POPMUSIC
Partial Optimization Metaheuristic Under Special Intensification Conditions

Éric D. Taillard †
Stefan Voß‡

†University of Applied Sciences of Western Switzerland
ei·vd campus at Yverdon
Route de Cheseaux 1
CH-1400 Yverdon-les-Bains, Switzerland

‡Technische Universität Braunschweig
Institut für Wirtschaftswissenschaften, Informationsmanagement
Abt-Jerusalem-Straße 7,
D - 38106 Braunschweig, Germany

Abstract

This article introduces POPMUSIC, a meta-heuristic that has been successfully applied to various combinatorial optimization problems. This meta-heuristic is especially useful for designing heuristic methods for large combinatorial problems that can be partially optimized. The basic idea is to optimize sub-parts of solutions until a local optimum is reached. Implementations of the technique to large centroid clustering and to the problem of balancing mechanical parts are shown to be very efficient.

1 Introduction

When presented with large combinatorial optimization problems to be solved, a natural reflex is to decompose these large problems into independent sub-problems that are solved with an appropriate procedure. In this way, large problems can be efficiently tackled since the complexity of the global method grows slowly, typically in $O(n)$ or $O(n \cdot log(n))$, where n is the problem size. However, proceeding like this may lead to solutions of moderate quality since the sub-problems might have been created in a somewhat arbitrary fashion. Indeed, it is not easy to find an appropriate way to decompose a problem *a priori*. The basic idea of POPMUSIC is to locally optimize, sub-parts of a solution, *a posteriori*, once a solution to the problem is available. These local optimizations are repeated until a local optimum is found. So, POPMUSIC can be seen as a local search working with a special, large neighbourhood. This is why POPMUSIC has been called LOPT (standing for *Local Optimizations*) in the seminal work of [23, 24] and LNS (standing for *Large Neighbourhood Search*) in the work of [19].

In contrast to other meta-heuristics that have been inspired by a natural process (such as simulated annealing, genetic algorithms or ant systems) which metaphor is translated for designing an optimization technique, POPMUSIC has been synthesized by analyzing principles used in heuristic methods designed for various combinatorial optimization problems. Therefore, other methods than those quoted above are based on POPMUSIC principles. The aim of this paper is to introduce the basic concept of this method and to show that it may successfully be applied to some known combinatorial optimization problems outperforming previous methods from the literature. Furthermore, we show that a number of different methods, presented under various names such as LNS [19], shuffle [2], MIMAUSA [14], VNDS [11] or hybrid tabu search/branch & bound [6] share similar ideas within the same general framework. Therefore, this paper tries to extract the essence of these methods and to synthesize a number of ideas under the same framework that are very close, in a fashion similar to what was done for AMP (*Adaptive Memory Programming*) [24, 25] for meta-heuristics working with a memory.

The paper is organized as follows. In Section 2, POPMUSIC principles are presented. Then, Section 3 reviews methods that are based on these principles. Application for vehicle routing problems [22, 16, 17, 19] and clustering [23] are reviewed. Moreover, similar ideas have been already used in other applications, for example in scheduling [2] (*Shuffle* procedure), direct flight network [6] (*Hybrid tabu search/branch & bound* application). Finally, a new application to the balancing of mechanical parts is proposed in Section 4 and shown to be efficient.

2 POPMUSIC

Local search methods have been used for a long time to improve the quality of a given solution of a combinatorial optimization problem. They consist in defining a subset of *neighbour* solutions for each feasible solution of the problem under consideration. Having defined a neighbourhood, local searches attempt to find a good solution, starting from an initial solution (whose quality is not necessarily good) by moving in the solution space from one solution to a neighbour one.

Very often, local search methods use neighbourhoods whose size are polynomial in the problem size. To illustrate, let us suppose that we want to solve a clustering problem consisting in classifying a number n of elements into p classes. The elements have characteristics (e.g. co-ordinates on the Euclidean plane). These characteristics are used to define a (dis-)similarity measure between elements (e.g. the distance on the plane). The objective of a clustering problem is to find classes, or clusters that are homogeneous and well separated (e.g. classes that minimize the sum of all distances between elements belonging to the same class).

A very well known method for clustering problems is the k-means algorithm [10] that consists in moving one element from one class into another class. This neighbourhood is relatively small (in $O(np)$). However, for problem instances of large size (n and p), this neighbourhood contains a majority of solutions that are useless to enumerate. Let us illustrate this by the solution to a Euclidean problem presented in Figure 1 (in this figure, small disks are entities and large ones are centres of clusters). Considering the two classes on the top of the figure (in dark grey), it is clear that moving an element into the other class cannot improve the solution since both clusters are homogeneous and well separated.

On one hand, the k-means neighbourhood is too large. On the other hand, it is too small to generate solution of good quality [4]. The aim of POPMUSIC is precisely to suggest how to generate neighbourhoods that are better adapted to problems of large size. These neighbourhoods are large and are not explicitly enumerated. Generally, they are implicitly enumerated with the help of an optimization procedure, either an exact one or a heuristic one.

