038	17.67	9356.63	9359.96	0.049	38.69	9360.84	9365.39	0.108	1484.56	**9355.34**	9356.19	0.009	26.69
250	**7737.72**	7746.02	0.107	10.67	7737.78	7740.95	0.039	42.39	7747.07	7748.34	0.143	1917.64	7738.33	7741.16	0.036	31.70
300	6617.27	6623.79	0.099	32.89	6616.98	6619.51	0.121	56.33	6623.95	6628.54	0.252	2336.65	**6611.60**	6613.42	0.029	39.07
350	5719.30	5726.81	0.131	28.56	5720.18	5725.42	0.101	54.43	5729.59	5735.62	0.289	4009.06	**5719.08**	5724.48	0.098	43.78
400	**5006.75**	5010.11	0.067	249.22	5010.14	5017.82	0.217	54.84	5023.60	5032.17	0.486	4271.78	**5006.75**	5010.09	0.044	36.07
450	4473.36	4478.36	0.112	193.67	4483.27	4488.35	0.440	58.08	4486.74	4489.98	0.480	3568.48	**4468.29**	4468.31	0.021	39.64
500	4046.86	4049.41	0.063	79.11	4048.84	4051.19	0.104	69.92	4052.90	4055.94	0.225	3930.11	**4046.85**	4048.81	0.047	72.36

TABLE V
NUMERICAL RESULTS FOR PCB3038 INSTANCES

	PBS				multiVNS				ALCMA			
p	best	MED	%err	time	best	MED	%err	time	best	MED	%err	time
10	**1213082.03**	1213082.03	0.000	0.33	**1213082.03**	1213082.03	0.000	217.67	**1213082.03**	1213082.03	0.000	182.07
20	**840844.53**	840844.53	0.000	5.00	**840844.53**	840844.53	0.000	126.68	**840844.53**	840844.53	0.000	147.83
30	**677272.22**	677404.52	0.020	549.00	**677272.22**	677805.67	0.046	106.21	**677272.22**	677272.22	0.034	130.51
40	**571887.75**	571887.75	0.000	15.44	**571887.75**	571887.75	0.000	92.43	**571887.75**	571887.75	0.000	84.65
50	**507558.16**	507640.13	0.016	70.11	**507558.16**	507558.16	0.010	84.62	**507558.16**	507558.16	0.000	86.55
60	**460771.87**	460794.48	0.005	30.00	**460771.87**	460771.87	0.000	86.91	**460771.87**	460771.87	0.000	75.69
70	**426068.24**	426140.81	0.017	41.11	**426068.24**	426068.24	0.000	98.30	**426068.24**	426068.24	0.000	71.61
80	397491.10	397572.43	0.020	39.22	**397488.90**	397512.27	0.005	103.55	**397488.90**	397488.90	0.001	79.67
90	**373241.86**	373476.07	0.063	35.44	**373241.86**	373380.30	0.031	108.58	**373241.86**	373241.86	0.000	69.79
100	352612.79	352737.86	0.035	35.67	**352609.60**	352644.54	0.022	115.67	**352609.60**	352621.05	0.003	80.37
150	281181.68	281309.07	0.045	86.22	281183.09	281276.08	0.041	177.22	**281163.12**	281190.31	0.007	117.08
200	238353.32	238423.75	0.030	259.22	**238344.20**	238379.53	0.018	240.14	**238344.20**	238344.20	0.002	120.47
250	209218.48	209323.19	0.050	95.11	209223.78	209274.68	0.032	296.09	**209206.90**	209212.78	0.003	116.95
300	187691.38	187759.60	0.036	153.44	187704.09	187712.75	0.016	376.11	**187689.40**	187696.05	0.004	144.86
350	170942.24	171045.07	0.060	121.44	170949.41	170967.58	0.033	126.89	**170919.46**	170919.46	0.002	58.97
400	157029.10	157071.22	0.027	289.11	157038.90	157044.88	0.020	141.17	**157027.21**	157029.23	0.001	63.76
450	145375.40	145417.03	0.029	422.86	145370.84	145391.85	0.025	147.17	**145362.91**	145363.44	0.000	57.81
500	135466.28	135506.75	0.030	1093.00	135471.98	135491.71	0.037	137.72	**135447.39**	135451.66	0.002	63.15
550	126838.11	126895.63	0.045	1449.11	126838.66	126877.54	0.050	155.66	**126825.24**	126828.31	0.002	59.02
600	119089.89	119133.75	0.037	1113.33	119099.25	119128.15	0.053	153.97	**119059.77**	119059.77	0.000	57.27
650	112029.04	112074.34	0.040	1518.56	112055.00	112089.40	0.077	170.17	**112017.65**	112018.35	0.001	61.59
700	105844.60	105888.23	0.041	821.33	105865.68	105882.80	0.060	164.31	**105823.96**	105824.09	0.000	69.65
750	100353.68	100390.90	0.037	1307.44	100364.76	100400.81	0.078	181.14	**100331.25**	100339.45	0.008	92.32
800	95405.89	95431.64	0.027	1325.00	95428.72	95455.21	0.096	189.30	**95374.17**	95377.37	0.003	85.80
850	91000.61	91032.43	0.035	1369.78	91080.24	91108.07	0.132	190.53	**90983.64**	90988.48	0.005	90.88
900	86973.11	87021.68	0.056	1619.56	87053.96	87097.56	0.139	181.75	**86972.78**	86976.00	0.003	98.51
950	83275.69	83298.62	0.028	1830.56	83316.62	83362.69	0.119	210.87	**83267.38**	83271.61	0.011	95.83
1000	79853.82	79869.25	0.019	1633.67	79889.53	79927.95	0.109	189.07	**79842.01**	79845.72	0.007	113.33

other exception is about HC. Due to the high complexity of the 2-opt operator in HC, numerical results of HC for PCB3038, RL5934, and RL11849 instances are not presented.

2) Since both ALCMA and PBS can always achieve the optimality over all the ORLIB, SL, and GR instances, the experimental results over these instances are not presented. The experimental results over the RW and TSPLIB instances are shown in Tables IV–VIII.
3) It is not straightforward to compare the running time of the algorithms, particularly for PBS. First, PBS is a distributed algorithm, tested in a much higher performance environment than in this study.[2] Second, in [30], over each instance, the author only reports the time that PBS achieves the best solution, which makes it not appropriate for direct comparison. As a result, we only report the average time obtained on ten CPUs in [30] and do not conduct the scaling task. However, we can observe that, over instances with a large number of medians (e.g.,

[2]PBS is tested on a Linux cluster with up to 30 CPUs that, when executing the DIMACS Machine Benchmark (dmclique, ftp://dimacs.rutgers.edu in /pub/dsj/clique) on one CPU, requires 0.31 s for r300.5, 1.93 s for r400.5, and 7.35 s for r500.5. In our environment, the three values are 0.46, 2.85, and 10.79.

