Yugoslav Journal of Operations Research
34 (2024), Number 3, 457–474
DOI: https://doi.org/10.2298/YJOR230215042P

A PROPOSAL TOWARDS A VNS-BASED DECISION SUPPORT TOOL FOR LARGE SCALE LOCATION-COVERING-TYPE PROBLEMS

In memory of my regretted friend Nenad Mladenović

Frank PLASTRIA
Prof. Em. BUTO, Vrije Universiteit Brussel, Belgium
Frank.Plastria@vub.be

Received: February 2023 / Accepted: November 2023

Abstract: We describe a number of typical features, constraints and objectives, frequently appearing in spatial system-design of maximum covering-type, such as emergency response systems. We then indicate some heuristic local search methods topped by a Variable Neighbourhood Search to construct and search for good solutions.

These ideas were intended to form the basis for a possibly multi-objective spatial decision support system.

Keywords: Location, covering, multiple objective, decision support, greedy heuristics, VNS.

MSC: 90B80 90C29 90C59.

1. INTRODUCTION

In 2005 I participated in a joint research project for the Belgian government involving several Belgian universities and a GIS-oriented private company in order to develop a decision support system for the evaluation and (re)optimization of the fire fighting services in Belgium, as reported in [1]. During the research discussions within the team and with several fire fighting actors as well as government officials many types of covering objectives and constraints were suggested, considered and tested, mainly using mixed integer linear programming solvers. These fell short on the one hand due to the sheer size of the problems with almost $20,000$ datapoints (the highest precision statistical sectors for which detailed data are available), which therefore had to be subdivided into 12 subregions (provinces and Brussels region), and for these subregions on the other hand due to model complexity when combining, as requested, several types of constraints. It was therefore clear that flexible heuristic methods were needed.

Around the same period I collaborated with Nenad Mladenović and Dragan Urošević, extending the variable neighbourhood metaheuristic developed by the first, see [2]. Convinced about the power and versatility of this method and its successful application in several kinds of location models (e.g. [3, 4]), I wrote a proposal for research into a variable neighbourhood approach (VNS) to the kind of covering problems we faced. By lack of time during the project, and because of direct availability of another metaheuristic (simulated annealing) my proposal was abandoned as unmature at that time and it was not rekindled since then. Being retired now for many years already I do not intend to do so, at least not on my own.

The present note is a somewhat updated version of this proposal, but remains just that: totally incomplete and untested, but yet another opportunity for showing the power and flexibility of Nenad's successful metaheuristic idea. Anyone is free to pursue these suggestions, but in this case please refer to this note, and feel free to contact me.

2. GENERAL NOTIONS AND NOTATIONS

In our covering type location problems there is a base set A of things to be covered (e.g. statistical sectors). There is also a set of potential sites S. Any solution will consist of a subset $C \subset S$ (of **C**hosen sites) satisfying certain conditions and/or optimising certain objectives.

Choosing a site $s \in S$ implies some fixed costs, described by the function

$$c \ : S \to \mathbb{R}_+ \ : s \mapsto c_s \tag{1}$$

This directly extends additively to solutions $C \subset S$ in the following evident way:

$$c(C) := \sum_{s \in C} c_s \tag{2}$$

The simplest case is obtained taking for c the constant cost-function with values 1, leading to $c(C) = |C|$, i.e. one just counts the number of chosen sites.

As soon as a site s is chosen, this implies that according to certain quality standards $q \in Q$ certain things $a \in A$ will be covered by it, as given by the set $A_q(s) \subset A$. A typical example of a quality standard is the combination of a distance measure d between sites and things and a threshold distance value $t_d^q > 0$: $A_q(s)$ then consists of all things which may be reached within d-distance t_d^q from s:

$$A_q(s) := \{ \ a \in A \mid d(s,a) \leq t_d^q \ \} \tag{3}$$

Different quality standards are then obtained when varying the threshold t_d^q and/or the distance measure d.

The most common example in emergency services is when d is a shortest path time-distance on a network (one might have to consider several such to cope with dayly/seasonal traffic-variations) and t_d^q an extraneously fixed maximal allowed service-time (one might again have to consider several such thresholds, because different types of risks may require different intervention times, and/or to consider second-order interventions, etc.). Note that the thresholds might also differ among things (different levels of protection). Further one frequently encounters additional regional limitations to coverage, due to administrative regulations.

Other examples where d is rather euclidean distance are the siting of wireless antennas, drone bases, alarm sirens or water sprinklers.

In many cases, in particular for mixed integer programming formulations (which will not be discussed here, but see e.g. [5, 6]), it is more convenient to work with the 'inverse' (or dual) covering sets, which carry the same information as the $A_q(s)$, but organized differently. For each quality $q \in Q$ and each thing $a \in A$ there is correspondingly the set $S_q(a)$ of sites which are able to cover a at quality q, defined as:

$$S_q(a) = \{ \ s \in S \mid a \in A_q(s) \ \} = \{ \ s \in S \mid d(s,a) \leq t_d^q \ \} \tag{4}$$

In fact any quality setting q may be represented by its characteristic function, a 0-1 matrix on the index set $A \times S$, with rows corresponding to things $a \in A$ and columns to sites $s \in S$, and each cell (a, s) containing either 1 or 0 according to whether a is covered by s at quality q or not. Row a then gives the characteristic function of $S_q(a)$, while column s gives the characteristic function of $A_q(s)$. Although often used this representation is, however, rather heavy since the matrix is usually sparse.