For large problems, it is often possible to consider that the solutions are composed of parts, sometimes called chunks [26]. Coming back to our clustering example, it is logical to consider a cluster as a part of a solution. Later, we are going to give other examples of part definitions for other combinatorial optimization problems.

So, let us suppose that a solution S can be represented as a set of parts $s_1, \dots, s_p$. Moreover, some parts are more in relation with some other parts. In Figure 1, the well separated clusters in dark grey at the top are very loosely connected and the clusters in medium grey around the cluster in black at the bottom left of the figure are closely related. So, let us suppose that a relatedness measure can be defined between two parts. The central idea of POPMUSIC is to select a part s_i, called *seed part* and a number $r < p$ of parts $s_{i_1}, \dots, s_{i_r}$ that are mostly related with seed part s_i to form a sub-problem called R_i. For the clustering example, the sub-problem R_i is itself a clustering problem, but smaller than the initial one.

If parts and sub-problems are defined in an appropriate way, to every improvement of the sub-problem corresponds an improvement of the solution of the whole problem. So, if all these conditions can be met for the combinatorial optimization problem considered, it is possible to state a local search optimization frame that consists in trying to improve all sub-problems that can be defined,

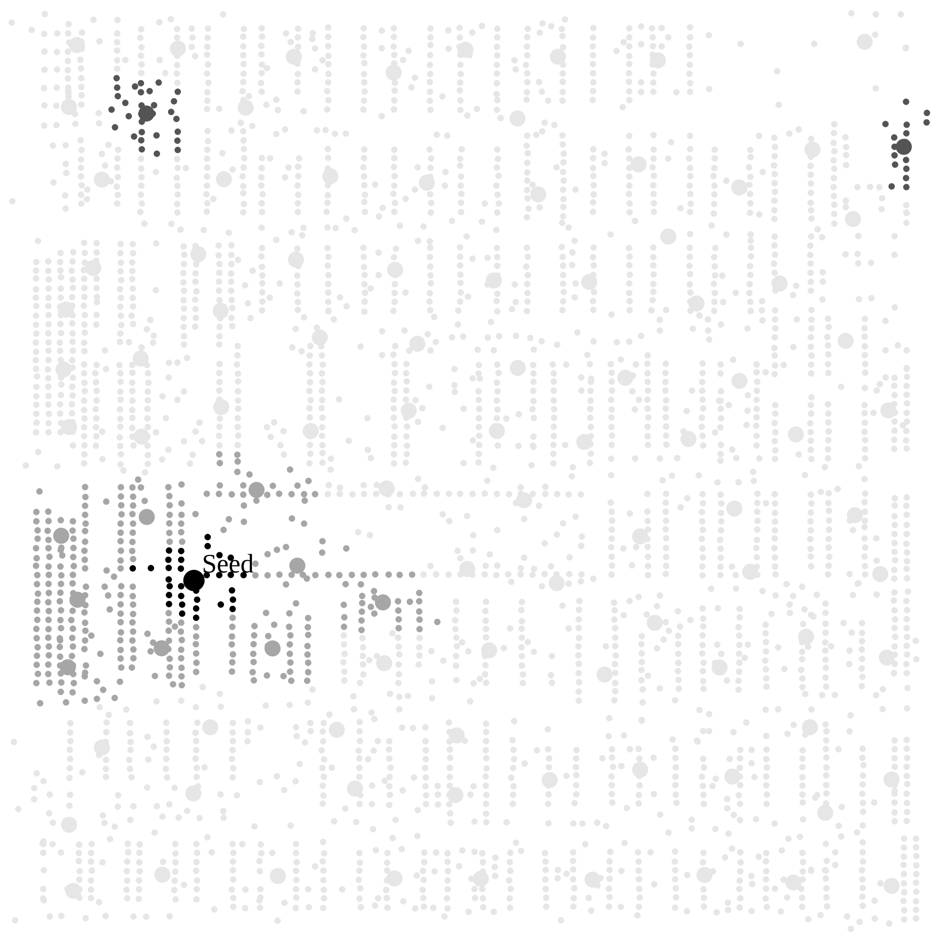

Figure 1: Example of two parts that are very loosely related (on the top, in dark grey) and a sub-problem created with 10 parts around a Seed part (in black) for the best solution known to a minimum sum-of-squares problem instance with 3038 entities and 100 clusters

until the solution does not contain a sub-problem that can be improved. In the POPMUSIC frame that follows, the set O of part corresponds precisely to seed parts that have been used to define sub-problems that have been unsuccessfully optimized. Once O contains all the parts of the complete solution, then all sub-problems have been examined without success and the process stops.

