Graph Partitioning using Genetic Algorithms with ODPX

A. Cincotti, V. Cutello, M. Pavone
Dept. of Mathematics and Computer Science
University of Catania
V.le A. Doria 6,
95125 Catania, Italy
{cincotti, cutello, mpavone}@dmi.unict.it

Abstract - In this paper, we will study approximate solutions to the extension of the Maximally Balanced Connected Partition Problem, whose corresponding decision problem is known to be $\mathcal{NP}$-complete. We will introduce a genetic algorithm with a new crossover operator and we will compare the results of our algorithm to a well known deterministic approximation algorithm.

Keywords— **Approximation algorithms, Graph Partition, Genetic Algorithm.**

I. Introduction

The Maximally Balanced Connected Partition Problem is an optimization problem defined on connected graphs. The problem, whose corresponding decisional problem is known to be $\mathcal{NP}$-complete (see [1]), is defined as follows:

PROBLEM 1 (MBCP) Let $G = (V, E)$ be a connected graph. Let $w : V \to \Re^+$ be a positive weight function defined on the set of vertices. Find a partition V_1, V_2 of the set of vertices V such that

- The induced graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ are connected, and
- the value $\min\{\sum_{v\in V_1} w(v), \sum_{v\in V_2} w(v)\}$ is maximized

Basically, the problem asks for a partition of the given graph into two connected subgraphs whose total sums of weights are (optimally) equal. The MBC Problem is a particular instance of the graph partitioning problem (GPP) which can be defined as follows:

PROBLEM 2 (GPP) Let $G = (V, E)$ be a graph. Let $w_1 : V \to \Re^+$ be a positive weight function defined on the set of vertices, and $w_2 : E \to \Re^+$ a positive weight function defined on the set of edges. Find subsets $V_1, V_2, \ldots, V_k$ of the set of vertices V such that

- $\bigcup V_i = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$.
- $W(i) \equiv W/k$ where W and $W(i)$ are, respectively, the total sum of weights of the vertices in V and $W(i)$ the total sum of weights of the vertices in V_i.
- the sum of the weights of edges crossing between subsets, called *cut-size*, is minimized.

The GP problem has been extensively studied and many approximation algorithms have been produced. One of the key heuristics was proposed by [8] and many other local and global improvements methods are extension of such heuristics. The basic idea of the heuristic is quite simple: given an initial bisection, the heuristic tries to find a sequence of node pair exchanges that leads to improvement. The multilevel graph partitioning schemes described in [5], [6], [7], are based on the above idea as well. The final product is a very fast program called **METIS**, which we will use to compare the results obtained by our algorithm.

Let us, now, introduce the problem we will work on, called k-Connected Partition:

PROBLEM 3 (k-CP) Let $G = (V, E)$ be a connected graph. Let $k \geq 2$ be a given integer, find a partition $V_1, V_2, \ldots, V_k$ of the set of vertices V such that

- The induced graphs $G_i = (V_i, E_i)$, for all values $i = 1, \ldots, k$ are connected, and
- the following value is maximized

$$\min\{| V_1 |, | V_2 |, \ldots, | V_k |\}$$

II. A genetic algorithm for graph partitioning

Genetic Algorithms are a population-based optimization strategy that have been successfully applied to many real world problems [3]. The crossover operator plays a very important role in genetic algorithms since it generates the exchange of information between individuals during the search. It has been shown [2], [4], [9] that a careful design of the crossover operator is essential in avoiding loss of information. In this paper we use the k-CP Problem as an example of an application of genetic algorithms where the relative order preservation of the structure of the configuration is extremely important when the diversity of the individuals is small.

The first step of the algorithm is to create an initial population P where every element p, that we call a chromosome, is a permutation of the $|V|$ integers representing the vertices of a given graph. At this point, the problem is to associate to each chromosome a partition of the graph. We proceeds as follows.

The first k elements of the chromosome represent the leaders of the k partitions, i.e. two leaders can not be contained in the same partition. The remaining $|V|-k$ vertices that are assigned to one of the k partitions by the following deterministic algorithm:

Algorithm Create_Partitions
Input: A connected graph $G = (V, E)$, an integer $1 < k < |V| = n$, and a permutation p of V
Output: Connected and pairwise disjoint k subgraphs of G
for $i = 1$ **to** k **do**
  assign p_i to the partition i.
end_for
while there are free vertices **do**
  for $i = k + 1$ **to** n **do**
   if p_i is free **then**
   assign p_i to the smallest adjacent partition, if any;
  end_for
end_while
end Algorithm

By inspecting the above algorithm, it is clear that once the initial k vertices are fixed the partition is deterministically determined by the vertex ordering given by p. Such an algorithm is very fast and will be used by the proposed Genetic Algorithm as a subroutine to compute fitness functions of individuals. As an example, consider the graph in figure 1 and the permutation $p \equiv 7\ 6\ 3\ 10\ 5\ 1\ 11\ 9\ 4\ 12\ 8\ 2$ with $k = 3$.

