Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research xxx (2007) xxx–xxx
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Local search algorithms for political districting

Federica Ricca *, Bruno Simeone

Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", Italy

Received 1 July 2005; accepted 1 August 2006

To the memory of Mario Lucertini

Abstract

Electoral district planning plays an important role in a political election, especially when a majority voting rule is adopted, because it interferes in the translation of votes into seats. The practice of gerrymandering can easily take place if the shape of electoral districts is not controlled.

In this paper we consider the following formulation of the political districting problem: given a connected graph (territory) with n nodes (territorial units), partition its set of nodes into k classes such that the subgraph induced by each class (district) is connected and a given vector of functions of the partition is minimized. The nonlinearity of such functions and the connectivity constraints make this network optimization problem a very hard one. Thus, the use of local search heuristics is justified. Experimentation on a sample of medium-large real-life instances has been carried out in order to compare the performance of four local search metaheuristics, i.e., Descent, Tabu Search, Simulated Annealing, and Old Bachelor Acceptance. Our experiments with Italian political districting provided strong evidence in favor of the use of automatic procedures. Actually, except for Descent, all local search algorithms showed a very good performance for this problem. In particular, in our sample of regions, Old Bachelor Acceptance produced the best results in the majority of the cases, especially when the objective function was compactness. Moreover, the district maps generated by this heuristic dominate the institutional district plan with respect to all the districting criteria under consideration. When properly designed, automatic procedures tend to be impartial and yield good districting alternatives. Moreover, they are remarkably fast, and thus they allow for the exploration of a large number of scenarios.

Keywords: Political districting; Local search; Multicriteria optimization

1. Introduction

The present work concerns an application of local search techniques to a graph–theoretic formulation of the political districting problem. As we shall see, the ensuing network optimization model is computationally very hard to solve; thus it makes sense to look at heuristics, in particular local search ones, in order to find "good" feasible solutions with a modest computational effort. Our main objective is the comparison of different local search techniques, both in terms of "quality" of the solution found and computation time. Different districting alternatives were evaluated according to three

* Corresponding author.
E-mail addresses: federica.ricca@uniroma1.it (F. Ricca), bruno.simeone@uniroma1.it (B. Simeone).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.08.065

commonly accepted criteria and our experimental results allowed us to obtain interesting insights on the actual trade-offs that exist between conflicting criteria.

In the last 40 years political districting has attracted the attention of many researchers who proposed a variety of models and algorithms. In the seminal paper (Hess et al., 1965), the districting problem is formulated as a discrete location problem, with binary variables and with linear constraints and objective function. Hess et al. propose a heuristic districting method based on traditional transportation and location/allocation techniques. Districting models are often integer linear programs (ILP's), or integer nonlinear programs (INLP's), according to the objective function under consideration. Garfinkel and Nemhauser (1970) adopt this kind of models and suggest an algorithm based on a set partitioning technique. Starting from these two pioneering papers, many authors got interested in the districting problem and in combinatorial models and algorithms for its formulation and solution, respectively. Generally speaking, districting methods follow either a divisive strategy or an agglomerative one. Several authors (Vickery, 1961; Liitschwager, 1973; Bodin, 1973; Arcese et al., 1992) use a multikernel growth approach, while others (Mills, 1967; Robertson, 1982; Hojati, 1996) adopt location techniques. Merhotra et al. (1998) propose a branch and price algorithm embodying column generation and heuristic search. In a number of papers (Bourjolly et al., 1981; Browdy, 1990; Arcese et al., 1992; Bussamra et al., 1996; Bozkaya et al., 2003) local search algorithms are also considered as a tool for political districting. For a detailed description of the main algorithms and models, see (Grilli di Cortona et al., 1999).

The present paper is a follow-up of the algorithmic work on political districting that was started in the early 1990s by a research group led by Mario Lucertini and one of us. The resulting districting procedure ADEN (Arcese et al., 1992; see also Grilli di Cortona et al., 1999) consists of four integrated algorithmic modules, the last of which is a basic Descent procedure. One aim of the present paper was to obtain a fully automatic procedure that would prove to be superior (with respect to the criteria adopted) to the institutional districting plan used in the Italian 1994, 1996 and 2001 political elections. We felt (and the results of this paper corroborate this idea) that this goal could be achieved if within ADEN the final Descent module were replaced by a more powerful local search heuristic. At the same time, we wanted to undertake a systematic experimental comparison of the performance of some main local search heuristics when applied to real-life political districting problems – a comparison which, as far as we know, was lacking in the literature.

The plan of the paper is as follows. In Section 2 we present the multicriteria connected graph partitioning model for political districting. In Section 3 four basic metaheuristics (Descent, Tabu Search, Simulated Annealing, Old Bachelor Acceptance) are recalled, and details about their implementation for the above model are provided. Section 4 describes the experimental plan we have chosen, and in Section 5 the results of our computational experiments are discussed. Final conclusions are presented in Section 6.

2. The mathematical model

Some of the authors mentioned above adopt a graph–theoretic model for political districting. In fact, if a territory is divided into n elementary units (counties, townships, wards, ...), it can be always represented as a connected n-node graph, where an edge between two nodes exists if and only if the two corresponding units are neighboring. The nodes of the graph can be weighted with the population of the associated territorial units, while arc-weights represent distances between two units. These distances are assumed to be *road* distances, so as to take into account orography and other geographical barriers. In political districting the territory must be subdivided into a fixed number k, $k < n$, of parts (called districts) such that each elementary territorial unit belongs to only one district and it cannot be split between two different districts (*integrity* constraint). A district satisfies the *contiguity* constraint if it is formed by a set of geographically contiguous units. Formally, a district is a connected subset of nodes and a *district map* is a connected partition of the graph. This means that each class of the partition induces a connected subgraph. For example, in Italy each regional territory is divided into townships and each township corresponds to a territorial unit, except for large townships, such as the city of Rome, which must be further divided into wards. In our application we consider five territories corresponding to different Italian Regions. In the input data, each Region is

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

divided into a number of elementary units corresponding to census tracts.

2.1. Criteria

Let k be the number of districts and n the number of territorial units. Let p_i be the population of unit i, $i = 1, 2, \ldots, n$. Suppose that the electoral district plan must be drawn for an election where a majority rule is adopted. In this case, the most important criterion is *population equality*. Population equality holds when each district has exactly the same population as any other district, that is, each district population is equal to the average district population $\bar{p}$. Let P_j denote the population of district j, $j = 1, 2, \ldots, k$, and P the total population (w.r.t. the whole territory). Obviously, we have $P = \sum_{i=1}^{n} p_i$ and, therefore, $\bar{p}$ can be easily obtained as $\bar{p} = \frac{P}{k}$.

Generally, population equality measures are computed as normalized norms of the vector $(P_1 - \bar{p}, P_2 - \bar{p}, \ldots, P_k - \bar{p})$. For example, index U_1 is given by the L_1 norm divided by $k\bar{p}$ (Arcese et al., 1992):

$$U_1 = \frac{\sum_{j=1}^{k} |P_j - \bar{p}|}{k\bar{p}}, \tag{1}$$

and it takes values in the interval $\left[0, \frac{2(k-1)}{k}\right]$.