Our POPMUSIC frame has one parameter, r, that controls the size of the sub-problems to be optimized and can be sketched as follows:

POPMUSIC(r)

```
1. Input: Solution $S$ composed of parts $s_1, \dots, s_p$
2. Set $O = \emptyset$
3. While $O \neq \{s_1, \dots, s_p\}$ repeat
    (a) Select $s_i \notin O$
    (b) Create a sub-problem $R_i$ composed of the $r$ parts $s_{i_1}, \dots, s_{i_r}$ most related to $s_i$
    (c) Optimize $R_i$
    (d) If $R_i$ has been improved, update $S$ (and corresponding parts) and set $O \leftarrow \emptyset$
        Else set $O \leftarrow O \cup \{s_i\}$
```

Basically, the technique is a gradient method that starts from a given initial solution and that stops in a local optimum relative to a large neighbourhood structure. Indeed, the neighbourhood structure contains all solutions S' that differ from S only by sub-problem $R_i, i = 1, \dots, p$, i.e., the size of the neighbourhood is the number of solutions to the sub-problems. Naturally, this number may be huge and grows exponentially with parameter r (for $r = p, R_i = S$ and the technique just optimizes S directly). So, the optimization step cannot be performed by explicit enumeration of the neighbourhood (as in "classical" local search) but either by implicit enumeration methods (such as branch & bound) or by a heuristic method. The optimization procedure may also contain a number of parameters $p_1, p_2, \dots$. In this case, POPMUSIC has additional parameters, stated as POPMUSIC($r, p_1, \dots$) in the following.

Let us comment on the general POPMUSIC framework step by step. First, the technique can be seen as a "simple" local search. So, it can be embedded in a meta-heuristic that uses a local search as building block. Typical examples are AMP or MIMAUSA. In fact, the method of Rochat and Taillard [17] for vehicle routing problems is precisely an AMP that embeds an optimizer based on POPMUSIC. In MIMAUSA, the sub-problems are optimized using an exact method based on branch & bound and the value of POPMUSIC parameter r is automatically tuned in such a way to optimize the largest sub-problems that the exact method is able to solve in a reasonable computing time.

When designing a method based on POPMUSIC for a given problem, the first choice that has to be made is to define the *parts* of a solution. To illustrate what a part can be, let us take the example of the vehicle routing problem. A solution to this problem is a set of vehicle tours visiting customers. Each customer belongs to one, and only one tour. Therefore, the tours are disjoint and it is quite natural to consider a tour as a part of a solution. This approach was adopted in [22, 16] where the optimizers used are heuristic methods based on tabu search. In [19], the optimizer used is an exact algorithm based on constraint logic programming and a part of a solution can be defined as one customer.

Once the parts of a solution have been defined, POPMUSIC tries to improve the solution by locally improving a sub-problem composed of a *seed part* and few parts that mostly interact with the chosen seed part. The number of parts considered for optimizing the sub-problem so defined is given by r, the parameter of the method.

The mechanism proposed for avoiding to try to improve a sub-problem that has already been tried to be improved without success is to maintain a set O of seed parts that has defined a sub-problem already treated without success. Once all parts of a solution are contained in O, the solution is a local optimum and the process stops. In case a sub-problem R_i has been successfully improved, a number of parts from $s_{i_1}, \ldots, s_{i_r}$ have been changed and the solution is perhaps not locally optimized in the neighbourhood of the parts that have been modified. Therefore, O is emptied before continuing the process.

A meta-heuristic just provides guidelines for designing optimization methods. The art of the designer is to find how to implement the points that are not specified precisely in the meta-heuristic for the specific problem he wants to solve. The main points that are left free by POPMUSIC are the following:

1. Definition of the *parts* of a solution
2. *Selection* procedure of a part in O
3. *Relatedness* function between parts
4. Subproblem *optimizer*

In the next section, we review some methods that are using POPMUSIC principles by specifying the choices made for the four free points quoted above. This allows a very synthetic description of the methods.

3 Some methods using POPMUSIC principles

3.1 Vehicle routing

The most basic vehicle routing problem (VRP) is to find a set of vehicle tours starting from and coming back to a depot, in such a way that the tours cover all the customers once and only once, with the constraint that the sum of customers demands on a tour is not higher than the capacity of the vehicles (see [3] for more details about the VRP). There are mainly two possibilities for defining a part of a solution: In [22, 16, 17] a part is a vehicle tour and in [19] a part is a customer. So, a solution can be seen either as a set of tours or as a set of travel from one customer to another. Depending on the point of view, quite different methods based on POPMUSIC principles can be derived.

In case a vehicle tour defines a part, it is clear that POPMUSIC can be used only for problem instances involving several tours (at least 3, but typically more than 6). In the problem instances considered in [22, 16, 17], each customer has a co-ordinate on the Euclidean plane. The relatedness between two tours has been defined as follows: First, the rectangular (x, y) co-ordinates of the centre of gravity of each tour is computed. Then the rectangular co-ordinates are transformed into polar ones (ρ, ϕ), the depot being the reference point. The relatedness between two vehicle tours with centre of gravity (ρ_1, ϕ_1) and (ρ_2, ϕ_2) is then defined by the difference in their angles: $(\phi_1 - \phi_2)$ modulo 2π. A sub-problem is created by randomly selecting a seed tour and by considering then an independent VRP containing only the customers of the r closest tours of the seed tour. Since the aim of [22] was to design a parallel method, several independent sub-problems were considered at a time and optimized using a basic tabu search.

