INTERNATIONAL JOURNAL OF ECONOMICS AND STATISTICS
DOI: 10.46300/9103.2022.10.15 Volume 10, 2022

Competitive location: new models and methods and future trends

José Fernández*, Juana L. Redondo†, Pilar M. Ortigosa‡, Aránzazu G. Arrondo*
*Dpt. of Statistics and Operations Research, University of Murcia, Spain
†Dpt. of Computer Architecture and Technology, University of Granada, Spain
‡Dpt. of Informatics, University of Almería, Spain

***Abstract*—Two new models for sitting a new facility in a competitive environment are introduced. Both the location and the quality of the new facility are to be found, so as to maximize the profit obtained by the locating firm. The patronizing behavior of customers is assumed to be probabilistic, i.e., they split their demand among all the existing facilities in the area, proportionally to the attraction they feel for them. The attraction is determined both by the distance between the demand point and the facility and by the quality of the facility. Contrarily to what is commonly done in literature, the demand is not fixed, but varies depending on the location of the facilities. The first model assumes a static scenario, whereas in the second one a competing chain reacts by location a single new facility too, leading to a Stackelberg (or leader-follower) problem. The new continuous location models lead to hard-to-solve global optimization problems. A new evolutionary algorithm called UEGO was used to deal with those problems. The computational results showed its usefulness and robustness. Parallel implementations of UEGO are also presented to cope with large instances. The efficiency and scalability of the parallel algorithms were shown through a computational study. Future trends which will allow the construction of an expert system for facility location are also discussed.**

***Keywords*—Global Optimization, Evolutionary Computing, High Performance Computing, Decision Support Systems.**

I. Introduction

Location science deals with the location of one or more facilities in a way that optimizes a certain objective (minimization of transportation costs, minimization of social costs, maximization of the market share, etc.). For an introduction to the topic see [1], [2], [3]. Depending on the space where a location problem is explicitly defined it is possible to distinguish three types of problems: *continuous* location problems, where the facilities to be sited can be placed anywhere on a region of the plane, *network* location problems, where any point on a network is suitable for location, and *discrete* location problems, where the facilities can be located only at a limited number of eligible points. Furthermore, the feasible set may be restricted by the introduction of "forbidden zones", i.e. areas in which facilities should not be located [4] or other types of constraints. From the optimization point of view, the techniques used to cope with the problems also differ. Continuous location problems are, in most of cases, nonlinear optimization problems, while discrete and network location problems usually lead to integer programming/combinatorial optimization problems. It is customary to differentiate between single-facility location problems and multi-facilities problems. In the former, only one facility is to be located, while in the latter, several facilities are to be sited.

Two types of location models can be distinguished depending on whether a single player or multiple players in the market are considered. They are referred to as *non-competitive* and *competitive* models, respectively. A detailed taxonomy can be found in survey papers [5], [6], [7]. In many location models it is assumed that the decision maker, who plans the location of his facilities, faces an empty space without any similar or competing facilities. Nevertheless, in most cases, similar facilities already exist in the region and the task is to add new ones in an optimal way [5], [8], [9]. The existing facilities may belong to the decision maker's own chain or to a competitor's chain [4]. When a competition takes place, it may be *static*, which means that the competitors are already in the market, the owner of the new facility knows their characteristics and no reaction is expected from them, or *with foresight*, in which the competitors are assumed to react after the new facility enters. Furthermore, if the competitors can change their decisions, the model is considered *dynamic*, as it is characterized by the major concern of the existence of *equilibria*. In this context, Hakimi [10] introduced the well known Stackelberg problems. The scenario considered in these kind of problems is that of a *duopoly*. A chain (the leader) wants to set up p new facilities in the market, where similar facilities of a competitor (the follower), and possibly of its own chain, are already present. The follower will react by locating r facilities after the leader locates its own facilities. Hakimi introduced the terms *medianoid* for the follower problem, and *centroid* for the leader problem. More precisely, an $(r|X_p)$ medianoid problem refers to the follower's problem of locating r new facilities in the presence of p leader's facilities located at a set of points X_p. And an $(r|p)$ centroid problem refers to the leader's problem of locating p new facilities, knowing that the follower will react positioning r new facilities by solving an $(r|X_p)$ medianoid problem.

Regarding customers, clients can be either distributed according to some distribution function over a given set, or located at specific points (named *demand points*, see [11]) in the plane or at vertices in a network, which is the common approach in literature.

Another important feature is the so called *demand*. Demand

can be either fixed (*exogenous*) or may vary (*endogenous*). In the first case, the demand is known with certainty. This is usually the case when the goods are *essential* to the customers, and then, they will buy the goods independently of the distance to the facility or the price. In the second case, goods are *inessential* to the customers, then, demand can vary depending on prices, distances to the facilities, etc.

Another important question to take into account is whether customers are free to choose the facility from which they are served. If so, knowing how customers buy goods among the existing facilities helps to estimate the market share captured by each facility [12]. The patronizing behavior of the customers is usually assumed to be either *deterministic*, when the full demand of the customer is served by the facility to which he/she is *attracted* most (leading to Hotelling-type models) or *probabilistic*, when the customer splits his/her demand among all the existing facilities (leading to Huff-type models).

Furthermore, location problems can be defined as *pure location* problems, where the aim is to determine only the optimal sites for the new facilities, or as *mixed* problems, in which, apart from the location, a decision has to be made about other variables. In mixed problems, besides the location of the services, some "active" interactions have to be determined (see [13]). These interactions can be expressed by an *attraction (or utility) function* of a customer towards a given facility. For instance, in competitive location problems, it usually depends on the distance between the customer and the facility, and on other characteristics of the facility which determine its *quality*. The *market share* captured by the facilities depends on all those factors.

