Les Cahiers du GERAD ISSN: 0711–2440

An Algorithm for Multiobjective Optimization in Graph Theory

G. Caporossi

G–2013–13

February 2013

An Algorithm for Multiobjective Optimization in Graph Theory

Gilles Caporossi

GERAD & HEC Montréal
Montréal (Québec) Canada, H3T 2A7

gilles.caporossi@hec.a

February 2013

Les Cahiers du GERAD
G–2013–13

Abstract: In this paper, we propose an algorithm to solve multi objective optimization problem where the objects under study are graphs. The proposed algorithm is designed to handle the problem of finding extremal graphs when more than one graph invariant is considered, but also for finding graphs which have vertices that are Pareto optimal solutions for more than one vertex related value.

Résumé : Dans cet article, nous proposons un algorithme pour résoudre des problèmes d'optimisation multi objectifs dans lesquels les objets étudiés sont des graphes. L'algorithme proposé est conçu pour résoudre des problèmes dans lesquels on recherche à identifier des graphes extrêmes quand plus d'un invariant est considéré, mais aussi des graphes qui ont des sommets qui sont des solutions Pareto optimales pour plus d'une valeur associée à des sommets.

Acknowledgments: This work was supported by NSERC (Canada).

Indroduction

A graph invariant is defined to be a property preserved under all possible isomorphisms of a graph. Some of the most common graph invariants are the number of vertices, the number of edges, diameter, chromatic number, independence number, etc.

Graph invariants provide a powerful analytical tool for investigation of structures of graphs. They, combined in convenient algebraic relations, contain global and general information about a graph and its particular substructures such as cycle structures, factors, matchings, colorings, coverings, etc. The discovery of these relations is the primary problem of graph theory.

So, the way to study some particular graph invariant from the mathematical point of view is to describe it or bound it by a function of other invariants. Automatic search for those relations can be achieved by parametric optimization in which this particular invariant is bounded by a constraint while the other ones are optimized.

Depending on the application, in some situations a global description of a graph is not needed nor useful. For example, in the study of social networks one could focus only on vertices and characterize them with quantities. Among those vertex quantities, the most common are the degree $d(v)$ of the vertex v or the transmission t_v which is defined to be the sum of geodesic (the shortest) distances from v to all other vertices of G. As it is the case for invariants, vertex quantities are often better described using functions of other quantities. In this case the search for extremal values could not be achieved by parametrization since any constraint on a value will apply to all vertices of G. In order to efficiently find conjectures on vertex quantities, one needs a special optimization routine that handles more than one single vertex quantity at the same time.

In this paper we present how the optimization module of AGX could be modified and adapted in order to handle multi objective problems when invariants and vertex related values are considered. The first section provides some background information, the second describes the optimization algorithm based upon the Variable Neighborhood Search (VNS) [6, 5], used in AGX, the third exposes some possible ways to handle multi objective optimization with invariants, the fourth stresses the particular difficulties that occur when vertex related values are involved. The chosen algorithm (that was implemented in AGX) is presented in the fifth section, a comparison analysis is achieved in Section 6 and the last section concludes.

1 Preliminaries

Let $G = (V, E)$ be a connected undirected graph without loops or multiple edges. Note $\mathcal{G}^n$ the set of all graphs on $|V| = n$ vertices, and let $i : \mathcal{G}^n \to \mathbb{R}$ be a graph invariant.

The problem of finding extremal graphs could be described as follows, without lose of generality:

$$\begin{aligned} \max \quad & i(G) && (1)\\ subject\ to & \\ i_k(G) \quad & \leq \quad 0 && (2)\\ i_l(G) \quad & = \quad 0 && (3)\\ G \quad & \in \quad \mathcal{G}^n. && (4) \end{aligned}$$