On the other hand, one may adopt the following index U_2 based on the L_2 norm:

$$U_2 = \frac{\sum_{j=1}^{k} (P_j - \bar{p})^2}{k\bar{p}}. \tag{2}$$

In our application we consider index (1). Actually, although in general indicators taking values in the interval $[0, 1]$ are preferred, index U_1 has the advantage of being particularly sensitive to small differences in population equality between two maps. This index has also an interesting interpretation as the average percent deviation of the district populations from $\bar{p}$. Finally, it must be noticed that it is quite rare that U_1 takes values greater than 1. However, in order to obtain an index which varies between 0 and 1, we can always refer to the normalized version of U_1:

$$U_3 = \frac{\sum_{j=1}^{k} |P_j - \bar{p}|}{2(k-1)\bar{p}}. \tag{3}$$

Notice that, unless the district map is extremely out of balance, that is, there are many small districts formed by a single territorial unit and just a few big districts collecting the remaining ones, U_1 always takes values smaller than 1. On the other hand, U_1 distinguishes between couples of balanced maps better than U_3, providing a sound indicator for the evaluation of the alternative maps under the population equality point of view.

Consider the total number of seats S and the total population P, the ratio $\Gamma = P/S$ gives a rough evaluation of the population per seat and, ideally, each seat should correspond to Γ votes. Actually, when a single-member district map satisfies population equality perfectly ($P_j = \bar{p}, \forall j$) the ratio between the number of voters and the number of seats in each district perfectly coincides with $\Gamma(P_j/S_j = \bar{p}/1 = P/S)$. The same can be required in multi-member district maps. In fact, *fair apportionment* can be considered just as an extension of population equality. A district map satisfies the fair apportionment criterion if each district population size is a multiple of Γ (see Grilli di Cortona et al., 1999, Section 10.2). Thus, the population equality and fair apportionment criteria embody the principle of one-man-one-vote in the single- and multi-member district case, respectively.

Even when the principle one-man-one-vote is reasonably satisfied, still manipulation of the district shape (*gerrymandering*) may occur. One of the most powerful criteria for preventing such malpractice is *compactness*, which forces the districts to have regular geometric shapes, so as to avoid octopus or banana shaped districts to be drawn. In our application we adopt a compactness measure which is quite complex and highly nonlinear (Arcese et al., 1992; Grilli di Cortona et al., 1999, Paragraph 10.3). This index is given by the sum of compactness indices computed over each district separately. For a given district C it can be briefly described as follows. Let d_{ij} be the distance between unit i and unit j. For each unit $i \in C$ compute its eccentricity

$$d(i) = \max_{j \in C} d_{ij}$$

and set

$$\delta = d(s) = \min_{i \in C} d(i).$$

By definition, s is the center of district C and the compactness in district C is measured by:

$$I = \frac{\sum_{i \in C} p_i}{\sum_{j \in D} p_j},$$

where $D = \{j \in V : d_{js} \leqslant \delta\}$.

The last criterion is *conformity to administrative boundaries* (or simply *administrative conformity*),

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

which requires district maps to take into account administrative boundaries that already exist, such as Regions, Provinces, Counties, and health-care districts. In this application, we adopt the administrative conformity index proposed in (Arcese et al., 1992). It is defined on the basis of the number of those territorial units that produce discrepancies between the administrative districts and the electoral ones. These units are called *misplaced units*. The more the units of this type, the worst the resulting value of the administrative conformity index. Consider a given type of administrative areas. If an electoral district is completely included into an administrative one, then it is ignored. On the other hand, consider an electoral district that shares portions of its land with different administrative areas of a given type. In this case, among all the units in the district, we compute the percentage of those that have at least one adjacent unit belonging to the same electoral district, but to a different administrative area (misplaced units). The global index, which varies between 0 and 1, is obtained by averaging over all types of administrative boundaries and over all the electoral districts in the map. A detailed description of the index is reported in (Grilli di Cortona et al., 1999) and in (Ricca and Simeone, 1997). Notice that conformity to administrative boundaries is a very important criterion in view of the advantages brought forth in terms of organization and management of the election over the territory: respecting administrative boundaries can be useful to simplify electoral procedures, such as the identification of the electoral body and other organization issues.

2.2. The network optimization model and its computational challenges

In this paper we shall consider the criteria discussed in the previous section as objectives. Unfortunately, these criteria are often in conflict; therefore, optimizing a certain objective generally results in a degradation of at least one of the others. Since it is impossible to reach the best for all the objectives simultaneously, in practice, one must be satisfied with a good compromise between the objectives' values in the final solution which, in the best case, corresponds to a *Pareto-optimal solution*. Given p objectives, a solution s is *Pareto-optimal* if any other solution s', which is better than s for some objective, is necessarily worse than s for at least one other objective.

We are now in position to state a precise mathematical formulation of the network optimization model under consideration.

Given:

- a connected (usually planar) graph $G=(V,E)$ with n nodes (the contiguity graph),
- an integer k, $1 \leqslant k \leqslant n$ (the number of districts),

a *connected k-partition* of G is a partition π of V into k subsets $C_1,\ldots,C_k$ such that each C_h $(h=1,\ldots,k)$ induces a connected subgraph of G. These subsets are called *districts*. The collection of all connected k-partitions of G will be denoted by Π_k (G).

One wants to solve the following vector-minimization problem:

$$\min_{\pi\in\Pi_k(G)} \{f_1(\pi), f_2(\pi), f_3(\pi)\}, \tag{4}$$

where $f_1(\pi), f_2(\pi), f_3(\pi)$ measure population *in*equality, *non*compactess, and *non*conformity to administrative boundaries, respectively, as discussed in the previous section. Solving (4) consists in finding some Pareto-optimal connected k-partition of G: it is a very difficult and challenging computational task, due to the following aspects:

1. *Connectivity constraints*
   Such constraints make the problem much more difficult than other partitioning problems in combinatorial optimization, such as coloring or frequency assignment. A clustering problem with connectivity constraints was investigated by Hansen et al. (2003). They provide a binary linear programming model with exponentially many constraints and a row generation exact algorithm for instances up to about 600 nodes. However, their model, unlike ours, has a single and linear objective function. In the special case of trees, a binary linear programming formulation with polynomial size is available and its continuous relaxation turns out to be remarkably tight (Lari et al., 1998). Mehrotra et al. (1998) use column generation for the exact solution of a political districting instance with 50 nodes, but they do not take into account administrative conformity and use a much simpler compactness indicator.
2. *Presence of multiple (conflicting) objective functions*
   Dell' Amico and Maffioli (1996) exhibit examples showing that maximizing a weighted average of

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

two objective functions is NP-hard even when one can separately maximize both objective functions in polynomial time.

3. *Nonlinearity of all the objective functions*.
   All the criteria $f_1(\pi)$, $f_2(\pi)$, $f_3(\pi)$ are nonlinear functions of the variables x_{ih} representing the presence/absence of node i in district h; one of them, the noncompactness $f_2(\pi)$, is not even given by an explicit analytical expression, but, rather, by an *oracle*, i.e., a subroutine that returns the value $f_2(\pi)$ for each $\pi \in \Pi_k(G)$.
4. *Size of the graph*
   In real-life applications, the size of contiguity graphs may reach one thousand nodes and more.

Finding Pareto-optimal solutions for the three above criteria is computationally hard. In fact, even minimizing the simplest one among the three criteria, namely, the population inequality $f_1(\pi)$, turns out to be an NP-complete problem already when the graph G is a spider, i.e., a tree with at most one node with degree 3 or more (De Simone et al., 1990).

2.3. Pareto optimality and local search

Local search algorithms generally are able to produce only "locally Pareto-optimal" solutions. Given a multiobjective program with p objective functions, a solution is *locally Pareto-optimal* if there is no local perturbation that makes an objective improve without worsening some other objective. Local Pareto-optimality depends on the topology of the space of feasible solutions and in particular on the specific structure of the neighborhood.