The continuous competitive single-facility location and design problem with endogenous demand with a probabilistic patronizing behavior of customers introduced in [14], as well as its corresponding Stackelberg model [15], are revised in Section II. They both are hard-to-solve global optimization problems. Although branch-and-bound methods can solve small instances of those problems (see [12], [16]), when dealing with real-life instances, with much more demand points, heuristic methods are required. Evolutionary algorithms have proved to be good alternatives at solving moderate-sized problems. In particular, the Universal Evolutionary Global Optimizer algorithm (UEGO) has been successfully applied to both problems [14], [15]. Its main steps will be explained in Section III. Still, when solving large instances, UEGO takes a great amount of CPU time and has large memory requirements. High performance computing approaches are then the best way forward, as it will be discussed in Section IV. The integration of those models and methods within a Graphical Information System (GIS) seems to be the next step, so that decision-makers can have a user-friendly expert system to assist them at taking decisions.

II. Competitive location models

One of the major questions that a retail chain has to face when it considers entering or extending its presence in a market is 'where to locate' the new facility (or facilities) to be opened. If other facilities offering the same goods already exist in the area, the new facility will have to *compete* for the market. Many competitive location models are available in the literature, see for instance the survey papers [5], [9], [17], [18]. In most competitive location literature, it is assumed that the demand is fixed regardless the conditions of the market. Although this may be appropriate for essential goods, in other cases this is mainly due to the difficulty of the problems to be solved: even with fixed demand, the corresponding location models may be hard-to-solve global optimization problems. However, sometimes demand is elastic, that is, it varies depending on several factors. For instance, as already stated in [19], consumer expenditures on products or services offered by the facilities may increase for a variety of reasons related to location of the new facility: opening new outlets may increase the overall utility of the product; the marketing expenditures resulting from the new facilities may increase the overall 'marketing presence' of the product, leading to increased consumer demand; or some consumers who did not patronize any of the facilities, perhaps because none were close enough to their location, may now be induced to do so. On the other hand, the quality of the facilities may also affect consumer expenditures, since a better service usually leads to more sales. To our knowledge [14] seems to be first paper where the influence of the fixed demand assumption in the optimal location and quality of new facilities to be located is investigated. The following notation, borrowed from [14], will be used throughout this paper:

Indices

i	index of demand points, $i = 1, \ldots, n$.
j	index of existing facilities, $j = 1, \ldots, m$.

Variables

x	location of the new facility, $x = (x_1, x_2)$.
α	quality of the new facility ($\alpha > 0$).

Data

p_i	location of demand point i ($i = 1, \ldots, n$).
f_j	location of existing facility j ($j = 1, \ldots, m$).
d_{ij}	distance between demand point p_i and facility f_j.
a_{ij}	quality of facility f_j as perceived by demand point p_i.
$g_i(\cdot)$	a non-negative non-decreasing function.
u_{ij}	attraction that p_i feels for f_j (or utility of f_j perceived by the people at p_i), $u_{ij} = \dfrac{a_{ij}}{g_i(d_{ij})}$.
γ_i	weight for the quality of the new facility as perceived by demand point p_i.
$d_i^{\min}$	minimum distance from p_i at which the new facility can be located.
$\alpha_{\min}$	minimum level of quality.
$\alpha_{\max}$	maximum level of quality.
S	region of the plane where the new facility can be located.

Miscellaneous

$d_i(x)$	distance between demand point p_i and the new facility.
u_{i0}	attraction that p_i feels for the new facility, $u_{i0} = \dfrac{\gamma_i \alpha}{g_i(d_i(x))}$.

It is assumed that $g_i(d_{ij}) > 0 \ \forall i, j$. Following the framework of *spatial interaction models* introduced by Huff [20],

it is considered that the patronizing behavior of customers is probabilistic. Based on these assumptions, if we denote $\widehat{w}_i$ the fixed demand (or buying power or total expenditure) at p_i, the market share captured by the chain is

$$M(x,\alpha) = \sum_{i=1}^{n} \widehat{w}_i \frac{u_{i0} + \sum_{j=1}^{k} u_{ij}}{u_{i0} + \sum_{j=1}^{m} u_{ij}}.$$

and the problem of profit maximization is described by

$$\begin{cases} \max & \Pi(x,\alpha) = F(M(x,\alpha)) - G(x,\alpha) \\ \text{s.t.} & d_i(x) \geq d_i^{\min} \ \forall i \\ & \alpha \in [\alpha_{\min}, \alpha_{\max}] \\ & x \in S \subset \mathbb{R}^2 \end{cases} \quad (1)$$

where $F(\cdot)$ is a strictly increasing differentiable function which transforms the market share into expected sales, $G(x,\alpha)$ is a differentiable function which gives the operating cost of a facility located at x with quality α, and $\Pi(x,\alpha)$ is the profit obtained by the chain. The parameter $d_i^{\min} > 0$ is a given threshold, which guarantees that the new facility is not located on top of demand point p_i. By S we refer to the region of the plane where the new facility can be located. In [14] it is assumed that $F(M(x,\alpha)) = c \cdot M(x,\alpha)$, where c is the income per unit of goods sold, and that function G should increase as x approaches one of the demand points (since it is rather likely that the operational cost of the facility will be higher around those locations) and be a convex function in the variable α, since the more quality we expect from the facility the higher the costs will be, at an increasing rate. The following choices were made: $G(x,\alpha) = G_1(x) + G_2(\alpha)$, where $G_1(x) = \sum_{i=1}^{n} \Phi_i(d_i(x))$, with $\Phi_i(d_i(x)) = \widehat{w}_i/((d_i(x))^{\phi_{i0}} + \phi_{i1})$, $\phi_{i0}, \phi_{i1} > 0$ and $G_2(\alpha) = e^{\frac{\alpha}{\beta_0} + \beta_1} - e^{\beta_1}$, with $\beta_0 > 0$ and β_1 given values. Other possible expressions for F and G can be found in [21], [12].