TABLE VI
NUMERICAL RESULTS FOR RL5934 INSTANCES

	PBS				multiVNS				ALCMA			
p	best	MED	%err	time	best	MED	%err	time	best	MED	%err	time
10	**9794951.00**	9794973.65	0.000	24.78	9794973.65	9794973.65	0.000	1377.68	9794973.65	9794973.65	0.000	1776.59
20	**6718848.19**	6719017.83	0.003	98.56	**6718848.19**	6718848.19	0.000	784.21	**6718848.19**	6718848.19	0.005	961.19
30	**5374936.14**	5375923.84	0.018	76.56	**5374936.14**	5374936.14	0.000	730.18	**5374936.14**	5374936.14	0.000	820.91
40	**4550327.09**	4550499.03	0.004	70.67	**4550327.09**	4550327.09	0.000	541.63	**4550327.09**	4550327.09	0.000	613.61
50	**4032379.97**	4032621.78	0.006	85.44	**4032379.97**	4032449.39	0.004	505.16	4032416.19	4032449.39	0.001	553.44
60	**3642064.70**	3642870.18	0.022	95.89	**3642064.70**	3642064.70	0.008	545.46	**3642064.70**	3642064.70	0.000	526.39
70	3343617.76	3344853.43	0.037	122.78	**3343541.36**	3343592.13	0.011	477.66	**3343541.36**	3343682.70	0.003	480.71
80	3094507.17	3095359.40	0.028	139.22	3094467.91	3095224.66	0.023	535.72	**3094451.30**	3094451.30	0.000	453.70
90	2893234.39	2894843.70	0.056	733.78	**2893204.11**	2893204.11	0.026	495.76	**2893204.11**	2893204.11	0.000	425.02
100	2725020.51	2725500.78	0.018	171.67	2724933.42	2724996.37	0.014	467.23	**2724740.61**	2724740.61	0.003	419.64
150	2147817.00	2148721.38	0.042	166.11	**2147756.39**	2147910.69	0.010	561.21	**2147756.39**	2147756.39	0.001	403.70
200	1808010.37	1808643.12	0.035	273.22	1807430.63	1807946.93	0.027	695.91	**1807411.06**	1807411.06	0.001	466.16
250	1569830.31	1570412.75	0.037	293.11	1569712.98	1569919.18	0.015	710.39	**1569724.89**	1569724.89	0.000	371.90
300	1394045.57	1394341.18	0.021	378.44	1393961.27	1393995.94	0.004	999.41	**1393951.74**	1393951.74	0.001	438.99
350	1256775.39	1257084.61	0.025	606.00	1256620.47	1256746.10	0.010	1222.28	**1256599.00**	1256599.00	0.000	490.56
400	1145631.85	1145942.24	0.027	467.11	1145314.67	1145587.41	0.026	1318.17	**1145271.28**	1145311.91	0.003	550.68
450	1053303.41	1053717.23	0.039	638.11	1053169.87	1053200.69	0.020	1611.69	**1053024.59**	1053034.68	0.001	546.90
500	973982.54	974230.36	0.025	765.89	973953.99	974133.53	0.020	1708.86	**973920.22**	973932.77	0.001	516.87
600	848266.43	848492.37	0.027	721.00	848232.39	848298.49	0.011	412.32	**848217.37**	848217.37	0.000	226.24
700	752047.96	752259.40	0.028	674.33	751972.31	752039.19	0.013	515.50	**751972.31**	751974.98	0.000	245.66
800	676778.00	676952.90	0.026	1396.56	676735.47	676774.00	0.010	419.88	**676724.24**	676724.24	0.000	226.33
900	613352.32	613494.23	0.023	1444.78	613317.62	613401.84	0.014	492.38	**613308.88**	613309.47	0.000	246.99
1000	558801.23	558941.68	0.025	1770.22	558813.33	558847.55	0.012	459.70	**558783.56**	558783.56	0.001	270.06
1100	511804.96	511923.31	0.023	1892.89	511772.60	511803.98	0.006	470.63	**511778.61**	511778.61	0.000	239.21
1200	470290.06	470408.35	0.025	2392.11	470277.12	470304.92	0.011	518.21	**470264.28**	470265.65	0.000	250.32
1300	433587.06	433675.29	0.020	1573.11	433567.90	433589.84	0.011	598.68	**433543.91**	433543.91	0.000	267.69
1400	401829.22	401931.31	0.025	3778.00	401881.32	401928.11	0.030	572.10	**401820.03**	401820.03	0.000	307.89
1500	374001.06	374051.86	0.014	6409.11	374032.29	374075.08	0.022	626.38	**373987.02**	373987.02	0.000	266.43

TABLE VII
NUMERICAL RESULTS FOR RL11849 INSTANCES

	PBS	ALCMA			PBS	ALCMA	
p	best	best	time	p	best	best	time
100	**5850644**	5851738	2416.64	600	2112973	**2112870**	2039.77
200	**4003810**	4003881	1600.58	700	1923108	**1922903**	2073.27
300	**3194525**	3194619	2309.20	800	1766618	**1766512**	2149.72
400	**2696221**	2696236	2005.35	900	1636303	**1636053**	2595.33
500	**2357273**	2357328	1709.16	1000	1523174	**1522955**	2418.70

PCB3038 with $p \geq 300$ and RL5934 with $p \geq 500$), the time for PBS to achieve the best solutions is longer than that for ALCMA to terminate. Moreover, as for HC and multiVNS, we can observe that, over many instances (53 out of 105 for HC and 120 out of 161 for multiVNS), their running time is longer than that of ALCMA. For those instances over which ALCMA is slower, the time is in the same order of magnitude with HC and multiVNS. These, to some extent, validate the solution quality comparison between these algorithms and ALCMA.

Now, we shall discuss the computational results. Over TSPLIB instances with a small number of medians (e.g., PCB3038 and RL5934 with $p < 100$), ALCMA is able to achieve solutions as good as PBS. When the number of medians grows larger (PCB3038 and RL5934 with $p \geq 100$), ALCMA usually obtains better solutions than PBS. Over all the 84 TSPLIB instances, ALCMA totally achieves 52 new best solutions and 13 currently best known solutions. For the TSPLIB instances over which ALCMA cannot achieve currently best known solutions, the average percentage errors are always less than 0.1%. It should be noted that the performance of ALCMA over RW instances is not so satisfying. The reason may be that these instances are randomly generated [34]; thus, the underlying structural information of these instances is hard to be extracted. With few backbone (fat) variables obtained, ALCMA will then degrade into multiVNS. In summary, over the total 171 benchmark instances, ALCMA is able to achieve 52 new best solutions and locate 92 of the currently best known/optimal solutions.