Where $i_k(G)$ and $i_l(G)$ are sets of invariants corresponding to sets of possible additional constraints. Note $\mathcal{G}^n_r$ the set of all the realizable solutions of the problem (1)-(4), i.e. graphs $G \in \mathcal{G}^n$ that respect these constraints. An alternative definition of the problem is:

$$\begin{aligned} \max \quad & i(G) \\ subject\ to & \\ G \quad & \in \quad \mathcal{G}^n_r. \end{aligned}$$

In the case of multi objective optimization, the objective function is $I(G) \in \mathbb{R}^q$, a vector whose components are graph invariants $i_j(G)$, $j = 1, \ldots, q$. Let us now consider, again without lose of generality, the multi

objective optimization problem that consists in maximizing all the components of $I(G)$. This problem could be described as:

$$\max \quad I(G) \tag{5}$$
$$subject \quad to$$
$$G \in \mathcal{G}_r^n. \tag{6}$$

Consider two graphs $G_1, G_2 \in \mathcal{G}_r^n$ and the vector $I = I(G_1) - I(G_2)$.

- If $I \in \mathbb{R}^{q+}$ and $I \neq 0$ (I has some non 0 components), then G_1 dominates G_2, and we write $I(G_1) > I(G_2)$.
- If I has both strictly positive and strictly negative components, or $I = 0$ (all components of I are 0), then G_1 and G_2, are not comparable, and we write $G_1 \sim G_2$.

The set $\mathcal{G}^* \subset \mathcal{G}_r^n$ of Pareto optimal solutions for the problem (5)–(6) is such that:

1. $\forall G_2 \in \mathcal{G}_r^n \backslash \mathcal{G}^* \; \exists G_1 \in \mathcal{G}^*$ so that $G_1 > G_2$ and
2. $\forall G_1, G_2 \in \mathcal{G}^*$, $G_1 \sim G_2$.

2 The variable neighborhood search algorithm in AGX

The basic principle of the variable neighborhood search is to successively apply local searches and perturbations of various magnitude [6, 5]. In AGX, both the local search and the perturbations involve transformations of the graph.

Let the neighborhood $t(G)$ of G be the set of graphs that could be obtained from G by applying the transformation t. Examples of transformations could be to add an edge, to remove an edge, to apply a 2-opt, etc. The transformations used in AGX are mainly based upon the replacement of a subgraph on 4 vertices by another one. Indeed, $t(G)$ is too large to be completely explored. Therefore, some heuristics are used to consider the most promising graphs $G' \in t(G)$.

The choice of the graphs to be considered in $t(G)$ is achieved by a learning algorithm which uses probabilities that are updated during the optimization process. The basics of this learning algorithm are provided in [1], but a much improved version is described in [4].

The VNS algorithm involves a local search described by the Algorithm 1, and a perturbation described in the Algorithm 2.

Algorithm 1 Local Search in AGX

```
1: function LS(G)
2:     Let G be an initial graph.
3:     Note i(G) the value associated to G                ▷ To be maximized
4:     repeat
5:         Let G' ∈ t(G) ∩ G_r^n so that i(G') > i(G)
6:         if G' exists then,
7:             G ← G'.
8:     until G' doesn't exist.                            ▷ G is a local optimum according to t
9: return G
```

The Variable Neighborhood Search implementation used in AGX is described by the Algorithm 3, where the stopping criterion could either be the maximum number of evaluations of the objective function or the maximum time.

Algorithm 2 Perturbation in AGX

```
1: function PERTURB(G,k)
2:     for step = 1 → k do
3:         Randomly choose G' ∈ t(G).
4:         G ← G'
5: return G
```

Algorithm 3 Variable Neighborhood Search in AGX

```
1: Define k_max, the maximum perturbation magnitude.
2: Let G be an initial graph                          ▷ Usually a random graph
3: Set G* ← G the best known solution
4: Set k ← 1
5: repeat
6:     G' ← PERTURB(G, k)
7:     G'' ← LS(G')
8:     if i(G'') > i(G*) then
9:         Set G* ← G''
10:        Set k = 1
11:    else
12:        Set k = k + 1 mod(k_max)
13: until the stopping criterion is reached
```

3 Multi objective optimization using graph invariants

In the case of multi objective optimization, the problem is not to find an optimal solution, but a set of Pareto optimal solutions $\mathcal{G}^*$. Various approaches may be considered for this problem. The first one, that was generally used because it does not require any special implementation as long as the software may handle constraints, consists in optimizing an objective function, while the others are expressed as constraints in which they are bounded by parameters.