In [16], a POPMUSIC based method is embedded in a kind of variable neighbourhood search (VNS, [11]) which is itself used as the intensification mechanism of a tabu search. In this application, a part is a vehicle tour, the customers have co-ordinates on the Euclidean plane and the angle of the centre of gravity of the tours is also used for defining the relatedness between tours. First, the POPMUSIC method is called with parameter $r = 4$. Once all sub-problems have been optimized with a basic tabu search, the POPMUSIC method is called with parameter r increased by 2 units and so on till the sub-problem created contains all the customers of the original problem. At this point, a regular tabu search working on the whole problem runs for a given number of iterations. Then the VNS-POPMUSIC intensification mechanism is called again. In [17], the whole method is embedded in an adaptive memory programme and used as optimizer.

The approach of [19] is quite different from [22, 16, 17] since a part is a customer (or, more precisely, the trips arriving to and departing from a customer). The relatedness between two customers is defined in two ways, depending on the objective function to optimize. If the goal is to minimize the total distance travelled by the vehicles, then the relatedness between two customers is the inverse of the distance as defined in the original problem. If the objective is to minimize the number of vehicles used to service all customers (with total distance minimization as secondary objective), a penalty to the ordinary distance is added in case both customers belong to different tours. In order to add some randomness in the sub-problem created, the computed relatednesses are randomly perturbed.

The choice of the seed customer is always random. Once a subset of customers have been chosen, they are removed from the problem and optimally re-inserted with constraint logic programming. The process is also embedded in a VNS frame: At the beginning, parameter r is set to 1. Once a sub-problems have been unsuccessfully solved (where a is a parameter), r is incremented. Note that none of the methods discussed here uses exactly the stopping criterion described in the POPMUSIC frame. An interesting research issue would be to study methods more closely related to the frame proposed in this article and to try to use other relatedness measures for building sub-problems to optimize.

3.2 Clustering

The basic foundation of POPMUSIC was given by Taillard [23] for centroid clustering problems where the method was introduced under the acronym LOPT, standing for *Local OPTimizations.* Cluster analysis is to partition a set of entities into subsets, or clusters, such that the subsets are homogeneous and separated from one another, considering measurements describing the entities. This problem is very old and appears in a very large number of practical applications (see, e. g. [18]). A class of centroid clustering problems can be stated as follows: Given n entities e_i with weights $w_i(i = 1, \dots, n)$ it is searched p centres $c_j(j = 1, \dots, p)$ minimizing $\sum_{i=1}^{n} [\min_j w_i \cdot d(e_i, c_j)]$, where $d(e_i, c_j)$ measures the dissimilarity between e_i and c_j. Different centroid clustering problems are commonly used: If the centres can be placed only on entities, the problem is known as the p-median problem. If the entities are given by co-ordinates on the Euclidean plane, the distance measure being the Euclidean distance and the centres can be placed anywhere on the plane, this is the multi-source Weber problem. If the entities are given by co-ordinates in a Euclidean space, the distance measure being the square of the Euclidean distance and the centres can be placed anywhere in the space, this is the minimum sum-of-squares clustering problem.

For clustering problems, POPMUSIC-based methods have been designed by making the following choices: A part of a solution is a cluster of entities. For centroid clustering, the relatedness between two parts is defined as the inverse of the distance between the centre of the clusters. The selection of a seed part not in O is done at random. Finally, the optimization procedure is a descent method that uses a candidate list strategy and that stops after a given number of iterations I. This heuristic method has a parameter. Therefore, our POPMUSIC method has two parameters: r, as described in the frame, and I,

	Best	Quality [% above best]				CPU time [s. SPARC10]			
p	known	RVNS	VNDS	POP1	POP2	RVNS	VNDS	POP1	POP2
100	47685934.0	2.34	0.73	1.19	0.44	153	1132	145	505
150	30524769.8	3.13	1.44	1.16	0.58	153	1676	111	355
200	21875113.9	2.49	1.10	1.07	0.50	160	2124	96	262
250	16621446.4	2.56	1.34	1.35	0.76	182	2954	89	234
300	13289633.4	2.50	1.57	1.58	0.78	229	3151	82	205
350	11019171.4	2.60	1.36	1.69	0.76	231	3760	75	179
400	9362179.2	3.35	1.82	1.40	0.66	165	3446	72	170
450	8101618.7	3.47	1.71	1.61	0.80	242	4152	69	163
500	7102678.4	2.85	1.86	1.70	0.90	204	4060	68	156

Table 1: Comparison of RVNS, VNDS, POPMUSIC(6, 40) (POP1) and POPMUSIC(10, 100) (POP2) for minimum sum-of-squares clustering instances with 3038 entities

the parameter of the optimization procedure.

