An Evolutionary Approach to Computer-Aided Orchestration

Grégoire Carpentier[1], Damien Tardieu[1], Gérard Assayag[1], Xavier Rodet[1], and Emmanuel Saint-James[2]

[1] IRCAM-CNRS, UMR-STMS 9912, 1 place Igor Stravinsky, F-75004 Paris, France
[2] LIP6-CNRS, 8 rue du Capitaine Scott, F-75015 Paris, France

Abstract. In this paper we introduce an hybrid evolutionary algorithm for computer-aided orchestration. Our current approach to orchestration consists in replicating a target sound with a set of instruments sound samples. We show how the orchestration problem can be viewed as a multi-objective 0/1 knapsack problem, with additional constraints and a case-specific criteria formulation. Our search method hybridizes genetic search and local search, for both of which we define ad-hoc genetic and neighborhood operators. A simple modelling of sound combinations is used to create two new mutation operators for genetic search, while a preliminary clustering procedure allows for the computation of sound mixtures neighborhoods for the local search phase. We also show in which way user interaction might be introduced in the orchestration procedure itself, and how to lead the search according to the users choices.

1 Introduction

In the last decades Computer-Aided Composition (CAC) software have provoked a growing interest among contemporary music composers and have become a core element in the development of their works. Originally, motivation for the design of such tools was to provide composers with the ability to easily manipulate musical symbolic objects, such as notes, chords, melodies, polyphonies... Simultaneously, another main branch in computer music research concentrated its efforts on sound analysis, sound synthesis, and sound processing, leading to a finer comprehension of many aspects of the wide sound phenomenon, and among them, the timbre of musical instruments.

In the meantime, contemporary composers have slowly started - since the beginning of the 1970s - to move away from purely combinatorial aspects of musical structures, and have drawn their attention to the spectral properties of sound. This turning point in western orchestral music set up a new aesthetic direction that has been carried on by later and today's composers. Simultaneously, the parallel evolution of CAC made these pioneers and their successors dream of a composition tool that could cope with rich timbre information to help them in their orchestration tasks. Unfortunately, such a tool required that techniques from various fields of music research achieved a certain degree of

maturity. Today, the tremendous knowledge inherited from the analysis of instrumental sounds, the breakthrough in timbre research, the accessibility of large sound databases and the computer performance allow for the bridging of the gap between traditional CAC systems and the current potential of sound analysis and manipulation.

In a previous paper [1] we had presented a new tool for computer-aided orchestration, with which composers can specify a target sound and replicate it with a given, pre-determined orchestra. The development of this tool was driven by the wish to consider globally the complex mechanism of timbre perception and to allow the discovery of large, non-trivial solutions. Unfortunately, the NP-hardness of the problem discouraged us to expect orchestrations involving more than two or three instruments. In the present paper we introduce an hybrid, genetic/local-search algorithm designed to face the huge combinatorial problem arising in our orchestration procedure. This algorithm is mainly inspired by Jaszkiewicz's MOGLS [2], and has been significantly adapted to our specific case. The paper is organized as follows. Section 2 reports on previous work in the field of computer-aided orchestration. Section 3 recalls the main paradigms of our system and show why the orchestration procedure can be considered as a multi-objective knapsack problem. Our orchestration algorithm itself as well as specific genetic and neighborhood operators are then presented in Sect. 4. Last, conclusions and future work are discussed in Sect. 5.

2 Previous Works in Computer Orchestration

Computer-aided orchestration is a relatively new topic of interest in the computer music domain, and the literature in this field is somehow poor. In our precedent paper we had reviewed the three previous works that aim at designing orchestration tools. We briefly recall them here, for more details see [1].

