Optimization of a Two-Dimensional Cutting Problem through Parallel Cooperation of Multi-objective Evolutionary Algorithms

Jesica de Armas, Coromoto León, Gara Miranda, and Carlos Segura

Av. Astrofísico Fco. Sánchez, s/n, 38271 La Laguna, Spain
{jdearmas,cleon,gmiranda,csegura}@ull.es
http://mereida.deioc.ull.es

Abstract. This paper presents a multi-objective implementation of the two-dimensional cutting problem. The objectives are to minimize the length of the material required and the total number of cuts needed. Only the guillotine approach is studied. In order to simplify the code development, a skeleton tool has been used. The ULL::A-Team allows the resolution of multi-objective optimization problems in sequential and in parallel through the cooperation of different evolutionary algorithms. Currently, evolutionary algorithms incorporated in the tool are SPEA, SPEA2 and NSGA-II. The genetic algorithm with better behaviour from the used to solve the specific problem in this case was the NSGA-II. The computational results, over a cluster of PCs, show the good behaviour of the approach presented here.

Key words: Cutting Problems, Evolutionary Algorithms, Parallel Multi-objective optimization

1 Problem Formulation

Characteristics of the specific problem to be resolved fit in the definition of the two-dimensional guillotine cutting problem. In this case, we want to minimize the length of the material required and the total number of cuts needed to get all parts.

To represent the gene has been used a postfix notation, so that pieces and operators are concatenated to determine the layout of such parts on the material. The pieces are identified by numbers, while operators can be H (horizontal) or V (vertical).

It has been also considered that each piece has two possible states or orientations: horizontally, denoted by '0', and vertical, denoted by '1'. This information has been added at the end of the gene and increased the number of possible solutions to the problem.

* This work has been supported by the EC (FEDER) and the Spanish Ministry of Education and Science inside the 'Plan Nacional de I+D+i' (TIN2005-08818-C04-04). The work of Gara Miranda has been developed under grant FPU-AP2004-2290.
2 Implementation using a Team Algorithm

To simplify the code development has been used a skeleton tool. The ULL: A-Team allows optimizations multi-objective problems in sequential and parallel environments through the cooperation of different evolutionary algorithms. Setting the tool it is possible to carry out executions of different models of parallel multi-objective optimization. Thus, it is possible to adapt the tool to carry out executions by islands homogeneous model, heterogeneous islands model or model of self-adaptive islands, which are granted more resources to the most promising algorithms as execution progresses. It is also possible to carry out executions combining the different previous models.

Currently evolutionary algorithms incorporated in the tool are: SPEA (Strength Pareto Evolutionary Algorithm), SPEA2 (Strength Pareto Evolutionary Algorithm 2) and NSGA-II (Nondominated Sorting Genetic Algorithm 2).

The genetic algorithm used to solve the specific problem in this case was the NSGA-II, as it is similar to that used in the article [1].

To carry out implementation using the Team Algorithm has been necessary to develop a set of methods, kept always to the data provided by the article [1]:

- **bool init (const vector &params)**: Initialization method which determines the number of parameters of the problem, the number of variables and their ranges, and objectives of the problem.
- **void evaluate (void)**: Method to evaluate the objectives of the problem. Whether to evaluate the overall length as the number of necessary cuts, we have made use of a stack and the postfix notation with which represents the gene. It should be noted at this point that if it is detected that the width of the chain of parts exceeds the width of the material, vertical operator responsible for this is replaced with a horizontal operator.
- **void mutation (double pm)**: Method that specifies the gene mutation. The technique has been followed:
  - Pick up two elements of the gene at random, p1 and p2 respectively, where p1 is nearer to the left of the gene:
    - If both are part number or operator, they are swapped.
    - If p1 is operator and p2 is a part, they are swapped.
    - If p1 is a part and p2 is operator, they are swapped only when:
      \[ 1 \leq no \leq np - 1 \]
      Where ‘no’ is the number of operators to the left plus itself and ‘np’ is the number of parts to its left. This condition must be kept after the swap.
  - Choose any operator randomly and flip it based upon the probability of mutation.
  - Choose any location of the orientation and flip its value from 0 to 1 or vice versa depending upon the probability of mutation.
- **void crossover (Individual* ind)**: Method that specifies crossover. In this case it has been used PMX (Partially Mapped Crossover), since it has been demonstrated that it is better behaved [1]. It consists of:
Optimization of 2DCP

- Choose randomly two crossing points.
- Swap these 2 segments on the children that are generated.
- The rest of the chains are obtained by mapping between 2 parents:
  * If a value is not contained in the swapped segment, remains the same.
  * If a value is contained in the swapped segment, then be replaced by the value that takes this segment in the other.
- void restart(void): Method by which an individual is generated randomly. At this point it should be borne in mind that a pattern may always be added to the gene smoothly, while an operator can only be added if it meets the condition \( 1 <= no <= np - 1 \).
- Individual* clone (void) const Method by which an individual is cloned.

3 Computational Results

The aim is to compare the results obtained using the Team Algorithm, with the results presented in the article [1].

We have tested the algorithm with sequential NSGA-II and Pareto fronts, using a smaller number of generations, have been better than Pareto fronts presented in the article to a higher number of generations.

Given the results, one would expect greater improvement if it makes use of parallelism provided by the Team Algorithm.

4 Conclusions

For this multi-objective implementation of the two-dimensional guillotine cutting problem, where the objectives are to minimize the length of the material required and the total number of cuts needed, we compare the results obtained with the results presented in the article [1]. The code developed follows the guidelines indicated in this article, which says that the optimization strategy used is similar to NSGA-II. The computational results obtained with our sequential NSGA-II implementation show better behaviour that the results presented in the article [1]. It is the reason because of hopefully greater improvement if we make use of parallelism provided by ULL::A-Team.

References