E. Statistical Tests

In this section, statistical tests are introduced to compare the average results of ALCMA and the other algorithms and draw confident conclusions whether one algorithm outperforms another. Following [12], in this section, we consider the average error rates (i.e., $\%err$ defined in (15), with respect to the nine independent runs of each algorithm). More specifically, we first employ Friedman's test to detect the potential differences in performance among algorithms. Then, the two-sided Wilcoxon's signed rank tests are conducted between ALCMA and other algorithms so as to analyze both the strength and the weakness of the framework. In this section, we consider the 95% confidence level (i.e., the $p - values$ below 0.05 are considered significant) for both Friedman's test and Wilcoxon's test.

As required by the statistical tests, we first specify the test instances. Due to the reason that HC is not applicable to all the instances, we divide the instance set into two subsets. More detailed, Subset 1 consists of the ORLIB, GR, SL, RW, and FL1400 instances, which have relatively small number of facilities. Subset 2 comprises the PCB3038 and RL5934 instances, which have larger number of facilities. Accordingly, we

TABLE VIII
NUMERICAL RESULTS FOR RW INSTANCES

		PBS				multiVNS				HC				ALCMA			
	p	best	MED	%err	time	best	MED	%err	time	best	MED	%err	time	best	MED	%err	time
100	10	**530**	530	0.000	0.02	**530**	533	0.460	0.35	**530**	530	0.106	0.33	**530**	530	0.147	0.48
	20	**277**	277	0.000	$< \epsilon$	**277**	277	0.119	0.57	**277**	277	0.000	0.40	**277**	277	0.000	0.55
	30	**213**	213	0.000	$< \epsilon$	**213**	213	0.000	0.47	**213**	213	0.000	0.11	**213**	213	0.000	0.36
	40	**187**	187	0.000	$< \epsilon$	**187**	187	0.000	0.42	**187**	187	0.000	0.04	**187**	187	0.000	0.34
	50	**172**	172	0.000	$< \epsilon$	**172**	172	0.000	0.40	**172**	172	0.000	0.03	**172**	172	0.000	0.33
250	10	**3691**	3691	0.000	0.15	**3691**	3722	1.409	1.41	**3691**	3698	0.710	15.16	**3691**	3722	0.530	1.62
	25	**1360**	1364	0.294	0.11	1364	1387	2.149	0.99	1364	1374	1.111	12.38	1364	1366	1.021	1.17
	50	**713**	713	0.000	0.49	717	720	1.060	1.42	715	720	0.997	6.89	**713**	713	0.311	1.28
	75	**523**	523	0.000	0.20	**523**	523	0.042	1.35	**523**	523	0.021	1.61	**523**	523	0.000	0.80
	100	**444**	444	0.000	0.01	**444**	444	0.000	1.34	**444**	444	0.000	0.59	**444**	444	0.000	0.85
	125	**411**	411	0.000	0.01	**411**	411	0.000	1.40	**411**	411	0.000	0.45	**411**	411	0.000	0.86
500	10	**16108**	16108	0.000	4.48	**16108**	16283	1.218	6.15	**16108**	16246	0.848	115.56	**16108**	16225	0.814	6.83
	25	**5626**	5681	0.978	13.46	5683	5831	3.444	2.95	5745	5807	3.274	115.07	5681	5764	2.247	3.40
	50	**2626**	2628	0.076	8.09	2645	2699	2.518	2.88	2649	2678	1.802	137.98	2644	2653	1.219	2.89
	75	**1757**	1757	0.000	3.10	1761	1773	1.341	4.64	1762	1775	1.056	56.36	**1757**	1769	0.651	3.96
	100	**1379**	1380	0.073	5.17	1380	1384	0.427	4.38	1382	1389	0.604	51.36	**1379**	1384	0.314	3.93
	150	**1024**	1024	0.000	22.04	**1024**	1025	0.087	4.96	1025	1026	0.195	16.86	**1024**	1024	0.000	2.48
	200	**893**	893	0.000	0.04	**893**	893	0.000	5.06	**893**	893	0.000	9.36	**893**	893	0.000	2.92
	250	**833**	833	0.000	0.01	**833**	833	0.000	6.10	**833**	833	0.000	2.24	**833**	833	0.000	3.39
1000	10	**67811**	67811	0.000	7.43	**67811**	68963	1.717	43.15	68136	68136	0.682	2454.69	**67811**	68136	0.858	54.97
	25	**24896**	24896	0.000	14.62	25076	25502	2.548	22.62	25140	25341	1.879	4144.14	24964	25228	1.270	24.33
	50	**11259**	11259	0.322	37.58	11443	11654	3.127	15.62	11376	11550	2.435	4908.51	11327	11486	2.002	14.40
	75	**7134**	7134	0.477	101.55	7278	7323	3.102	15.59	7218	7348	2.897	2281.48	7220	7288	2.151	14.93
	100	**5210**	5210	0.302	51.97	5296	5414	3.751	15.37	5250	5314	2.122	455.80	5254	5300	1.779	12.88
	200	**2704**	2704	0.000	33.89	2716	2727	0.822	18.96	2723	2731	0.990	835.31	2705	2716	0.366	14.39
	300	**2018**	2018	0.000	38.29	**2018**	2019	0.071	26.52	2020	2021	0.176	362.45	**2018**	2018	0.005	11.67
	400	**1734**	1734	0.000	0.51	**1734**	1734	0.000	26.13	**1734**	1735	0.058	155.82	**1734**	1734	0.000	13.65
	500	**1614**	1614	0.000	0.09	**1614**	1614	0.000	26.66	**1614**	1614	0.000	37.56	**1614**	1614	0.000	13.06

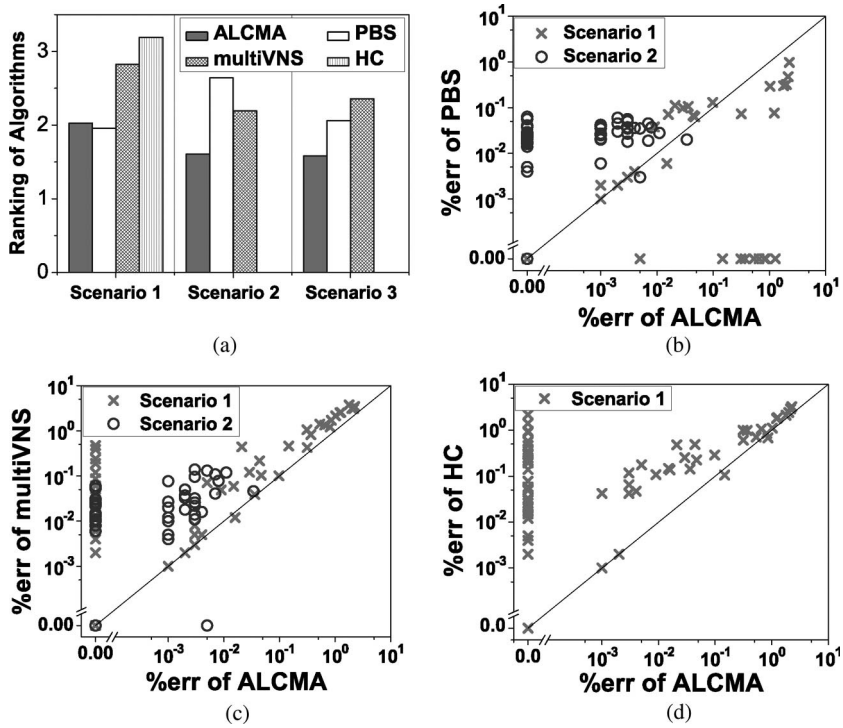

Fig. 3. Comparisons between ALCMA and other algorithms. (a) Average ranking of algorithms. (b) ALCMA versus PBS. (c) ALCMA versus multiVNS. (d) ALCMA versus HC.