Rose and Hetrik [3] propose a Singular Value Decomposition (SVD) -based algorithm that allows either the analysis of a given orchestration or the proposition of new orchestrations that approach a target sound. Another method proposed by Psenicka [4] addresses this problem by performing the search on instruments, not directly on sounds. In a Lisp-written program called SPORCH (SPectral ORCHestration), the author uses an iterative matching on spectral peaks to find a combination of instruments that best fit the target sound. The third system is proposed by Hummel [5]. The principle is similar to Psenicka's, except that it works on spectral envelopes rather than on spectral peaks. The program first computes the target's spectral envelope, then iteratively finds the best approximation.

All these methods present the significant advantage of requiring relatively low computation times. However, as they all rely on spectrum decomposition techniques (invoking either SVD or matching-pursuit methods), they implicitly consider a sound target replication procedure as filling a LIFO stack in which "bigger" elements are introduced first. Roughly speaking, these methods can been seen as kind of greedy algorithms which are known to achieve only ap-

proximations of the best solutions in most problems. On the other hand, they fail in considering the timbre perception as a complex, multidimensional mechanism, as the optimization process is driven by a unique objective function. Our orchestration system and algorithm were designed to overtake these limitations.

3 Orchestration Viewed as a Multi-Objective Knapsack Problem

3.1 Overview of our Orchestration System

The general framework of our orchestration system is shown on Fig. 1. As presented in [1], one of the core concepts of our tool is the *target* object. This target is a set of audio and symbolic features that describe different aspect of the sound to be "reproduced" with an orchestra. These features may come either from the analysis of a pre-recorded sound, or from a compositional process.

The number of features is not fixed yet and will in all probability increase as research goes on. Currently we use little and static spectral data as audio description, and a set of pitches as symbolic features. This might seem somehow poor, however our purpose is not to build an exhaustive sound description, but rather to design a general framework which can be easily extended by adding new features when needed.

The target being defined, an orchestration engine uses an instrumental knowledge database (features database) created by the analysis and structuring of large sound sample databases, to suggest instruments notes combinations (orchestration proposals) that "realize" the target. More precisely, the procedure searches for combinations whose features best match the target's features. The orchestration proposals may afterwards be edited, transformed, or simulated.

3.2 The Multi-Objective Knapsack Approach

Let E be the set of all sounds potentially produced by any individual instrument in a given orchestra, $P(E)$ the power set of E, T a target object, and $S(T)$ the set of elements of $P(E)$ that "sound" as close to the target as possible. Starting from an initial point K_0 of $P(E)$, our goal is modify K_0's elements in order to converge into $S(T)$. In other words, the elements of K_0 may be (at will) removed, substituted, or completed by other elements, provided that the total number of elements does not exceed the orchestra's size. As stated, the problem is extremely close to the Binary Knapsack Problem (KP-0/1), well known in operational research. The KP-0/1 is usually formulated as follows:

$$(\text{KP-0/1}) \begin{cases} \max z(x) = \sum_{i=1}^{n} p_i x_i \\ \\ \text{s.t. } x_i \in \{0;1\} \\ \quad \sum_{i=1}^{n} w_i x_i \leq C \end{cases} \tag{1}$$

where n is the size of the items set, w_i is the weight of item i, p_i the profit generated by inserting item i in the knapsack, and C is the total capacity of the

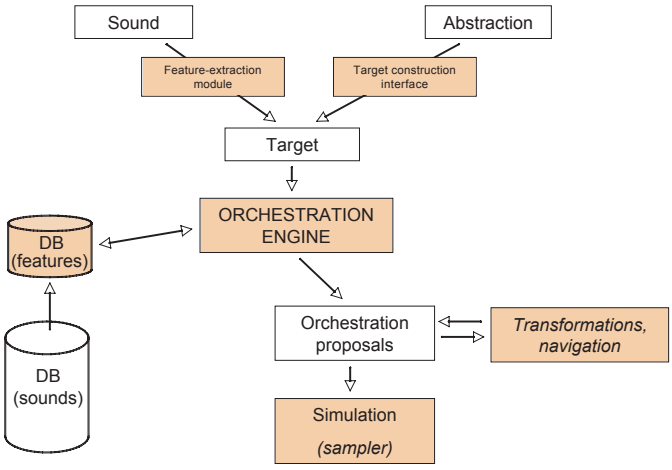

Fig. 1. General architecture of our orchestration tool.

knapsack. In our orchestration context, the items are the sounds of the database, all the weights w_i are all equal to one and the capacity is the size of the orchestra. The definition of the profits is less straightforward and will be discussed in the next section.