The problem (5)–(6) becomes:

$$\begin{aligned} \max \quad & i_p(G) && (7)\\ subject \quad to & \\ i_s \quad & \geq \quad a_s \forall s = 1 \dots q, s \neq p && (8)\\ G \quad & \in \quad \mathcal{G}_r^n, && (9) \end{aligned}$$

where a_s is a bound for the s^{th} objective, used as a parameter.

Another approach to handle the multi objective optimization issue is to adapt the VNS algorithm, as well as the local search. In the algorithm 1, line 3 must be replaced by "Note $i(G)$ the value associated to G" and line 5 by "Let $G' \in t(G) \cap \mathcal{G}_r^n$ so that $I(G') > I(G)$ ". The Algorithm 3 should be replaced by the Algorithm 4.

4 Multiobjective optimization for vertex related values

Beyond invariants, which consists in a single value for a graph, the problem of multi objective optimization could be extended to values computed for each vertex. In terms of optimization, the solution of such a problem is a set of vertices associated to their corresponding graph.

When working on individual vertices, the objective function, i.e. the vector $I(G)$ is no more appropriate, but should be replaced by a set of vectors, $I(v)$, $\forall v \in G$.

The approach that consists in solving the multi objective optimization problem by a parametric optimization of a single objective function cannot be applied anymore.

Algorithm 4 VNS for multi objective optimization in AGX

1: Define k_{max} the maximum perturbation magnitude
2: Let G be an initial graph ▷ Usually a random graph
3: Set $\mathcal{G}* = \{G\}$ the set of best known solutions
4: Set $k \leftarrow 1$
5: **repeat**
6: Randomly chose $G \in \mathcal{G}^*$
7: $G' \leftarrow PERTURB(G, k)$
8: $G'' \leftarrow LS(G')$
9: **if** G'' is not dominated by any graph $G \in \mathcal{G}^*$ **then**
10: Set $\mathcal{G}^* \leftarrow \{G''\} \cup \mathcal{G}^*$
11: Remove all dominated solutions from $\mathcal{G}^*$
12: Set $k = 1$
13: **else**
14: Set $k = k + 1 mod(k_{max})$
15: **until** the stopping criterion is reached

Indeed, it is possible that some vertices are not considered because they belong to the same graph as vertices that don't respect some of the constraints added for parametric optimization.

The optimization would then be achieved over a subset of the realizable solutions, which is clearly suboptimal. Therefore, it is mandatory to develop a proper strategy in order to solve this problem. An approach to handle this problem is to modify the definition of domination (used in the Algorithm 1, line 5) and eventually adapt the local search algorithm.

Definition 1 *Domination in the strong sense: We say that the graph G_2 dominates the graph G_1 if $\exists v \in G_2$ so that $I(v) > I(u) \forall u \in G_1$.*

This definition is very restrictive. In the context of a local search, it leads to poor results because it is very easy to find a graph G which is not dominated by any $G' \in t(G)$. In that case the local search stops even if G is not a good solution.

Definition 2 *Domination in the weak sense: We say that G_2 dominates G_1 if $\exists v \in G_2$ such that $\nexists u \in G_1$ so that $I(u) > I(v)$.*

According to this definition, graph G_2 dominates graph G_1 if there is a vertex in G_2 that is not dominated by any vertex of G_1. However, it is possible that a vertex v of G_2 is not dominated by any vertex of G_1 and there also exists a vertex u of G_1 that is not dominated by any vertex of G_2. In such case u and v are not comparable. Implementing this definition in the Algorithm 1 would cause it to cycle.