compare the performance of the algorithms in three scenarios. In Scenario 1, we compare ALCMA with PBS, multiVNS, and HC over the instances from Subset 1, since HC is only tested over these instances. Then, in Scenario 2, we compare ALCMA with PBS and multiVNS over the instances from Subset 2. Finally, in Scenario 3, we compare ALCMA with PBS and multiVNS over all the instances.

In Fig. 3(a), we present the average ranking with respect to Friedman's test [12]. In the subfigure, each column represents the ranking of the corresponding algorithm, and lower columns

TABLE IX
RESULTS OF WILCOXON'S TESTS

	ALCMA $vs.$ PBS	ALCMA $vs.$ multiVNS	ALCMA $vs.$ HC
Scenario 1	0.004 (PBS)	7.78×10^{-10} (ALCMA)	7.35×10^{-11} (ALCMA)
Scenario 2	5.72×10^{-10} (ALCMA)	2.15×10^{-9} (ALCMA)	-
Scenario 3	0.025 (ALCMA)	$<2.20\times10^{-16}$ (ALCMA)	-

indicate better performance. From Fig. 3(a), we can observe that, in Scenario 1, PBS obtains better ranking than ALCMA, while in both Scenarios 2 and 3, ALCMA obtains the best ranking. In all the scenarios, Friedman's test detects significant differences in performance among the algorithms (with $p-values = 4.69\times10^{-11}$, 4.73×10^{-11}, and 5.19×10^{-11} for the three scenarios). However, since ALCMA does not always achieve the best ranking, we proceed to compare ALCMA with other algorithms in a pairwise paradigm so as to analyze both the strength and the weakness of ALCMA.

To gain a more intuitive impression about the pairwise comparisons, in Fig. 3(b)–(d), we present the relative comparison between algorithms in both Scenarios 1 and 2. For example, Fig. 3(b) shows the comparison between ALCMA and PBS. In the subfigure, the x-axis and the y-axis indicate the (log scale) $\%err$ of ALCMA and PBS, respectively. More specifically, each point (x, y) in the subfigure indicates that there are one or more instances over which ALCMA's $\%err$ and PBS's $\%err$ are x and y, respectively. Since the instance set is separated into two subsets (with respect to Scenarios 1 and 2), these instances are represented by different point types. For clarity, we plot the reference line $y = x$. Consequently, a point above the line implies that, over the corresponding instance(s), ALCMA outperforms PBS, since ALCMA obtains smaller $\%err$. Companion with Fig. 3(b)–(d), the results of Wilcoxon's tests are presented in Table IX, which are organized as follows. The first column indicates the three scenarios, as described earlier. Then, in columns 2–4, the results of Wilcoxon's tests are reported in each scenario. The comparison results consist of the $p-value$ as well as the name of the algorithm that achieves the better performance.

From Fig. 3(b)–(d) and Table IX, several interesting phenomena can be observed.

1) The result of the comparison between ALCMA and PBS is scenario dependent. In Scenario 1, there exist more points that lie below the reference line, which implies that PBS outperforms ALCMA in this scenario. This observation is confirmed by Wilcoxon's test ($p-value = 0.004$). We attribute this observation to the fact that, in Scenario 1, the graph-based (ORLIB, GR, and SL) instances are relatively easy, which cannot tell differences between ALCMA and PBS, while the RW instances are randomly generated, from which the structural information such as backbone and fat is hard to be extracted. Contrarily, in Scenario 2, ALCMA significantly outperforms PBS ($p-value = 5.72\times10^{-10}$). This observation demonstrates that our framework is very effective in solving those instances with a large number of facilities.
2) In Fig. 3(c), the majority of the points lie above the reference line, which reveals that ALCMA outperforms multiVNS ($p-value < 1.00\times10^{-8}$ in Scenarios 1 and 2). Since both ALCMA and multiVNS employ the same embedded heuristic (VNS(k) in this study), this observation demonstrates the effectiveness of the proposed framework.
3) Similar to 2), ALCMA statistically outperforms HC ($p-value = 7.35\times10^{-11}$), which is shown in Fig. 3(d).
4) Finally, in Scenario 3, when we compare ALCMA with PBS and multiVNS over the whole instance set (recall that, since HC is not tested over all the instances, ALCMA is not compared with HC in this scenario), Wilcoxon's tests indicate that ALCMA outperforms both PBS ($p-value = 0.025$) and multiVNS ($p-value < 2.20\times10^{-16}$). Considering the family-wise error rate [12], we can deduce that ALCMA performs the best among ALCMA, PBS, and multiVNS with a $p-value$ less than $1-(1-0.025)\times(1-2.20\times10^{-16}) < 0.026$.

As a brief summary, in this section, statistical tests are introduced to evaluate the effectiveness of the framework. Through various comparisons, we demonstrate that ALCMA is able to obtain competitive results, particularly over those instances with a large number of facilities. However, if the instances are unstructured, ALCMA does not have much advantage over the state-of-the-art algorithm. Furthermore, considering all the instances, the overall performance of ALCMA is better than that of the other algorithms (PBS and multiVNS).

F. Configuration Landscape Analysis

In this section, the configuration landscape analysis is conducted over typical ORLIB instances (pmed15 and pmed40) and TSPLIB instances (PCB3038 with $p = 400$ and RL5934 with $p = 600$) to investigate the underlying reasons why ALCMA works over these instances. We demonstrate that ALC is able to effectively alter the structure of the search space, and several design strategies in ALCMA are necessary, such as the dynamic maintenance of the pseudo upper bound and the exponential adjustment of α. As a comparison, we carry out the landscape analysis over an RW instance (RW1000 with $p = 100$) so as to explain why ALCMA performs poorly over RW instances. In addition, we also present the configuration landscape analysis for the HC-S1 reduction within HC so as to investigate why ALCMA outperforms HC and its variants [36]–[39].