As previously said, the target object is a set of features, and each feature is to be seen as a specific dimension of timbre. As we aim at capturing the timbre perception mechanism globally, all dimensions have to be considered jointly in the objective function. Unfortunately we cannot predict the relative contribution of each dimension, because we do not know *a priori* which target's characteristics the composer would like to reproduce, and most of the times neither does he (she).

The multi-objective approach is therefore mandatory. In multi-objective optimization the final output is not a *unique* solution but *a set of* efficient solutions, also called *Pareto-optimal solutions*. A solution is said *Pareto-optimal* when no other solution achieves better values on *every* criteria. For more details see for instance [2]. Formally, the Multi-Objective Knapsack Problem (MOKP-0/1) is stated as follows:

$$(\text{MOKP-0/1}) \begin{cases} \max z_k(x) = \sum_{i=1}^{n} p_i^k x_i \\ k = 1, ..., K \\ \text{s.t. } x_i \in \{0; 1\} \\ \quad \sum_{i=1}^{n} w_i x_i \leq C \end{cases} \tag{2}$$

where p_i^k is the profit of item i relative to the dimension (or criterion) k.

3.3 Constraints and Limitations

The MOKP-0/1 formulated in Eq. 2 cannot be applied directly to the orchestration problem. First, it is virtually impossible to define the profits without knowing *in advance* all the elements in the combination. In other words, the profits are correlated; they are not anymore a function of a *single* index i, but of *all* indices $1, ..., n$. For instance, let x be an instrument sound sample, and K_1 and K_2 two sound mixtures defined as $K_1 = \{x\}$ and $K_2 = \{x, x\}$. Its is straigtforward that K_1 and K_2 have the same spectral features, as adding x to K_1 just increase its loudness. Figure 2 show how this problem can be overcome. Criteria are jointly computed each time a sound combination is created or changed. First, an *aggregation method* computes the combination features from the individual sounds' features. Then a set of *distance functions* compute the relative distances (along each timbre dimension) between the combination and the target. Distance are relative for homogeneity reasons. With such a formulation, the orchestration problem turns into a goal attainment problem, because we wish to minimize the distances to the target along each timbre direction, with an ideal value of zero for each criteria (when the goal is attained).

The other problem is related to orchestra's limitations. Re-using notations introduced in Sect. 3.2, a lot of elements of $P(E)$ are *not physically playable* by the orchestra, simply because a combination with two trombone sounds requires at least two trombone players, which is not necessary the case. We therefore introduce additional constraints for discarding non-feasible solutions. Let J be the number of different instruments in the orchestra and $I_j(i)$ a binary function, taking the value 1 if sound i is played by instrument j, 0 otherwise. The formulation of the Multi-Objective Orchestration Problem (MOOP) is now possible:

$$\text{(MOOP)} \begin{cases} \min z_k(x) = D_k(T, x_1, ..., x_n) & \\ k = 1, ..., K & \\ \text{s.t. } x_i \in \{0; 1\} & (a) \\ \quad \sum_{i=1}^{n} x_i \leq N & (b) \\ \quad \forall j \in \{1, J\}, \sum_{i=1}^{n} x_i I_j(i) \leq N_j & (c) \end{cases} \tag{3}$$

where N is the size of the orchestra, N_j the total number of instruments of type j, and $D_k(T, x_1, ..., x_n)$ the distance function between the target and a combination along dimension k.