Let us make the more realistic assumption that the demand at p_i is affected by the perceived utility of the facilities, given by the vector $u_i = (u_{i0}, u_{i1}, \ldots, u_{im})$. Making the simplifying assumption that *the utility is additive*, then $U_i = u_{i0} + \sum_{j=1}^{m} u_{ij}$ represents the total utility perceived by a customer at p_i provided by all the facilities. Hence, it is natural to assume that the actual demand at p_i is a function of U_i. If we denote the maximum possible demand at p_i by $w_i^{\max}$, and the minimum possible demand at p_i by $w_i^{\min}$, then the actual demand w_i at p_i is a function of the utility vector u_i only through the total utility U_i, i.e., $w_i(U_i) = w_i^{\min} + incr_i \cdot e_i(U_i)$, where $incr_i = w_i^{\max} - w_i^{\min}$. Here, $e_i(U_i)$ is a non-negative and non-decreasing function of U_i that must not exceed 1 (notice that w_i cannot exceed $w_i^{\max}$). Function $e_i(U_i)$ can be interpreted as the share of the maximum possible increment that a customer decides to expend under a given location scenario. There are different possible expressions for this:

- Linear expenditures: $w_i^{\min} = 0$, $e_i(U_i) = c_i U_i$, with c_i a given constant such that $c_i \leq 1/U_i^{\max}$, where $U_i^{\max}$ is the maximum utility that can possibly be perceived by a customer at i, see [19].
- Exponential expenditures: $w_i^{\min} = 0$, $e_i(U_i) = 1 - \exp(-\rho_i U_i)$, where $\rho_i > 0$ is a positive constant, see [19]. A similar model, with $e_i(U_i) = 1 - \rho_{i1}^{-\rho_{i2} U_i}$, was presented in [22].
- Affine expenditures: similar to linear expenditures, but with $w_i^{\min} \neq 0$, [14].
- Logit expenditures: $e_i(U_i) = \frac{1}{1+\exp(\rho_{i1} + \rho_{i2} \frac{1}{U_i})}$, where $\rho_{i1} \in \mathbb{R}$ and $\rho_{i2} > 0$ are given constants [23].

The corresponding model with variable demand to be solved is (1), where specifically:

1) In function M, $\widehat{w}_i$ is changed to $w_i(U_i)$.
2) In the cost function $G(x,\alpha)$, for $\widehat{w}_i$ a (fixed) average value is used. Notice that $\widehat{w}_i$ is not replaced by $w_i(U_i)$ in function G. In particular, this means that it is assumed, on the one hand, that the cost of obtaining a given level of quality, as given by G_2, does not depend on the level of demand in the market. This can be realistic in many cases, especially when $incr_i$ is not too high. On the other hand, it also implies that the location cost does not depend on the level of demand either. This is especially true if the cost of buying or renting the place for the location is paid in advance, before opening the new facility. In this way the scenario which determines the cost of the location is not affected by the 'variation' in the demand produced by the location of the new facility, but just by the expected average demand.

As shown in [14] the loss in profit when assuming fixed demand (difference in objective function value of the problem with variable demand between the optimal solution point obtained for that problem when assuming fixed demand and the optimal solution) may be on average more than 50% for problems with 1000 demand points. This clearly shows how important to take the variability of the demand into account may be.

The corresponding $(1|1)$ centroid problem (see [15]) is much harder to solve, with many local maxima and in some instances with very different objective values at quite close feasible points. Notice that to evaluate its objective function at a given point, the corresponding medianoid (follower) problem (as described above) has to be solved in order to obtain the corresponding location and quality of the follower. Furthermore, in order to compute the objective value of the leader accurately, the corresponding follower problem has to be precisely solved since otherwise, the error of the approximate value can be considerable. And it is important to mention that to solve a single centroid problem, many medianoid problems have to be solved, as many points have to be evaluated. As a result, in [15] only problems with up to 500 demand points could be solved with the adapted sequential algorithm UEGO.

III. Evolutionary algorithms

From a modeler point of view, it is interesting to have general optimization methods able to solve many types of problems, so that one can concentrate on the modeling of the problem, and not to worry about how to solve it later on. In [12] new general branch-and-bound methods which make use of interval analysis [24] are developed, and are successfully applied to location problems. Those methods were used in [14] to solve the follower problem described in the previous section. Unfortunately, interval B&B methods, besides being

very time-consuming, can only solve small size problems (less than 200 demand points), as the computer runs out of memory.

In literature bigger instances are usually solved with ad-hoc heuristics, i.e., specifically designed for the particular type of problem at hand. Although they usually provide good results quickly, they require a very good knowledge of the mathematical properties of the problem and a good expertise to implement them.

During the last twenty years meta-heuristics such as tabu search [25], simulated annealing [26], genetic algorithms [27], or variable neighborhood search [28] have also been applied to different (usually discrete and network) location problems, (see [29], [30], [31], [32]). Applications to continuous location problems is scarcer, but we can also find several examples (see for instance [33], [34]). A meta-heuristic can be defined as a high-level algorithmic framework that can be specialized to solve different optimization problems. They are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic, or when it is not practical to implement such a method. They are not problem-specific, but they may make use of domain-specific knowledge in the form of procedures or secondary heuristics that are controlled by an upper level strategy. That is why they are a good alternative for a modeler.