Vertices 7, 6 and 3 will belong to different partitions. Applying the algorithm Create_Partitions we can see that in the first run, 10 is not assigned, 5 is assigned to 6, 1 is assigned to 3, 11 is assigned to 7, 9 to 7. 4 could be assigned to 3 and to 6 since they have both the same cardinality. To make the algorithm deterministic, 4 gets assigned to the group whose leader comes first in the permutation, in this case 6. 12 gets assigned to 7 via 11. 8 gets assigned to 3 and 2 to 3 as well. After the first round of the while loop, only 10 has not been assigned yet. In the second run, it gets assigned to 7. In fig. 2, the obtained partitions are shown.

Obtained partitions will be judged according to the

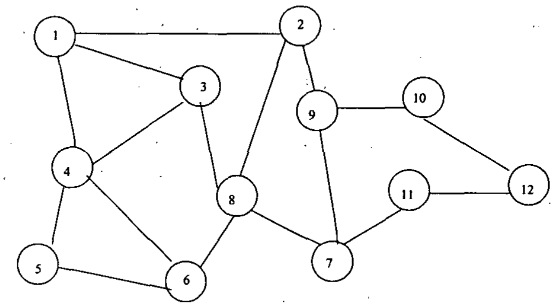

Fig. 1. A graph example

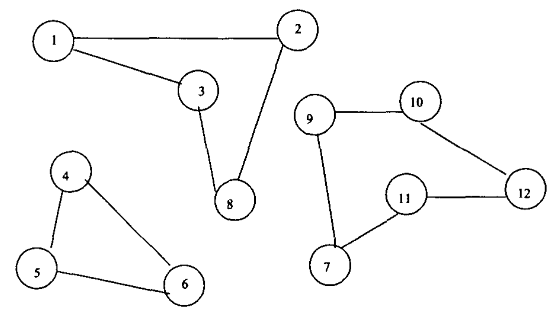

Fig. 2. The obtained partition

measure introduced in METIS:

$$\frac{k \cdot M}{|V|} \tag{1}$$

where k is the number of partitions and M is the number of vertices of the largest partition. That is to say, (1) will give us the fitness value of a chromosome (vertex permutation) p. By definition $\frac{k \cdot M}{|V|} \geq 1$ and the goal of the algorithm is to minimize such a value.

A. The crossover operator

After selection, a new crossover operator, called *order and distance preserving crossover* (ODPX), is applied. We define the distance between two chromosomes as the number of leaders that are contained in one chromosome but not in the other. Thus, two chromosomes which have the same k initial elements, even if they are in different order, have distance zero. Two chromosomes instead, that have no elements in common among the first k, have distance k.

The aim of the ODPX is to produce an offspring which has the same distance to each of its parents as one parent

0-7803-7282-4/02/$10.00 ©2002 IEEE

to the other. Now, in the first chromosome we swap all the leaders that are not in common with the second chromosome with some of $|V| - k$ elements chosen in a random way. Afterwards, we repeat the same operation with the second chromosome. Finally, we must exchange the information about the $|V| - k$ elements of the two chromosome but preserving the order. Suppose that

$$C_{k+1}C_{k+2}\dots C_{|V|}$$
$$D_{k+1}D_{k+2}\dots D_{|V|}$$

are the second parts of the two chromosomes. We consider the list

$$C_{k+1}D_{k+1}C_{k+2}D_{k+2}\dots C_{|V|}D_{|V|}$$

and use it to create two new chromosomes as follows: we start from the first element and, if the element does not belong to the new first chromosome, we put in there, otherwise we put it in the second chromosome. Analogously, we can consider the list

$$D_{k+1}C_{k+1}D_{k+2}C_{k+2}\dots D_{|V|}C_{|V|}$$

and generate two other offspring.

Procedure ODPX
Input: Vectors p_a, p_b which are permutations of the set of vertices V.
Output: Vectors q_1, q_2, q_3, q_4 which are permutations of the set of vertices V.

compute the intersection I of the first k elements of p_a and p_b
for $i = 1$ **to** k **do**
  if $p_a[i]$ is not in I **then**
   choose randomly $j > k$
   swap($p_a[i]$, $p_a[j]$)
end for
for $i = 1$ **to** k **do**
  if $p_b[i]$ is not in I **then**
   choose randomly $h > k$ such that $p_b[h]$ is not in J
    Comment: J is the set of the first k elements of p_a;
   swap($p_b[i]$, $p_b[h]$)
end for
copy the first k elements of p_a in q_1 and q_3
copy the first k elements of p_b in q_2 and q_4
Comments: we create now two vectors L, L' with $2(n-k)$ elements each, as follows:
for $j = 1$ **to** $2(n-k)$ **do**
  if (j *mod* $2 = 1$) **then**
   $L[j] = p_a[k + (j+1)/2]$
   $L'[j] = p_b[k + (j+1)/2]$
  else
   $L[j] = p_b[k + j/2]$
   $L'[j] = p_a[k + j/2]$
end for
for $j = 1$ **to** $2(n-k)$ **do**
  if $L[j]$ is not in $q1$ then copy $L[j]$ in q_1
  else copy $L[j]$ in q_2
  if $L'[j]$ is not in $q4$ then copy $L'[j]$ in q_4
  else copy $L'[j]$ in q_3
end for
end Procedure