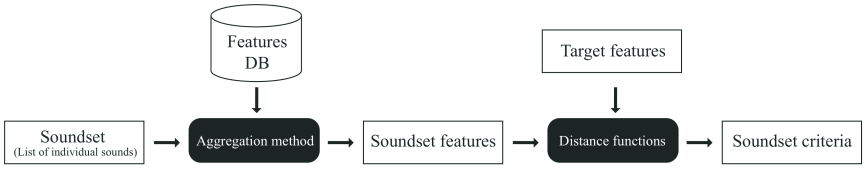

Fig. 2. Soundset criteria computation flowchart.

4 An Hybrid, MOGLS-Inspired Algorithm

The operational research literature counts many efficient exact methods (such as Branch-and-Bound or dynamic programming) for solving the knapsack problem, either uni- or bi-objective. For a complete review of these methods see [6]. However it is generally admitted that exact methods fail when the number of objectives is greater than two. Moreover, Branch-and-Bound techniques require the knowledge of items profits for the calculation of bounds, and are therfore inapplicable to the problem formulated in Eq. 3. Consequently, the use of heuristics methods is mandatory in our case.

Jaszkiewicz has proposed in [2] an efficient hybrid algorithm (called MOGLS) for multi-objective optimization, and has proved in [7] the superiority of MOGLS upon other methods for the MOKP-0/1. Basically, MOGLS is an evolutionary method that alternates genetic search and local search over the iterations, exploiting the fact that genetic and local search heuristics have different and complementary effects on populations of solutions. Here again, some preliminary concepts and operators need to be introduced before presenting our MOGLS-inspired algorithm in Sect. 4.5.

4.1 Preprocessing

Two preprocessing operations are performed before the orchestration procedure itself. First, the target is analyzed with a multi-f0 extraction method in order to discard a large set of candidates, as explained in [1]. Assume for instance the target's partial set can be explained by two pitches, C3 and Eb4. The candidate selection procedure will here keep only database sounds whose pitch belongs to C3 or Eb4 harmonic series, i.e. C3, C4, G4, C5, E5... and Eb4, Eb5, Bb5, Eb6, G6... respectively. We call "pools" these harmonic groups and "state" the harmonic rank in a given group. With such a terminology, a G4-pitched sound will belong to the C3 pool with state 3. Thanks to this concept we define ad-hoc genetic operators in section 4.3. After the multi-f0 extraction procedure a new database is created by filtering items by pitch and by instrument. The database items are sorted by increasing pitches, and within each pitch group, by increasing spectral centroid.

In a second step, the database items are clustered to form groups of sounds close to each other. The criterion used in the categorization phase is a euclidean distance on sounds contributions to the target's most important partials (for more details see [1]). In other words, this spectral distance is *target-specific*. We call "domain" of a given sound the set of all sounds belonging to the same cluster. This notion will be exploited in the local search procedure (see Sect. 4.4).

4.2 Modeling Sound Combinations

Sound combinations are modeled by a "soundset" object, containing two main slots: "elements" and "features". The elements field is made out of four vectors: the items indices in the sound database, the pools and states indices, and the

instrument group vector. This last information is used to handle orchestra's limitations (constraint (c) in Eq. 3). The pools vector is modified only when sounds are added or removed of the set (see Sect. 4.3).

Each time a sound combination is created or modified, features and criteria are computed as shown in Fig. 2. Then, a masking test procedure is invoked to "clean" the set by removing all non-perceptible components.

As suggested in [7] and [2] the combinations fitness is computed as a weighted aggregation of criteria with a weighted Tchebycheff function:

$$F(T, x_1, ..., x_n) = \max_k \lambda_k D_k(T, x_1, ..., x_n) \tag{4}$$

where $(\lambda_k)_{1 \leq k \leq K}$ are the aggregation weights, randomly drawn on each iteration.

4.3 Genetic Operators

We use the conventional binary string representation for the genetic encoding, and the crossover operator is the classic 1-point crossover. Mutation, however, is defined differently than the traditional bit-flip. In fact, we use three different kinds of mutations:

1. Horizontal shift: The sound is replaced by another one with same pool and state, but the instrument, the playing style and the dynamics may differ. The substitute sound is chosen to have a spectral centroid value as close as possible to the original.
2. Vertical Shift: The sound is replaced by another one with same instrument and pool, but the state may differ. The probability of an up shift is lower than the probability of a down shift.
3. Addition or Deletion.