Metaheuristics can be classified into *trajectory methods* and *population based methods*. In the first group, the search process is characterized by a trajectory in the search space (tabu search of simulated annealing belong to this category). These methods usually allow moves to worse solutions to be able to escape from local optima. Population-based metaheuristics (PBM) [35] are algorithms that work on a *set of solutions* (i.e. a population) at the same time rather than on a single solution. At first glance, it might be seen that this idea does not really provide anything new, since the previous algorithms could run several times to increase the probability of arriving at the global optimum. But there is an additional component that can make PBMs essentially different from other solving methods: the concept of *competition* among solutions in a population. That is, they simulate the evolutionary process of competition and selection so that the candidate solutions in the population fight for room in future generations. In this way, PBMs provide a natural, intrinsic way for the exploration of the search space.

UEGO is a PBM which falls in the category of *evolutionary algorithms*. As it has been applied to several continous location problems [33], [34], including the two ones described in the previous section (after adapting it to the problem at hand, see [14], [15]), next we briefly describe its main characteristics.

Its main structure (see Algorithm 1) is very much like the sequence of procedures of other evolutionary algorithms, where there is a population initialization phase and an iterative loop aimed at evolving the population towards the optima.

A key notion in UGEO is that of a *species*. A species is equivalent to an individual in a usual evolutionary algorithm. A species is not a point, but a hypersphere defined by its center and a radius. The center is a solution, and the radius indicates its attraction area, which covers a region of the search space and hence, multiple solutions. The radius of the species is neither constant along the execution of UEGO nor the same for each species. The radius of a species created at iteration

Algorithm 1 UEGO

```
1: Init_population
2: for t = 1 to L
3:     Create_species(new_t)
4:     Selection(max_spec_num)
5:     Optimize_species(n_t)
6:     Selection(max_spec_num)
```

t, R_t, decreases as the index level (or cycles or generations) increases (for a detailed description of how to compute the radius at each level of the algorithm see [36]). Species with small radii examine a relatively small area, their motion in the space is slower, but they are able to differentiate between good solutions which are very close. On the contrary, species with large radii study a somewhat larger region, they may move greater distances and discover new promising areas, which may be analyzed conscientiously in later stages of the algorithm.

In UEGO every species is intended to occupy a local maximizer of the fitness function, without knowing the total number of local maximizers in the fitness landscape. This means that when the algorithm starts it does not know how many species there will be at the end. Consequently, the population consists of a list of species whose size is continuously varying due to the operators into the loop, namely, the *Create_species, Selection* and *Optimize_species* procedures. However, it is important to mention that there exist a maximum population size (max_spec_num), which is imposed by the user.

In the *Create_species* method, for every species on the list, a set of possible new candidates is computed, fused and evaluated in order to find new promising species, thereby increasing the species list. The parameter new_t refers to the function evaluation budget for the current population list. Notice that each species generates new ones by itself, independently of the remaining ones. In contrast, the *Selection* procedure requires the knowledge of the whole species list to be able to measure the distances among them. This method fuses the species that are close enough in the search space domain, and if there are still more than the maximum allowed, it deletes the surplus.

The *Optimize_species* mechanism, on the other hand, performs a local optimization on each species, and the obtained local optima replace the caller species. Notice that each species is optimized without considering the remaining ones. Here, the function evaluation budget per species is given by n_t/max_spec_num. It is mainly in this step where UEGO differs from one type of problem to another. For instance, for the follower problem, a Weiszfeld-like algorithm is used in [14]. For the leader problem things become more difficult. Local optimizers usually assume that the configuration of the problem during the optimization process does not change. However, this is not the case for the centroid problem, since every time the leader's facility changes, so does the follower's facility. Thus, the value of the objective function of the leader's problem may change if the new configuration is taken into account. This means that the new follower's facility should be computed every time the leader's facility changes. A variant of the SASS algorithm [37] is employed in [15], which uses the Weiszfeld-like algorithm of [14] to recompute the new follower's facility.

Another important issue to take into account is that the evaluation of a single species in UEGO when applied to the leader problem requires intensive computational effort, since it involves the execution of another optimization algorithm to obtain the optimal location of the follower (by solving the corresponding medianoid problem), namely, another UEGO algorithm. For this reason UEGO, when applied to the leader problem, was designed to maintain a small-size population, by including a *'fuse'* process just after the creation of candidate solutions, therefore, only the resulting ones are evaluated.

IV. High performance computing

Parallel computing operates on the principle that large problems can almost always be divided into smaller ones which can be solved concurrently ("in parallel"). A parallel program is intrinsically more complex than its serial counterpart. *Sequential programming* keeps a single process (i.e. a unique flow of control), while *parallel programming* uses *concurrent implementations*, where two or more processes work together to perform a single task and most of the times they need to communicate and synchronize among them. Communication and synchronization among different sub-tasks is typically one of the greatest barriers in obtaining good parallel program performance.