To each $a \in A$ and quality q one may also attach an attribute, which we will call *q-weights* $w_q(a)$, which allows to measure how much covering a at quality q is valued (e.g. in terms of population at home at a, people at work at a, risk of some incident at a, etc.).

To each solution $C \subset S$ we may now associate following sets and values:

- $A_q(C) := \cup_{s \in C} A_q(s) = \{ \ a \in A \mid S_q(a) \cap C \neq \emptyset \ \}$ the set of things covered by C at quality q
- $w_q^1(C) := w_q(A_q(C)) = \sum_{a \in A_q(C)} w_q(a)$ is the total q-weight covered by C at quality q

More generally, for any integer $k > 0$, one may associate with C the set of things covered at least k-times by C at quality q:

$$A_q^k(C) := \{ \ a \in A \mid |S_q(a) \cap C| \geq k \ \}$$

and the corresponding k-times covered q-weight

$$w_q^k(C) := w_q(A_q^k(C)).$$

Note that $A_q(C) = A_q^1(C)$, justifying the notation $w_q^1(C)$ introduced above.

In practice one will usually be confronted with multiple covering questions only for very small k, e.g. $k \leq 3$.

3. CONSTRAINTS AND OBJECTIVES

3.1. Constraints

Several typical constraints may be considered. The following list covers many of the proposals found in the literature on covering problems, either by themselves or combined with others.

3.1.1. Full covering

Full k-tuple q-covering expresses that all things should be covered at least k times at quality q, or formally

$$A_q^k(C) = A$$

3.1.2. Mandatory covering

Mandatory k-tuple q-covering of some given $A' \subset A$ expresses that certain things (those of A') must be covered at quality q. Formally

$$A_q^k(C) \supset A'$$

Full covering is equivalent to mandatory covering of $A' = A$.

3.1.3. Mandatory weight covering

Mandatory weight k-tuple q-covering of some given $A' \subset A$ means one requires that the total q-weight that is k times covered at level q within A' should be at least some given value W. Formally

$$w_q(A_q^k(C) \cap A') \geq W \tag{5}$$

The value W will very often more conveniently be expressed as a percentage $p \in [0,1]$ of the total q-weight of A': $W = pw_q(A')$. Choosing $p = 100\%$ (and assuming that weights are always strictly positive) will then be equivalent to mandatory k-tuple q-covering of A'. Hence this type of constraint encompasses both previous ones.

3.1.4. Fixed site choices

It may be required that some of the sites $F \subset S$ are fixed in advance as to be chosen. Formally

$$C \supset F$$

One might be tempted to try to 'simplify' by getting rid of fixed choices, simply deleting them from the set of possible sites. But this would call for an in depth adaptation of all other covering data concerning all things covered by F and it will also rather mess up the evaluations of covered weights. So I feel this might be a bad idea. Adding the fixed sites should simply be a first step in any heuristic, while no $c \in F$ should be allowed to be deleted, see section 4.

3.1.5. Budgets

Budgets express that what is chosen should not be too expensive. But there may be a single global and/or several regional budgets involved, related to possibly distinct budgetary aspects (recall that my country Belgium is regionalized in a quite intricate way).

Letting R be the set of (possibly overlapping) regions to be considered, we may define the regional possible sites $S_r \subset S$ ($r \in R$) and corresponding budgets B_r relative to the costfunction c_r defined on S_r. The budget constraints are then

$$\forall r \in R : c(C \cap S_r) \leq B_r \tag{6}$$

3.1.6. Regional minima and maxima

Political reasons sometimes compell the minimal presence of a number of service stations within regions, not in terms of covered things (which would be handled through mandatory weight covering constraints), but rather in terms of chosen sites. In their simplest form this means specifying minimal numbers n_r for each region $r \in R$ and require

$$|C \cap S_r| \geq n_r$$

Clearly taking $n_r = 0$ simply drops any regional minimum requirement on region r.

Similarly, environmental reasons may impose that within some regions not too many sites are chosen. This may be handled by a regional budget, using unit costs.

3.1.7. 'Enough' and 'No more' constraints

We can classify the constraints into two *opposing* types, akin to upper and lower bounds.

Enough Mandatory weight constraints (and thus also Full and Mandatory covering constraints) as well as Fixed site choices and Regional minima all call for having 'enough' sites chosen. More precisely they have the property that if violated by a choice C any subset of C still violates it, but there always exists a larger choice $C' \supset C$ that satisfies it, while (as a consequence) when satisfied by C any larger $C' \supset C$ will also satisfy it.

Note that an Enough constraint is non contradictory (i.e. admits valid solutions) if and only if the full choice $C = S$ of all sites satisfies it.

No more Budget constraints (and thus also regional maxima) rather call for reducing the chosen sites: if violated by a choice C any superset of C still violates it, but there always exists a smaller choice $C' \subset C$ that satisfies the constraint; it follows that when satisfied by C any smaller $C' \subset C$ will also satisfy it.

Any combination of several (non-contradictory) Enough constraints can simply be seen as a single Enough constraint and will never lead to a contradiction. Indeed the union of any family consisting of a feasible choice for each single constraint will satisfy the combination.

Combining several No more constraints can in principle also be seen as a single No more constraint for which the intersection of individual feasible choices will be feasible. This intersection might be empty though (i.e. none of the sites is chosen) which is usually not considered as acceptable.