It seems impossible to extend or modify the definition of domination of a graph by another and get an efficient local search. If the values of objective function used in the optimization refer to vertices, then the individual vertices associated to the corresponding graph must be considered. Therefore, we cannot conclude that a graph dominates another, but that it dominates a vertex of a graph. The corresponding definition could be as follows:

Definition 3 *A graph G_2 dominates the pair (G_1, u) if there $\exists v \in G_2$ so that $I(v) > I(u)$.*

By using this definition, G_2 will implicitly be associated to the vertex v, thus forming the pair (G_2, v). So, if we use this definition instead of Definition 2, we solve the cycling problem and would theoretically produce the appropriate results.

5 The new multi objective optimization algorithm in AGX

None of the definitions proposed in the Section 4 is completely appropriate, the two first definition would lead to inefficient algorithms from a theoretical point of view. Indeed, the implementation of the algorithm using the Definition 3, if it is technically adapted for the optimization in the case of vertex related values, but it involves the implementation of special data structures to identify the vertex associated to a given solution. Furthermore, the use of such a structure allows only a single supposed Pareto optimal vertex after each local search, which implies a lot of local searches in order to identify a complete set of Pareto optimal solution.

Using a different strategy, an efficient algorithm would not only involve a single graph, but a set of graphs, each of which having at least one vertex that is not dominated by any vertex found during that local search.

As the algorithm is based upon vertices and involved a set of graphs, to simplify the notation, we will write $v \in \mathcal{G}$ to identify a vertex $v \in G$ of a graph $G \in \mathcal{G}$.

The resulting algorithms are Algorithm 5 and Algorithm 6.

Algorithm 5 New Local Search in AGX

function LSPARETO(G)
  Let G be an initial graph.
  Let $\mathcal{G} \leftarrow \{G\}$ be the current set of solutions
  Note $I(v)$ the value associated to a vertex $v \in G$ ▷ To be maximized
  repeat
    for $G \in \mathcal{G}$ **do** ▷ Test all graphs
      Let $G' \in t(G) \cap \mathcal{G}_r^n$ so that $\exists v \in G', I(v) > I(u) \forall u \in \mathcal{G}$
      if G' exists **then**
        $\mathcal{G} \leftarrow \mathcal{G} \cup \{G'\}$
        remove of $\mathcal{G}$ all graphs whose vertices are all dominated
  until G' doesn't exist. ▷ $\mathcal{G}$ is a set of local optimum according to t
return $\mathcal{G}$

The Algorithm 4 must be slightly adapted, and described as in Algorithm 6.

Algorithm 6 New VNS algorithm in AGX

Define k_{max} to be the maximum perturbation magnitude
Let G be an initial graph ▷ Usually a random graph
Set $\mathcal{G}* = \{G\}$ the set of best known solutions
Set $k \leftarrow 1$
repeat
  Randomly chose $G \in \mathcal{G}^*$
  $G' \leftarrow PERTURB(G, k)$
  $\mathcal{G} \leftarrow LSpareto(G')$
  if $\exists v \in \mathcal{G}, \nexists u \in \mathcal{G}^*$ so that $I(u) > I(v)$ **then**
    Set $\mathcal{G}^* \leftarrow \mathcal{G} \cup \mathcal{G}^*$
    remove of $\mathcal{G}^*$ all graphs whose vertices are all dominated
    Set $k = 1$
  else
    Set $k = k + 1 mod(k_{max})$
until the stopping criterion is reached

6 Numerical results

The only suitable algorithms for vertex related values are the Algorithm 4, in which the Definition 3 is implemented, and the Algorithm 6.

The steps in the Algorithm 4 implementing the Definition 3 for domination are the same as those of the Algorithm 4 except that the comparison is based upon vertex values (which implies a linear time additional computation time, a constant time for each vertex for each objective). This additional computation time is not significant compared to the computation of the objective functions.