Concurrent programming includes both the programming of multiprocessors (*Shared memory programming*) and multicomputers or distributed systems (*Distributed memory programming*). In shared memory programming, the whole memory is directly accessible to all the processes with an intent to provide communication among themselves. Depending on context, programs may run on the same physical processor or on separate ones. There exist several ways to deal with parallelism in a shared memory model. *OpenMP* (Open Multi-Processing) (`www.openmp.org`) is an Application Programming Interface (API) that supports multi-platform shared memory multiprocessing programming in C/C++ and FORTRAN on many architectures (including Unix and Microsoft Windows platforms). It consists of a set of compiler directives, library routines, and environment variables that are used to express shared-memory parallelism. Jointly defined by a group of major computer hardware and software vendors, OpenMP is a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications. The programmers use the OpenMP directives to tell the compiler which parts of the program must be executed concurrently and to specify synchronization points.

Distributed programming is based on the message-passing mechanisms. One of the standards to implement it is MPI (Message-Passing Interface), a language-independent communications protocol [38]. Processes are written in a sequential language (C, C++, FORTRAN), and communications and synchronizations are made by calling functions from the MPI library.

One of the main goals in parallelism consists of increasing the performance of an application with respect to its execution on a uni-processor. The commonly used metric to measure the performance of a parallel implementation on homogeneous processors is the *speedup*, which is defined as the ratio between the execution time on a uni-processor $T(1)$ and the execution time on P processors $T(P)$:

$$Spd(P) = \frac{T(1)}{T(P)}$$

Linear speedup (or ideal speedup) is obtained when $Spd(P) = P$. However, obtaining an ideal speedup is not always possible, since there are many factors which can increase the value of $T(P)$ and hence reduce the corresponding speedup. Among others, these factors can be: work load unbalance, sequential parts of the algorithm, communication overheads and synchronization among processors. Another metric is *efficiency*, computed as:

$$Eff(P) = \frac{Spd(P)}{P} = \frac{T(1)}{P \cdot T(P)}$$

Note that when linear speedup occurs efficiency is 1, which is called linear efficiency (or ideal efficiency).

An important concept in parallelism is that of *scalability*. It can be understood as the ability of a system, network, or process, to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.

Literature contains many examples of successful parallel models for being applied to population-based algorithms. For instance, *master-slave* is a communication model where one processor (the master) has unidirectional control over one or more processors (the slaves). This technique is called "global parallel model" too, since the management of the population is global (i.e. all the individuals in the population are considered when selection or crossover procedure is carried out). Usually, the master takes charge of performing it. In this model, the parallelism comes from the evaluation of the individuals in the population. This is because the fitness of an individual is independent of the rest of the population, and there is no need to communicate during this phase. The evaluation of individuals is parallelized by assigning a fraction of the population to each available processor. Communications occur only as each slave receives its subset of individuals for evaluation and when the slaves return the fitness values. The execution time of a master-slave model has three basic components: the time used in computations, the time used to communicate information among processors, and the waiting time due to the synchronization points. The master-slave method does not require a particular computer architecture, and it can be implemented efficiently on shared and distributed memory computers. On a shared-memory multiprocessor, the population can be stored in shared memory and each processor could read a fraction of the population and write back the evaluation results. On a distributed-memory computer, the population is stored in one processor (the master), which is responsible for sending the individuals to the other processors (the slaves) for evaluation, collecting the results, and applying the genetic operators to produce the next generation. The difference with a shared-memory implementation is that the master has to send and receive messages explicitly.

On the contrary, in a coarse-grain model, each processor executes an algorithm independently of the remaining ones during most of the time. The idea is that different processors work with smaller and different subpopulations in such a way that, when merging all the subpopulations, a population similar

to that of the sequential version can be obtained. Nevertheless, some information can migrate from a processor to another one (following a given topology) according to a migratory policy, which is controlled by the following parameters:

- *Interval of migration*: It establishes how often the migration of a certain amount of individuals will be conducted from a processor to another.
- *Rate of migration*: It indicates the number of individuals that have to communicate with other processors when the migration interval is fulfilled.
- *Selection criterion*: It determines the policy that will be applied for the selection of migratory individuals.

Surprisingly, despite the fact that the processing time and computational requirements needed to solve some location problems (as the ones considered in this paper) may be considerable, the use of high performance computing techniques in location science is rather scarce, with hardly a dozen papers dealing with the topic, usually, based on distributed programming paradigms executed on multicomputers (see for instance [39], [40], [41], [42], [43], [44]).

However, current standalone multicore personal computers present tremendous power, thanks to technological and architectural advances [45]. They have been successfully used in other fields to accelerate sequential codes [46], [47], but they still have not been fully exploited in the location field. In [48] the four cores of a Intel Core 2 Quad CPU are put to work in parallel to solve an uncapacitated warehouse location problem, although no special strategy is used there: the same heuristic algorithm is run in the four cores, as a kind of multi-start strategy. As far as we know, [49] is the first paper in which a parallel model, namely, a master-slave parallelization of the UEGO algorithm described in [14] and implemented using OpenMP, is used to solve a location model (the follower problem describe in Section II) using the eight Intel Xeon cores of a standard PC. The parallelism comes from the concurrent execution of two procedures: the creation and optimization methods. There exists a *synchronization point* imposed by the *selection* procedure. In such a procedure, only one thread works with the whole population to maintain coherence in the data. Even so, to reduce the waiting time, partial selections are carried out concurrently, although in the end a global one performed in the 'selection' procedure is required for a correct performance of the algorithm in terms of quality in the solutions. The computational results in [49] showed that the parallel algorithm has a nearly ideal efficiency for up to 8 processors for problems with 10000 demand points. Remarkably, the sequential algorithm UEGO was not been able to solve problems with 5000 or more demand points.

