European Journal of Operational Research 241 (2015) 622–630

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

On the unified dispersion problem: Efficient formulations and exact algorithms

Ting L. Lei*, Richard L. Church

Department of Geography, University of California, Santa Barbara, Santa Barbara, CA 93106-4060, United States

ARTICLE INFO

Article history:
Received 24 June 2013
Accepted 12 October 2014
Available online 17 October 2014

Keywords:
Location
Facility dispersion

ABSTRACT

Facility dispersion problems involve placing a number of facilities as far apart from each other as possible. Four different criteria of facility dispersal have been proposed in the literature (Erkut & Neuman, 1991). Despite their formal differences, these four classic dispersion objectives can be expressed in a unified model called the partial-sum dispersion model (Lei & Church, 2013). In this paper, we focus on the unweighted partial sum dispersion problem and introduce an efficient formulation for this generalized dispersion problem based on a construct by Ogryczak and Tamir (2003). We also present a fast branch-and-bound based exact algorithm.

1. Introduction

Facility dispersion problems involve maximizing the separation between facilities to minimize the negative impact they have on each other, minimize potential interaction among hazardous facilities, or enhance the reliability of service and logistic systems. Facility dispersal can be used in a variety of applications including military defense, franchise location, transportation of hazardous materials, layout planning for explosive chemicals (Curtin & Church, 2006) and telecommunication network design (Kim, 2012). Curtin and Church (2007) demonstrate that patterns in the classic central place theory can be replicated using facility dispersal. Facility dispersal has also been deemed as a means to promote the robustness and reliability of critical facilities when it is integrated with other classic location models (Maliszewski, Kuby, & Horner, 2012).

To capture the dispersive quality in different applications, multiple models for facility dispersal have been proposed. Four basic constructs have been developed in the literature to disperse facilities from each other. The first was suggested by Shier (1977) in which p-facilities are located while maximizing the minimum distance separating any two facilities. This problem has been called the p-dispersion problem (Moon & Chaudhry, 1984) and has also been termed the Max–Min–Min problem (Erkut & Neuman, 1991). Moon and Chaudhry (1984) proposed a second form of facility dispersal, which involved defining the minimum separation distance for each located facility. They proposed to locate p-facilities in order to maximize the sum of these minimum separation distances. Moon and Chaudhry called this the p-defense problem and Erkut and Neuman classified this problem as a Max–Sum–Min, as this involves MAXimizing the SUM of MINimum separation distances. Kuby (1987) introduced a third form of p-facility dispersal that he called p-dispersion sum. This problem involved maximizing the sum of all separation distances, which has been called the Max–Sum–Sum problem as it involves MAXimizing the SUM (over each facility) of SUMs (the sum of all separations distances for that facility) (Erkut & Neuman, 1991). The last of the four classic facility location dispersal problems was suggested by Erkut and Neuman (1991). This problem deals with the location of p-facilities while maximizing the smallest of the facility defined sums (a facility sum represents the sum of separation distances from a specific facility location to all other facilities) and has been classified as the Max–Min–Sum problem (Erkut & Neuman, 1991).

Curtin and Church (2006) added a new dimension to facility dispersal by conceptualizing a multi-type dispersion metric recognizing that the extent to which two facilities ought to be dispersed should depend on their types. The strength of interaction between facilities is modeled using both the traditional separation distance and a type-specific "repulsion" factor in which a smaller repulsion measure reflects a stronger inter-facility interaction. Curtin and Church developed four dispersion models based on the four basic dispersion metrics classified in Erkut and Neuman (1991) and the multi-type metric where each multi-type dispersion model is a multi-type extension of a classic dispersion model.

Fernández, Kalcsics, and Nickel (2013) proposed an extended dispersion problem involving locating multiple groups of facilities and applied the multi-group dispersion model to the location of recycling facilities. A dispersion metric corresponding to the Max–Min–Min criterion is maximized for each group of facilities. To ensure even distribution of workload, each facility is assigned a weight value and the sum of weights for each group is constrained to be within a range of

* Corresponding author. Tel.: +1 8058934519.
E-mail addresses: tinglei@geog.ucsb.edu (T. L. Lei), church@geog.ucsb.edu (R. L. Church).

http://dx.doi.org/10.1016/j.ejor.2014.10.020
0377-2217/

a pre-specified target weight value. Equity issues in dispersion modeling have also been addressed in Prokopyev, Kong, and Martinez-Torres (2009).

Dispersion problems are related to the maximum independent set problem and the maximum clique problem in graph theory. The *p*-dispersion problem, for example, can be reduced to the maximum independent set problem by removing edges longer than a given length *r* and find the maximum independent set on the converted graph (Erkut, 1990). The *p*-dispersion problem can be solved by solving a series of maximum independent set problems with increasing *r* values until the size of the maximum independent set is *p*. Solution methods for the maximum independent set problem have been explored by many researchers. Feo, Resende, and Smith (1994) proposed a parallel Greedy Randomized Adaptive Search Procedure (GRASP) for the maximum independent set problem and found it to be superior in performance to tabu search and simulated annealing methods. Hifi (1997) developed a genetic algorithm for the weighted maximum independent set program, which can be used also to solve equivalent problems such as the maximum clique problem. Gamarnik and Goldberg (2010) presented complexity results of greedy randomized algorithms on constant degree regular graphs.

With few exceptions, the majority of research on facility dispersion in the past two decades has involved the four basic dispersion models classified by Erkut and Neuman (1991). Lei and Church (2013) recently proposed a generalized dispersion model based on the concept of partial sums. At the individual facility level, the partial-sum dispersion metric involves accounting for the sum of the smallest *L* distances to neighboring facilities, weighted by a set of propulsion factors that can be used, for example, to emphasize the interaction of closely located facilities. At the system level, the model considers the sum of the *K* smallest facility-defined partial sums. This general construct is called the MaxPSumPSum problem (where PSum stands for partial sum). Lei and Church provided an integer linear programming formulation of this general partial-sum dispersion problem based on assignment variables which tracked specific separation distances.

The partial sum dispersion metric is a compromise of the one-or-all approach found in the four basic dispersion models and makes more sense in modeling many types of inter-facility interactions. For example, in franchise branch location, stores from the same franchise chain should be located away from each other to minimize cannibalization within the same organization. However, existing dispersion models either considers competition from the nearest store and ignores competition from other nearby stores, or considers competition from all stores in the franchise chain including remote sites that have little effect on a store. It would make more sense to consider a number of closest stores using the partial-sum dispersion metric. As another example, in military defense, it is common wisdom to locate assets such as missile silos apart from one another to minimize the chance that two facilities are destroyed at the same time. However, if avoiding the simultaneous loss of two facilities leads to low average spacing between sites, from management's perspective, it may be desirable to devise a facility layout that minimizes the chance of losing three or more facilities simultaneously.