One should however be aware that Enough constraints may contradict with No more constraints, in the sense that there simply is no solution at all that satisfies all constraints. Combining these two kinds of constraints in a same model should therefore be done carefully. One must make sure that choices C remain that satisfy them all. A well designed decision support system should be able to warn the user in the contrary case, but this is not an easy task, and will not be attempted.

3.2. Objectives

Most models will therefore rather consider one of these types of constraints as an objective, where the left hand side of an Enough constraint is to be maximized, while the left hand side of a No more constraint is to be minimized.

Typically one wants to cover 'everything' or 'as much as possible' at 'affordable' or 'least possible' cost. Thus there are two typical opposing kinds of objectives, which might be considered separately or simultaneously in a multi-objective way:

Maximising k-tuply q-covered weight:

$$\max w_q^k(C)$$

Minimising total cost:

$$\min c(C)$$

When only a single objective is present there should be at least one constraint of the opposing type present, otherwise the optimal solution would be trivial: when maximising without No more constraint at optimality all sites will be chosen ($C = S$), contrarily when minimising without Enough constraints at optimality none of the sites will be chosen ($C = \emptyset$).

With two (or more) objectives one may distinguish two cases:

- when both (or all) objectives are of the same kind they are usually treated in a lexicographic way: they are (totally) ordered by highest preference and optimized in this sequence as long as there remain several optimal solutions.
- otherwise with opposing objectives we should aim at non-dominance, see section 4.3.

3.3. Model

A model is a combination of constraints and at least one explicit objective opposing at least one of the constraints.

We will denote the set of all constraints of our covering model by $\mathcal{C}$ with generic element γ. This is subdivided into the set of Enough constraints $\mathcal{E}$ with generic element ϵ, and the set of No more constraints $\mathcal{N}$ with generic element ν.

For a constraint $\gamma \in \mathcal{C} = \mathcal{E} \cup \mathcal{N}$ we denote the value at the current solution C of the left-hand side by $l_\gamma(C)$ and the (fixed) right-hand side by r_γ. Note that typically l_γ is increasing, i.e. $C \subset C' \implies l_\gamma(C) \leq l_\gamma(C')$

An Enough (resp. No more) constraint ϵ (resp. ν) of inequality (5) (resp. (6)) will thus be satisfied at current solution C iff $l_\epsilon(C) \geq r_\epsilon$ (resp. $l_\nu(C) \leq r_\nu$), and violated iff $l_\epsilon(C) < r_\epsilon$ (resp. $l_\nu(C) > r_\nu$).

A maximization (resp. minimization) objective α (resp. ι) is assumed to always consist of an expression $l_\alpha(C)$ (resp. $l_\iota(C)$) similar to the left hand side of inequality (5) (resp. (6)).

Implicit objective. When no maximization objective is explicitly present we will assume that in fact there always is the unstated aim to cover the most possible things at least once at least quality level.

Similarly when no explicit minimization objective is given there always is the unstated aim to use (choose) the least possible number of sites.

We call these aims *implicit objectives.*

3.4. A few examples

Let us mention here a few references to seminal work that falls into the described framework. In view of the large and fastly growing literature on models of covering type any attempt at completeness would be an almost impossible task. Besides most of the recent research nowadays includes additional features (e.g. [7, 8, 9, 10, 11, 12, 13]. There are several survey papers treating (parts of) this material which, taken together, give a quite exhaustive picture, see e.g. (chronologically) [14, 15, 16, 17, 6, 18, 19] as well as the recent book [5] entirely devoted to the subject.

The earliest model is the Set Covering Problem, asking for full covering under a single quality by a minimum number of sites. It dates back to the 50's in the context of graph theory, see [20], and was introduced in emergency location in [21]. Since there usually are many optimal choices this was extended in [22] to maximize the number of twice covered things among these optimal solutions, i.e. two lex-max objectives.

Maximising the covered weight with a given number of sites was first introduced in [23], and suggested to be combined with mandatory covering of certain sites in [6] .

Backup coverage models require multiple covering to protect against failures and/or overflow in sites at the time of call. Ambulance location requiring double coverage are discussed in [11]. Another kind of application involves location of sensors that need to be able to localize the precise site of an incident, requiring detection by at least two sensors on a network, three sensors in planar environment (triangulation), and four sensors in 3-dimensional space.

The models proposed to the user in our fire fighting DSS for Belgium [1] combined the minimization of the number of chosen sites with a series of mandatory (single) weight covering objectives with differing characteristics corresponding to several kinds of incidents.

4. HEURISTICS

Many kinds of (meta)heuristics have already been applied with success to certain classical and more involved covering models, e.g. [24, 25, 4, 26, 27, 28, 29, 30, 40]. The aim of this section is to devise a number of easy and sufficiently general methods to search for 'good' solutions, which may be applied in many, if not all, situations falling in our framework described in section 3 .

4.1. Add and Drop steps

All heuristic ideas proposed in what follows consist of the combination of the two following basic kind of steps, starting from some initial choice C:

Add i.e. *adding* a site $s \in S \setminus C$ to C, resulting in the new choice $C_s^+ := C \cup \{ s \}$

Drop i.e. *deleting* a site $c \in C \setminus F$ from C, resulting in the new choice $C_c^- := C \setminus \{ c \}$.

Every heuristic is determined by the initial C, the particular choices of the added s or dropped c, and the sequence in which these operations are executed.