Even if local search algorithms are generally used with a single objective, they can be adjusted for the multiobjective case as follows. Suppose a minimization problem with p objectives $(z_1, z_2, \ldots, z_p)$ is given and consider z_1 as the "target" objective function. A percentage $\alpha_i \geqslant 0$ of the maximum acceptable worsening for each objective z_i, $i \neq 1$ must be fixed. If during an iteration a solution π changes into a new solution π', then for each z_i, $i \neq 1$ the following must hold:

$$z_i' \geqslant (1 - \alpha_i) z_i$$

or, equivalently

$$(z_i - z_i') \leqslant \alpha_i z_i,$$

where z_i', $i \neq 1$ are the new values (corresponding to solution π') for the $p - 1$ objectives which are not directly optimized. This means that, except for the target z_1, each other objective can worsen up to its percentage α_i. Notice that the value α_i, $i \neq 1$, is strategically important because it determines how much 'control' one has on the complete set of objectives. In general, when the size of the problem is very large, local search heuristics produce a final solution that rarely is (even locally) Pareto-optimal. In any case, when the size of the neighborhood is "small", it is easy to verify if a solution is locally Pareto-optimal or not by enumerating all the neighboring alternatives. Given an instance of size L of problem P, let S be the set of solutions to such an instance and let $N(s)$ be the set of solutions in the neighborhood of $s \in S$. A neighborhood structure can be defined *good* if, for some polynomial p, one has

$$|N(s)| \leqslant p(L), \quad \forall s \in S,$$

that is, the size of the neighborhood of any solution is smaller than a polynomial in the variable L. Clearly, if the neighborhood structure is good, verifying if a solution is locally Pareto-optimal or not can be done in polynomial time. The computational complexity of local search has been investigated in (Johnson et al., 1988).

Another way to transform the multiobjective model into a single objective one is to consider a convex combination of all the objective functions, weighted by appropriate weights. In our application we follow this approach because the other method turned out to be not effective within local search. Typically, local search needs to pass through bad points in order to reach good local optima. Using fixed thresholds prevents from excessive deterioration of the objectives, however, in many cases, a strong worsening of an objective could be very useful in order to find good search directions for the other objectives. We observed that adopting thresholds for all the objectives generally reduces the power of local search techniques, providing poor final solutions. Actually, the multiobjective approach discussed above cannot be combined with local search algorithms if the thresholds are kept constant. On the other hand, a good knowledge of the topology of the space of the solutions can be exploited to design threshold updating procedures. In this case it would be necessary to study appropriate (probably nonmonotone) updating schemes for each α_i to enhance the search of good points for more than one objective simultaneously. In this

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

context, the knowledge of the relations that intrinsically exist between two criteria (trade-off analysis) can be very useful to inspect the different regions of the solution space.

3. Local search algorithms

3.1. Generalities

We consider four different local search approaches and we are interested in evaluating their performance for the districting problem and in comparing them in order to understand if one is actually better than the others for this specific application. The algorithms considered are: a basic Descent algorithm, Tabu Search (Glover, 1989; Glover, 1990), Simulated Annealing (Kirkpatrick et al., 1983; Cerny, 1985) and the Old Bachelor Acceptance algorithm (Hu et al., 1995). The last procedure belongs to the class of *threshold algorithms*. We were particularly interested in its performance and in the comparison with Tabu Search and Simulated Annealing, which are generally known as good heuristic methods for many combinatorial problems. The Descent algorithm has been considered as a benchmark for the evaluation of the performance of the others.

Our purpose was *not* to single out a best possible heuristic for the problem under consideration. Rather, we wanted to compare the different metaheuristic approaches in their quintessential form, because we were interested in basic patterns of behaviour of the different approaches, in order to understand which ones have the greatest potential in solving the problem. For this comparison to be fair, we have chosen to implement *streamlined* versions of all these algorithms, removing as far as possible, common enhancements that would have marred our conclusions. Notice that sophisticated implementations tend to be hybrid and might not lead to robust results. A fair analysis should take into account not only the quality of the solutions obtained by the different algorithms, but also the amount of computational resources (especially processing time) spent for getting such solutions. For this reason, we have compared the best solutions produced by the four algorithms after 20,000, 40,000, 60,000, and 80,000 iterations. We have added a multistart feature to Descent, which tends to be easily trapped in local optima. When this algorithm stops prematurely, it is restarted from another initial feasible solution, and so on, until the desired number of iterations is obtained. Also one of our implementations of Tabu Search makes use, although much less frequently, of the same feature. As we shall see later, also this algorithm makes use of exact neighborhoods, a feature that is likely to lead to local optima. In this case, extreme diversification tactics such as multistart may help finding improved solutions at a later stage.

3.2. Common implementation features

As customary, the input contiguity graph $G = (N, E)$ is given through its adjacency list. When a local search algorithm must be implemented, the topology of the neighborhood of a given solution s is particularly important. More precisely, the definition of a neighboring solution of s and the rule used for generating neighboring solutions must be specified. For our districting problem on a graph $G = (N, E)$ consider two solutions (connected partitions) s and s'. Each solution consists of k subsets of nodes in N, so that $s = \{C_1, C_2, \ldots, C_k\}$ and $s' = \{C'_1, C'_2, \ldots, C'_k\}$. Solutions s and s' are neighboring if there exist two subsets, C_l, C_q, and a node $x \in C_l$ such that

$$C_h = C'_h, \quad \forall h = 1, 2, \ldots, k, \quad h \neq l, q$$

and

$$C'_l = C_l \setminus \{x\}, \quad C'_q = C_q \cup \{x\}.$$

In other words, s' is a neighbor of s if either one can be obtained from the other by a single node migration. Notice that this neighborhood is good according to the definition given in Section 2.2. In fact, given that k is the number of districts and n the number of nodes, in order to generate all the solutions in $N(s)$ at most nk migrations are necessary. Moreover, only those nodes belonging to the boundary of a district can move from the district to another; therefore, the actual number of necessary moves is generally much smaller than nk.

We shall call *feasible solution* any connected k-partition of G. A move generates a feasible solution (and, therefore, it will be called a *feasible move*) if it does not disconnect any district. In particular, since a move involves only two districts, the feasibility condition means that both the origin-district and the destination-district must not be disconnected by the move. Equivalently, the migrating node:

(a) must be *adjacent* to some node of the destination-district;

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

(b) must not be a *cut-node* for the origin-district.

For a given district C_h the *boundary* of the district is given by the subset of nodes that are adjacent to some node belonging to a different district, while its *cocycle* is the set of those edges having an endpoint in C_h and the other outside C_h. In order to test condition (a) efficiently, both the boundary and the cocycle of the districts are stored and dynamically maintained after each move. In order to test condition (b) efficiently, all the nodes adjacent to the migrating one, say, x, are previously marked and a *breadth-first* search is started from one of them. Clearly, x is not a cut-node iff every marked node is reached during the search. As soon as this condition is satisfied, the search stops. Since G is planar, this usually happens at an early stage of the search, resulting in considerable time savings.

The origin-district is always chosen at random. The choice of the migrating node within the boundary of the origin-district depends on the heuristic at hand. In our implementation, the Descent algorithm and the first of our Tabu Search implementations make use of *exact* neighborhoods, that is, they look at *all* the feasible solutions in the neighborhood (equivalently, all the nodes in the boundary) in order to find the *best* one. As we shall see, this is not the case for our second implementation of Tabu Search, and for Old Bachelor Acceptance and Simulated Annealing, which rather choose *random* solutions in the neighborhood of the current one.