In general, the fitness landscape analysis [45] is used either to understand the mechanism of algorithms [4], [46] or to evaluate the performance of algorithms [9], [26]. In addition, some researchers employ the fitness landscape to design new algorithms [22], [31]. As a special fitness landscape, the configuration landscape is developed by Zhang [51] for Max SAT to investigate the characteristics of search spaces. To explain why

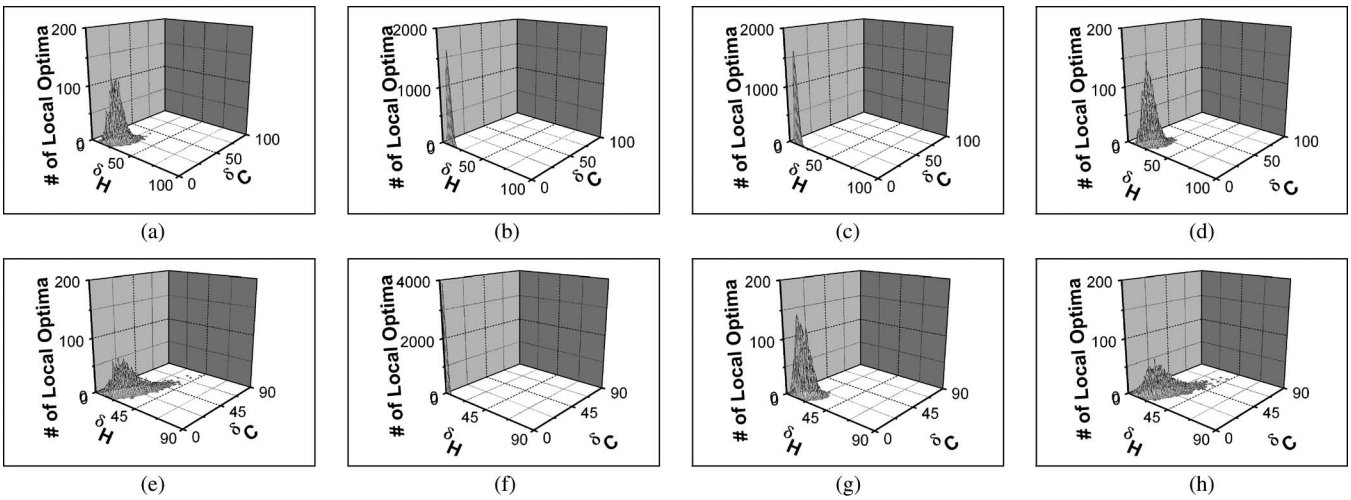

Fig. 4. Configuration landscapes generated by the interchange operator with/without the ALC reduction over ORLIB instances. (a) pmed15 (without ALC). (b) pmed15 ($\alpha = 0.001$). (c) pmed15 ($\alpha = 0.01$). (d) pmed15 ($\alpha = 0.5$). (e) pmed40 (without ALC). (f) pmed40 ($\alpha = 0.001$). (g) pmed40 ($\alpha = 0.01$). (h) pmed40 ($\alpha = 0.5$).

the ALC reduction works, we empirically compare the configuration landscapes generated by the interchange operator under two conditions. In the first case, the configuration landscape over the original search space is generated with 10 000 local optima, each of which is randomly initialized and improved by the interchange operator. In the second case, over the reduced search space derived by ALC, its configuration landscape is generated as follows. First, we follow the **Initialization** phase of ALCMA to generate an upper bound with the interchange operator and then employ the procedure of LR to achieve the lower bound LB_{Π}. With the t^*, λ^*, β^*, and I^* returned from LR in computing LB_{Π}, we can achieve the pseudo backbone (fat) with the specified values of α ($\alpha = 0.001, 0.01, 0.5$ in this study) with respect to criteria (13) and (14). Then, the search space can be reduced by fixing (trimming out) the pseudo backbone (fat). Within the reduced search space, 10 000 local optima are produced. Each local optimum is achieved by employing the interchange operator to improve a random initial solution, which includes two parts, i.e., the pseudo backbone and $p - \gamma$ facilities randomly chosen from all the facilities except the pseudo backbone and the pseudo fat, where γ denotes the number of facilities in the pseudo backbone.

In the generation method, the reasons we choose the interchange operator rather than VNS(k) are as follow. From the computational complexity perspective, VNS(k) is much more time consuming than the interchange operator, which makes it impractical to generate 10 000 local optima using VNS(k). Meanwhile, since the interchange operator is the key component of VNS(k), the distribution of local optima by the interchange operator can, to some extent, reflect the behavior of VNS(k). For example, in [15], the fitness landscapes generated by sampling 10 000 local optima of the interchange operator are employed to capture the properties of VNS.

Given the 10 000 local optima described earlier and a reference solution (provided by the best known solution; see Section IV-D), we can plot the configuration landscape for every instance. In the landscape, the three dimensions represent the Hamming distance and the cost difference from the reference solution, as well as the number of the local optima that are clustered together. The Hamming distance and the cost difference are defined as follows [15]:

$$\delta_H(J, J^*) = |J \backslash J^*| \tag{16}$$

$$\delta_C(J, J^*) = C_J(F, U, D, p) - C_{J^*}(F, U, D, p) \tag{17}$$

where J denotes a local optimum and J^* denotes the reference solution. A point (x, y, z) in the configuration landscape indicates that, among all the sampled local optima, z local optima are clustered together with the same Hamming distance x and the same cost difference y from the solution J^*.

The configuration landscapes of typical ORLIB instances over the original search spaces, and the reduced search spaces derived by ALC with $\alpha = 0.001, 0.01, 0.5$, are shown in Fig. 4. As shown in Fig. 4, the following observations of ALC can be achieved.

First, with an appropriate value of α, ALC is able to substantially alter the search space structure of these instances and restrict the search over a very small region of the search space that contains high-quality solutions. For example, over the instance pmed15, after the ALC reduction with $\alpha = 0.001$ [see Fig. 4(b)], most of the Hamming distances between the sampled local optima and the reference solution lie within $[0, (p/5)]$, and the cost differences lie within [0, 20]. While over the original search space of pmed15 [see Fig. 4(a)], the Hamming distances are within the interval $[0, (2p/5)]$, and the cost differences are within the interval [0, 50].