For a constraint-set $\mathcal{C}on$ we say that the current choice C is $\mathcal{C}on$-valid if C satisfies all constraints of $\mathcal{C}on$, otherwise C is $\mathcal{C}on$-invalid, i.e. C violates at least one constraint of $\mathcal{C}on$.

Concerning the full validity status of the current choice C we may then distinguish the four following situations:

improper i.e. C is both $\mathcal{E}$-invalid and $\mathcal{N}$-invalid

proper $\mathcal{E}$-invalid i.e. C is $\mathcal{E}$-invalid, but $\mathcal{N}$-valid

proper $\mathcal{N}$-invalid i.e. C is $\mathcal{E}$-valid, but $\mathcal{N}$-invalid

feasible i.e. C is both $\mathcal{E}$-valid and $\mathcal{N}$-valid.

Evidently the final solution(s) found should be feasible.

When C is $\mathcal{N}$-invalid any Add step will yield $\mathcal{N}$-invalidity again, while a Drop step on a $\mathcal{E}$-invalid choice always yields $\mathcal{E}$-invalidity.

It follows that any Add or Drop step on an improper C just yields another improper choice, so will never lead to feasibility. Since all our heuristics are Add/Drop based, improper choices must therefore be avoided.

If not stated otherwise, in all what follows it will be assumed that C is a proper (i.e. not improper) choice. Note that any time C is initialized one should therefore try to make sure it is proper. As soon as some feasible choice is known to exist, this choice is certainly proper, as well as the extreme choices $C = \emptyset$ and $C = S$.

When C is proper $\mathcal{E}$-invalid we will try to obtain feasibility by way of Add steps. All Enough constraints that were valid will remain so, while the new chosen site(s) should be aimed at making the violated Enough constraints valid.

Similarly, when C is proper $\mathcal{N}$-invalid we will try to gain feasibility by way of Drop steps. This guarantees that the resulting choices remain valid for all already valid No more constraints, and should be aimed at making the violated No more constraints valid.

Feasible choices C might possibly still be improved in view of the objective(s), so may be used to start a heuristic improvement attempt. This latter may start by using a series of Add (resp. Drop) steps, thereby possibly leading to violation of some No more (resp. Enough) constraints (but not to an improper choice set), which will then need to be brought back to feasibility by another sequence of steps of opposite type. This back and forth strategy may (need to) be repeated, see section 4.5.

4.2. Site-choice

For any Add or Drop step a site has to be chosen, an $s \in S \setminus C$ for Adding or a site $c \in C$ for Dropping. One must however make sure that such a step does not produce an improper choice.

We will call a site $s \in S \setminus C$ *addable* (to the $\mathcal{N}$-valid choice C) if C_s^+ is still proper. This means that C_s^+ may only become $\mathcal{N}$-invalid in case it has gained $\mathcal{E}$-validity.

A site $c \in C \setminus F$ is called *droppable* (from the $\mathcal{E}$-valid choice C) if C_c^- is still proper. In other words, if C_c^- has become $\mathcal{E}$-invalid it must at the same time have become $\mathcal{N}$-vaild.

If there is no addable (droppable) site we say that C is Add- (resp. Drop-) *blocked*.

4.2.1. Validity gain evaluation

When an addable site $s \in S \setminus C$ is considered for an Add step to C one may evaluate a *worst case gain* in covering validity as follows.

For each $\epsilon \in \mathcal{E}$ that is violated by C (i.e. $l_\epsilon(C) < r_\epsilon$) we calculate how much s would contribute to validity when added to C:

$$gain_\epsilon(s) := \min(l_\epsilon(C_s^+), r_\epsilon) - l_\epsilon(C) \geq 0$$

and take the worst case gain in relative terms to what is to be achieved for each such ϵ

$$gain(s) := \min\{ \frac{gain_\epsilon(s)}{r_\epsilon - l_\epsilon(C)} \mid \epsilon \in \mathcal{E} \text{ violated by } C \}$$

Note that $0 \leq gain(s) \leq 1$ (except when C is already $\mathcal{E}$-feasible, and then all gains are undefined). Reaching the upper bound of 1 indicates that full $\mathcal{E}$-validity would be gained by adding s. On the other hand $gain(s) = 0$ only means that at least one violated $\mathcal{E}$-constraint does not improve.

Similarly, when a droppable $c \in C \setminus F$ is considered for being Dropped a *worst case gain* in $\mathcal{N}$-validity may be calculated as follows:

For each $\nu \in \mathcal{N}$ that is violated by C (i.e. $l_\nu(C) > r_\nu$) we calculate how much c would contribute to validity when dropped from C:

$$gain_\nu(c) := l_\nu(C) - \max(l_\nu(C_c^-), r_\nu) \geq 0$$

and take the worst case gain in relative terms to what is to be achieved for each such ν

$$gain(c) := \min\{\ \frac{gain_\nu(c)}{l_\nu(C) - r_\nu}\ |\ \nu \in \mathcal{N} \text{ violated by } C\ \} \tag{7}$$

Note again that $0 \leq gain(c) \leq 1$ (except when C is already $\mathcal{N}$-feasible and gains are undefined), where reaching the upper bound of 1 indicates that full $\mathcal{N}$-validity would be gained by Dropping c, while value 0 indicates that at least one violated $\mathcal{N}$- constraint does not improve.