From a theoretical point of view, the MaxPSumPSum construct unifies all four classic dispersion models as special case problems. This means that the MaxPSumPSum problem not only defines a family of new (PSum) dispersion models, but also makes it possible to solve all four existing facility dispersal problems as special case instances. It should be noted that partial-sum dispersion should not be confused with similar partial-sum metrics in median location problems, which involve ***minimizing*** partial sums of demand-to-facility distances either at the individual facility level (Weaver & Church, 1985) or at the system level (Nickel & Puerto, 1999). The reader is referred to Lei and Church (2014) for a median location problem that considers such location criteria in a unified construct.

Facility dispersion problems often have high computational complexities. It is well-known that both the *p*-dispersion problem and the *p*-dispersion sum problem are NP-hard (see e.g. Erkut, 1990; Pisinger, 2006). In addition, no polynomial-time heuristic procedure (or approximation algorithm) can guarantee to obtain a near optimal solution for the *p*-dispersion problem that is within any fixed percentage of the optimal value (Tamir, 1991). In the special case where the separation distances satisfy the triangle inequality, a heuristic for the *p*-dispersion problem exists with an approximation ratio of 2 (Tamir, 1991) and this ratio is the best possible (Ravi, Rosenkrantz, & Tayi, 1994). An approximation algorithm for the *p*-dispersion sum problem also exists with an approximation ratio of 4 (Ravi et al., 1994).

Since it subsumes the four classic dispersion problems as special-cases, the generalized dispersion problem should have a computational complexity that is no less than the special-case problems. Lei and Church (2013) report high computation costs of the generalized dispersion problem in their experiments. Certain medium-sized instances of the MaxPSumPSum problem cannot be solved using their ILP formulation in hours. In fact, one example presented in Lei and Church (2013) involving 55 candidate sites and locating 10 dispersive facilities took a week and half to solve. In many applications, such high computational costs may well prevent the model from being applied in practical analysis.

This article aims at operationalizing the generalized dispersion model by developing improved model formulations and efficient, specialized solution procedures. In particular, we focus on a special case of the partial-sum dispersion model defined without the propulsion factors. It should be noted that this version of the partial-sum dispersion problem is still very general because the four classic dispersion models are defined without the propulsion factors and therefore are its special cases. To avoid ambiguity, we refer to the general partial-sum dispersion problem as the *weighted* partial-sum dispersion problem and the version without propulsion factors as the *unweighted* partial-sum dispersion problem. We demonstrate that a compact and efficient formulation of the unweighted partial-sum dispersion model can be developed by using a linear program by Ogryczak and Tamir (2003) twice within the model formulation. Our computational experiments show that for the same problem instances solved by Lei and Church (2013), the new formulation can solve the partial dispersion problems faster in a majority of the cases. Moreover, we develop and present an interchange heuristic inspired by Teitz and Bart (1968) in conjunction with a branch and bound search, which are orders of magnitude faster than the integer linear programming approach. As will be shown in the experiment section, the branch and bound procedure can solve the same 55-node problem instance that took the ILP formulations a week and a half in about 3 minutes, which makes the partial-sum dispersion model suitable for location analysis in day-to-day operations.

2. Model formulation

The unweighted partial-sum dispersion problem can be formally defined as follows:

> *Locate a set of p facilities such that the sum of K worst-case facility-based sum of distances is maximized, where each facility-based sum of distances is the partial sum of L smallest distances from its neighboring facilities.*

To illustrate the concept of the partial sum dispersion problem, consider the following example in Fig. 1, in which three dispersive facilities are located among five candidate sites using a partial sum dispersion metric with $K = 1$, and $L = 2$. In the example, the coordinates for candidate sites (from *A* to *E*) are labeled. Distances between sites (in Table 1) are Euclidean distance rounded to the nearest integer.

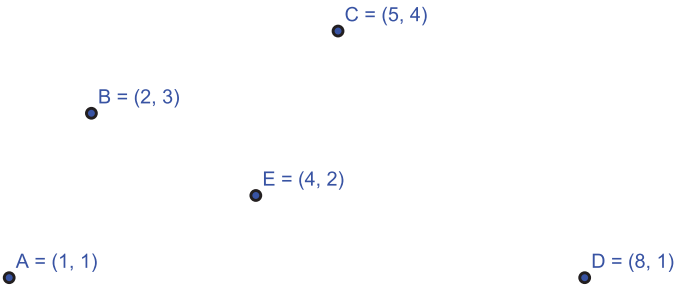

Fig. 1. Partial sum dispersion metric: a numeric example.

Table 1
Distances between candidate sites in Fig. 1.

Distance	A	B	C	D	E
A	0	2	5	7	3
B	2	0	3	6	2
C	5	3	0	4	2
D	7	6	4	0	4
E	3	2	2	4	0

For a particular three-facility set such as $\{A, C, D\}$, the partial sum metric is calculated by computing the sum of smallest two inter-facility distances for each facility. Referring to Table 1, the partial sums of distances to $L = 2$ neighbors for A, C and D are 12, 9 and 11, respectively. Adding the $K = 1$ partial sums, the partial-sum dispersion metric for this facility set is 9. Enumerating all 10 possible 3-facility combinations, one can verify that the facility set $\{A, C, D\}$ is actually the greatest possible, and $\{A, C, D\}$ is an optimal solution for this small example.

In a companion paper (Lei & Church, 2013), we developed an assignment formulation (MaxPSumPSumA) of the partial dispersion problem. Here, we show that a more compact formulation of the partial-sum dispersion problem (called MaxPSumPSumC) can be developed using a linear program by Ogryczak and Tamir (2003) that maximizes the sum of the smallest k elements in a set of numbers. The following notation is needed for MaxPSumPSumC:

I is a finite set of points representing potential facility sites.
i, j are indices used to represent facility sites, where $i \in I$, $j \in I$, $I = \{1, 2, \ldots, n\}$.
d_{ij} = separation distance between sites i and j.
d_{ij} = the sum of the smallest $L + 1$ distance assignments for site i, and
$M_i = L \cdot \max_{j \in I} d_{ij}$ is a sufficiently large number (which is chosen to be larger than any possible partial sum value for each i).

The decision variables are:

$y_i = 1$, if a facility is located at site i, and 0 otherwise.
t and u_i are auxiliary variables used in Ogryczak and Tamir (2003)'s program to compute the sum of the K smallest facility based partial sums. As will be explained shortly, u_i is non-zero if i is associated with one of the K smallest partial sums, and zero otherwise.

u_i and z_{ij} are similar auxiliary variables. z_{ij} is non-zero when j is one of the $L + 1$ closest assignments to site i, and zero otherwise.