6.1 Description of the test problems

To compare Algorithms 4 and 6, the following invariants were considered:

- $J(G)$: the *Balaban index* [2] is defined as follows:

$$J(G) = \frac{m}{m-n+2} \sum_{(u,v)\in E} \frac{1}{\sqrt{t_u \times t_v}},$$

  where t_u, the transmission of the vertex u is the sum of geodesic distances between u and the other vertices of the graph.
- $Kf(G)$: the *Kirchhoff index* (or resistance) is defined as follows:

$$Kf(G) = n \sum_{k=1}^{n-1} \frac{1}{\mu_k},$$

  where μ_k is the k^{th} eigenvalue of the laplacian matrix of G.
- $E(G)$: the energy is defined as:

$$E(G) = \sum_{i=1}^{n} |\lambda_i(G)|,$$

  where, $\lambda_i(G)$ is the i^{th} eigenvalue of the adjacency matrix of G.
- $\lambda_1(G)$: the spectral radius is the largest eigenvalue of the adjacency matrix of G.

The problems used to compare the algorithms were built by considering all pairs of invariants, each of them being maximized.

To evaluate the performance of Algorithms 4 and 6, they were both programmed in AGX. To avoid artifacts related to the unsure measure of the CPU time on modern computers (specially when multi core computers are used), it was decided that the stopping criterion would be the number of evaluations of the objective function (100 000 evaluations), and 100 runs were performed on each problem and the tests were run on graphs with 10 vertices. The test problems used involved the maximization of two invariants which always has non negative values, so that the dominated surface (1) could be used as a criterion.

The Table 1 describes the results obtained. The worst solution obtained with the Algorithm 6 is quite systematically better than the best solution from the Algorithm 4, the two exceptions being $J(G)$ *vs* $\lambda_1(G)$, which is not really a multi objective problem as it yields only one optimal solution, and the problem $J(G)$ *vs* $E(G)$ which has 20 Pareto optimal solutions. Comparing the Pareto fronts obtained, shows that a wide portion of the front is completely missed by the Algorithm 4. An explanation could be that it fails to diversify its search, while this task is made easier by the storage of a variety of solutions in the Algorithm 6.

It is also clear that the Algorithm 6 performs much better than the Algorithm 4 when the number of Pareto optimal solutions increases. This result was expected as the Algorithm 4 cannot generate more than one solution per local search, which is not the case for the Algorithm 6 that may generate a large number of them. The computational effort required for each Pareto optimal solution is thus much smaller when the Algorithm 6 is applied.

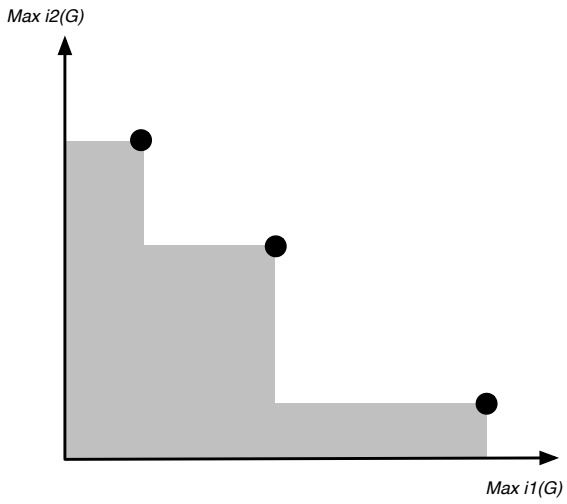

Figure 1: Representation of the dominated surface with 3 solutions when both objective functions are maximized.

Table 1: Performance comparison between the dominated surface after Algorithm 4 and Algorithm 6 were performed for each pair of invariants on 100 runs (*average*, minimum, maximum).