Note. One might also consider other types of gain evaluations, e.g. average, median or best case gain replacing min by the average, median or max operator, but then the just mentioned interpretation of $gain = 1$ will not hold anymore as soon as there are more than one Enough (resp. No more) constraints.

4.2.2. Greedy site-choice

A greedy site choice means that we try to make the best possible step towards feasibility. However, in presence of opposing objective(s) (which might be implicit, see section 3.3), we must take into account that each site choice involves a possibly different deterioration of these objectives. This may be done in a way similar to the 'bang for bucks' trick for continuous knapsack problems:

- For an Add step we consider all addable sites $s \in S \setminus C$ and choose a site s which maximizes

$$\frac{gain(s)}{\max\{\ l_\iota(C_s^+) - l_\iota(C)\ |\ \iota \text{ min objective }\}} \tag{8}$$

- For a Drop step we consider all droppable sites $c \in C \setminus F$ and choose a site c which maximizes

$$\frac{gain(s)}{\max\{\ l_\alpha(C) - l_\alpha(C_c^-)\ |\ \alpha \text{ max objective }\}} \tag{9}$$

Note that there might be (and often will be, in particular with implicit objective) ties, so that a greedy choice of site is not necessarily unique. Resolving such ties is most easily done randomly, but better (and only somewhat harder) by iterated max choice (i.e. for equal first max, take second max, etc.).

4.3. Reduction to smallness

Considering the two objectives mentioned in section 3.2, we say that a solution C *dominates* solution C' as soon as C covers more weight at less costs, i.e. $w_q^k(C) \geq w_q^k(C')$ and $c(C) \leq c(C')$ with at least one strict inequality. Clearly in such a case C is to be preferred to C'. Therefore we are only really interested in non-dominated solutions.

However, it is in general quite hard to check non-dominance exactly. It would call for solving to optimality some set-covering type combinatorial optimization problem, which is known to be NP-hard, being an extension of the classical NP-hard set covering problem [31], thus requiring a mixed binary linear programming formulation (e.g. using standard tools such as described in [32, 33, 34]) and branch and bound or other solution techniques that do not perform well for large scale problems.

We may, however, eliminate certain more easily detected (but much less frequent) cases of dominance using the following simple observations:

- We will call a feasible solution *small* if no proper subset remains feasible for all Enough constraints and does not reduce any maximization objective.
- Clearly any solution C that is not small is dominated, so only small solutions may be non-dominated, so be of possible interest.
- C is quite easily checked to be small. Indeed one only has to check that dropping any $c \in C \setminus F$ will either invalidate the choice or decrease a max objective.
- Contrarily, if some $c \in C \setminus F$ exists such that C_c^- remains feasible and has same max objective values, we know that C is not small and the just found *superfluous* c may be Dropped.

As a first operation, any non-small solution C may be stepwise reduced to a small solution, by iterative dropping of a superfluous site $c \in C$, until no such may be found. This is a fairly easy task that does not require validity gain calculations.

Therefore any solution seriously considered as possibly adequate should undergo this operation.

Note, however, that it is not uniquely defined: there may be several superfluous sites in C, but which may not be all simultaneously dropped without losing feasibility or max objective value. Therefore different drop-sequences may lead to different small solutions. In such a case the (possibly implicit) min objective may guide the dropping choice(s).

4.4. Greedy sequences

A greedy sequence is a series of successive greedy site-choices of same type (Add or Drop resp.) as long as applicable.

More precisely:

1. Start from some initial solution C. For an Add-sequence C must be proper $\mathcal{E}$-invalid, resp. for a Drop-sequence proper $\mathcal{N}$-invalid.
2. Repeat successive greedy Add (resp. Drop) steps (see section 4.2.2) as long as possible.

The end result can be of three kinds: feasible, proper oppositely invalid, or blocked.

4.5. Local search

Successive back and forth greedy sequences yield a form of local search starting from some infeasible (but proper) solution C, as follows

1. Apply an appropriate greedy sequence (Add when $\mathcal{E}$-invalid, Drop when $\mathcal{N}$-invalid).
2. In case the resulting choice is feasible we have obtained a solution, which we reduce to smallness (section 4.3) and stop.
3. In case the new choice has become valid, but of opposite invalidity the local search is continued by an opposite greedy sequence.
4. In case of block we are in trouble. If no feasible solution is known as yet, this situation seems to indicate that the model might be infeasible, but this conclusion is far from certain and needs further confirmation. Anyway, the local search terminates here without result.

If in step 3. the same site is uniquely chosen by two successive Add and Drop steps, the local search would enter an endless loop, so in this exceptional case one should stop without result. Anyway, just to be certain that the local search always ends one may limit the number of back and forth steps, and when this limit is reached terminate without result.

4.6. Greedy construction and destruction

Greedy construction consists of starting with $C = \emptyset$, followed by Add steps of all fixed sites $s \in F$ (if present), followed by local search.

Greedy destruction starts with $C = S$, followed by local search.

4.7. Variable Neighbourhood Search

Variable neighbourhood Search (VNS) methods (see [35, 36, 37]) have been quite successful for tackling many kinds of location problems (e.g. [4, 38, 39, 40, 41]), along many other optimization problems. They rely on the definition of neighbourhoods for any solution on several levels of 'closeness'.

Many variants may be devised based on the basic scheme delined below by varying the specification details of the elements indicated in italic, each of which are afterwards discussed separately together with one or some possibilities.