The variable z_{ij} serves a similar role as variable u_i except that it pertains to the sum of $L + 1$ closest assignments to site i. Note that we need the sum of the $L + 1$ closest facilities to i because it will include the zero self-distance from i to itself (i.e. $d_{ii} = 0$). That is, the sum of $L + 1$ smallest distances will equal the sum of distances to the L closest *neighboring* facilities. With this notation, the model formulation is as follows:

MaxPSumPSumC:

$$\text{maximize } Kt - \sum_{i \in I} u_i \tag{1}$$

subject to:

$$t - u_i \le q_i \quad \text{for each } i \in I \tag{2}$$

$$q_i = (L+1)s_i - \sum_{j \in I} z_{ij} \quad \text{for each } i \in I \tag{3}$$

$$s_i - z_{ij} \le y_i d_{ij} + M_i(2 - y_i - y_j) \quad \text{for each } i \in I,\ j \in I \tag{4}$$

$$\sum_{i \in I} y_i = p \tag{5}$$

$$u_i \ge 0,\ z_{ij} \ge 0 \quad \text{for each } i \in I, j \in I \tag{6}$$

$$y_i \in \{0, 1\}. \tag{7}$$

To understand this formulation, we need to explain the linear program of Ogryczak and Tamir (2003) which maximizes the sum of the K smallest quantities in an array. This linear program has actually been employed twice here. First of all, it is applied at the system level to choose the sum of the K smallest of all facility-based partial sums $\{q_i\}$, $i = 1, 2, \ldots, n$. More specifically, the system-level program is:

$$\text{maximize} \quad Kt - \sum_{i=1}^{n} u_i \tag{8}$$

subject to:

$$t - u_i \le q_i \quad \text{for all } i \in I \tag{9}$$

$$u_i \ge 0 \quad \text{for all } i \in I \tag{10}$$

A formal proof of the correctness of the linear program (8)–(10) can be found in Ogryczak and Tamir (2003). One can also see why (8)–(10) can select the sum of K smallest facility-based partial-sum $\{q_i\}$ as follows.

Let $q^{(s)}$ be the sth smallest member of $\{q_i\}$ and let $\pi(s)$ be the index of $q^{(s)}$. By (9), the maximizing objective function in (8) clearly satisfies the following inequality:

$$Kt - \sum_{i=1}^{n} u_i \le Kt - \sum_{s=1}^{K} u_{\pi(s)} \le \sum_{s=1}^{K} q^{(s)}.$$

Thus, the sum of K smallest elements of $\{q_i\}$ (the right hand side) is an upper bound on the objective (the left hand side). But this upper bound can be achieved easily when one sets $u_{\pi(s)} = 0$, for $s > k$ and $u_{\pi(s)} = t - q_{\pi(s)}$, for $s \le k$ on the left hand side. Therefore, the optimal objective value of (8) is equal to the k smallest facility-based partial-sums $\sum_{s=1}^{k} q^{(s)}$. This construct is especially valuable in that it is possible to identify the sum of the K smallest elements without ordering them by value and summing the first K smallest elements.

Secondly, the same construct (8)–(10) can be modified and used to compute each facility-based partial sum q_i. Here, instead of maximizing the sum of K smallest partial sums, we maximize for each site i the sum of the smallest distances to L neighboring facilities. Considering the zero self-distance $d_{ii} = 0$, we need to maximize the sum of smallest $L + 1$ distances to any given facility $i \in I$, which amounts to solving the following "lower-level" maximization problem for each $i \in I$:

$$\text{maximize} \quad q_i = (L+1)s_i - \sum_{j \in I} z_{ij} \tag{11}$$

subject to:

$$s_i - z_{ij} \le y_i d_{ij} + M_i(2 - y_i - y_j) \quad \text{for all } j \in I \tag{12}$$

$$z_{ij} \geq 0 \quad \text{for all } j = 1, 2, \ldots, n. \tag{13}$$

Compared with (8)–(10), in the lower-level program (11)–(13), the number of elements to sum is $L+1$ instead of K; the elements to be summed are distances to site i (i.e., d_{ij}) instead of the facility-based partial sums (q_i); and the objective value being maximized is an individual facility-based partial sum q_i instead of the partial sum of the smallest q_i's. Of note, is that in the right hand side of (12), in order to compute the facility-based partial sum we have to select from distances to open facilities only and not just any d_{ij}. To achieve this, we resort to using a sufficiently large value M_i defined for each site i. The right hand side (RHS) of (12) equals d_{ij} only if both sites i and j are selected for facility location (i.e., $y_i = 1, y_j = 1$). Otherwise, the RHS of (12) is at least M_i and will not be selected into the smallest $L+1$ distance values by the definition of M_i.

Thirdly, at the higher level we need to maximize the K smallest q_i's obtained from (11) through (13). This amounts to solving the higher level program (8)–(10) with the understanding that q_i is defined by the optimal value of (11)–(13). Since the higher-level program also aims to maximize the q_i's, we can simplify the two-level maximization problem into one-level maximization by dropping the "maximize" operator in (11) to obtain (3) and therefore derive the MaxPSumPSumC formulation in (1)–(7).

Of note, the Ogryczak and Tamir (2003) program can be used to find the smallest k numbers in an array without explicitly sorting distances. This construct suffices for the partial-sum dispersion problem considered here. However, as discussed in Lei and Church (2013), the multi-level closest assignment (MLCA) constraints (Lei & Church, 2011) will be needed to define the specific ordering of neighboring facilities, if propulsion factors are used based on the relative rank of distances from neighboring facilities. This assignment-based formulation (called MaxPSumPSumA and listed in Appendix A) consequently has a higher problem dimension and involves L times more continuous variables and about 30 percent more constraints than MaxPSumPSumC. The reader is referred to Lei and Church (2013) for properties of the assignment formulation.

3. A branch and bound algorithm for the unified dispersion problem

The general partial-sum dispersion problem (i.e. MaxPSumPSum) is NP-hard since one of its special case problems, the p-dispersion problem, is NP-hard. This means that, in general, no polynomial bounded time algorithm will likely exist to solve the problem optimally. Our experiments, presented in the next section, show that although the MaxPSumPSumC formulation can be used to solve many medium sized problems optimally using commercial Integer-linear programming solvers, the computational cost for the general dispersion problem is still relatively high. The complexity of the general dispersion problem calls for efficient, specialized solution procedures. To address this issue, we have developed a branch and bound solution algorithm for the general dispersion problem. Our experiments show that the method is effective and orders of magnitude faster than the integer linear programming formulation described in the previous section or the assignment formulation in Lei and Church (2013). A heuristic is used to jump-start the branch and bound procedure. We will describe the heuristic first.

The heuristic procedure we propose is based on a greedy drop strategy and the interchange heuristics that has been widely used in the operations research literature, including location-allocation problems such as the p-median problem (see e.g. Drezner & Drezner, 2007; Rosing & ReVelle, 1997; Teitz & Bart, 1968; Whitaker, 1982). The work flow is,

1. Initially, all sites are added to the set S of selected facilities. For any given S, the dispersion objective Z is calculated according to (1) by summing the smallest L distances for each facility and adding the K smallest partial sums.
2. (Greedy drop) Each facility is tentatively removed from S. The facility that, if removed, generates the greatest PSum dispersion objective Z is deleted from S. This step is repeated until p facilities are left in S.
3. (Interchange) For each facility x in S,
   Remove x from S
   For each site y in $I \backslash S$,
   Add y into S

   Using (1) to compute the dispersion objective Z associated the p-facility set S, and the new dispersion objective Z_y associated with $S \cup \{y\} \backslash \{x\}$. Define the net change δ_y in the dispersion objective as $\delta_y = Z_y - Z$.