	$J(G)$ vs $Kf(G)$	$J(G)$ vs $E(G)$	$J(G)$ vs $\lambda_1(G)$	$Kf(G)$ vs $E(G)$	$Kf(G)$ vs $\lambda_1(G)$	$E(G)$ vs $\lambda_1(G)$
Algorithm 4	*105.52*	*80.1348*	*54.7297*	*489.368*	*191.199*	*136.434*
	81.84	50.6016	54.7297	358.47	127.311	75.3986
	149.55	90.7557	54.7297	979.147	281.961	147.468
Algorithm 6	*810.72*	*116.805*	*54.7297*	*2427.87*	*796.545*	*176.962*
	606.87	57.2514	54.7297	1438.58	755.708	176.795
	863.19	118.61	54.7297	2537.91	798.032	176.971
Best known solution	**863.195**	**118.61**	**54.7297**	**2538.19**	**798.032**	**196.971**
Number of solutions	48	20	1	84	83	20

7 Conclusion

The new proposed algorithm performs significantly better than the classical approach that was illustrated by the Algorithm 4.

A deeper analysis of the results indicates that the Algorithm 4 fails to diversify the search and the solution often remains on the same portion of the Pareto front. This result is not surprising and a similar effect was already noticed in AGX when optimizing invariants that could only take few values, such as diameter or maximum degree (this difficulty is known as *plateau*). The algorithm then fails to find any improved solution, because it does not explore other solutions which are equivalent to the incumbent. In the present context, this phenomena is the following: when a graph G is found, the Algorithm 4 simply ignores alternative graphs $G' \in t(G)$ that don't dominate G, even if some of them are not dominated by G either (they are equivalent to the incumbent). In the Algorithm 6, those graphs are stored in the current set of Pareto optimal solutions $\mathcal{G}^*$, whose neighborhood is likely to be explored later.

Analyzing the results from the new algorithm, it is a promising approach to handle the optimization problem related to *plateau*, as the drawbacks from the Algorithm 4 are being rather well corrected by the Algorithm 6. Therefore, the use of the Algorithm 6 can be extended to the classical optimization in AGX. However, the case of two non comparable solution is different from the case of two graphs with the same objective value, as in this last situation, one needs to avoid that both graphs are not isomorphic, otherwise the set $\mathcal{G}^*$ could yield a huge number of isomorphic graphs. It is possible to reduce the number of graphs in $\mathcal{G}^*$ by checking the isomorphism, which would be computationally intensive. An alternative to the use of isomorphism test could be to use a discriminating invariant such as the Balaban index, $\mathcal{G}^*$ being the set of graphs with the best known value for the invariant under study and different values for the discriminating invariant.

In the present paper, a VNS algorithm for finding Pareto optimal solution for vertex related values objectives is proposed. The problem has a special characteristic as many Pareto optimal solutions (vertices) may be associated to the same graph. In this case, it was shown that keeping a variety of non dominated solutions during the local search is better than the natural approach which consists in performing a local search on a single solution that is eventually added to the Pareto front only at the end of the local search. Indeed, the classical approach cannot add more than a single new solution to the Pareto front after each local search.

References

[1] M. Aouchiche, J.-M. Bonnefoy, A. Fidahoussen, G. Caporossi, P. Hansen, J. Lacheré, A. Monhait Variable neighborhood search for extremal graphs. 14. The AutoGraphiX 2 System. In L. Liberti and N. Maculan, editors, *Global Optimization. From Theory to Implementation* (2006) Springer Science, New-York.

[2] A. T. Balaban, Distance Connectivity Index, *Chem. Phys. Lett.* **89**: 399–404 (1982).

[3] G. Caporossi, P. Hansen, Variable Neighborhood Search for Extremal Graphs. I: The AutoGraphiX System, *Discrete Mathematics* **212**:29–44 (2000).

[4] G. Caporossi, P. Hansen, A Learning Optimization Algorithm in Graph Theory. Versatile Search for Extremal Graphs Using a Learning Algorithm, *Lecture Notes in Computer Science* **7219**:16–30 (2012).

[5] P. Hansen, N. Mladenović, Variable neighborhood search: Principles and applications, *European J. Oper. Res.* **130**:449–467 (2001).

[6] N. Mladenović, P. Hansen, Variable neighborhood search, *Comput. Oper. Res.* **24**:1097–1100 (1997).