A simple example follows. Let us consider a graph with $|V| = 12$ vertices and suppose $k = 3$. Suppose we want to apply ODPX to the following two chromosomes:

5 3 7 | 4 11 8 2 12 1 9 6 10
7 6 3 | 10 5 1 11 9 4 12 8 2

The chromosomes have distance one because they have two leaders, namely 3 and 7, in common. We swap the remaining leader with one randomly chosen. Suppose, for instance, that 2 is chosen for the first chromosome and the 4 for the second. We obtain:

2 3 7 | 4 11 8 5 12 1 9 6 10
7 4 3 | 10 5 1 11 9 6 12 8 2

Now, we consider the list

4 10 11 5 8 1 5 11 12 9 1 6 9 12 6 8 10 2

we reset the chromosomes

2 3 7 | 0 0 0 0 0 0 0 0 0
7 4 3 | 0 0 0 0 0 0 0 0 0

and create the offspring

2 3 7 | 4 10 11 5 8 1 12 9 6
7 4 3 | 5 11 1 9 12 6 8 10 2

In a symmetric way, we consider the list

10 4 5 11 1 8 11 5 9 12 6 1 12 9 8 6 2 10

and generate the other two offspring

2 3 7 | 4 11 5 1 12 9 8 6 10
7 4 3 | 10 5 11 1 8 9 12 6 2

B. The algorithm

We present here for completeness the pseudo-code of the algorithm.

```
The GA for graph partitioning
Input: A connected graph G = (V, E) with | V |= n and
an integer 1 < k < n/4
Output: Connected k subgraphs of G whose total cardi-
nality is "close" to the average value.
    initialize randomly a population P of 2×n elements.
    for i = 0 to MAX_GEN do
     compute_fitness
     sort the population with respect to fitness values
     delete half of population with lower fitness
     crossover
    end_for
end Algorithm
```

The procedure compute_fitness formalizes how the fitness value of each chromosome is computed.

k	50	100	200	400	600	800	1000
PMETIS	1.01	1.02	1.08	1.18	1.34	1.40	1.44
ODPX	1.00	1.00	1.02	1.04	1.08	1.12	1.20

TABLE I

```
Procedure compute_fitness
for each p ∈ P do
    call Create_Partitions(p);
    fitness(p) = k·M(p)/n;
    Comment: M(p) is the partition of maximal cardi-
    nality among the k partitions created given the per-
    mutation p;
end for
end Procedure
```

Below we have pseudo-code for the crossover function.

```
Procedure crossover
for i = 1 to n/2 do
    select two parents p_a, p_b ∈ P randomly;
    add the 4 individuals producted by ODPX(p_a, p_b)
    to P
end_for
end Procedure
```

III. Results and Future work

Table I shows the results of our GA compared to an iterative version of METIS. The shown results refer to average results on randomly generated graphs with 5000 vertices and 100000 edges. For each value of k 5 different experiments were conducted.

The results prove that our approach produces better results than one of the best existing algorithms although it is slower. We intend now to apply this idea to the MBC Problem. In particular, we are also trying to explore some specific heuristics to apply to planar graphs, which represent an interesting instance of the MBC Problem, which is somewhat harder due to the implicit *sparseness* of planar graphs.

Acknowledgments: We thank the anonymous referees for the very valuable comments.

References

[1] J. Chlebikova. Approximating the Maximally Balanced Connected Partition Problem in graphs. *Information Processing Letters*, 60:225-230, 1996.

[2] B. Freisleben and P. Merz A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman

Problems *In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, (Nagoya, Japan), pp. 616-621, 1996.

[3] D.E. Goldberg Genetic algorithms in search, optimization and machine learning *Addison-Wesley*, MA, 1989

[4] M. Gorges-Schleuter ASPARAGOS96 and the Traveling Salesman Problem *In Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Press (1997)

[5] G. Karypis and V. Kumar. Multilevel Algorithms for Multi-Constraint Graph Partitioning. *In Proceedings of Supercomputing*, 1998.

[6] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, pages 269–278, 1996.

[7] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1), 1998.

[8] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49:291-307, 1970.

[9] P. Moscato On genetic crossover operators for relative order preservation *Caltech concurrent computation program*, C3P Report 778 (1989)