   If δ_y is positive and $\delta_y > \delta_{y'}$ for all previously tested site y', let $y_{\max} = y, x_{\max} = x$.
4. Repeat Step 3 until no improvement can be made (i.e. no positive δ_y can be found). Stop and return $S \cup \{y_{\max}\} \backslash \{x_{\max}\}$ as the heuristic solution.

The exact method we propose is a depth-first branch and bound process inspired by Pisinger (2006). Pisinger (2006) derived multiple upper bounds for the p-dispersion problem and the p-dispersion sum problem, respectively. This includes, in particular, a tight upper bound for the p-dispersion sum problem based on relaxing the cardinality constraint (5) and a symmetric property of the p-dispersion sum problem. More specifically, Pisinger observed that in the p-dispersion sum problem, when the distance d_{ij} from i to j is in the objective value (i.e. when both i and j are selected), d_{ji} must also be in the objective function. Consequently, the distance d_{ij} can be changed to $d_{ij} + \lambda_{ij}$, as long as λ_{ij} satisfy $\lambda_{ij} + \lambda_{ji} = 0$. Based on this property, Pisinger (2006) was able to develop tightened upper bounds by minimizing bounds over all λ_{ij}, subject to $\lambda_{ij} + \lambda_{ji} = 0$. Unfortunately, the generalized dispersion problem does not have this symmetric property as the p-dispersion sum problem does, because d_{ji} is not necessarily in the partial sum of j when d_{ij} is in the partial sum of i. We develop an upper bound based on relaxing the requirement that each facility can be only linked to open facilities. This requirement corresponds to relaxing the Balinski constraint (20) in the assignment formulation in Appendix A (Lei & Church, 2013), which requires that only distances to open facilities are counted in partial sums.

The branch and bound procedure starts from an initial solution, which in our case is the output of the interchange heuristic. The objective function of the initial solution is used as a lower bound on the optimal objective value as it is a feasible solution and any optimal solution will be at least as good as the initial feasible solution. As the branch and bound process progresses, the lower bound is updated with the best/greatest feasible objective value.

In our branch and bound tree, a site at any node of the tree can only be in one of the three states: fixed in, fixed out or free. If a site is fixed out, it is excluded from further consideration in any descendants of the current node. If it is fixed in, it must be included in the final solution in any descendants. Otherwise, the site is free. Since the Balinski condition is relaxed, a site i can be assigned to any other site j as long as j is not fixed out. There are two cases that need to be handled. In order to compute an upper bound u_i for the partial sum s_i for a given site i,

a) For sites that are fixed in (i.e. in the final solution), we can safely choose the smallest distances from them to i to include in u_i. If a free site that is closer than all current fixed-in sites is fixed in later, the dispersion objective will only be reduced.
b) For sites that are free, we should choose the larger distances from them to obtain an upper bound.

Therefore, to calculate u_i, we first sort the fixed-in sites in increasing distance order with respect to i. If the number of fixed-in sites

Table 2
Computational results for solving MaxPSumPSumC on the Kuby dataset using CPLEX.

p K L	Objective	Model C times[a]	Iterations	Nodes	Sites	Model A times[a]
5 1 1	35.22783	0.7	9895	1354	[1 5 19 21 25]	1.6
5 1 2	76.925823	1.6	29387	6402	[1 4 17 20 25]	10
5 1 4	236.805411	6.3	159176	49762	[3 18 19 23 25]	0.9
5 2 1	70.45566	1.0	15882	1822	[1 5 18 21 25]	3.8
5 2 2	154.751114	2.2	51748	8994	[1 5 18 21 25]	12
5 2 4	475.583077	14	340602	75963	[3 18 19 23 25]	3
5 5 1	198.575002	6.9	202369	39179	[2 16 19 20 25]	26
5 5 2	439.860115	12	401851	60857	[1 3 19 20 25]	57
5 5 4	1242.004422	31	1448895	106959	[1 2 18 19 25]	62
7 1 1	26.24881	0.7	14826	1603	[3 13 16 19 21 23 25]	1.5
7 1 2	59.933448	1.8	40605	8031	[2 5 16 19 20 22 25]	21
7 1 3	104.006959	4	101940	24373	[3 4 18 19 20 24 25]	58
7 1 6	323.541149	55	2219997	462318	[1 2 16 18 19 24 25]	1.4
7 2 1	52.49762	0.9	23202	1912	[3 13 16 19 21 23 25]	4.2
7 2 2	119.966774	2.7	68478	10289	[2 5 16 19 20 22 25]	50
7 2 3	210.747014	6	144815	23744	[2 3 16 19 20 24 25]	62
7 2 6	651.22826	122	4198715	727373	[1 2 16 18 19 24 25]	6.5
7 7 1	213.27974	49	1507882	326151	[1 3 13 16 19 20 25]	367
7 7 2	461.740276	79	2934791	493377	[1 4 16 19 21 23 25]	993
7 7 3	810.595348	95	3689721	519538	[2 3 16 19 21 23 25]	1156
7 7 6	2404.275342	284	14094445	921784	[1 2 16 18 19 24 25]	1325

[a] Solution times are in seconds. Note the average time in solving Model C was 36.94 seconds and Model A was 201.04 seconds.

Table 3
Computational results for solving MaxPSumPSumC on the Swain dataset using CPLEX.

p K L	Objective	Model C times[a]	Iterations	Nodes	Sites	Gap	Model A times[a]
5 1 1	27.6586	13	99116	11248	[12 24 39 51 52]	Optimal	233
5 1 2	56.7275	56	383227	51644	[12 35 39 51 52]	Optimal	451
5 1 4	150.556	1155	10210845	2883575	[12 24 39 51 52]	Optimal	31
5 2 1	55.3172	15	130510	8463	[12 24 39 51 52]	Optimal	723
5 2 2	114.9512	74	682615	101161	[12 24 39 51 52]	Optimal	615
5 2 4	302.6227	3113	32435549	4527186	[12 24 39 51 52]	Optimal	65
5 5 1	143.2394	309	2775724	596561	[12 24 39 51 52]	Optimal	17404
5 5 2	301.1074	1956	15352591	3123854	[14 24 39 51 52]	Optimal	21600
5 5 4	787.146	18035	150112823	6839916	[14 24 39 51 52]	Optimal	21600
7 1 1	21.095	14	136383	17728	[24 27 28 40 42 50 51]	Optimal	847
7 1 2	42.4726	86	813320	158091	[24 27 28 40 42 50 51]	Optimal	15023
7 1 3	70.6179	433	3025500	764901	[12 14 35 39 48 50 52]	Optimal	18823
7 1 6	204.1384	21600	166922752	52787992	[14 24 27 35 40 51 52]	52.89 percent	25
7 2 1	42.19	18	192710	22022	[24 27 28 40 42 50 51]	Optimal	1490
7 2 2	85.2902	166	1870434	280370	[2 27 28 35 40 50 51]	Optimal	16072
7 2 3	142.2925	831	6433249	1282540	[12 14 35 39 48 50 52]	Optimal	21600
7 2 6	408.6465	21600	259333291	39091179	[14 24 27 35 40 51 52]	134.13 percent	742
7 7 1	157.4154	15340	124478128	25176600	[3 14 24 28 39 51 52]	Optimal	21600
7 7 2	330.6336	21600	234756652	43651734	[2 14 24 28 39 51 52]	57.33 percent	21600
7 7 3	542.6349	21600	265376882	38997972	[14 24 28 39 50 51 52]	87.9 percent	21600
7 7 6	1505.2686	21600	376934102	7641083	[12 14 24 35 39 51 52]	424.2 percent	21600