Second, over these instances, the distribution of the local optima over the reduced search space exhibits a consistent trend as α grows. For example, over pmed40, after the ALC reduction with $\alpha \leq 0.01$ [see Fig. 4(f) and (g)], the cost differences between the sampled local optima and the reference solution are always less than 30. As α grows to 0.5 [see Fig. 4(h)], the maximum cost difference also increases to 79, and the configuration landscape is almost the same as that over the original search space [see Fig. 4(e)]. This observation confirms the prediction in Section IV that, with α increasing from a very

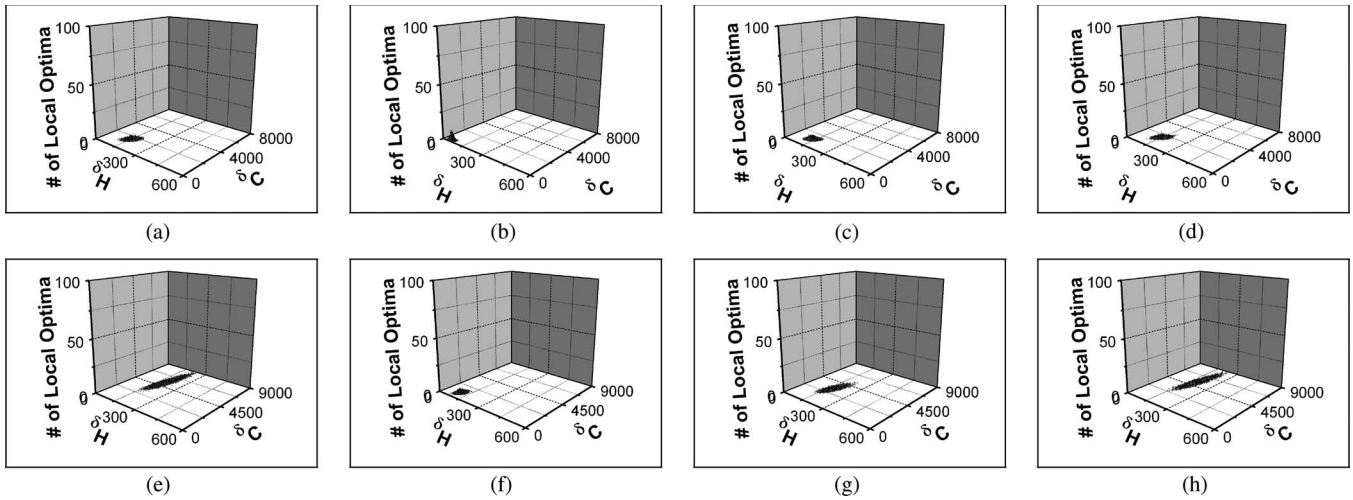

Fig. 5. Configuration landscapes generated by the interchange operator with/without the ALC reduction over TSPLIB instances. (a) PCB3038 ($p = 400$ and without ALC). (b) PCB3038 ($p = 400$ and $\alpha = 0.001$). (c) PCB3038 ($p = 400$ and $\alpha = 0.01$). (d) PCB3038 ($p = 400$ and $\alpha = 0.5$). (e) RL5934 ($p = 600$ and without ALC). (f) RL5934 ($p = 600$ and $\alpha = 0.001$). (g) RL5934 ($p = 600$ and $\alpha = 0.01$). (h) RL5934 ($p = 600$ and $\alpha = 0.5$).

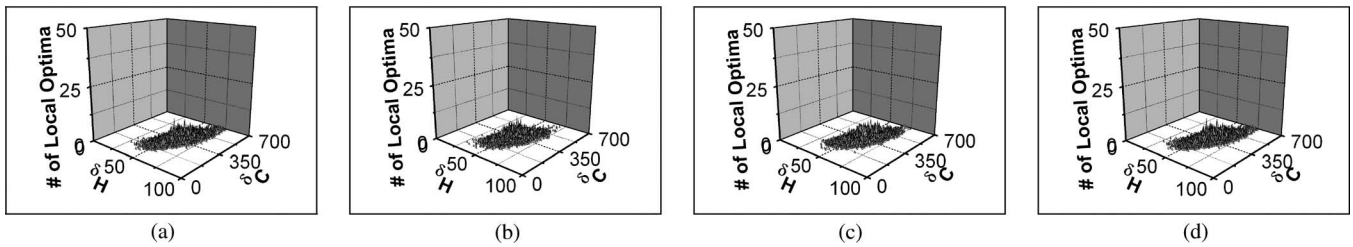

Fig. 6. Configuration landscapes generated by the interchange operator with/without the ALC reduction over an RW instance. (a) RW1000 ($p = 100$ and without ALC). (b) RW1000 ($p = 100$ and $\alpha = 0.001$). (c) RW1000 ($p = 100$ and $\alpha = 0.01$). (d) RW1000 ($p = 100$ and $\alpha = 0.5$).

small positive value to one, the reduced search space gradually grows from a highly restricted subspace toward the original search space. On the other hand, with the same value of α, the comparison between the configuration landscapes over different instances implies that ALC is instance dependent. For example, as α increases from 0.001 to 0.01, the configuration landscape over pmed40 [see Fig. 4(f) and (g)] changes more drastically than that over pmed15 [see Fig. 4(b) and (c)]. This also explains why we dynamically adjust the value of α.

In order to examine whether the aforementioned phenomena hold for other classes of instances, we proceed to conduct the configuration landscape analysis over typical TSPLIB and RW instances. The generation method is the same as those over the ORLIB instances. From Fig. 5, similar observations can be drawn over the TSPLIB instances, i.e., ALC with the appropriate value of α is able to effectively restrict the search within a promising region of the search space, and α is instance dependent. However, from Fig. 6, it is obvious that, over RW1000 with $p = 100$, ALC fails to alter the search space structure significantly, even with $\alpha = 0.001$. This observation partially explains the reason for the poor performance of ALCMA over RW instances.

In the remaining part of this section, we intend to investigate the reason why ALCMA outperforms HC, by analyzing the configuration landscapes with/without the HC-S1 reduction. We first produce the configuration landscape over the original search space by sampling 10 000 local optima, each of which is randomly initialized and improved by the 2-opt operator. For comparison, we also achieve 10 000 local optima within the reduced search space derived by the HC-S1 reduction as follows. First, we follow the work in [38] to generate 15 local optima using the interchange operator. Hereafter, for clarity, let InS and UnS be the intersection and the union of these 15 solutions, respectively. Then, the concentration set is formed with UnS. Within this concentration set, a reduced search space can be derived, and 10 000 local optima are produced. Each local optimum is achieved by applying the 2-opt operator to a random initial solution composed of two parts, i.e., InS and $p - \gamma$ facilities randomly chosen from $UnS \backslash InS$, where $\gamma = |InS|$. Due to the computational complexity of the 2-opt operator, the configuration landscapes with/without the HC-S1 reduction are only generated over typical ORLIB instances. In this experiment, the configuration landscapes are generated by sampling local optima of the 2-opt operator, rather than that of the interchange operator. The reason is that, in HC, the interchange operator and the 2-opt operator are applied during the HC-S1 reduction stage and the HC-S2 solving stage, respectively. Meanwhile, in [38], it is claimed that the heuristics used during the HC-S1 stage and the HC-S2 stage should not be the same [38].