Inspired from this POPMUSIC method, designed by Taillard [23] in 1996, Hansen and Mladenović [11] have proposed variable neighbourhood decomposition search (VNDS) by varying the value of POPMUSIC parameter r. In Table 1, we compare our POPMUSIC heuristic (with two parameter sets) with the reduced variable neighbourhood search (RVNS) and variable neighbourhood decomposition search of [11]. Comparisons are restricted to minimum sum-of-squares instances with 3038 unit weight entities located on the Euclidean plane, named pcb3038 in the TSPLIB compiled by Reinelt [15]. Further computational results are provided in [23]. Various centroid clustering problems with non-unit weight instances, larger instances up to 88900 entities, are considered in this reference. The initial solution provided to our method is obtained with DEC [23], a fast heuristic that decomposes the problem into a number of sub-problems. DEC produces solutions from 5 to 8% above best ones known. The computation times provided in Table 1 take into account the time to generate the initial solution. Each problem instance was solved 10 times with our method and the values are averaged in the table.

In Table 1, it is quite clear that our POPMUSIC implementation is more efficient than RVNS and VNDS. Even for short runs with $r = 6$ and $I = 40$ the method produces solutions of similar quality to VNDS while taking on the average 34 times less computation time. By enlarging the sub-problem size ($r = 10$) and increasing the optimization effort ($I = 100$), it is possible to halve the average gap to the best solutions known. That is, for the second parameter setting (POP2) the solution quality improves considerably while having computation times that are well below those of VNDS in all cases and even below those of RVNS for $p \geq 300$. All best solution values reported in the table have been found with our POPMUSIC implementation.

4 POPMUSIC for balancing mechanical parts

4.1 Problem definition

The problem of balancing mechanical parts may be sketched as follows: n mechanical parts of weight $w_i (i = 1, \ldots, n)$ have to be fixed on a support. Each part may occupy any of n positions on the support, given by their co-ordinates $(x_i, y_i, \ldots)$ $(i = 1, \ldots, n)$, where typically the problem is considered in one, two or three dimensions. Only one part may be fixed on a given position. The goal is to position each mechanical part on the support in such a way that the centre of gravity of all the parts is as close as possible to an ideal centre of gravity, say $(0, \ldots, 0)$. Mathematically, the objective is to find $\pi \in \Pi$, the set of the permutations of n elements, that minimizes the function $f(\pi)$:

$$min_{\pi \in \Pi} f(\pi) = \frac{1}{\sum_{i=1}^{n} w_i} \cdot \left\| \left(\sum_{i=1}^{n} w_i \cdot x_{\pi_i}, \sum_{i=1}^{n} w_i \cdot y_{\pi_i}, \ldots \right) \right\|$$

Note that it is possible to cover the case that the number of positions differs from the number of parts by adding dummy parts of weight 0 (if the number of positions is greater than the actual number of parts) or dummy positions at the ideal centre of gravity (if the number of parts is greater than the actual number of positions).

The balancing problem is of major practical importance. Typical applications are the balancing of rotating parts [9, 7, 8] such as turbine rotors. A turbine rotor is composed of an axis around which a number of (almost) identical blades are fixed. Due to mechanical, thermic and chemical constraints, the blades are made in a special alloy that is difficult to manufacture. The result is that the blades have not exactly the same weight. The weight difference of the blades induces vibrations in the turbine that strongly depend on the position of the centre of mass of the blades. The more the centre of mass is away from the centre of the axis of the turbine, the higher the vibrations are. Therefore, it is very important to find an arrangement of the blades around the axis in such a way that the centre of gravity is as close to the rotation centre of the rotor as possible.

Another practical application of the problem is the balancing of the load of boats, airplanes or lorries.

In the literature [12, 20, 13], the balancing problem is commonly explored as a special case of the quadratic assignment problem, which is NP-hard. This, however, does not settle its complexity, since transforming a problem $\mathcal{P}$ into a hard one does not establish the complexity of $\mathcal{P}$.

The balancing of mechanical parts can be shown to be NP-hard by the polynomial transformation of the bi-partition problem. The latter consists in finding a partition of $2n$ elements of weight $w_i, (i = 1, \ldots, 2n)$ into two subsets S_1 and S_2 of n elements each, in such a way that $\sum_{i \in S_1} w_i = \sum_{i \in S_2} w_i$. Let us now consider a balancing problem in one dimension with $2n$ parts with the same weight as the weight of the partition problem and a support with n positions at co-ordinate (-1) and n positions at co-ordinate (1). There is a solution of cost 0 to this balancing problem if, and only if a bi-partition exists.

The approach of modeling the balancing problem as a special case of the quadratic assignment problem may have some additional shortcoming. Although a large number of efficient tabu search approaches have been developed for the problem, the computation of the objective function takes a time proportional to n^2 with the quadratic assignment formulation while it can be computed in $O(n)$ with the original formulation.