4.7.1. VNS scheme

The basic VNS scheme for optimization is as follows

```
• Start at *some solution* $C$, and set $n = 1$.
• Repeat
    – *Search a bit* within some *n-th level neighbourhood* of $C$.
    – If this produced a better solution, set $C$ to this new solution
      and $n = 1$
    – Otherwise increase $n$
  Until some *stopping rule* is reached.
• The final $C$ (if a feasible one was found) is then exhibited as can-
  didate optimal solution. Otherwise the problem is declared to be
  infeasible.
```

Previous scheme is easily adapted tot multiple-objective situations, where the aim is to produce a list of good candidate nondominated solutions. A provisional list of currently nondominated solutions is kept, containing all feasible solutions found that are not dominated by any of the previously inspected feasible solutions.

```
• Start at *some solution* $C$, set $List = \emptyset$ and set $n = 1$.
• Repeat
    – *Search a bit* within some *n-th level neighbourhood* of $C$.
    – If this produced a feasible solution nondominated by any ele-
      ment of the *List*, set $C$ to this new solution and $n = 1$. Also
      add $C$ to *List* and delete all elements of the *List* dominated by
      $C$.
    – Otherwise increase $n$
  Until some *stopping rule* is reached.
• The final *List* contains all candidate nondominated solutions. This
  might still be empty, in which case the problem is declared to be
  infeasible.
```

4.7.2. 'some solution'

Any feasible solution may best be used as starting solution. This might for example be some solution found by a greedy construction or destruction.

If no feasible solution is known one may select any random solution at the risk that it is improper.

4.7.3. 'n-th level neighbourhood'

We propose the two following types of neighbourhoods. The first type corresponds to quite traditional types of neighbourhood used in VNS schemes for general combinatorial problems, but totally disregards the particular spatial distance based setting that underlies location covering problems. The second type explicitely takes this spatial setting into account, and is probably better suited to the models considered here. In fact both types may be mixed in a random way, which might produce even better results, similar to the formulation space idea (see [42, 43]).

set neighbourhoods

The n-th level set neighbourhood of a solution $C \subset S$ consists of all solutions differing from C by at most n elements:

$$N_n(C) := \{ C' \subset S \mid |C \Delta C'| \leq n \}$$

where Δ is the symmetric difference defined by $C\Delta C' = (C \cup C') \setminus (C \cap C') = (C \setminus C') \cup (C' \setminus C)$.

Thus, $N_1(C)$ consists of all solutions obtained by dropping a site from C or by adding a new site to C. $N_2(C)$ additionally contains all solutions obtained by either dropping two sites from C, or adding two new sites to C, or dropping a site from C and adding another site to C.

A (random) selection in $N_n(C)$ is obtained by (randomly) selecting with possible repetition n elements of S and among these Dropping all those that are in C while Adding all the others.

One may also define set neighbourhoods $N_{\ell,m}(C)$ by specifying separately that at most ℓ sites may be dropped from C and at most m new ones added (ℓ stands for ℓess, m for $m$ore). In fact this is just a further refinement of previous neighbourhoods since $N_n(C) = \bigcup\{ N_{\ell,m}(C) \mid \ell + m = n \}$.

spatial neighbourhoods

When the model includes covering for some quality q defined through a distance d and threshold t_d^q as in (3), (4) we may define spatial neighbourhoods of C as sets of rearrangements of C within some region of d-size governed by n. More precisely, a n-th level spatial neighbourhood of C consists of all solutions that differ from C only in sites inside a region of radius e.g. n times the threshold t_d^q. Formally:

$$N_n^d(C) := \{ C' \subset S \mid \exists s \in S : \forall c \in C' : d(c,s) > nt_d^q \implies c \in C \}$$

A (random) selection in $N_n^d(C)$ consists of choosing a (random) $s \in S$, fixing all sites $c \in C$ with $d(c,s) > nt_d^q$ and adding a (random) selection out of $\{ c \in S \mid d(c,s) \leq nt_d^q \}$.

4.7.4. 'Search a bit'

Any solution in the neighbourhood may be taken as the starting point of local search. In case of a blocked situation the randomization will hopefully unblock it.

One may decide to try a single random neighbour, or a given number of these. For small n a complete search of the neighbourhood may be attempted.

4.7.5. 'stopping rule'

'Time is up' is the usual rule.

Another more natural one, but probably unreachable in large scale situations, is when n cannot be increased.

5. SOME IMPLEMENTATION CONSIDERATIONS

Since Add and Drop operations are to be used very frequently it is important to be able to implement them and evaluate their effect efficiently.

This means that one must be able to update rapidly the basic information pertaining to the evaluation of the covering when performing an Add or Drop step. This basic information mainly consists of the sets $A_q^k(C)$ and their evaluations $w_q^k(C)$ for all relevant q and k. For greedy steps one must be able to efficiently choose the best site, so to evaluate all the relevant gains.

Storing additional information may allow to speed up considerably feasibility testing and gain calculations.

For example, keeping track for each $a \in A$ and $q \in Q$ of the number of times $C_q(a)$ that a is covered at quality q by the current site-choice C will be extremely useful. Adding (Dropping) the site $s \in S \setminus C$ $(s \in C)$ calls for a simple updating of the $C_q(a)$: for each q and $a \in A_q(s)$ add (subtract) 1.