When exact neighborhoods are adopted, there is the possibility that the algorithms stop in a local optimum before the stopping condition is met. Therefore, whenever a local optimum has been attained for this reason, the algorithms are restarted from a random initial feasible solution.

In order to get one such feasible solution, the following procedure is used:

1. a spanning tree T of G is randomly generated;
2. k centers (one per district) are randomly chosen among $2k$ candidate centers (usually, these are the most populous towns);
3. k-1 randomly chosen edges of T are cut, in such a way that each of the resulting k subtrees contains exactly one center. This is accomplished as follows: starting from T, and as long as there are two (or more) centers in some current subtree, cut a randomly chosen edge along the unique path connecting them in T;
4. the k initial districts are the node-sets of the k subtrees obtained in this way.

3.3. The metaheuristics

In this section we briefly outline the four local search metaheuristics examined in this work, giving a schematic description for each of them.

Consider a problem in which the objective function f must be minimized and let s be a feasible solution. Let $N(s)$ be the set of solutions in the neighborhood of s and $\Delta(t) = f(t) - f(s)$, $t \in N(s)$ the variation of the objective function when going from s to t. We will call s^* a *best solution* in $N(s)$ if we have $\Delta(s^*) = \min\Delta(t)$, the minimum being taken among all feasible solutions $t \in N(s)$. Starting from an initial feasible solution, at each iteration the Descent algorithm searches for a best solution in the neighborhood of the current solution. Clearly, the Descent algorithm stops when there are no more feasible moves that make the objective function improve, and this means that the algorithm often remains entrapped in bad local optima. This kind of problem is generally alleviated by a multiple start approach (*Multistart Descent algorithm*) according to which the search is re-started from a new initial (random) solution. Many restarts produce many new possible directions for the search, but do not guarantee that those directions are good. Let M denote the total number of iterations which was fixed to define the stopping condition. In general, when M is increased, the performance of the Descent algorithm improves, due to the fact that, in this case, many additional solutions are evaluated. Indeed, if M is very large, the Descent algorithm can occasionally perform even better than other, more sophisticated, local search algorithms. According to our approach, we adopt a multistart Descent algorithm, in order to guarantee that the fixed total number of iterations ($M = 80{,}000$) is performed even in this case.

We consider the Descent algorithm as a benchmark in order to compare the performance of the other local search algorithms, namely, Tabu Search, Simulated Annealing and Old Bachelor Acceptance.

First of all, we consider the well known Tabu Search algorithm. The basic steps of Tabu Search are described below.

We provide two alternative implementations of the Tabu Search algorithm, basically differing in the choice (*best* vs. *random*) of the move from the

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

current solution. In both cases we consider a fixed length tabu list. For the stopping condition, besides the total number of iterations, we fix a maximum tolerable number of successive disadvantageous moves. When either stopping condition is met the algorithms stop. Although both implementations are streamlined versions of Tabu Search, we also implemented some particular features which make provision for some special (particularly advantageous) conditions, such as the possibility of performing a forbidden move (*aspiration criterion*). Actually, in our algorithms, a tabu move can be performed when it generates a new solution that is strictly better than the best solution found up to the current iteration. Since the tabu status of a move may produce premature stops of the algorithm, in our first implementation we basically follow the procedure described in Fig. 1, but we adopt a multistart approach as a search diversification tool (*Exact Tabu Search*). On the other hand, in our second version of Tabu Search, given a current solution s, we implement the random choice of a solution in the neighborhood of s (*Random Tabu Search*).

We implemented a basic version of Simulating Annealing. Let ε and T_0 be two fixed positive thresholds. The Simulated Annealing algorithm can be briefly described as in Fig. 2.

Old Bachelor Acceptance is a heuristic which belongs to the class of threshold algorithms, but it is characterized by a special procedure for updating the threshold. At each step the threshold value specifies the maximum acceptable change in the objective function. When the algorithm goes from a solution s to a new solution s' in the neighborhood of s, the objective function may improve, otherwise it may worsen within the threshold limit. Each time the threshold is automatically updated in a *nonmonotonic* way, with even the possibility that it reaches negative values. In particular, the threshold decreases after an improvement in the objective function and increases when the objective function worsens. Such an updating strategy has been shown to be an efficient way to avoid premature arrests in bad local optima (Hu et al., 1995). Actually, when the threshold changes, the search has the possibility of finding new promising Descent directions, especially when the current solution is far from a local optimum. In practice, the algorithm becomes pretentious when improving moves are easily found, while it has no pretensions in the neighborhood of locally optimal solutions, where it is hard to find improving moves. In the latter case, the algorithm has the ability to escape from bad local minima by increasing the threshold value, while, during the ambitious phase, successive threshold decreases cause the algorithm to accelerate along steep Descents towards a local minimum point. In Fig. 3, the main steps of Old Bachelor Acceptance are reported. Functions $\Delta^+(i)$ and $\Delta^-(i)$ are both positive and they are involved in the threshold updating, while the stopping condition is established on the basis of a fixed number of total iterations.

In our application $\Delta^+(i)$ and $\Delta^-(i)$ are linear functions of the quantity $\left(1-\frac{i}{M}\right)$, where i denotes the current iteration. More precisely, when the last steps of the algorithm are performed – that means a low value for $(1 - i/M)$ – both $\Delta^+(i)$ and $\Delta^-(i)$ become small. This corresponds to a strategy of maximum exploitation of the last iterations in order to find some additional local optima. Notice that the updating of the thresholds $\Delta^+(i)$ and $\Delta^-(i)$ is symmetric in order to avoid undesired unbalanced threshold adjustments. Moreover, one must take into account that an extremely high threshold prevents from any move and may provoke the

1.	Select an initial feasible solution $\bar{s}$
2.	**repeat**:
	generate the set of all feasible moves producing the corresponding set of feasible solutions in the neighborhood $N(s)$ of the current solution s
2.1.	**if** there is at least a feasible non-tabu move
	select a feasible non-tabu move leading to a best solution $s' \in N(s)$
	update the tabu-list
2.2.	**else** [all possible moves are either infeasible or tabu]
	STOP (a local optimum is found)
2.3.	**end if**
3.	**until** the stopping condition is met
4.	The final solution is the best local optimum found s^*

Fig. 1. Tabu Search.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Simulated Annealing [Kirkpatrick et al., 1983; Cerny, 1985]

1. Select an initial feasible solution $\bar{s}$
2. Let $T = T_0$ be the initial temperature
3. **while** $T \geq \varepsilon$, do the following steps
3.1. **for** L times do the following steps
select at random a feasible move producing a neighborhood feasible solution s' of the current solution s
3.1.1. **if** $\Delta = \Delta(s') < 0$ (*downhill move*)
the move is performed
3.1.2. **else** [$\Delta = \Delta(s') \geq 0$] (*uphill move*)
the move is performed with probability $p = e^{-\Delta/T}$
3.1.3. **end if**
update the temperature T
3.2. **end for**
4. **end while**
5. The final solution is the best local optimum found s^*

Fig. 2. Simulated Annealing.