4.2 POPMUSIC adaptation

The problem of balancing mechanical parts has been successfully approached with tabu search-based methods working on the quadratic assignment formulation [20]. So, we decided to adapt a basic tabu search due to Taillard [21] to the original formulation of the problem and to use this method as the optimizer of a POPMUSIC application. The neighbourhood used in this tabu search method is to swap the positions of two parts. Naturally, swapping two parts of equal weight is forbidden since such a move does not change the position of the centre of gravity. It is quite natural to consider a part of a solution as a mechanical part. The parts are initially ranked by decreasing weights. So, let us suppose that the i^{th} heaviest part has weight w_i. With such a numbering of the parts, we define the set of the r closest parts of i as parts $i, i+1, \dots, i+r-1$ (the additions are made modulo n, so, the lightest part is close to the heaviest one). The rationale behind this definition of sub-problems is the following: The aim of the optimization of a sub-problem is to modify the position of r parts in such a way to move the centre of gravity in the direction of its ideal position. It is observed, for problem instances commonly used in the literature, that the swap of two parts often considerably modifies the centre of gravity, meaning also a considerable variation in the objective function evaluation. Since the sub-problems created in our POPMUSIC implementation are composed of parts having similar weights, it is possible to obtain small moves of the centre of gravity. Note that few sub-problems are composed of the lightest and heaviest parts. So, large moves of the centre of gravity are also possible. The seed parts are simply chosen in the order $1, 2, \dots, n, 1, \dots$.

The initial solution provided to the method is just a random one. After the solution is optimized with POPMUSIC, a short tabu search (500 iterations) working on the whole neighbourhood tries to improve it.

4.3 Numerical results

In order to show the efficiency of our POPMUSIC implementation for the balancing of mechanical part, we have considered a set of 18 problem instances of the literature with size ranging from 30 to 100 parts. These instances corresponds to the balancing of turbine rotors where the parts are blades to be regularly arranged around the rotor axis. The first set of problems, linear30 …, linear80 are artificial instances with blades of weight $1, 2, 3, \ldots, n$. The second set is composed of two real problems obtained from an airline company, named mas38 and mas40, used by Mason and Rönnqvist [13]. The last set is composed of ten 100-parts instances randomly generated with the same mean and variances of blade weights as observed in airplane turbines, also originating from [13].

Unfortunately, it is not possible to use the numerical results published in the literature for making significant comparisons with our POPMUSIC heuristic. First of all, it would be unfair to compare our implementation (with $O(n)$ objective function evaluation) with QAP-based methods using an evaluation in $O(n^2)$. In our tabu search implementation, the complexity of one iteration is $O(n^2)$ and performing I iteration takes a time in $O(I \cdot n^2)$. In contrast, the reverse elimination and star shape diversification methods implemented in [20] take a time in $O(n^3 + I^2 n^2)$ to perform I iterations.

Besides the difficulty to compare methods with different time complexity, there is another major difficulty : The landscape associated with the neighbourhood that swaps two mechanical parts is extremely rugged (see [1] for details about landscapes and ruggedness). So, a huge difference between a good and a bad run can be observed. For example, looking at 300 tabu search executions for problem instances linear30, …, linear80, the worst solution is typically 250-300% above the average one, meaning that a single bad run may influence a lot the average objective function statistic usually used in the literature. This implies that a number of numerical results published are meaningless. So, significant comparisons cannot be done.

In order to get the numerical results given in Table 2, we have proceeded as follows. First, we run a basic tabu search for 10000 iterations and looked at the computing time (expressed in seconds on Sun Sparcstation 5 in the Table) and we try to find parameters for our POPMUSIC heuristic in such a way that its computational time is similar. We found that choosing $r = 22$ parts in the sub-problems and performing about 1070 tabu search iterations for optimizing the sub-problems was convenient. Then, we run both POPMUSIC and tabu search 300 times with the chosen stopping criterion. In Table 2, we provide first the name of the instance, then the number of blades, then the absolute average solution value obtained with POPMUSIC (multiplied by 10^{10} to simplify the number listed in the table), then the average solution value of tabu search relatively to POPMUSIC ones, then the computational time of both methods and finally a confidence measure of the superiority of POPMUSIC, as explained below. We see that POPMUSIC produces better solutions than tabu search for problem instances with at least 40 mechanical parts. For linear80, the average solutions produced by tabu search are nearly 50% above those produced by POPMUSIC. For the smallest instances linear30 and mas38, POPMUSIC average solution value is slightly worse than basic tabu search ones.