[a] Solution times are in seconds. A 6 hour time limit (21,600 seconds) was imposed.

is equal to L (i.e. when a leaf node is reached), we select and add them (excluding the zero self-distance) to obtain the partial sum. If the number of fixed in sites is less than L, we add distances to all other fixed-in sites, and then sort the free sites in decreasing distance order and add the greatest distances until we have L entries to make the partial sum.

Our branch and bound procedure traverses the tree in solution space in the depth-first order. If the upper bound computed for any node is less than the best known lower bound, this branch is excluded from further consideration (fathomed). In choosing the order of which tree node to expand first, we try to fix in or fix out the lower indexed sites first. We have experimented with other orders of tree expansion such as alternating between high and low indexed sites, but the solution times were not significantly different. We have also experimented with using other heuristic strategies such as random starts, rather than greedy drop, but they did not appreciably add to the quality of the heuristic results.

4. Computational experiments

In this section, we present experimental results for the proposed integer linear programming formulation, the interchange heuristic, and the exact solution procedure. As with Lei and Church (2013), we use the 25 node dataset in Kuby (1987) and the 55 node dataset of Swain (1971), which are widely used in the literature. All computations were performed on an Intel i7 3770 3.4 gigahertz CPU with 32 gigabytes of system memory, which is the same configuration with the machine used in Lei and Church (2013). The operating system was Microsoft Windows Server 2008 R2. The heuristic and exact branch and bound method were programmed in standard C++ (C++ 2011) and compiled using GNU Compiler Collection (GCC) version 4.8. For solving the MaxPSumPSumC formulation we used IBM ILOG CPLEX 12.4.

Computational results for solving the MaxPSumPSumC formulation using CPLEX applied to the Kuby dataset are presented in

Table 4
Computational results for the heuristic and exact methods on the Kuby dataset.

p K L	Heuristic objective	Heuristic times[a]	Objective	Exact times[a]	Iterations	Nodes	Sites
5 1 1	20.6155	0.0156	35.2278	0.0312	2209	1104	[1 5 19 21 25]
5 1 2	58.2462	0	76.9258	0.0624	6021	3010	[1 4 19 20 25]
5 1 4	227.211	0	236.805	0.0156	547	273	[3 18 19 23 25]
5 2 1	41.2311	0.0156	70.4557	0.0312	2209	1104	[1 5 19 21 25]
5 2 2	144.733	0	154.751	0.0624	8083	4041	[1 5 19 21 25]
5 2 4	428.284	0	475.583	0.0156	1231	615	[3 18 19 23 25]
5 5 1	198.575	0.515	198.575	0.546	3599	1799	[2 16 19 20 25]
5 5 2	426.65	0	439.86	0.078	9077	4538	[1 3 19 20 25]
5 5 4	1238.3	0	1242	0.031	1999	999	[1 2 18 19 25]
7 1 1	11.1803	0.749	26.2488	0.811	15039	7519	[3 13 16 19 21 23 25]
7 1 2	50.7131	0	59.9334	0.078	11837	5918	[2 5 16 19 20 22 25]
7 1 3	86.6213	0	104.007	0.156	25093	12546	[3 4 18 19 20 24 25]
7 1 6	297.496	0	323.541	0.0156	877	438	[1 2 16 18 19 24 25]
7 2 1	22.3607	0.624	52.4976	0.7332	15039	7519	[3 13 16 19 21 23 25]
7 2 2	106.741	0	119.967	0.14	15155	7577	[2 5 16 19 20 22 25]
7 2 3	194.416	0	210.747	0.25	28357	14178	[2 3 16 19 20 24 25]
7 2 6	624.814	0.0156	651.228	0.047	1995	997	[1 2 16 18 19 24 25]
7 3 1	76.7001	0.733	81.0069	0.78	9169	4584	[2 5 16 19 21 23 25]
7 3 2	156.011	0	182.413	0.109	19275	9637	[1 4 16 19 20 22 25]
7 3 3	306.034	0	323.139	0.25	34795	17397	[2 3 16 19 20 24 25]
7 3 6	965.728	0.0156	988.128	0.031	4067	2033	[1 2 17 18 19 24 25]
7 7 1	199.953	0.0156	213.28	0.265	25637	12818	[1 3 13 16 19 20 25]
7 7 2	460.227	0	461.74	0.499	61345	30672	[1 4 16 19 21 23 25]
7 7 3	785.401	0	810.595	0.53	77073	38536	[2 3 16 19 21 23 25]
7 7 6	2392.26	0	2404.28	0.17	10197	5098	[1 2 16 18 19 24 25]

[a] Solution times are given in seconds.

Table 5
Computational results for the branch and bound method on the Swain dataset.