Old Bachelor Acceptance [Hu et al., 1995]

1. Select an initial feasible solution $\bar{s}$
2. Define an initial threshold T_1
3. **repeat**
select at random a feasible move producing a neighborhood feasible solution s' of the current solution s
3.1. **if** $\Delta(s') < T_i$
perform the move
3.1.1. **if** $\Delta(s') < 0$ decrease the threshold: $T_{i+1} := T_i - \Delta^-(i)$
3.2. **otherwise** [$\Delta(s') \geq T_i$]
increase the threshold: $T_{i+1} := T_i + \Delta^+(i)$
3.3. **end if**
4. **until** the stopping condition is met
5. The final solution is the best local optimum found s^*

Fig. 3. Old Bachelor Acceptance.

algorithm to find bad local optima. On the other hand, a threshold which decreases too fast tends to become negative, thus producing a premature stop of the algorithm. This is similar to what happens with the Descent algorithm, since in this case it becomes very difficult to find good moves. We calibrated $\Delta^+(i)$ and $\Delta^-(i)$ in order to allow a long and diversified search, increasing the chance of finding many different local optima. If the final set of encountered local optima is large, the final solution is likely to be a good one.

4. Experimental plan

The performance of the local search techniques for political districting was studied over a sample of five Italian Regions, namely, Abruzzi, Latium, Marches, Trentino–Upper Adige (for the sake of brevity, we will refer to this Region as Trentino) and Piedmont. In order to make such sample as representative as possible, the territories were chosen so as to be very different in extension, characteristics and shape. The main features of the corresponding graphs are summarized in Table 1.

Notice that they are all planar graphs, since they are *contiguity graphs*. The density of a planar graph with n nodes varies from a minimum of $\frac{n-1}{n}$ (this is the case of a tree) to a maximum of $\frac{3n-6}{n}$ (triangulated graphs). In Table 1 for each region the maximum density values are reported. All our graphs have high densities (greater than 2), but Piedmont nearly reaches its maximum.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Table 1
Graphs of Italian regions

Region	Nodes	Edges	Density	$(3n-6)/n$	Districts
Abruzzi	305	847	2.78	2.980	11
Latium	374	1006	2.69	2.983	19
Marches	246	674	2.74	2.975	12
Piedmont	1208	3527	2.92	2.995	28
Trentino	339	938	2.78	2.982	8

The number of nodes varies from a minimum of 246 (Marches) to a maximum of 1208 (Piedmont) and also the number of edges and the number of districts to be drawn are very different from region to region. Notice that the number of districts is related to the number of nodes of the graph very loosely. For example, Trentino has about one hundred nodes more than Marches and, however, the number of its districts is smaller (only 8 for Trentino and 12 for Marches). The number of districts in each region was fixed according to the Italian electoral law of 1992.

Table 2
Parameter's values for local search

Algorithm	Parameters' values	
Tabu Search	Tabu list length	24
	Maximum number of successive disadvantageous moves	208
Simulated Annealing	Initial temperature T_0	1
	Final threshold ε for the temperature	10^{-5}
	Cooling rate	0.998
Old Bachelor Acceptance	Granularity value for population equality	10^{-8}
	Granularity value for compactness	10^{-3}
	Granularity value for administrative conformity	10^{-7}
	Granularity value for target mixture	10^{-6}

All the algorithms start from a random initial solution which is generated by selecting a spanning tree for the graph G at random and obtaining k subtrees by the random selection of their roots. The trees are explored in breadth-first search order.

Table 3
Target values at iterations 20,000, 40,000, 60,000 and 80,000 for Abruzzi

ABRUZZI	0 iterations	Algorithm	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations
Population equality	0.752	Descent (4619)	0.310	0.310	0.310	0.310
		Exact Tabu Search (2332)	0.574	0.456	0.456	0.399
		Random Tabu Search	0.046	**0.013**	**0.013**	**0.013**
		Old Bachelor Acceptance	**0.042**	0.030	0.030	0.030
		Simulated Annealing	0.201	0.044	**0.013**	**0.013**
Compactness	0.766	Descent (2805)	0.345	0.236	0.236	0.176
		Exact Tabu Search (2401)	**0.128**	0.128	0.097	0.097
		Random Tabu Search	0.455	0.405	0.405	0.405
		Old Bachelor Acceptance	0.135	**0.092**	**0.092**	**0.089**
		Simulated Annealing	0.553	0.384	0.382	0.382
Administrative conformity	0.338	Descent (1900)	0.127	0.053	0.053	0.053
		Exact Tabu Search (2459)	**0.045**	**0.045**	**0.045**	**0.045**
		Random Tabu Search	0.081	0.081	0.081	0.081
		Old Bachelor Acceptance	0.079	0.054	0.054	0.048
		Simulated Annealing	0.200	0.135	0.059	0.059
Target mixture (0.5;0.3;0.2)	0.674	Descent (2487)	0.474	0.474	0.474	0.392
		Exact Tabu Search (2408)	0.478	0.478	0.478	0.461
		Random Tabu Search	**0.275**	**0.273**	**0.252**	**0.252**
		Old Bachelor Acceptance	0.309	0.285	0.285	0.285
		Simulated Annealing	0.434	0.329	0.320	0.320

For Descent and Exact Tabu Search the number of repeated restarts is specified in brackets.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

The possible objective functions are population equality, compactness and conformity to administrative boundaries, but a mixture of the three is also considered. In this case the objective function is given by a convex combination of the objectives with weights equal to 0.5 for population equality, 0.3 for compactness and 0.2 for conformity to administrative boundaries, according to the relative importance accounted by a panel of experts to the tree objectives in political districting problems. From now on we will refer to this *mixed* objective function as *target mixture*.

As stated above, the total number of iterations was fixed to 80,000 for all the heuristics. This number of iterations has shown to be sufficient to guarantee that all the heuristic achieve their best solution.

Excluding Descent, all the algorithms considered are characterized by their own typical parameters which are strategic for the efficiency of the search (see Table 2). Therefore, a careful tuning of these parameters is extremely important in order to find good local optima.

For Tabu Search we set the length of the tabu list equal to 24 and we fixed a maximum number of successive disadvantageous moves equal to 208.

Simulated Annealing depends on the initial temperature, the cooling rate and a final threshold ε for the temperature. We fixed the initial temperature equal to 1 and experimentally calibrated the cooling rate (0.998) and the final value of the temperature ($\varepsilon = 10^{-5}$). The values adopted for these parameters are seen to guarantee the convergence of Simulated Annealing. In addition, these values allowed Simulated Annealing to perform a total number of iterations approximately equal to M, making possible the comparison between the performances of all the metaheuristics at the same breakpoints.

Old Bachelor Acceptance is characterized by a parameter – called *granularity* – which is

Table 4
Target values at iterations 20,000, 40,000, 60,000 and 80,000 for Latium

LATIUM	0 iterations	Algorithm	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations
Population Equality	0.897	Descent (6511)	0.489	0.489	0.489	0.489
		Exact Tabu Search (3920)	0.736	0.736	0.736	0.736
		Random Tabu Search	0.257	0.237	0.197	0.197
		Old Bachelor Acceptance	**0.123**	**0.123**	**0.123**	**0.123**
		Simulated Annealing	0.253	0.179	0.179	0.179
Compactness	0.744	Descent (3898)	0.368	0.368	0.368	0.368
		Exact Tabu Search (3693)	0.172	0.172	0.172	0.172
		Random Tabu Search	0.463	0.463	0.463	0.463
		Old Bachelor Acceptance	**0.129**	**0.101**	**0.087**	**0.083**
		Simulated Annealing	0.406	0.356	0.356	0.356
Administrative Conformity	0.356	Descent (3251)	0.100	0.100	0.100	0.100
		Exact Tabu Search (3641)	**0.048**	0.048	0.048	0.048
		Random Tabu Search	0.109	0.093	0.074	0.071
		Old Bachelor Acceptance	0.061	**0.042**	**0.040**	**0.037**
		Simulated Annealing	0.206	0.079	0.079	0.079
Target Mixture (0.5;0.3;0.2)	0.743	Descent (3818)	0.520	0.520	0.520	0.520
		Exact Tabu Search (3665)	0.602	0.602	0.602	0.602
		Random Tabu Search	0.429	0.410	0.390	0.350
		Old Bachelor Acceptance	**0.374**	0.342	**0.329**	**0.329**
		Simulated Annealing	0.387	**0.337**	0.337	0.337