This information might even be more important than the $A_q^k(C)$, while allowing linear time retrieval of these latter sets because of the fact that $C_q(a) = |C \cap S_q(a)|$. Indeed $C_q(a) = 0$ if and only if a is not q-covered by C, and otherwise a belongs to $A_q^k(C)$ exactly for all $k \leq C_q(a)$. So (without going into all details) the gain of either Adding $s \in S \setminus C$ or Dropping $c \in C$ will have to be updated only at each such a for which $C_q(a)$ is at the constraint limit.

It view of this it should be worthwhile to investigate in further detail the trade-off between increase of memory use against flexibility of Add and Drop operations, as well as which type of datastructures are most efficient.

6. CONCLUDING REMARKS

Other types of constraints may arise in practice, in particular capacities. Indeed, each thing covered calls for sufficient resources at the covering sites in order to enable the service expressed by the covering. It might be that limits are imposed on these resources, possibly rendering some solutions unfeasible. Such constraints complicate matters considerably, because the relatively simple notion of covering is then replaced by an allocation question which is nontrivial.

Also most applications will require an allocation rule to the closest chosen site, in order to maximize the service quality, but this information is not fully available with the simple covering sets we used here. For that one will have to revert to the full distance information, which is usually either too voluminous to store in an easily searchable format, and not immediately reconstructible when not stored.

However, when some sites have been fixed in advance, as often happens in practice because one seldom can redesign the whole system from scratch, such capacities might be defined as supplementary constraints of type discussed higher. This remains to be investigated, along many other details.

A different, but related type of objective is the optimization of quality, usually under restriction of a fixed number of chosen sites and a single covering constraint. With full (single) covering requirement this yields the well-known p-center problem [44, 5], with full k-tuple covering we obtain the more recently studied k-neighbour p-center problem, [45]. VNS has been successfully tuned to both such problems [4, 46]. With mandatory weight covering we obtain an apparently yet unstudied model, asking for the highest quality at which one may cover at least a given weight of things by p sites. Strictly speaking, all these models fall outside the scope of the framework presented here, which is rather aimed at multiple constraint types, possibly mixing several fixed qualities. However, one possible general solution approach consists of a binary search among fixed qualities replacing the covering constraint by a maximal covering-type objective, yielding subproblems of the type discussed here.

Funding. This research received no external funding.

REFERENCES

[1] P. Chevalier, I. Thomas, D. Geraets, E. Goetghebeur, O. Janssens, D. Peeters, and F. Plastria, "Locating fire stations: An integrated approach for Belgium," *Socio-Economic Planning Sciences*, vol.46, no. 2, jun 2012, pp. 173–182.

[2] N. Mladenović, F. Plastria, and D. Urošević, "Reformulation descent applied to circle packing problems," *Computers & Operations Research*, vol. 32, no. 9, 2005, pp. 2419–2434.

[3] P. Hansen and N. Mladenović, Variable neighborhood search for the p-median. *Location Science*, vol. 5, no. 4, pp. 207–226, dec 1997.

[4] N. Mladenović, M. Labbé, and P. Hansen, Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, vol. 42, no. 1, pp. 48–64, 2003.

[5] R. L. Church, and A. Murray, *Location Covering Models*. Springer International Publishing, 2018.

[6] S. García and A. Marín, Covering location problems. In Gilbert Laporte, Stefan Nickel, and Francisco Saldanha da Gama, editors, *Location Science*, chapter 5, pages 93–114. Springer International Publishing, 2015.

[7] L. A. McLay, A maximum expected covering location model with two types of servers. *IIE Transactions*, 41(8):730–741, jun 2009.

[8] A. Başar, B Çatay, and T. Ünlüyurt, A multi-period double coverage approach for locating the emergency medical service stations in Istanbul. *Journal of the Operational Research Society*, vol. 62, no. 4, pp. 627–637, 2011.

[9] E. Aktaş, Ö. Özaydın, B. Bozkaya, F. Ülengin, and Ş. Önsel, Optimizing fire station locations for the Istanbul metropolitan municipality. *Interfaces*, vol. 43, no. 3, pp. 240–255, 2013.

[10] Z. Drezner, V. Marianov, and G. O. Wesolowsky, Maximizing the minimum cover probability by emergency facilities. *Annals of Operations Research*, vol. 246, no. 1-2, pp. 349–362, 2014.

[11] G. Mullaoğlu and G. Sariyer, Variations of double coverage ambulance location model using call volume and population data. *Journal of Industrial and Production Engineering*, vol. 36, no. 8, pp. 533–545, 2019.

[12] V. Blanco, R. Gázquez, and F. S. da Gama, Multi-type maximal covering location problems: Hybridizing discrete and continuous problems. *European Journal of Operational Research*, oct 2022.

[13] J. Xu, A. T. Murray, R. L. Church, and R. Wei, Service allocation equity in location coverage analytics. *European Journal of Operational Research*, vol. 305, no. 1, pp. 21–37, 2023.

[14] D. A. Schilling, C. Revelle, J. Cohon, and D. J. Elzinga, Some models for fire protection locational decisions. *European Journal of Operational Research*, vol. 5, no.1, pp. 1–7, 1980.

[15] S. Ceria, P. Nobili, and A. Sassano, Set covering. In M.Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliography in Combinatorial Optimization*, chapter 23. J. Wiley andSons, 1997.