The UEGO algorithm adapted to the leader problem described in Section II has also recently been parallelized [50]. In fact, three parallelizations were studied:

- The first one is a distributed memory programming model based on message-passing mechanisms, suitable for being used in multicomputers. It is a master-slave technique implemented using MPI. The master processor executes UEGO sequentially. The parallelism comes from the simultaneous evaluation of the new candidate solutions in the creation function, and from the concurrent execution of the local search procedure (see [15]). Basically, when the creation takes place, the master obtains a new offspring of candidate solutions for the leader's facility. The evaluation of the objective function at those candidate solutions is carried out in a parallel way: the master processor divides the list of candidate solutions by the number of processors and delivers the resulting sublists among them all (including itself). Each processor receives a species sublist from the master and evaluates the objective function at each of its elements. The optimization procedure behaves similarly. The efficiency is nearly ideal for up to 8 processors, and more than 0.7 for 64 processors.
- The second one is a share memory programming model, suitable for being used in multiprocessors (as most of todays PCs are). The parallel model can also be considered a master-slave technique, but contrarily to the previous parallel strategy, no messages are required to communicate processors, though a mechanism to coherently share memory data is necessary. For the implementation of the problem at hand, OpenMP was selected. The efficiency is nearly ideal for up to 8 processors.
- The third one is a hybrid of the two previous models, suitable for being used in clusters, where several nodes are interconnected according to a given topology (as multicomputers), and each node contains several processors sharing some memory (such as multiprocessors). The parallel programming combines message-passing mechanisms among nodes with shared memory parallelization inside each node. MPI and OpenMP have been considered tools to implement the parallel version. The parallel model links a coarse-grain model with a pseudo master-slave strategy. Each node executes UEGO. The population size and the total number of function evaluations for the whole optimization process are internally divided by the number of nodes. Concerning the migration procedure, two types of nodes (collectors and workers) are considered. Half of the nodes act as collectors and the other half as workers. At each communication, each node behaves either as a worker (sender) or as a collector (receiver), although, they interchange their roles at the next communication. The mechanism works as follows: The nodes are supposed to be connected in a ring topology and run independent of the remaining ones. In a communication stage, node i is a worker and sends its sublist to the next node $i+1$ (collector). Node $i+1$ fuses this list with its own sublist and distributes the resulting list between both nodes. In the next communication stage, node i will be a collector and will receive a sublist from node $i-1$, while node $i+1$ will be a worker and will send the sublist to the node $i+2$. The migration process is carried out at the first half of the levels of the algorithm. The communications among nodes are implemented using MPI.
  Notice that with the previous parallel strategy only, the computational resources inside each node are not fully exploited, since only a single processor would

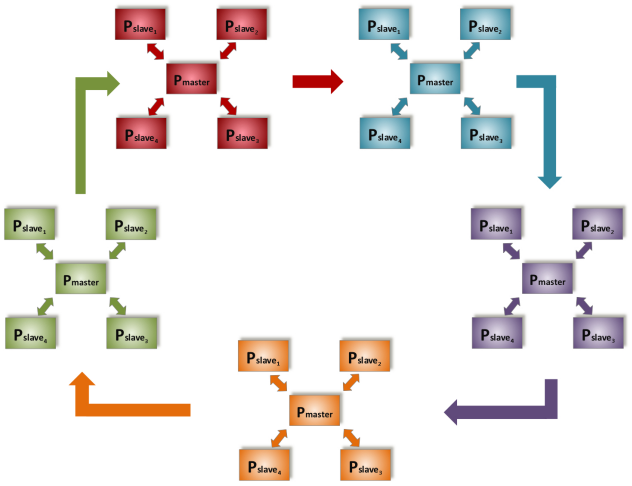

Fig. 1. Coarse-grain model + master-slave model

be used. To make use of the whole set of processors and improve the efficiency of the parallel version, inside each node the share memory programming implementation of UEGO is used (see Figure 1). The efficiency is 0.6 for 8 nodes with 8 processors each (i.e., 64 processors altogether).

Additionally, the scalability of the three implementations was demonstrated by solving problems with different computational loads, and checking that the efficiency increases with the size of the problems.

V. New trends

Location problems are ubiquitous: shopping centers, fast food restaurants, petrol stations, schools, fire stations, nuclear plants, military facilities, garbage dumps, ... Both public and private decision makers are demanding intelligent systems which assist them at selecting the locations for the new facilities.

The first step is modeling the problems as close to reality as possible. As shown in Section II, neglecting some of the factors that play role in the problem (as the variability of the demand, for instance) may lead to models whose solutions (although optimally obtained) may be far from the real optimum. Determining the key factors of a problem should be done with care. Some of the existing facility location models do not include some of them in an attempt to make them computationally solvable.

On the other hand, considering a mathematical model including all the factors that play a role may make the second step, solving the model, more difficult, as the mathematical formulation of the location problems usually leads to hard-to-solve global optimization problems (in continuous space) or NP-had problems (in networks or discrete spaces). In this regard, general methods, able to solve a wide range of problems (maybe by tuning some specific procedures) are required. Both exact methods (as the interval B&B methods described in [12]) and metaheuristics (such as UEGO) are required. And it is rather likely that in order to solve large instances high performance computing approaches (at least to make use of all the computing power of today's multicore PCs) will be a must.

But there is still a third step needed: to communicate with the decision maker in a friendly way. In this sense, the integration of models and solution procedures within Graphical Information Systems (GIS), such as GRASS (http://grass.osgeo.org/), is still to come. Whereas the applications of GIS cover many fields, its use in facility location is still scarce. We can find in literature some examples (see for instance [51], [52], [53], [54], [55] to name a few) but they are specific applications for some problems, and have not lead to modules for reuse. Those modules will close the construction of an expert system for facility location.