The configuration landscapes generated by the 2-opt operator are shown in Fig. 7. As shown in Fig. 7, the HC-S1 reduction fails to alter the structure of the search space. Over each instance in this experiment, the configuration landscape of the

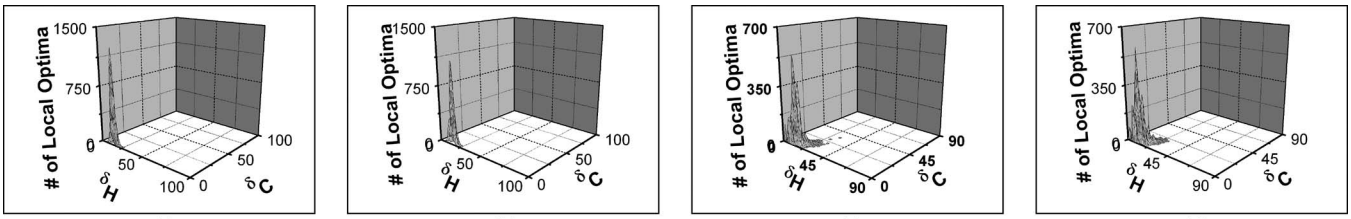

Fig. 7. Configuration landscapes generated by the 2-opt operator with/without the HC-S1 reduction over ORLIB instances. (a) pmed15 (without HC-S1). (b) pmed15 (with HC-S1). (c) pmed40 (without HC-S1). (d) pmed40 (with HC-S1).

original search space is almost the same as that of the reduced search space. As a result, the performance of HC mainly benefits from the shrinking of the search space.

VI. Conclusion

The main contributions of this paper can be summarized into the following aspects. First, in this paper, we propose an LC framework searching for the backbone (fat) variables. Based on the LC framework, a novel algorithm ALCMA is proposed, and numerical results over the benchmark instances indicate that our algorithm can achieve very competitive solutions. Over the benchmark instances, several new best solutions are achieved. Finally, the configuration landscape analysis is introduced to investigate the underlying reasons why ALCMA works. For future research, we plan to investigate several potential directions. As a straightforward extension of our work, we are interested in incorporating some other existing heuristic algorithms into ALCMA, e.g., TS [35], ACO [23], and SA [28]. In addition, there exist various studies following VNS, such as decomposition [14] and parallelization [11]. It will be interesting to test whether these ideas can be incorporated into the ALCMA framework. Second, since both LC and ALC are sensitive to the upper (lower) bound, it could be an important direction in the future work on how to obtain tight bounds, so that more exact backbone (fat) variables can be identified. As a long-term plan, we intend to investigate how to employ ALCMA to solve more optimization problems.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

[1] J. E. Beasley, "A note on solving large p-median problems," *Eur. J. Oper. Res.*, vol. 21, no. 2, pp. 270–273, Aug. 1985.

[2] K. D. Boese, "Cost versus distance in the traveling salesman problem," Univ. California, Los Angeles, CA, Tech. Rep. CSD-950018, 1995.

[3] I. T. Christou, "Coordination of cluster ensembles via exact methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 2, pp. 279–293, Feb. 2011.

[4] S. Choi and B. Moon, "Normalization for genetic algorithms with nonsynonymously redundant encodings," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 604–616, Oct. 2008.

[5] S. Climer and W. Zhang, "Searching for backbone and fat: A limit-crossing approach with applications," in *Proc. 18th Assoc. Adv. Artif. Intell.*, 2002, pp. 707–712.

[6] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Manage. Sci.*, vol. 23, no. 8, pp. 789–810, Apr. 1977.

[7] W. Cook and P. Seymour, "Tour merging via branch-decomposition," *Inf. J. Comput.*, vol. 15, no. 3, pp. 233–248, Jul. 2003.

[8] M. Daskin and L. Dean, "Location of health care facilities," *Oper. Res. Health Care*, vol. 70, pp. 43–76, 2005.

[9] M. Gallagher and B. Yuan, "A general-purpose tunable landscape generator," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 590–603, Oct. 2006.

[10] F. Garcıacute;a-López, B. Melián-Batista, J. A. Moreno-Pérez, and J. M. Moreno-Vega, "Parallelization of the scatter search for the p-median problem," *Parallel Comput.*, vol. 29, no. 5, pp. 575–589, 2003.

[11] F. Garcıacute;a-López, B. Melián-Batista, J. Moreno-Pérez, and J. Moreno-Vega, "The parallel variable neighborhood search for the p-median problem," *J. Heurist.*, vol. 8, no. 3, pp. 375–388, May 2002.

[12] S. Garcıacute;a, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heurist.*, vol. 15, no. 6, pp. 617–644, Dec. 2009.

[13] L. Greco, M. Gaeta, and B. Piccoli, "Sensor deployment for network-like environments," *IEEE Trans. Autom. Control*, vol. 55, no. 11, pp. 2580–2585, Nov. 2010.

[14] P. Hansen, N. Mladenovi, and D. Perez-Britos, "Variable neighborhood decomposition search," *J. Heurist.*, vol. 7, no. 4, pp. 335–350, Jul. 2001.

[15] P. Hansen and N. Mladenovi, "Variable neighborhood search for the p-median," *Loc. Sci.*, vol. 5, no. 4, pp. 207–226, Dec. 1997.

[16] H. Hoos and T. Stützle, *Stochastic Local Search: Foundations and Applications*. San Mateo, CA: Morgan Kaufmann, 2005, pp. 158–171.

[17] C. M. Hosage and M. F. Goodchild, "Discrete space location-allocation solutions from genetic algorithms," *Ann. Oper. Res.*, vol. 6, no. 2, pp. 35–46, Feb. 1986.

[18] O. Kariv and S. Hakimi, "An algorithmic approach to network location problems. II: The p-medians," *SIAM J. Appl. Math.*, vol. 37, no. 3, pp. 539–560, Dec. 1979.

[19] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Optimal placement of replicas in trees with read, write, and storage costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 6, pp. 628–637, Jun. 2001.

[20] P. Kilby, J. Slaney, S. Thiébaux, and T. Walsh, "Backbones and backdoors in satisfiability," in *Proc. 20th Nat. Conf. Artif. Intell.*, 2005, pp. 1368–1373.

[21] P. Kilby, J. Slaney, and T. Walsh, "The backbone of the travelling salesperson," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, 2005, pp. 175–180.

[22] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2005.

[23] Y. Kochetov, T. Levanova, E. Alekseeva, and M. Loresh, "Large neighborhood local search for the p-median problem," *Yugosl. J. Oper. Res.*, vol. 15, no. 1, pp. 53–63, 2005.

[24] A. Linhares, "Synthesizing a predatory search strategy for VLSI layouts," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 147–152, Jul. 1999.