p K L	Heuristic objective	Heuristic timesa	Objective	Exact times[a]	Iterations	Nodes	Sites
5 1 1	16.1245	0.0312	27.6586	0.374	17731	8865	[12 24 39 51 52]
5 1 2	41.7919	0.0156	56.7275	0.546	26741	13370	[12 35 39 51 52]
5 1 4	127.966	0.0156	150.556	0.109	3101	1550	[12 24 39 51 52]
5 2 1	32.249	0.0156	55.3172	0.39	17731	8865	[12 24 39 51 52]
5 2 2	88.7477	0.0156	114.951	0.827	46989	23494	[12 24 39 51 52]
5 2 4	250.382	0.0312	302.623	0.218	5769	2884	[12 24 39 51 52]
5 5 1	114.717	0.0156	143.239	0.28	9623	4811	[12 24 39 51 52]
5 5 2	301.107	1.1232	301.107	1.716	46271	23135	[14 24 39 51 52]
5 5 4	770.413	0.0312	787.146	0.20	6043	3021	[14 24 39 51 52]
7 1 1	13.8924	0.0156	21.095	1.31	98621	49310	[24 27 28 40 42 50 51]
7 1 2	28.2627	0.0156	42.4726	2.2	204545	102272	[24 27 28 40 42 50 51]
7 1 3	56.597	0.0156	70.6179	4.74	418135	209067	[12 14 35 39 48 50 52]
7 1 6	178.546	0.0312	204.138	0.561	19915	9957	[14 24 27 35 40 51 52]
7 2 1	27.7848	0.0156	42.19	1.17	98621	49310	[24 27 28 40 42 50 51]
7 2 2	65.6087	0	85.2902	3.32	310873	155436	[2 27 28 35 40 50 51]
7 2 3	122.977	0.0156	142.292	7.63	699139	349569	[12 14 35 39 48 50 52]
7 2 6	391.428	0.0156	408.647	0.686	33429	16714	[14 24 27 35 40 51 52]
7 3 1	41.8205	0.0312	63.5676	1.326	123161	61580	[24 27 28 40 42 50 51]
7 3 2	95.9869	0.0156	130.538	3.51	434797	217398	[1 14 28 35 39 51 52]
7 3 3	200.439	0.0156	216.674	7.78	1082147	541073	[14 28 35 39 47 51 52]
7 3 6	601.223	0.0156	617.577	0.61	71955	35977	[12 14 21 24 39 51 52]
7 7 1	138.463	0.0156	157.415	1.28	93433	46716	[3 14 24 28 39 51 52]
7 7 2	293.088	0	330.634	6.536	509957	254978	[2 14 24 28 39 51 52]
7 7 3	506.347	0.0156	542.635	18.21	1427981	713990	[14 24 28 39 50 51 52]
7 7 6	1505.27	2.5896	1505.27	4.945	116459	58229	[12 14 24 35 39 51 52]

[a] Solution times are in seconds.

Table 2. Column 1 tabulates the problem parameters p, K and L. We tested the same problems instances as those in Lei and Church (2013). This includes two sets of parameters for dispersing five and seven facilities, respectively. For $p = 5$, problem instances in the first, third, seventh and ninth rows correspond to the p-dispersion problem, MaxMinSum, MaxSumMin and p-dispersion sum problems, respectively. Column 2 tabulates the optimal/best known objective value. Columns 3, 4 and 5 present the computational time in seconds, the number of iterations and the number of expanded nodes by CPLEX, respectively involving the MaxPSumPSumC formulation. Column 6 presents the optimal/best known dispersion pattern. Column 7 gives the computational time for the MaxPSumPSumA model reported in Lei and Church (2013). Compared with the MaxPSumPSumA model, the MaxPSumPSumC model solved 17 out of 21 problems faster than the MaxPSumPSumA model. For example, when $p = 7, K = 7, L = 3$, MaxPSumPSumC took 95 seconds to solve while MaxPSumPSumA took 1156 seconds. It is important to note that for problem instances where $L = p - 1$, the MaxPSumPSumA is often faster. For example, when $p = 7, K = 1, L = 6$, MaxPSumPSumC and MaxPSumPSumA took 55 and 1.4 seconds, respectively. On the average, Model C is

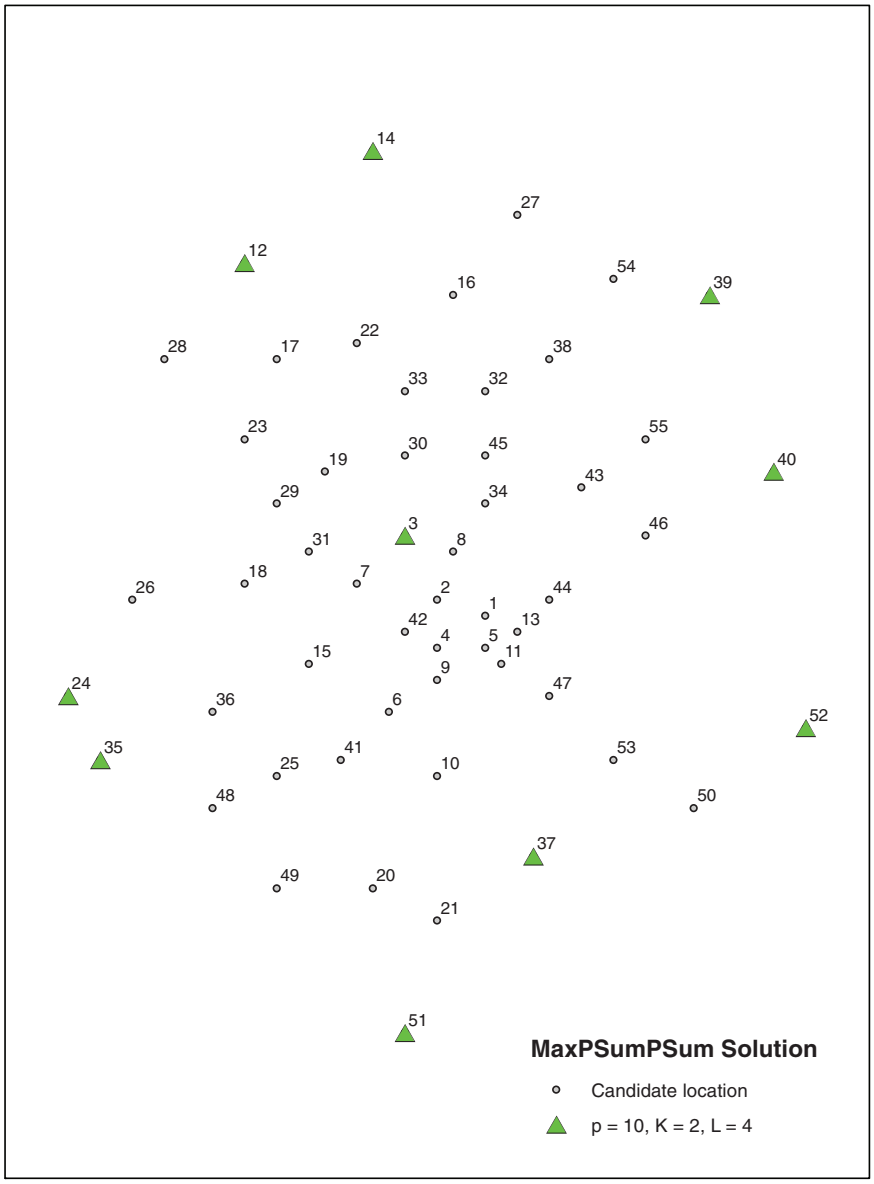

Fig. 2. An example solution for the partial sum dispersion problem.

considerably faster than the Model A results reported by Lei and Church (2013).