Acknowledgments

This work has been funded by grants from the Spanish Ministry of Economy and Competitiveness (ECO2011-24927, TIN2008-01117), Fundación Séneca (The Agency of Science and Technology of the Region of Murcia, 00003/CS/10 and 15254/PI/10), Junta de Andalucía (P08-TIC-3518 and P10-TIC-6002) and Program CEI from MICINN (PYR-2012-15 CEI BioTIC GENIL, CEB09-0010), in part financed by the European Regional Development Fund (ERDF). Juana López Redondo is a fellow of the Spanish 'Juan de la Cierva' contract program.

References

[1] Z. Drezner, Ed., *Facility location: a survey of applications and methods*. Springer, Berlin, 1995.

[2] Z. Drezner and H. Hamacher, *Facility location. Applications and theory*. Springer, Berlin, 2002.

[3] R. Francis, L. McGinnis, and J. White, *Facility layout and location: an analytical approach*. Prentice Hall, Englewood Cliffs, 1992.

[4] H. Eiselt and G. Laporte, "Objectives in location problems," in *Facility location: a survey of applications and methods*, Springer Series in Operations Research and Financial Engineering, Z. Drezner, Ed. Berlin: Springer, 1995, pp. 151–180.

[5] H. Eiselt, G. Laporte, and J. Thisse, "Competitive location models: a framework and bibliography," *Transportation Science*, vol. 27, no. 1, pp. 44–54, 1993.

[6] H. Hamacher and S. Nickel, "Classification of location models," *Location Science*, vol. 6, no. 1, pp. 229–242, 1998.

[7] C. ReVelle and H. Eiselt, "Location analysis: a synthesis and survey," *European Journal of Operational Research*, vol. 165, no. 1, pp. 1–19, 2005.

[8] M. Kilkenny and J. Thisse, "Economics of location: a selective survey," *Computers and Operations Research*, vol. 26, no. 14, pp. 1369–1394, 1999.

[9] F. Plastria, "Static competitive facility location: an overview of optimisation approaches," *European Journal of Operational Research*, vol. 129, no. 3, pp. 461–470, 2001.

[10] S. Hakimi, "On locating new facilities in a competitive environment," *European Journal of Operational Research*, vol. 12, no. 1, pp. 29–35, 1983.

[11] R. Francis, T. Lowe, and A. Tamir, "Demand point aggregation for location models," in *Facility location: application and theory*, Z. Drezner and H. Hamacher, Eds. Springer, 2002, pp. 207–232.

[12] B. Tóth and J. Fernández, *Interval methods for single and bi-objective optimization problems - applied to competitive facility location problems*. Saarbrücken: Lambert Academic Publishing, 2010.

[13] F. Plastria, "Continuous location problems," in *Facility location: a survey of applications and methods*, ser. Springer Series in Operations Research and Financial Engineering, Z. Drezner, Ed. Berlin: Springer, 1995, pp. 151–180.

[14] J. Redondo, J. Fernández, A. Arrondo, I. García, and P. Ortigosa, "Fixed or variable demand? Does it matter when locating a facility?" *Omega*, vol. 40, no. 1, pp. 9–20, 2012.

[15] ——, "A two-level evolutionary algorithm for solving the facility location and design $(1|1)$-centroid problem on the plane with variable demand," *Journal of Global Optimization*, To appear, (doi: 10.1007/s10898-012-9893-4).

[16] M. Sáiz, E. Hendrix, J. Fernández, and B. Pelegrín, "On a branch-and-bound approach for a Huff-like Stackelberg location problem," *OR Spectrum*, vol. 31, pp. 679–705, 2009.

[17] H. Eiselt and G. Laporte, "Sequential location problems," *European Journal of Operational Research*, vol. 96, no. 2, pp. 217–231, 1996.

[18] T. Drezner and H. Eiselt, "Consumers in competitive location models," in *Facility location: applications and theory*, Z. Drezner and H. Hamacher, Eds. Springer, 2001, pp. 151–178.

[19] O. Berman and D. Krass, "Locating multiple competitive facilities: Spatial interaction models with variable expenditures," *Annals of Operations Research*, vol. 111, no. 1, pp. 197–225, 2002.

[20] D. Huff, "Defining and estimating a trading area," *Journal of Marketing*, vol. 28, no. 3, pp. 34–38, 1964.

[21] J. Fernández, B. Pelegrín, F. Plastria, and B. Tóth, "Solving a Huff-like competitive location and design model for profit maximization in the plane," *European Journal of Operational Research*, vol. 179, no. 3, pp. 1274–1287, 2007.

[22] R. McGarvey and T. Cavalier, "Constrained location of competitive facilities in the plane," *Computers and Operations Research*, vol. 32, pp. 359–378, 2005.

[23] Z. Drezner, G. Wesolowsky, and T. Drezner, "On the logit approach to competitive facility location," *Journal of Regional Science*, vol. 38, no. 2, pp. 313–327, 1998.

[24] E. Hansen and G. W. Walster, *Global optimization using interval analysis*, second revised and expanded ed. Marcel Dekker, 2004.

[25] F. Glover and M. Laguna, *Tabu search*. Kluwer Academic Publisher, 1997.

[26] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[27] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. New York: Addison-Wesley, 1989.

[28] P. Hansen and N. Mladenović, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[29] S. Benati and G. Laporte, "Tabu search algorithms for the $(r|X_p)$-medianoid and $(r|p)$-centroid problems," *Location Science*, vol. 2, no. 4, pp. 193–204, 1994.