In order to see if there is a statistical difference in the solutions produced by both methods, we run a Mann-Whitney test (for unparametric statistics, see, e. g [5]). The latter can be used to test if POPMUSIC has a probability higher than 1/2 to produce a solution better than tabu search. Moreover, the degree of confidence of the assumption that POPMUSIC has a probability higher than 1/2 to produce a better solution than tabu search can be evaluated (a confidence near to 0% indicates that tabu search is significantly better). Note that the distribution functions of solution values for both methods differ not only in the location of the distribution. So, the Mann-Whitney test cannot be used for comparing the average solution values. For three problem instances (linear30, mas38 and mas40) we found that 300 runs of each method were not enough for finding a significant difference between both methods (even if the relative average solution values may differ from more than 12%). So, we run both methods 10000 times for these 3 instances. Then, we can have a significant confidence degree, but unusual conclusions : For Linear30, POPMUSIC produces solutions of worse average quality than the basic tabu search, but the probability for a POPMUSIC run to produce a solution better than a tabu search run is significantly above 1/2. This just translates the fact that both distribution functions differ, but not necessarily their averages. For mas38, POPMUSIC produces solutions of worse average quality and has a probability significantly below 1/2 to produce a solution better than tabu search. For mas40, POPMUSIC is better considering both criteria.

5 Conclusions

This article presents POPMUSIC, a meta-heuristic especially designed for optimizing the solutions of large instances of combinatorial problems. The meta-heuristic suggests how to create sub-problems from a given solution in order to optimize them. In addition, POPMUSIC includes a stopping criterion similar to those used in improvement methods. In essence, POPMUSIC works as an improvement procedure defined on a very large neighbourhood. The size of this neighbourhood can be modulated through a parameter. Therefore, POPMUSIC-based methods can be easily embedded in a variable neighbourhood frame, as done for the vehicle routing problem (before the introduction of the VNS terminology).

POPMUSIC formalizes and presents under a simple frame a number of ideas “in the air” and already used in various implementations, in particular for vehicle routing problems. But other problems have been approached similarly, such

		Quality		CPU [s. SPARC5]		Confidence (%)
Name	n	POP	TS/POP	POP	TS	$P(POP < TS) > 1/2$
linear30	30	57721	0.991	3.26	4.06	98.2
linear40	40	20613	1.169	6.92	6.75	99.996
linear50	50	9336	1.300	10.21	10.26	99.999
linear60	60	5080	1.437	14.11	15.13	99.999
linear70	70	3285	1.419	18.44	20.53	99.999
linear80	80	2109	1.494	21.88	26.71	99.999
mas38	38	224	0.878	4.67	5.92	0.000
mas40	40	1773	1.037	6.24	6.30	99.999
mas100-1	100	74	1.201	36.29	43.02	99.999
mas100-2	100	88	1.234	44.08	42.85	99.999
mas100-3	100	71	1.096	51.08	42.12	97
mas100-4	100	92	1.280	39.04	42.03	99.999
mas100-5	100	77	1.239	42.86	43.01	99.999
mas100-6	100	101	1.145	35.50	42.81	99.98
mas100-7	100	95	1.300	38.47	42.74	99.999
mas100-8	100	110	1.305	38.91	42.50	99.999
mas100-9	100	103	1.324	42.78	42.16	99.999
mas100-10	100	91	1.149	37.70	43.51	98.3

Table 2: Comparison of POPMUSIC(22, 1070) and Taboo(10000) for turbine rotor balancing instances

as the job-shop scheduling or the direct flight network problem.

Two methods based on POPMUSIC are reviewed in detail: First an application to centroid clustering and second an application to the problem of balancing mechanical parts. It is shown that the POPMUSIC application to the sum-of-squares clustering problem is much more efficient than other VNS-based methods, which are themselves much more efficient than methods frequently used in statistical software.

Second, the problem of balancing mechanical parts is introduced and shown to be NP-hard. A fast tabu search has been designed for this problem. This tabu search is embedded in a POPMUSIC frame and used as sub-problem optimizer. A careful statistical analysis discloses that the POPMUSIC-based method is more efficient than the fast tabu search for large instances of this problem.

Finally, let us speculate on other combinatorial optimization problems that could be tackled with a POPMUSIC-based method. A way to approach problems on a tree (steiner tree, capacitated minimum spanning tree, ...) would be to define parts as sub-trees. Then, the relatedness between two part may depend on the distance between two nodes belonging to different parts (either the minimum direct distance between any couple of nodes or the distance along the complete tree). The Steiner nodes most related to the considered parts and that are not used in the current solution should be added for creating a sub-problem. Another approach, more atomic, would be to consider any single node as a part and the relatedness between parts could just depends on the distance between nodes.

For graph colouring problems, a part could be a colour. A more atomic approach could be to consider one element to colour as a part. In the first case, the sub-problems are independent graph colouring problems with a limited number of colours. In the second case, the sub-problems have a limited number of elements to colour that are in relation with the other elements of the complete graph. So, the sub-problems would be colouring problems with additional constraints. Both approaches may be transposed for frequence allocation problems.

For scheduling problems, a classical approach is to consider a machine as a part. Sub-problems are then scheduling problems with a limited number of machines but additional constraints to take into account the schedule of the operations on the other machines. Another possibility that could be interesting to investigate is to consider a job as a part.