An illustration of all genetic operators is show on Fig. 3.

4.4 Adaptive Search for Local Exploration

Local search is used to search for better solution within the neighborhood of sound mixtures. The method is inspired by adaptive local search introduced by Codognet and al. [8]. Here again the objective function is a weighted Tchebycheff distance with random weights. The variables are the combination's items and the domains are their associated classes as built during the clustering phase (see Sect. 4.1). As we cannot define a marginal cost function for each sound in the set, the currently selected variable is the non-taboo item with the highest loudness value.

4.5 Orchestration Algorithm

MOGLS for MOOP Our MOGLS-like orchestration method uses a population of solutions, each of them modeling a sound combination as explained in

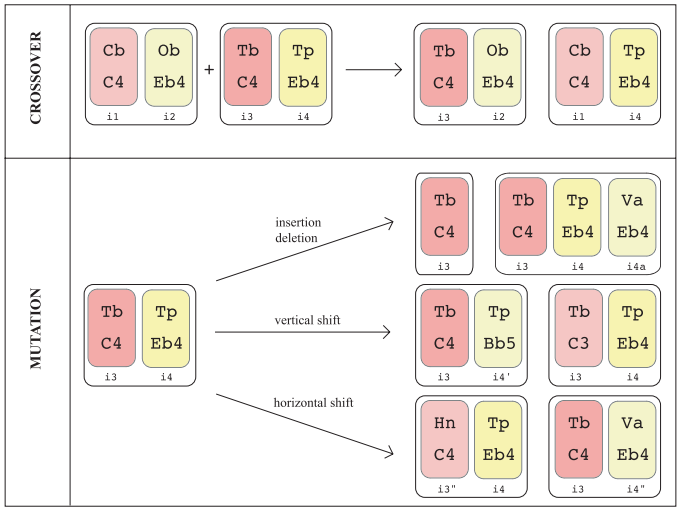

Fig. 3. Genetic operators for an orchestration algorithm.

Sect. 4.2. At each iteration, a set of weights is drawn randomly and fitness is computed. Then, individuals are chosen to fill the mating pool with a binary-tournament selection method. Genetic operators then apply and offsprings are inserted in the population if the corresponding mixtures are playable by the orchestra. Afterwards, a new set of weights is drawn and another set of N best individuals are selected on the basis of the new fitness values for local-search improvement. As recommended by Ishibushi and al. in [9], The genetic part and the local search part have been kept separated. Furthermore, the mating pool size and the number of local search iterations have been chosen to allow equal computations times to both phases, as also suggested by the authors in [9].

User Interaction As explained in [1], user interaction was thought as a fundamental paradigm in the design of our system. Interaction is introduced in the orchestration algorithm itself in the following way: When the algorithm reaches the maximum number of iterations, the user is asked to choose the solution of the Pareto set he (she) finds the closest of the target. New aggregation weights are then computed in ordered to rank the chosen solution first on the fitness scale. These weights reflect the user preferences and are calculated as in Eq. 5:

$$\lambda_k = \frac{D_k^{-1}(T, x_1, ..., x_n)}{\sum_k D_k^{-1}(T, x_1, ..., x_n)} \tag{5}$$

We then explore the direction of the search space implicitly suggested by the user's choice. The algorithm remains the same, but the weights have know fixed values, as calculated with Eq. 5. The overall orchestration procedure schema comes hereafter.