For Descent and Exact Tabu Search the number of repeated restarts is specified in brackets.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

fundamental for threshold updating when the objective function improves or worsens (functions $\Delta^+(i)$ and $\Delta^-(i)$ actually depend on granularity). For the granularity parameter, we observed that very low values are necessary in order to guarantee a good performance of Old Bachelor Acceptance. On the basis of our experimental analysis, granularity seems to be sensitive to the target objective formulated in the model. Notice that this does not mean that our choice for the granularity parameter is instance-dependent, since, in fact, different objective functions define different problems. If the target is compactness, granularity takes values of magnitude 10^{-2} or 10^{-3}. When conformity to administrative boundaries is considered, granularity becomes much smaller (10^{-7}), but it reaches the smallest value when population equality is the target criterion (10^{-8}). For the target mixture we obtain the best performance with a granularity value of magnitude 10^{-6}. In general, if the granularity value is too high, Old Bachelor Acceptance produces bad effects. In particular, according to the threshold schedule, the algorithms pass from too permissive values of the threshold to very small (even negative) ones. In the first case, all moves are acceptable, while no move is sufficiently good to be accepted in the second case. At any extent, since the threshold is strictly related to which objective function is considered in the model, granularity values must be accurately calibrated with respect to the specific model at hand.

5. Results

In this section we discuss the experimental results of our application. First of all we show (Tables 3–7) the results obtained in a single run of each metaheuristic in each region and with each possible target, starting from an initial random solution. Notice that the performance of those algorithms that adopt

Table 5
Target values at iterations 20,000, 40,000, 60,000 and 80,000 for Marches

MARCHES	0 iterations	Algorithm	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations
Population Equality	0.757	Descent (5422)	0.288	0.288	0.288	0.288
		Exact Tabu Search (2029)	0.419	0.419	0.419	0.419
		Random Tabu Search	**0.003**	**0.003**	**0.003**	**0.002**
		Old Bachelor Acceptance	0.010	0.010	0.010	0.010
		Simulated Annealing	0.160	0.014	0.014	0.014
Compactness	0.795	Descent (2974)	0.353	0.353	0.353	0.261
		Exact Tabu Search (1961)	0.303	0.195	0.195	**0.091**
		Random Tabu Search	0.460	0.384	0.384	0.384
		Old Bachelor Acceptance	**0.155**	**0.108**	**0.108**	0.108
		Simulated Annealing	0.465	0.346	0.346	0.346
Administrative Conformity	0.399	Descent (2098)	0.112	0.112	0.097	0.097
		Exact Tabu Search (2101)	**0.026**	**0.026**	**0.026**	**0.026**
		Random Tabu Search	0.095	0.090	0.070	0.068
		Old Bachelor Acceptance	0.077	0.066	0.054	0.054
		Simulated Annealing	0.151	0.073	0.073	0.073
Target Mixture (0.5;0.3;0.2)	0.697	Descent (2565)	0.456	0.456	0.456	0.401
		Exact Tabu Search (2008)	0.453	0.435	0.435	0.435
		Random Tabu Search	0.252	0.238	**0.229**	**0.226**
		Old Bachelor Acceptance	**0.249**	**0.232**	0.232	0.232
		Simulated Annealing	0.352	0.243	0.243	0.243

For Descent and Exact Tabu Search the number of repeated restarts is specified in brackets.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Table 6
Target values at iterations 20,000, 40,000, 60,000 and 80,000 for Trentino

TRENTINO	0 iterations	Algorithm	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations
Population Equality	0.896	Descent (3565)	0.348	0.348	0.348	0.348
		Exact Tabu Search (870)	0.541	0.536	0.536	0.536
		Random Tabu Search	0.167	**0.037**	0.036	0.036
		Old Bachelor Acceptance	**0.135**	0.065	0.065	0.065
		Simulated Annealing	0.149	0.047	**0.017**	**0.001**
Compactness	0.821	Descent (1494)	0.363	0.363	0.234	0.234
		Exact Tabu Search (774)	0.182	0.182	0.182	0.182
		Random Tabu Search	0.584	0.584	0.584	0.584
		Old Bachelor Acceptance	**0.125**	**0.103**	**0.098**	**0.096**
		Simulated Annealing	0.590	0.480	0.475	0.475
Administrative Conformity	0.319	Descent (789)	0.067	0.067	0.067	0.065
		Exact Tabu Search (908)	**0.032**	0.032	**0.013**	**0.013**
		Random Tabu Search	0.052	**0.029**	0.027	0.023
		Old Bachelor Acceptance	0.056	0.056	0.056	0.056
		Simulated Annealing	0.229	0.198	0.188	0.164
Target Mixture (0.5; 0.3; 0.2)	0.758	Descent (1385)	0.464	0.464	0.464	0.457
		Exact Tabu Search (817)	0.535	0.535	0.492	0.492
		Random Tabu Search	0.362	0.337	0.313	**0.216**
		Old Bachelor Acceptance	**0.330**	0.324	0.323	0.260
		Simulated Annealing	0.393	**0.269**	**0.233**	0.233

For Descent and Exact Tabu Search the number of repeated restarts is specified in brackets.

a multistart approach does not depend on the initial solution. On the other hand, this may be the case for the other heuristics. As we will see going on in this section, when a number of repeated runs is performed with any of these algorithms the variability of the objective function value is very small, hence leading to the conclusion that the initial solution does not affect the performance of any of our compared algorithms. Moreover, since the algorithms are very fast (few seconds), the running times are omitted.[1]

We consider all the possible combinations of a region and an objective function. We run the five algorithms and record the value of the best solution produced by each of them after 20,000, 40,000, 60,000, and 80,000 iterations (*breakpoints*) in order to have a deeper insight into the search behaviour of our heuristics. For each different objective, we report its value in the initial solution and the results obtained by the five local search algorithms. With respect to each breakpoint, the best value obtained for the objective function is shown in bold.

In order to compare the experimental results obtained by the five heuristics, with respect to both the quality of the solution and the amount of computational resources spent for getting such solutions, we provide three different types of indices which are able to measure different aspects of the performance of the algorithms.