Finally, POPMUSIC could also be used for on-line optimization problems where a new request have to be added to a solution computed with requests previously received. Indeed, a sub-problem can be built with the parts of the current solution that are most related to the new request. The sub-problem is then optimized on-line. Then, a POPMUSIC process can optimize the current solution off-line until the next request arrives.

References

[1] Angel E. and V. Zissimopoulos (1997), "On the landscape ruggedness of the quadratic assignment problem", presented at the *International Workshop on Combinatorics and Computer Science LIX–CNRS*, Palaiseau, France.

[2] Applegate D. and W. Cook (1991), "A computational study of the job-shop scheduling problem", *ORSA Journal on Computing 3*, 149–156.

[3] Ball M., T. Magnanti, C. Monma and G. Nemhauser (eds.) (1995), "Network Routing", *Handbooks in Operations Research and Management Science 8*, Elsevier, Amsterdam.

[4] Brimberg J., P. Hansen, N. Mladenović, É. D. Taillard (1997), "Improvements and Comparison of Heuristics for solving the Multisource Weber Problem", *Technical report IDSIA-33-97*, IDSIA, Lugano.

[5] Conover W. J. (1971), *Practical nonparametric statistics*, Wiley.

[6] Büdenbender K., T. Grünert and H.-J. Sebastian (1999) "A hybrid tabu search/branch and bound algorithm for the direct flight network design problem", *Technical Report*, Institut für Wirtschaftswissenschaften, Aachen, Germany. To appear in *Transportation Science.*

[7] Darlow M. S. (1989), *Balancing of High-Speed Machinery*, Springer, New York.

[8] Ehrich F. F. (1992), *Handbook of Rotordynamics*, McGraw-Hill, New York.

[9] Goodwin M. J. (1989), *Dynamics of Rotor-bearing Systems*, Unwin Hyman, London.

[10] Jancey R. C. (1966), "Multidimensional Group Analysis" *Australian Journal on Botany 14*, 127–130.

[11] Hansen P. and N. Mladenović (1999), "An introduction to variable neighborhood search", in S. Voss, S. Martello, I. H. Osman and C. Roucairol (eds.) *Meta-heuristics : Advances and Trends in Local Search paradigms for Optimization*, Kluwer, Boston, 422–458.

[12] Laporte G. and H. Mercure (1988), "Balancing hydraulic turbine runners : A quadratic assignment problem", *European Journal of Operational Research 35*, 378–381.

[13] Mason A. and M. Rönnqvist (1997), "Solution methods for the balancing of jet turbines", *Computers & Operations Research 24*, 153–167.

[14] Mautor T. and P. Michelon (1997), "MIMAUSA: a new hybrid method combining exact solution and local search", Extended abstracts of the 2^{nd} *International Conference on Metaheuristics*, Sophia-Antipolis, France, p. 15.

[15] Reinelt G. (1991), "TSPLIB : A Traveling Salesman Library", *ORSA Journal on Computing 3*, 376–384.
http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.

[16] Rochat Y. and F. Semet (1994), "A tabu search approach for delivering pet food and flour in Switzerland", *Journal of the Operational Research Society 45*, 1233–1246.

[17] Rochat Y. and É. D. Taillard (1995), "Probabilistic diversification and intensification in local search for vehicle routing", *Journal of Heuristics 1*, 147–167.

[18] Späth H. (1980), *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Ellis Horwood, Chichester.

[19] Shaw P. (1998), "Using constraint programming and local search methods to solve vehicle routing problems", *Technical report*, ILOG S.A., Gentilly, France.

[20] Sondergeld L. and S. Voß (1996), "A star-shaped diversification approach in tabu search", in I. H. Osman and J. P. Kelly (eds.) *Meta-heuristics : Theory & Applications*, Kluwer, Boston, 489–502.

[21] Taillard É. D. (1991), “Robust taboo search for the quadratic assignment problem”, *Parallel Computing 17*, 443–455.

[22] Taillard É. D. (1993), “Parallel iterative search methods for vehicle routing problems”, *Networks 23*, 661–673.

[23] Taillard É. D. (1996), “Heuristic methods for large centroid clustering problems”, *Technical report IDSIA-96-96*, IDSIA, Lugano, Switzerland.

[24] Taillard É. D. (1998), “Programmation à mémoire adaptative et algorithmes pseudo-gloutons: nouvelles perspectives pour les méta-heuristiques”, Habilitation thesis, University of Versailles, France.

[25] Taillard É. D., L. M. Gambardella, M. Gendreau and J.-Y. Potvin (1998), “Adaptive memory programming: A unified view of meta-heuristics”, *Tutorial and Research review booklet*, EURO XVI conference, Brussels, July 1998.

[26] Woodruff D. L. (1996), “Chunking Applied to Reactive Tabu Search”, in I. H. Osman and J. P. Kelly (eds.) *Meta-heuristics: Theory & Applications*, Kluwer, Boston, 555–569.