Table 3 presents results for solving the MaxPSumPSumC formulation using CPLEX involving the Swain dataset. The same problem parameters as Table 2 were tested. A 6 hour time limit was imposed, and the optimality gap is reported in column 6 of the table if a problem instance cannot be solved optimally. Computational times for the larger Swain dataset are greater. However, for MaxPSumPSumC, most (16 out of 21) instances can be solved to optimality. Column 7 presents the results of Lei and Church for the same problems using their MaxPSumPSumA model. The MaxPSumPSumA model was solved to optimality in 14 out of the 21 problem instances within the imposed maximum time limit. Compared with the assignment formulation, the MaxPSumPSumC formulation was able to be solved faster in 14 out of 18 comparable cases. For example, for the $p = 5$, $K = 1, L = 1$ case, MaxPSumPSumC and MaxPSumPSumA took 13 and 233 seconds respectively.

For more than two thirds of the tested instances in Tables 2 and 3, the MaxPSumPSumC formulation performs better than the assignment formulation. However, when the L value is large, the assignment formulation tends to perform better. While this property is difficult to prove in general, one possible explanation is as follows. In the MaxPSumPSumC formulation, the lower-level program (11)–(13) uses the "big M" method in formulating constraints (12), which is known to be integer unfriendly (for the y_i variables) and the larger the value of L is, the larger the natural gap is between the optimal and the relaxed problem. On the other hand, while the assignment formulation does not involve using "big M" based constraints, it does have a greater number of decision variables and constraints, as discussed earlier.

Overall, the MaxPSumPSumC is an improved formulation for the generalized dispersion problem, especially for small values of L. The new formulation is advantageous in facility dispersal applications such as franchise branch location, in which only the impact of a (small) number of nearby facilities need to be accounted for. For smaller values of K and L, medium sized problems can be solved optimally within 15 minutes using commercial solvers. However, computational times for larger K and L values may be long even if some of the problem instances were solved optimally within 3 hours. This reveals the combinatorial complexity of the generalized dispersion problem and the

limitations of solving them using a general purpose ILP solver. However, it should be noted that the formulation presented here appears to be distinctly better on the average than what has been reported in earlier work by Lei and Church (2013).

Table 4 presents the computational results for the interchange heuristic and the exact method on the same problem instances involving the Kuby dataset. Columns 2 and 3 tabulate the dispersion objective value obtained by the heuristic and the running time of the heuristic. Column 4 presents the objective values generated by the exact method and column 5 gives the running times for the exact method. In 22 out of the 25 tested instances, the interchange heuristic generated solutions that are greater than 70 percent of the optimal value (but less than the optimal value itself). Comparing Tables 2 and 4, the exact method generates the same optimal objective values as the ILP formulation. However, the overall solution time improved drastically. All tested instances were solved within a second, while the longest solution time was reduced from 284 seconds to 0.81 seconds when comparing the MaxPSumPSumC model to that of the exact tree search method. In most instances, the exact method is orders of magnitude faster.

Table 5 presents computational results of the heuristic and exact methods on the larger Swain dataset. In 20 out of the 25 tested instances, the heuristic generates dispersion objectives that are greater than 70 percent of the optimal. Solution times of the exact method are significantly shorter when compared to the ILP formulations. All problem instances can be solved optimally within 20 seconds. Overall, this represents a dramatic reduction in computational effort. This clearly shows the efficacy of the exact method.

For illustration, we show in Fig. 2 an optimal solution of MaxPSumPSum solution with 10 facilities and $K = 2$, $L = 4$ (adapted from Lei and Church, 2013). In this setting, the negative impact from the nearest four facilities is considered, and the two most impacted sites are considered in the dispersion metric. Note that for this example with relatively low K and L values, the assignment formulation took 9.5 days (819,858 seconds) to solve, whereas the branch-and-bound procedure obtained the same optimal value in just 195 seconds.

From Fig. 2, we can observe that the optimal solution to the partial sum dispersion problem differs from a traditional p-dispersion solution in that some facilities are allowed to locate relatively close to each other. This is the case, for example, for sites 24 and 35. However, the nearest four facilities are kept far away from any facility *on average*. This reduces the total negative impact from neighboring facilities, without counting distances of very far facilities in the dispersion objective. In locating missile silos, solutions like Fig. 2 increase the possibility of certain pair of facilities being taken out simultaneously, but reduce the risk that four (or L) facilities are lost simultaneously in one strike.

5. Conclusion

Facility dispersal is a basic optimization problem in operations research and location science. It has applications ranging from defensive location to increasing system robustness. In this paper, we begin with a short review of the facility dispersion literature. From the review, we can conclude that most models for facility dispersal are based on four dispersion criteria codified by Erkut and Neuman (1991). Lei and Church (2013) recently developed a generalized formulation that encapsulated the four basic dispersion criteria in a unified model. While it is theoretically possible to use their integer linear programming formulation to solve all four classic dispersion models, the solution times for their model can be rather long for certain problem instances. This may prevent the generalized dispersion model from being applied in practice.

This paper aims at developing efficient solution methods for the generalized dispersion problem. The contribution of the paper is twofold. First, we have developed an improved ILP formulation of the (unweighted) generalized dispersion problem based entirely on an efficient linear program by Ogryczak and Tamir (2003). This alternative formulation for the generalized dispersion problem not only requires fewer decision variables and constraints than previously developed models, but can be solved faster than previous formulations in a majority of tested instances.

Second, we present an interchange heuristic and an exact solution procedure based on depth first search branch and bound strategy. Our experiments reveal that the heuristic can generate solutions that are within 70 percent of the optimal value most of the time. Computationally, the heuristic and exact methods are significantly faster than what is required in using general purpose software applied to the integer-programming model presented here. All tested problem instances can be solved within a minute.

Overall, the computational methods proposed in this paper represent a significant improvement over the original formulation of the generalized dispersion problem, which can help operationalize models of facility dispersal in decision making and practical applications.

Acknowledgments

The authors would like to thank the two anonymous reviewers for their insightful comments and valuable suggestions, which greatly helped to improve the quality of the paper in terms of organization, presentation and experimental design. They are also grateful to Dr. Alan Murray and Dr. Tony Grubesic for their helpful discussion on the presented model during the 56th Annual North American Meetings of the Regional Science Association at San Francisco, CA, U.S.A.

Appendix A. The assignment formulation of the partial-sum dispersion problem

For comparison purposes, we present here a brief review of the assignment formulation of the partial sum dispersion problem called MaxPSumPSumA (Lei & Church, 2013). The following notation, in addition to what is presented in the beginning of Section 2 is needed:

- x_{ij}^l is a decision variable known as the assignment variable. It describes the closeness order of other facilities with respect to facility i. x_{ij}^l is 1 if sites i and j are selected as facilities and j is the lth closest facility to i. It is 0 otherwise. When x_{ij}^l is 1, we say that site j is assigned to site i at level l.
- $C_{ij} = \{q \neq i | d_{iq} < d_{ij}$ or $(d_{iq} = d_{ij}$ and $q < j)\}$ denotes the set of sites excluding i that are closer to i than j. A facility that is the same distance away from i with facility j but with a lower site index than j is also considered "closer" and included in this set. Altogether, the family of sets C_{ij} defines the closeness relation (or a topology) among candidate sites.