[30] T. Drezner, Z. Drezner, and S. Salhi, "Solving the multiple competitive facilities location problem," *European Journal of Operational Research*, vol. 142, no. 1, pp. 138–151, 2002.

[31] J. Jaramillo, J. Bhadury, and R. Batta, "On the use of genetic algorithms to solve location problems," *Computers and Operations Research*, vol. 29, no. 6, pp. 761–779, 2002.

[32] D. Pérez-Brito, J. Moreno-Pérez, and C. García-González, "Búsqueda por entornos variables: desarrollo y aplicaciones en localización," in *Avances en localización de servicios y aplicaciones*, B. Pelegrín, Ed. Servicio de Publicaciones Universidad de Murcia, 2004, pp. 349–381.

[33] J. Redondo, J. Fernández, I. García, and P. Ortigosa, "A robust and efficient global optimization algorithm for planar competitive location problems," *Annals of Operations Research*, vol. 167, pp. 87–106, 2009.

[34] ——, "Solving the multiple competitive facilities location and design problem on the plane," *Evolutionary Computation*, vol. 17, no. 1, pp. 21–53, 2009.

[35] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[36] P. Ortigosa, I. García, and M. Jelásity, "Reliability and performance of UEGO, a clustering-based global optimizer," *Journal of Global Optimization*, vol. 19, no. 3, pp. 265–289, 2001.

[37] F. Solis and R. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, 1981.

[38] M. P. I. Forum, "MPI: A Message-Passing Interface Standard," *International Journal of Supercomputer Applications*, vol. 8, no. 3-4, pp. 165–414, 1994.

[39] J. Campbell, G. Stiehr, A. Ernst, and M. Krishnamoorthy, "Solving hub arc location problems on a cluster of workstations," *Parallel Computing*, vol. 29, no. 5, pp. 555–574, 2003.

[40] T. Crainic, M. Toulouse, and M. Gendreau, "Synchronous tabu search parallelization strategies for multicommodity location-allocation with balancing requirements," *OR Spectrum*, vol. 17, no. 2-3, pp. 113–123, 1995.

[41] B. Gendron and T. Crainic, "A parallel branch-and-bound algorithm for multicommodity location with balancing requirements," *Computers and Operations Research*, vol. 24, no. 9, pp. 829–847, 1997.

[42] J. Redondo, J. Fernández, I. García, and P. Ortigosa, "Solving the facility location and design $(1|1)$-centroid problem via parallel algorithms," *The Journal of Supercomputing*, vol. 58, no. 3, pp. 420–428, 2011.

[43] J. Rosen and G.-L. Xue, "Computational comparison of two algorithms for the Euclidean single facility location problem," *ORSA Journal on Computing*, vol. 3, no. 3, pp. 207–212, 1991.

[44] A. de Silva and D. Abramson, "A parallel interior point method and its application to facility location problems," *Computational Optimization and Applications*, vol. 9, no. 3, pp. 249–273, 1998.

[45] J. Hennessy and D. Patterson, *Computer architecture, fourth edition: A quantitative approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[46] J. Agulleiro and J. Fernández, "Fast tomographic reconstruction on multicore computers," *Bioinformatics*, vol. 27, no. 4, pp. 582–583, 2011.

[47] J. Agulleiro, E. Garzón, I. García, and J. Fernández, "Vectorization with SIMD extensions speeds up reconstruction in electron tomography," *Journal of Structural Biology*, vol. 170, no. 3, pp. 570–575, 2010.

[48] T. Cura, "A parallel local search approach to solving the uncapacitated warehouse location problem," *Computers and Industrial Engineering*, vol. 59, no. 4, pp. 1000–1009, 2010.

[49] A. Arrondo, J. Fernández, J. Redondo, and P. Ortigosa, "An approach for solving competitive location problems with variable demand using multicore systems," *Optimization Letters*, 2012, (doi: 10.1007/s11590-012-0596-z).

[50] A. Arrondo, J. Redondo, J. Fernández, and P. Ortigosa, "High performance computing approaches for solving a continuous $(1|1)$-centroid problem with endogenous demand," submitted to *Journal of Global Optimization*.

[51] C. Jungthirapanich and T. Pratheepthaweephon, "A geographic information system-based decision support system (GISDSS) for facility location," in *International Conference on Engineering and Technology Management, Pioneering New Technologies: Management Issues and Challenges in the Third Millennium. IEMC '98 Proceedings*, pp. 82–87.

[52] W. Gu, W. Xin, and L. Geng, "GIS-FLSolution: A Spatial Analysis Platform for Static and Transportation Facility Location Allocation Problem." in *ISMIS*, Lecture Notes in Computer Science, J. Rauch, Z. W. Ras, P. Berka, and T. Elomaa, Eds., vol. 5722. Springer, 2009, pp. 453–462.

[53] G. Higgs, "Integrating multi-criteria techniques with geographical information systems in waste facility location to enhance public participation," *Waste managment & Research*, vol. 24, no. 2, pp. 105–117, 2006.

[54] L. Panichelli and E. Gnansounou, "GIS-based approach for defining bioenergy facilities location: A case study in Northern Spain based on marginal delivery costs and resources competition between facilities," *Biomass and Bioenergy*, vol. 32, no. 4, pp. 289–300, 2008.

[55] R. Suarez-Vega, D. Santos-Peñate, P. Dorta-González, and M. Rodríguez-Díaz, "A multi-criteria GIS based procedure to solve a network competitive location problem ," *Applied Geography*, vol. 31, pp. 282–291, 2011.