Interval Island Model Initialization for Permutation-based Problems

Malika Mehdi
6, rue Richard
Coudenhove-Kalergi
L-1359, Luxembourg
malika.mehdi@uni.lu

Nouredine Melab
INRIA Futurs, 40 avenue
Halley,
Bt. A, Park Plaza 59650
Villeneuve d’Asq, France
Nouredine.Melab@lifl.fr

El-Ghazali Talbi
INRIA Futurs, 40 avenue
Halley,
Bt. A, Park Plaza 59650 Villeneuve d’Asq, France
El-Ghazali.Talbi@lifl.fr

Pascal Bouvry
6, rue Richard
Coudenhove-Kalergi
L-1359, Luxembourg
pascal.bouvry@uni.lu

ABSTRACT
In the absence of a priori knowledge about global optima, initial populations in genetic algorithms (GAs) should at least be diversified, especially while dealing with large spaces. On the other hand, the use of parallel models for GAs helps to solve large instances. We will focus on the island model. In this paper we propose an island initialization technique for permutation-based problems. We exploit a virtual tree organisation commonly used in exact methods (Branch and Bound) to generate a fully disjoint and well distributed (over the search space) initial population in each island. This method can be used for all permutation-based problems (QAP, Flow-shop, Q3AP...). Experiments are performed over Q3AP benchmarks using a 10 island model. The results shows the efficiency of the proposed method especially for large instances.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Heuristic Methods; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence

General Terms
Algorithms

1. INTERVAL INITIAL POPULATION FOR PERMUTATION-BASED PROBLEMS
The basics of our initialization technique are inspired from a coding approach of permutation-trees proposed in [2].

This approach was initially used in a parallel exact method, the branch and bound algorithm (B&B). The principle of this approach was to implicitly enumerate all the solutions (leaves in the tree) and to assign a unique number (identifier) to each of them.

The number associated to any node in a permutation tree is calculated following Equation (1). The concepts of path, node weight and node rank define the number associated to any node. A path of a node $n$ is the set of nodes from the root to the node $n$, including both the root and $n$. The weight $weight(n)$ of a node $n$ is the number of leaves of the sub-tree of which $n$ is root, see Equation (2). The rank is the position of a node among its sibling nodes. Then, a set of nodes always forms an interval as shown in Figure 1, an example of a codage for a one-permutation tree. Assume that $m$ is the number of permutations in the tackled problem. The global interval $S$ is represented by two numbers: $S = [0, N^{!m}]$. In our example, $S = [0, 6]$.

$$number(n) = \sum_{i \in \text{path}(n)} \text{rank}(i) \times \text{weight}(i)$$ (1)

$$weight(n) = (P - \text{depth}(n))!$$ (2)

Figure 1: Illustration of node numbers and intervals

In this paper we reuse the numbering technique described above, to generate a diversified initial population for GAs. Our initialization technique can be used for all permutation-based problems. First, we need to generalize the equations defined above for more than one permutation. Considering a $m$-permutation problem, the concept of node weight defined in Equation (2) can be generalized as shown in Equation (3).

$$weight(n) = (P - \text{depth}(n))!^m$$ (3)
In [2] the ranks of nodes are directly obtained during the generation of the children of a given node (decomposition stage in a B&B algorithm). Nevertheless, from the meta-heuristic point of view, there is no notion of ordering while moving in the search space. Thus, we need to define the rank of any solution independently from the neighbors. We just consider a problem representation based on \( m \) permutations and a virtual permutation tree for the enumeration. Assume that \( n \) is a node in this tree. Then \( n \) can be identified by its path and its depth \( p \) in the tree. Let \( N \) be the size of the used permutation. Then, the rank of \( n \) noted \( rank_p(n) \) can be defined by Equation (4).

\[
rank_p(n) = \left\{ \begin{array}{ll}
\sum_{i=1}^{M-1} Missed(\phi_i[p]) \cdot (N - p) \\
+ Missed(\phi_M[p])
\end{array} \right.
\]  

(4)

In Equation (4), \( \phi_i \) is the \( i^{th} \) permutation. \( \phi_i[p] \) is the value assigned to the position \( p \) in the permutation. The idea is to count and sum, at each position or depth \( p \), the number of non yet assigned or missed variables that are inferior to the actual value in position \( p \) in each permutation. This indicates the position of the actual node among all those from the same parent. The number of missed variables prior to \( i \) in the depth \( p \) of any permutation \( \phi \), can be calculated using Equation (5).

\[
Missed(i) = \left\{ \begin{array}{ll}
\sum_{order(j) < order(i)} X_p \\
with : X_p = 0 \text{ if } j \in \phi_p, X_p = 1 \text{ otherwise}
\end{array} \right.
\]  

(5)

Then, this coding is used to generate a diversified initial population as illustrated in Figure 2. In this example, the tree is for a one-permutation problem of size \( N = 4 \). The global interval is \([0, 24]\). Let \( POP\_SIZE \) be the population size used in a sequential genetic algorithm. The basic idea is to split the global interval in \( I = POP\_SIZE \) disjoint sub-intervals: \([0, 4], [4, 8], \ldots, [20, 24]\). The next step is to generate a random number in each sub-interval and form the population pool where each individual is represented by a number: \( \{2, 5, \ldots, 21\} \). Finally, we need a decoding operator to generate the permutation-based representation for each individual using its associated number. Since the numbers are obtained using the concepts of weight, path and rank of a node, and both the weight, the path and the number are known, we need to deduce the rank in order to complete each position in the permutation form. Indeed, each assigned variable in the permutation corresponds to one internal node in the tree representation. This node is in the path of the final solution to be generated. The rank can be calculated using Equation (6).

\[
rank(n) = number(n)/weight(n)
\]  

(6)

In the above example, we consider a single population for a sequential GA. In the parallel island model, each island is working with an independent sub-population. The proposed approach consists of splitting the whole interval into sub-intervals and applying the previous procedure to initialize the population of each island within its associated interval as shown in Figure 3.

This method makes each island searching in a different region of the search space, at least in the first generations. The complexity in the worst case of our initialization technique is \( O(m.n^2) \), \( m \) is the number of permutations used in the problem and \( n \) is the size of the problem. The complexity of the random initialization is \( O(m.n) \).

2. ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several universities as well as other funding bodies (see https://www.grid5000.fr).

3. REFERENCES

[1] P. M. Hahn, B.-J. Kim, T. Stutzle, S. Kanthak, W. L. Hightower, Z. D. H. Samra, and M. Guignard. The quadratic three-dimensional assignment problem (Q3AP) [1]. Experiments are done using 10 machines belonging to Grid’5000. The obtained results show that our interval initialization technique performs better than a random island initialization, especially for large instances.