With these additional notation, the MaxPSumPSumA formulation is:

MaxPSumPSumA

$$\text{maximize} \quad Kt - \sum_{i \in I} u_i \tag{14}$$

s.t.

$$t - u_i \leq q_i \quad \text{for each } i \in I \tag{15}$$

$$u_i \geq 0 \quad \text{for each } i \in I \tag{16}$$

$$q_i = \sum_{l=1}^{L} \sum_{j \neq i} x_{ij}^l d_{ij} \quad \text{for each } i \in I \tag{17}$$

$$\sum_{j \neq i} x_{ij}^l \leq 1 \quad \text{for each } i \in I,\ l = 1, 2, \ldots, L \tag{18}$$

$$\sum_{l=1}^{L} x_{ij}^l \leq y_i \quad \text{for each } i \in I,\ j \in I \text{ where } i \neq j \tag{19}$$

$$\sum_{l=1}^{L} x_{ij}^l \leq y_j \quad \text{for each } i \in I,\ j \in I \tag{20}$$

$$\sum_{v \in C_{ij}} y_v + \sum_{l=1}^{L} (L + l - 1) x_{ij}^l \geq L(y_i + y_j - 1) \quad \text{for each } i, j \in I, \text{ where } i \neq j \tag{21}$$

$$\sum_{i=1}^{n} y_i = p \tag{22}$$

$$y_i \in \{0, 1\} \quad \text{for each } i \in I \tag{23}$$

$$0 \leq x_{ij}^l \leq 1 \quad \text{for each } l = 1, 2, \ldots, L,\ i \in I,\ j \in I \text{ where } i \neq j \tag{24}$$

Like the MaxPSumPSumC formulation (1)–(7), the MaxPSumPSumA formulation employs the Ogryczak and Tamir (2003) program (14), (15) and (16) to maximize the sum of the K smallest facility-based partial sums of distances. In fact, (14), (15) and (16) are identical to Eqs. (1)–(3). The MaxPSumPSumA formulation differs from the MaxPSumPSumC formulation in how the individual facility-based partial sums of distances are obtained. The MaxPSumPSumC formulation uses a lower-level maximization problem to maximize each facility-based sum of smallest L distances (q_i). By comparison, MaxPSumPSumA defines q_i in (17) using the multi-level closest assignment (MLCA) construct (18)–(21), which explicitly sort nearby facilities by distance, and selects the L smallest distance to make the facility-based partial sum. More specifically, constraints (18) maintains that only open facilities can be assigned to any given facility i. Constraints (19) and (20) are generalized Balinski constraints which maintain that only open facilities can be assigned to one another. Constraints (21) are MLCA constraints and maintain that if both i and j are open facilities (i.e. $y_i + y_j - 1 = 1$), $x_{ij}^l = 1$ if and only if j is the lth closest facility to i. The reader is referred to Lei and Church (2011) for detailed explanation of the construction of different forms of MLCA constraints and their properties.

References

Curtin, K. M., & Church, R. L. (2006). A family of location models for multiple-type discrete dispersion. *Geographical Analysis*, *38*, 248–270.

Curtin, K. M., & Church, R. L. (2007). Optimal dispersion and central places. *Journal of Geographical Systems*, *9*, 167–187.

Drezner, T., & Drezner, Z. (2007). The gravity p-median model. *European Journal of Operational Research*, *179*, 1239–1251.

Erkut, E. (1990). The discrete p-dispersion problem. *European Journal of Operational Research*, *46*, 48–60.

Erkut, E., & Neuman, S. (1991). Comparison of four models for dispersing facilities. *INFOR*, *29*, 68–86.

Feo, T. A., Resende, M. G., & Smith, S. H. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, *42*, 860–878.

Fernández, E., Kalcsics, J., & Nickel, S. (2013). The maximum dispersion problem. *Omega*, *41*, 721–730.

Gamarnik, D., & Goldberg, D. A. (2010). Randomized greedy algorithms for independent sets and matchings in regular graphs: Exact results and finite girth corrections. *Combinatorics, Probability & Computing*, *19*, 61–85.

Hifi, M. (1997). A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. *Journal of the Operational Research Society*, *48*, 612–622.

Kim, H. (2012). p-hub protection models for survivable hub network design. *Journal of Geographical Systems*, *14*, 437–461.

Kuby, M. J. (1987). Programming models for facility dispersion: The p-dispersion and maximum dispersion problems. *Geographical Analysis*, *19*, 315–329.

Lei, T. L., & Church, R. L. (2011). Constructs for multilevel closest assignment in location modeling. *International Regional Science Review*, *34*, 339–367.

Lei, T. L., & Church, R. L. (2013). A unified model for dispersing facilities. *Geographical Analysis*, *45*, 401–418.

Lei, T. L., & Church, R. L. (2014). Vector assignment ordered median problem: A unified median problem. *International Regional Science Review*, *37*, 194–224.

Maliszewski, P. J., Kuby, M. J., & Horner, M. W. (2012). A comparison of multi-objective spatial dispersion models for managing critical assets in urban areas. *Computers, Environment and Urban Systems*, *36*, 331–341.

Moon, I. D., & Chaudhry, S. S. (1984). An analysis of network location problems with distance constraints. *Management Science*, *30*, 290–307.

Nickel, S., & Puerto, J. (1999). A unified approach to network location problems. *Networks*, *34*, 283–290.

Ogryczak, W., & Tamir, A. (2003). Minimizing the sum of the k largest functions in linear time. *Information processing letters*, *85*, 117–122.

Pisinger, D. (2006). Upper bounds and exact algorithms for p-dispersion problems. *Computers & Operations Research*, *33*, 1380–1398.

Prokopyev, O. A., Kong, N., & Martinez-Torres, D. L. (2009). The equitable dispersion problem. *European Journal of Operational Research*, *197*, 59–67.

Ravi, S. S., Rosenkrantz, D. J., & Tayi, G. K. (1994). Heuristic and special case algorithms for dispersion problems. *Operations Research*, *42*, 299–310.

Rosing, K. E., & ReVelle, C. S. (1997). Heuristic concentration: Two stage solution construction. *European Journal of Operational Research*, *97*, 75–86.

Shier, D. R. (1977). A min-max theorem for p-center problems on a tree. *Transportation Science*, *11*, 243–252.

Swain, R. W. (1971, June). *A decomposition algorithm for a class of facility location problems*. Ithaca, New York: Cornell University.

Tamir, A. (1991). Obnoxious facility location on graphs. *SIAM Journal on Discrete Mathematics*, *4*, 550–567.

Teitz, M. B., & Bart, P. (1968). Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, *16*, 955–961.

Weaver, J. R., & Church, R. L. (1985). A median location model with non-closest facility service. *Transportation Science*, *19*, 58–74.

Whitaker, R. A. (1982). Some interchange algorithms for median location problems. *Environment and Planning B*, *9*, 119–130.