First of all, we consider the set of test problems given by the combination of a region, an objective function and a breakpoint. Then, we rank the five algorithms from the best to the worst, according to the quality of the solution found in each of these tests and we assign a score (the so called *Borda count*) equal to 4, 3, 2, 1 and 0 to the first, the second, the

[1] Our experiments were performed on a Pentium IV – 2.00 GHz.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Table 7
Target values at iterations 20,000, 40,000, 60,000 and 80,000 for Piedmont

PIEDMONT	0 iterations	Algorithm	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations
Population Equality	0.870	Descent (3158)	0.684	0.684	0.656	0.624
		Exact Tabu Search (844)	0.761	0.749	0.749	0.749
		Random Tabu Search	**0.142**	0.107	0.084	0.077
		Old Bachelor Acceptance	**0.142**	**0.077**	**0.063**	0.062
		Simulated Annealing	0.397	0.284	0.094	**0.026**
Compactness	0.918	Descent (1285)	**0.562**	0.562	0.562	0.562
		Exact Tabu Search (804)	0.564	0.564	0.564	0.540
		Random Tabu Search	0.796	0.768	0.745	0.745
		Old Bachelor Acceptance	0.669	**0.408**	**0.372**	**0.266**
		Simulated Annealing	0.857	0.767	0.679	0.632
Administrative Conformity	0.339	Descent (778)	0.200	0.177	0.177	0.177
		Exact Tabu Search (784)	0.167	**0.118**	0.118	0.118
		Random Tabu Search	0.163	0.134	0.131	0.130
		Old Bachelor Acceptance	**0.128**	0.120	**0.102**	**0.092**
		Simulated Annealing	0.301	0.301	0.301	0.105
Target Mixture (0.5;0.3;0.2)	0.778	Descent (1237)	0.643	0.643	0.631	0.627
		Exact Tabu Search (827)	0.658	0.655	0.655	0.655
		Random Tabu Search	0.459	**0.419**	**0.403**	0.398
		Old Bachelor Acceptance	**0.456**	0.420	0.420	0.406
		Simulated Annealing	0.560	0.478	0.424	**0.373**

For Descent and Exact Tabu Search the number of repeated restarts is specified in brackets.

third, the fourth, and the fifth algorithm, respectively. On the basis of these scores, we compute a *ranking score* given by the sum of the scores over all the regions. Tables 8–11 show the ranking scores of our algorithms with respect to each objective. In the last column we report the sum of the scores over the four different breakpoints, while in Table 12 we compute the sum of all the scores obtained by an algorithm over all the possible cases. This *total ranking score* can be used for an initial gross comparison between the five algorithms.

In order to understand more deeply how the algorithms work, we suggest two additional indices of performance: the first is related to the progression of the search, while the second measures the total gain obtained by each algorithm in terms of objective function value. Denote by t the generic breakpoint, $t = 1, 2, 3, 4$, and by A a generic algorithm.

Table 8
Total ranking scores w.r.t. popoulation equality (the same score was assigned to ex aequo cases)

Algorithm	Ranking score – population equality				
	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations	Sum
Descent	6	5	6	6	23
Exact Tabu Search	1	0	1	1	3
Random Tabu Search	15	**17**	**16**	15	63
Old Bachelor Acceptance	**19**	16	**16**	15	**66**
Simulated Annealing	13	12	15	**17**	57

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Table 9
Total ranking scores w.r.t. compactness

Algorithm	Ranking score – compactness				
	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations	Sum
Descent	12	9	9	7	37
Exact Tabu Search	16	14	14	15	59
Random Tabu Search	4	0	0	1	5
Old Bachelor Acceptance	**17**	**20**	**20**	**19**	**76**
Simulated Annealing	1	7	7	8	23

Table 10
Total ranking scores w.r.t. conformity to administrative boundaries

Algorithm	Ranking score – conformity to administrative boundaries				
	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations	Sum
Descent	6	5	5	3	19
Exact Tabu Search	**18**	**18**	**18**	**17**	**71**
Random Tabu Search	11	9	9	8	37
Old Bachelor Acceptance	15	14	15	16	60
Simulated Annealing	0	4	3	6	13

Table 11
Total ranking scores w.r.t. target mixture

Algorithm	Ranking score – target mixture (0.5; 0.3; 0.2)				
	20,000 iterations	40,000 iterations	60,000 iterations	80,000 iterations	Sum
Descent	4	4	4	5	17
Exact Tabu Search	1	1	1	0	3
Random Tabu Search	15	15	**17**	**17**	**64**
Old Bachelor Acceptance	**19**	**16**	15	14	**64**
Simulated Annealing	11	14	13	14	52

Table 12
Total ranking score

Algorithm	Total ranking score
Descent	96
Exact Tabu Search	136
Random Tabu Search	169
Old Bachelor Acceptance	**266**
Simulated Annealing	145

For each $t = 2,3,4$, define the following score function:

$$s_A(t) = \begin{cases} 1 & \text{if the value of the best solution found by } A \text{ at } t \text{ is better than the one found at } t-1 \\ 0 & \text{otherwise} \end{cases}$$

Then, for each algorithm A, we can compute a *progression score* as the sum of the scores $s_A(t)$ obtained for $t = 2, 3, 4$. Notice that a high progression score means that the algorithm is able to exploit the availability of additional iterations to improve the solution, even when many iterations have already elapsed. On the contrary, low values for this index suggest the tendency to fall into local optima. Table 13 reports the average progression scores for each algorithm and each objective, computed over the five different regions. The same table shows information about the quality of the solutions found. In order to understand if an algorithm with high progression is also very effective in terms of the quality of the solution produced, we compute the *gain* which is given by the percent difference between the values of the best solutions found after 80,000 and 20,000 iterations.

Table 14 shows the average progression scores computed with respect to all the different objective functions considered.

Notice that, when an algorithm works well during the first 20,000 iterations, the corresponding improvement in the objective function is not

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Table 13
Average progression score and average percent gain from 20,000 to 80,000 (over 5 tests – regions)

Algorithm	Objective							
	Population equality		Compactness		Administrative conformity		Target mixture (0.5;0.3;0.2)	
	Progression	Gain (%)	Progression	Gain (%)	Progression	Gain (%)	Progression	Gain (%)
Descent	0.40	−2	0.80	−22	0.80	−17	1.00	−7
Exact Tabu Search	0.80	−7	0.80	−20	0.40	−18	0.80	−3
Random Tabu Search	1.80	−51	0.80	−7	2.40	−28	2.80	−18
Old Bachelor Acceptance	1.00	−27	2.40	−37	2.00	−27	1.80	−12
Simulated Annealing	2.00	−81	1.80	−23%	1.60	−55	1.80	−29

Table 14
Average progression score (over 20 tests – criterion/region)

Algorithm	Average progression score
Descent	0.75
Exact Tabu Search	0.7
Random Tabu Search	1.95
Old Bachelor Acceptance	1.8
Simulated Annealing	1.8

detected by the previous indices. Therefore, in order to measure the global quality of the solution produced by each algorithm, we need to compute the gain reached after 80,000 iterations with respect to the initial objective function value. For each algorithm, Table 15 reports the average *global gain* with respect to each objective function.

To conclude our analysis, we show some results related to the application of local search techniques to political districting with real-life data. With respect to our selected five regions, Table 16 shows the results obtained when Old Bachelor Acceptance was applied to the Italian case. For this purpose, the Italian institutional district plan was chosen as initial solution. These results are on average over 50 repeated runs. Actually, we run the algorithm repeatedly in order to evaluate the effect of randomization. In fact, even if the initial solution is the same, each time Old Bachelor Acceptance starts again, the direction of the search may change, since it depends on the first moves performed. This implies the possibility of obtaining very different final solutions. To guarantee the robustness of these results, one must ensure a good quality of the final solution not only in a single (lucky) run, but in a large percentage of cases. In this sense, the robustness of the results is guaranteed by the fact that

Table 15
Average percent global gain

Algorithm	Objective			
	Population equality	Compactness	Administrative conformity	Target mixture (0.5;0.3;0.2)
Descent	−51%	−61%	−72%	−35%
Exact Tabu Search	−33%	−74%	−86%	−28%
Random Tabu Search	−93%	−37%	−79%	−61%
Old Bachelor Acceptance	−93%	−85%	−83%	−59%
Simulated Annealing	−95%	−46%	−72%	−59%

Table 16
Old Bachelor Acceptance improvements w.r.t. the institutional district plan

Region	Objective							
	Population equality		Compactness		Administrative conformity		Target mixture (0.5;0.3;0.2)	
	Initial value	% variation	Initial value	% variation	Initial value	% variation	Initial value	% variation
Abruzzi	0.08	−82	0.63	−76	0.21	−65	0.27	−12
Latium	0.06	−86	0.68	−43	0.20	−53	0.27	−30
Marches	0.05	−94	0.67	−75	0.17	−65	0.26	−12
Trentino	0.04	−94	0.70	−52	0.07	−18	0.24	−16
Piemonte	0.10	−90	0.88	−51	0.14	−35	0.34	−7

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

all the variances associated with these repeated runs are always very small (order of 10^{-3}).