Orchestration Procedure

1. Build a target object and perform mutli-f0 analysis on target.
2. Build new sound database, by filtering on pools and instruments.
3. Sort items and compute sound clusters.
4. Build initial combinations population.
5. Until a stopping criterion is met, do:
   - Draw random weights and compute fitness with Eq. 4.
   - Fill mating pool with a binary tournament selection scheme.
   - Apply crossover and mutation operators.
   - Draw new random weights and re-compute fitness.
   - Select N best individuals and improve them with local search.
6. Ask the user to choose *one best solution* in the current Pareto set.
7. Compute the user's weights with Eq. 5.
8. Goto step 5 with the new fixed weights. Repeat procedure at will.

4.6 Results

Evaluation methods are difficult to design in our case because there is no measure of performance adapted to the orchestration problem. Traditionally performances of multi-objective optimization methods are based on the size, shape, density, or homogeneity of the Pareto set, or the distance between the theoretic Pareto set and its approximation, when the former is known. In the orchestration problem the theoretic Pareto is only known when the target is *exactly playable* by the orchestra, for instance a mixture of the instrument database sounds. In that specific case the theoretic Pareto set is reduced to the target itself. Otherwise, Pareto sets obtained by our algorithm are difficult to score. In all probability the evaluation procedure will mostly depend on composers expectations.

Early experiments with our system however gave encouraging results. Examples of pre-recorded sound target orchestrations are available on the following web page:
http://recherche.ircam.fr/equipes/analyse-synthese/dtardieu/exemple_orchestration.html

5 Conclusions and Future Work

In this paper we have exposed how an orchestration task could be viewed as a variant of the Multi-Objective Knapsack Problem (MOKP-0/1). A refinement of criteria computation and a set of extra constraints have been introduced to define the Multi-Objective Orchestration Problem (MOOP). An evolutionary algorithm inspired of Jaszkiewiecz's MOGLS have then been proposed to address the MOOP. This algorithm is a hybrid method where genetic search and local search alternate. We have also explained how user interaction could be introduced in the search procedure itself, and how the user's choice among the current solutions could help in guessing its preferences.

Future research will focus on the design of evaluation measures and procedures for our method. Alternative approaches to MOGLS for multi-objective search might then be tested. In the meantime, Gaussian Mixture Models (GMM)

-based instruments models trained on large instruments sample databases should help on one hand in increasing the power of generalization of our system, on the other hand in defining appropriate combinations neighborhoods. This should significantly ease both genetic and local search procedures.

6 Acknowledgments

Once again the authors would like to deeply thank the composers Yan Maresz and Joshua Fineberg for their involvement in the project. Their everyday support is always a great source of motivation for future development and research.

References

1. Carpentier, G., Tardieu, D., Assayag, G., Rodex, X. and Saint-James,E.: Imitative and Generative Orchestrations Using Pre-analyzed Sound Databases. Proc. of Sound and Music Computing conference, Marseille, France (2006) 115–122 http://mediatheque.ircam.fr/articles/textes/Carpentier06a/
2. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization. European Journal of Operational Research (2002)
3. Rose, F. and Hetrick, J.: Spectral Analysis as a Ressource for Contemporary Orchestration Technique. Proc. of Conference on Interdisciplinary Musicology (2005)
4. Psenicka, D.: SPORCH: An Algorithm for Orchestration Based on Spectral Analyses of Recorded Sounds. Proc. of International Computer Music Conference (2003)
5. Hummel, T.: Simulation of Human Voice Timbre by Orchestration of Acoustic Music Instruments. Proc. of International Computer Music Conference (2005)
6. Martello, S. and Toth, P.: Knapsack problems : Algorithms and computer implementations. John Wiley & Sons, Chichester (1990)
7. Jaszkiewicz, A.: Comparison of local search-based metaheuristics on the multiple objective knapsack problem. Foundations of Computing and Design Sciences 26 (2001) 99–120
8. Codognet, P., Diaz, D., and Truchet C.: The Adaptive Search Method for Constraint Solving and its Application to Musical CSPs. 1st International Workshop on Heuristics (2002)
9. Ishibuchi, H., Yoshida, T., and Murata., T.: Balance between Genetic Search and Local Search in Hybrid Evolutionary MultiCriterion Optimization Algorithms (2002)