[25] H. Jiang, X. Zhang, G. Chen, and M. Li, "Backbone analysis and algorithm design for the quadratic assignment problem," *Sci. China Ser. F, Inf. Sci.*, vol. 51, no. 5, pp. 476–488, May 2008.

[26] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 337–352, Nov. 2000.

[27] N. Mladenovi, J. Brimberg, P. Hansen, and J. A. Moreno-Prez, "The p-median problem: A survey of metaheuristic approaches," *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 927–939, Jun. 2007.

[28] A. Murray and R. Church, "Applying simulated annealing to location-planning models," *J. Heurist.*, vol. 2, no. 1, pp. 31–53, 1996.

[29] A. Parkes, "Clustering at the phase transition," in *Proc. Nat. Conf. Artif. Intell.*, 1997, pp. 340–346.

[30] W. Pullan, "A population based hybrid metaheuristic for the p-median problem," in *Proc. IEEE World Congr. Comput. Intell.*, 2008, pp. 75–82.

[31] M. Qasem and A. Prügel-Bennett, "Learning the large-scale structure of the MAX-SAT landscape using populations," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 518–529, Aug. 2010.
[32] J. Reese, "Solution methods for the *p*-median problem: An annotated bibliography," *Networks*, vol. 48, no. 3, pp. 125–142, Oct. 2006.
[33] M. Resende and R. Werneck, "A hybrid heuristic for the *p*-median problem," *J. Heurist.*, vol. 10, no. 1, pp. 59–88, 2004.
[34] M. Resende and R. Werneck, "On the implementation of a swap-based local search procedure for the *p*-median problem," in *Proc. 5th workshop Algor. Eng. Exp.*, 2003, pp. 119–128.
[35] E. Rolland, D. A. Schilling, and J. R. Current, "An efficient Tabu search procedure for the *p*-median problem," *Eur. J. Oper. Res.*, vol. 96, no. 2, pp. 329–342, Jan. 1997.
[36] K. E. Rosing and C. S. ReVelle, "Heuristic concentration: Two stage solution construction," *Eur. J. Oper. Res.*, vol. 97, no. 1, pp. 75–86, Feb. 1997.
[37] K. E. Rosing, "Heuristic concentration and Tabu search: A head to head comparison," *Eur. J. Oper. Res.*, vol. 104, no. 1, pp. 93–99, Jan. 1998.
[38] K. E. Rosing, C. S. ReVelle, and D. A. Schilling, "A gamma heuristic for the *p*-median problem," *Eur. J. Oper. Res.*, vol. 117, no. 3, pp. 522–532, Sep. 1999.
[39] K. E. Rosing and M. J. Hodgson, "Heuristic concentration for the *p*-median: An example demonstrating how and why it works," *Comput. Oper. Res.*, vol. 29, no. 10, pp. 1317–1330, Sep. 2002.
[40] M. Ruffolo, M. S. Daskin, A. V. Sahakian, and R. A. Berry, "Design of a large network for radiological image data," *IEEE Trans. Inf. Technol. Biomed.*, vol. 11, no. 1, pp. 25–39, Jan. 2007.
[41] A. B. Stephens, Y. Yesha, and K. E. Humenik, "Optimal allocation for partially replicated database systems on ring networks," *IEEE Trans. Knowl. Data Eng.*, vol. 6, no. 6, pp. 975–982, Dec. 1994.
[42] J. Schneider, C. Froschhammer, I. Morgenstern, T. Husslein, and J. Singer, "Searching for backbones—An efficient parallel algorithm for the traveling salesman problem," *Comput. Phys. Commun.*, vol. 96, no. 2/3, pp. 173–188, Aug. 1996.
[43] J. Schneider, "Searching for backbones—A high-performance parallel algorithm for solving combinatorial optimization problems," *Future Gener. Comput. Syst.*, vol. 19, no. 1, pp. 121–131, Jan. 2003.
[44] E. Senne and L. Lorena, "Lagrangean/surrogate heuristics for *p*-median problems," in *Proc. Comput. Tools Model., Optim. Simul., Interf. Comput. Sci. Oper. Res.*, 2000, pp. 115–130.
[45] P. Stadler, "Landscapes and their correlation functions," *J. Math. Chem.*, vol. 20, no. 1, pp. 1–45, Mar. 1996.
[46] J. Tavares, F. B. Pereia, and E. Costa, "Multidimensional knapsack problem: A fitness landscape analysis," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 3, pp. 604–616, Jun. 2008.
[47] M. B. Teitz and P. Bart, "Heuristic methods for estimating the generalized vertex median of a weighted graph," *Oper. Res.*, vol. 16, no. 5, pp. 955–961, 1968.
[48] R. Whitaker, "A fast algorithm for the greedy interchange for large-scale clustering and median location problems," *INFOR*, vol. 21, no. 2, pp. 95–108, 1983.
[49] T. Ye, H. T. Kaur, and S. Kalyanaraman, "Large-scale network parameter configuration using an on-line simulation framework," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 777–790, Aug. 2008.
[50] B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot routing problem in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 9, pp. 1252–1261, Sep. 2007.
[51] W. Zhang, "Configuration landscape analysis and backbone guided local search: Part I: Satisfiability and maximum satisfiability," *Artif. Intell.*, vol. 158, no. 1, pp. 1–26, Sep. 2004.
[52] W. Zhang and M. Looks, "A novel local search algorithm for the traveling salesman problem that exploits backbones," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, 2005, pp. 343–348.

Zhilei Ren received the B.Sc. degree in software engineering from Dalian University of Technology, Dalian, China, in 2007. He is currently working toward the Ph.D. degree at Dalian University of Technology.

His research interests include metaheuristic algorithm design and data mining.

Mr. Ren is a student member of the China Computer Federation.

He Jiang (M'10) received the B.Sc. and Ph.D. degrees in computer science from the University of Science and Technology of China, Hefei, China, in 1999 and 2005, respectively.

He is currently an Associate Professor with the School of Software, Dalian University of Technology, Dalian, China. His research interests include computational intelligence and its applications.

Dr. Jiang is a member of the China Computer Federation.

Jifeng Xuan received the B.Sc. degree in software engineering from Dalian University of Technology, Dalian, China, in 2007. He is currently working toward the Ph.D. degree at Dalian University of Technology.

His research interests include search-based software engineering, mining software repositories, and machine learning.

Mr. Xuan is a student member of the China Computer Federation.

Zhongxuan Luo received the B.Sc. and M.Sc. degrees in computational mathematics from Jilin University, Changchun, China, in 1985 and 1988, respectively, and the Ph.D. degree in computational mathematics from Dalian University of Technology, Dalian, China, in 1991.

He has been a Full Professor with the School of Mathematical Sciences, Dalian University of Technology, since 1997. His research interests include multivariate approximation theory and computational geometry.