Our results show that the Italian district map can be improved with respect to different important aspects. Indeed, in our Old Bachelor Acceptance experiments we observed the simultaneous decrease of *all* three objectives. This also occurs with other local search algorithms, such as Tabu Search. This means that several maps providing better values for the three criteria taken into account can be easily built with the help of an automatic procedure.

6. Conclusions

In this study we compared the performance of five local search techniques for the political districting problem. The algorithms considered are Descent, Tabu Search (exact or random), Simulated Annealing and Old Bachelor Acceptance, Descent being considered as a benchmark for the evaluation of the performance of the others. The districting problem was formulated as a multicriteria graph partitioning problem under connectivity constraints and the application was performed over a sample of five medium-large Italian regions. The results showed a good performance for Old Bachelor Acceptance which produced the best results in the majority of the cases, especially when the objective function is compactness. We do not have, at the moment, a full explanation of the superior behaviour of such heuristic in this case. One partial explanation might be that all the three heuristics employing a random choice of the migrating node perform better than the remaining two, which make use of exact neighborhoods. Another reason might be the uncanny ability of Old Bachelor Acceptance to escape from local minima with its ''learning'' threshold adjusting procedure. On the other hand, the two implementations of Tabu Search have shown a complementary behaviour: Random Tabu Search behaves well when the objective function is population equality, while the Exact Tabu Search performs well for the other criteria. With respect to Simulated Annealing, we observed a very good performance when the objective function is population equality, while it seems that, within the observed range of iterations, Simulated Annealing is not able to reach a good local optimum for compactness. Finally, it can be noticed that, as one could expect, the Descent Algorithm has a poor performance, even if it is frequently restarted from different initial feasible solutions.

In conclusion, with respect to the political districting application, our experiments provided strong evidence in favor of the use of automatic procedures for this problem. In particular, we observed a very good performance of Old Bachelor Acceptance, the more so for larger regions, but we were able to obtain very good solutions for political districting with all the local search algorithms considered (except Descent). Moreover, by Old Bachelor Acceptance, we obtained significant improvements on institutional (manual) district plans related to real-life data. As a matter of fact, if properly designed, such procedures tend to be impartial and yield good – as measured by appropriate indicators – districting alternatives. Moreover, they are remarkably fast, and thus they allow for the exploration of a large number of scenarios. This does not mean that we advocate the use of fully automated procedures for political districting: we rather believe that in any case they need to be integrated by human judgment and common sense.

References

Arcese, F., Battista, M.G., Biasi, O., Lucertini, M., Simeone, B., 1992. Un modello multicriterio per la distrettizzazione elettorale: la metodologia C.A.P.I.R.E.A.D.E.N., unpublished manuscript, Roma.

Bodin, L.D., 1973. A district experiment with a clustering algorithm. Annals of the New York Academy of Sciences 219, 209–214.

Bourjolly, J.M., Laporte, G., Rousseau, J.M., 1981. Découpage electoral automatisé: Application a l'Ile de Montréal. INFOR 19, 113–124.

Bozkaya, B., Erkut, E., Laporte, G., 2003. A Tabu Search heuristic and adaptive memory procedure for political districting. European Journal of Operational Research 144, 12–26.

Browdy, M.H., 1990. Simulated annealing: An improved computer model for political redistricting. Yale Law & Policy Review 8 (146), 163–178.

Bussamra, N.M., Franca, P.M., Sosa, N.G., 1996. Legislative districting by heuristic methods. In: AIRO 96 Proceedings, pp. 640–641.

Cerny, V., 1985. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications 45, 41–51.

Dell' Amico, M., Maffioli, F., 1996. On some multicriteria arborescence problems: complexity and algorithms. Discrete Applied Mathematics 65 (1), 191–206.

De Simone, C., Lucertini, M., Pallottino, S., Simone, B., 1990. Fair dissection of spiders, worms and caterpillars. Networks 20 (3), 323–344.

Garfinkel, R.S., Nemhauser, G.L., 1970. Optimal political districting by implicit enumeration techniques. Management Science 16, 495–508.

Glover, F., 1989. Tabu search – Part I. ORSA Journal on Computing 1, 190–206.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065

Glover, F., 1990. Tabu search – Part II. ORSA Journal on Computing 2, 4–32.

Grilli di Cortona, P., Manzi, C., Pennisi, A., Ricca, F., Simeone, B., 1999. Evaluation and optimization of electoral systems, SIAM Monographs on Discrete Mathematics and Applications. SIAM, Society for Industrial and Applied Mathematics, Philadelphia.

Hansen, P., Jaumard, B., Meyer, C., Simeone, B., Doring, V., 2003. Maximum split clustering under connectivity constraints. Journal of Classification 20, 143–180.

Hess, W., Weaver, J.B., Siegfelatt, H.J., Whelan, J.N., Zitlau, P.A., 1965. Non partisan political redistricting by computer. Operations Research 13, 998–1006.

Hojati, M., 1996. Optimal political districting. Computers and Operations Research 23 (12), 1147–1161.

Hu, T.C., Kahng, A.B., Tsao, C.W.A., 1995. Old bachelor acceptance: A new class of non-monotone threshold accepting methods. ORSA Journal on Computing 7, 417–425.

Johnson, D.S., Papadimitriou, C.H., Yannakakis, M., 1988. How easy is local search?. Journal of Computer and System Sciences 37 79–100.

Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Lari, I., Maravalle, M., Simeone, B., 1998. A linear programming heuristic for a hard clustering problem on trees. In: Bock, H.-H., Rizzi, A., Vichi, M. (Eds.), Data Science, Classification, and Related Methods, Proceedings of IFCS 98. Springer, Rome, Heidelberg.

Liitschwager, J.M., 1973. The IOWA redistricting system. Annals of New York Academy of Sciences 219, 221–235.

Mehrotra, A., Johnson, E.L., Nemhauser, G.L., 1998. An optimization based heuristic for political districting. Management Science 44, 1100–1114.

Mills, G., 1967. The determination of local government electoral boundaries. Operational Research Quarterly 18 (3), 243–255.

Ricca, F., Simeone, B., 1997. Political Districting: Traps, Criteria, Algorithms, and Trade-offs. Ricerca Operativa 27, 81–119.

Robertson, I.M.L., 1982. The delimitation of local government electoral areas in Scotland: A semi-automated approach. Journal of Operational Research Society 33, 517–525.

Vickery, W., 1961. On the prevention of gerrymandering. Political Science Quarterly 76, 105–110.

Please cite this article in press as: Ricca, F., Simeone, B., Local search algorithms for political districting, European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.08.065