[16] L. V. Snyder, Covering problems. In H. A. Eiselt and Vladimir Marianov, editors, *Foundations of Location Analysis*, pages 109–135. Springer US, New York, NY, 2011.

[17] X. Li, Z. Zhao, X. Zhu, and T. Wyatt, Covering models and optimization techniques for emergency response facility location and planning: a review. *Mathematical Methods of Operations Research*, vol. 74, no. 3, pp. 281–310, 2011.

[18] A. T. Murray, Fire station siting. In H. A. Eiselt and Vladimir Marianov, editors, *Applications of Location Analysis*, pages 293–306. Springer International Publishing, Cham, 2015.

[19] Y. Liu, Y. Yuan, J. Shen, and W. Gao, Emergency response facility location in transportation networks: A literature review. *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 8, no. 2, pp. 153–169, 2021.

[20] M. L. Balinski, Integer programming: Methods, uses, computations. *Management Science*, vol. 12, no. 3, pp. 253–313, 1965.

[21] C. Toregas and C. ReVelle Optimal location under time or distance constraints. *Papers of the Regional Science Association*, vol. 28, no. 1, pp. 131–143, 1972.

[22] M. S. Daskin and E. H. Stern, A hierarchical objective set covering model for emergency medical service vehicle deployment. *Transportation Science*, vol. 15, no. 2, pp. 137–152, 1981.

[23] Church and ReVelle, The maximal covering location problem. *Papers of the Regional Science Association*, vol. 32, pp. 101–118, 1974.

[24] V. Chvatal, A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[25] M. Ohlsson, C. Peterson, and B. Söderberg, An efficient mean field approach to the set covering problem. *European Journal of Operational Research*, vol. 133, no. 3, pp. 583–595, 2001.

[26] N. Liu, B. Huang, and M. Chandramouli, Optimal siting of fire stations using GIS and ANT algorithm. *Journal of Computing in Civil Engineering*, vol. 20, no. 5, pp. 361–369, 2006.

[27] L. Yang, B. F. Jones, and S. H. Yang, A fuzzy multi-objective programming for optimization of fire station locations through genetic algorithms. *European Journal of Operational Research*, vol. 181, no. 2, pp. 903–915, 2007.

[28] D. Tong, A. Murray, and N. Xiao, Heuristics in spatial analysis: A genetic algorithm for coverage maximization. *Annals of the Association of American Geographers*, vol. 99, no. 4, pp. 698–711, 2009.

[29] A. Başar, B. Çatay, and T. Ünlüyurt, A taxonomy for emergency service station location problem. *Optimization Letters*, vol. 6, no.6, pp. 1147–1160, 2011.

[30] A. Elshaikh, S. Salhi, J. Brimberg, N. Mladenović, B. Callaghan, and G. Nagy, An adaptive perturbation-based heuristic: An application to the continuous p-centre problem. *Computers & Operations Research*, vol. 75, pp 1–11, 2016.

[31] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., USA, 1990.

[32] H. P. Williams, *Model building in mathematical programming.* Wiley, 5th edition, 2013.

[33] G. Sierksma, and Y. Zwols, *Linear and Integer Optimization : Theory and Practice, Third Edition.* Chapman and Hall/CRC, 2015.

[34] F. Plastria, Formulating logical implications in combinatorial optimisation. *European Journal of Operational Research*, vol. 140, no. 2, pp. 338–353, 2002.

[35] N. Mladenović and P. Hansen, Variable neighborhood search. *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[36] G. Caporossi, P. Hansen, and N. Mladenović, Variable neighborhood search. In *Metaheuristics*, pages 77–98. Springer International Publishing, 2016.

[37] P. Hansen, N. Mladenović, R. Todosijević, and S. Hanafi, Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, vol. 5, no. 3, pp. 423–454, 2017.

[38] I. Grujičić and Z. Stanimirović, Variable neighborhood search method for optimizing the emergency service network of police special forces units. *Electronic Notes in Discrete Mathematics*, vol. 39, pp. 185–192, 2012.

[39] N. Nikolić, I. Grujičić, and N. Mladenović, A large neighbourhood search heuristic for covering designs. *IMA Journal of Management Mathematics*, vol. 27, no. 1, pp. 89–106, 2014.

[40] L. Mrkela and Z. Stanimirović, A variable neighborhood search for the budget-constrained maximal covering location problem with customer preference ordering. *Operational Research*, vol. 22, no.5, pp. 5913–5951, 2021.

[41] A. Casado, N. Mladenović, J. Sánchez-Oro, and A. Duarte, Variable neighborhood search approach with intensified shake for monitor placement. *Networks*, dec 2022.

[42] N. Mladenović, F. Plastria, and D. Urošević, Formulation space search for circle packing problems. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, pages 212–216. Springer Berlin Heidelberg, 2007.

[43] N. Mladenović, J. Brimberg, and D. Urošević, Formulation space search metaheuristic. In *The Palgrave Handbook of Operations Research*, pages 405–445. Springer International Publishing, 2022.

[44] S. L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph, *Operations Research*, vol. 12, no. 3, pp. 450–459, 1964.

[45] D. Chen and R. Chen, Optimal algorithms for the α-neighbor p-center problem. *European Journal of Operational Research*, vol. 225, no. 1, pp. 36–43, 2013.

[46] S. R. Mousavi, Exploiting flat subspaces in local search for p-center problem and two fault-tolerant variants. *Computers & Operations Research*, vol. 149, 106023, 2023.