A Novel Evolutionary Formulation of the Maximum Independent Set Problem

Valmir C. Barbosa[*] and Luciana C. D. Campos

Universidade Federal do Rio de Janeiro
Programa de Engenharia de Sistemas e Computação, COPPE
Caixa Postal 68511
21941-972 Rio de Janeiro - RJ, Brazil

September 12, 2003

Abstract

We introduce a novel evolutionary formulation of the problem of finding a maximum independent set of a graph. The new formulation is based on the relationship that exists between a graph's independence number and its acyclic orientations. It views such orientations as individuals and evolves them with the aid of evolutionary operators that are very heavily based on the structure of the graph and its acyclic orientations. The resulting heuristic has been tested on some of the Second DIMACS Implementation Challenge benchmark graphs, and has been found to be competitive when compared to several of the other heuristics that have also been tested on those graphs.

Keywords: Maximum independent sets, evolutionary algorithms, genetic algorithms.

1 Introduction

Let $G = (N, E)$ be an undirected graph of node set N and edge set E such that $n = |N|$ and $m = |E|$. An *independent set* (or *stable set*) of G is a subset S of N containing no *neighbors* (nodes that are connected by an edge in G). The set $N \setminus S$ is a *node cover* of G, that is, a set of nodes that includes at least one of the two end nodes of every edge. We call every subset of N whose nodes are all neighbors of one another in G a *clique*. The *complement* $\bar{G}$ of G is an undirected graph of node set N in which two nodes are neighbors if and only if

[*]Corresponding author (`valmir@cos.ufrj.br`).

they are not neighbors in G. Clearly, S is an independent set of G if and only if it is a clique of $\bar{G}$.

We are concerned in this paper with the problem of finding a maximum independent set in G, that is, an independent set of maximum size. Equivalently, this problem can be viewed as asking for a minimum node cover in G or a maximum clique in $\bar{G}$. Finding independent sets (or any of the other equivalent structures) of extremal size has several important applications. We refer the reader to the volume that resulted from the Second DIMACS Implementation Challenge [38] for various examples of application areas, and to [25] for further examples from coding theory.

The problem of finding a maximum independent set in G is NP-hard (it is NP-complete when formulated as a decision problem [39, 26]), and remains NP-hard even if we settle for solving it approximately within $n^{1/4-\epsilon}$ of the optimum for any $\epsilon > 0$ [3, 4, 12]. That is, if $\alpha(G)$ is the size of a maximum independent set of G (the *independence number* of G), then finding an independent set of size at least $n^{1/4-\epsilon}\alpha(G)$ is NP-hard.

Our formulation of the maximum independent set problem is based on the notion of an *acyclic orientation* of G, i.e., an assignment of directions to the edges of G that leads to no directed cycles. Let $\Omega(G)$ denote the set of all the acyclic orientations of G. For $\omega \in \Omega(G)$, let D_ω be the set of all *chain decompositions* of the nodes of G according to ω, that is, each member of D_ω is a partition of N into sets that correspond to chains (directed paths) according to ω. For $d \in D_\omega$, let $|d|$ denote the number of chains in d. Our point of departure is the following equality, which relates the independence number of G to its acyclic orientations [23]:

$$\alpha(G) = \max_{\omega \in \Omega(G)} \min_{d \in D_\omega} |d|. \tag{1}$$

By (1), $\alpha(G)$ is the number of chains in the chain decomposition of N into the fewest possible chains, according to the acyclic orientation of G for which that number is greatest. This result is a refinement of Dilworth's theorem [24] and is illustrated in Figure 1, where two acyclic orientations of the same graph are shown alongside the corresponding minimum chain decompositions. In the figure, the bottommost acyclic orientation is the one whose minimum chain decomposition is greatest, thence $\alpha(G) = 2$ for the graph in question.

As we see it, the greatest significance of (1) is that it spells out how the set $\Omega(G)$ can be regarded as a set of individuals, the fittest of which yields the independence number of G.[1] Viewing the maximum independent set problem from this perspective is based on taking $\min_{d \in D_\omega} |d|$ as the measure of fitness for individual ω and on searching $\Omega(G)$ for an individual of maximum fitness.[2]

[1] A relation dual to the one in (1) indicates how to express the chromatic number of G (cf. [14]) in terms of its acyclic orientations [23]. The use of that other relation in an evolutionary approach to find the graph's chromatic number has been developed by one of us and collaborators [11].

[2] $\Omega(G)$ is a set of vast dimensions. For example, the number of distinct acyclic orientations of G is 2^m if G is a tree, $n!$ if it is a complete graph (all nodes connected to all others), and

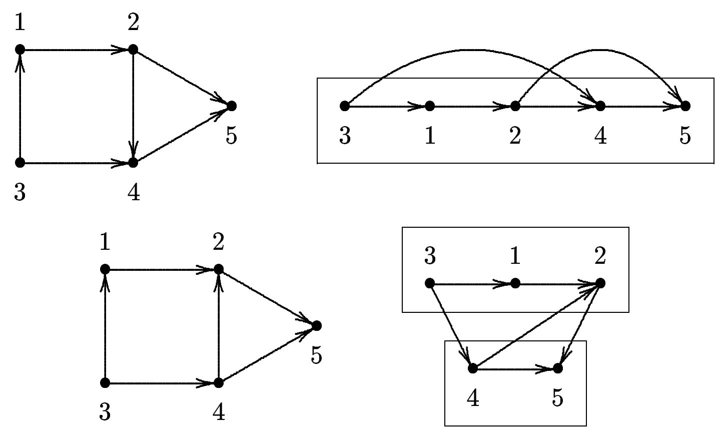

Figure 1: Two acyclic orientations and the corresponding minimum chain decompositions.

We develop this evolutionary approach in the remainder of the paper, starting in Section 2, where the details of the new formulation are introduced, including how to compute an individual's fitness and the evolutionary operators of crossover and mutation for $\Omega(G)$. We then continue in Section 3 with a description of the graphs to be used in our experiments, whose results are reported in Section 4. Conclusions are given in Section 5.

2 The formulation

The key to a better understanding of how to use (1) is that $\min_{d\in D_\omega} |d|$, the purported measure of fitness for individual $\omega \in \Omega(G)$, is in fact the size of an independent set that can be unequivocally obtained from orientation ω. In order to see this, we associate a directed graph, call it $D(G,\omega)$, with G oriented by ω. This directed graph has $2n+2$ nodes: two distinguished nodes, called s and t, and two nodes, called i' and i'', for each node $i \in N$. For every $i \in N$, $D(G,\omega)$ has an edge directed from s to i' and another from i'' to t. For every $(i,j) \in E$ that is directed by ω from i to j, in $D(G,\omega)$ an edge exists from i' to j''. Node s is therefore a *source* (a node with no edges directed inward) and node t a *sink* (a node with no edges directed outward). An illustration of this construction is given in Figure 2, where the directed graphs corresponding to the two acyclic orientations of Figure 1 are shown.

We now regard $D(G,\omega)$ as a flow network whose edges either have unit capacity (those leaving s or arriving at t) or infinite capacity (all others), and

$2^n - 2$ if it is a ring. In general, and remarkably, the number of members of $\Omega(G)$ is given by the absolute value of the chromatic polynomial of G (cf. [14]) applied to the negative unit [48].

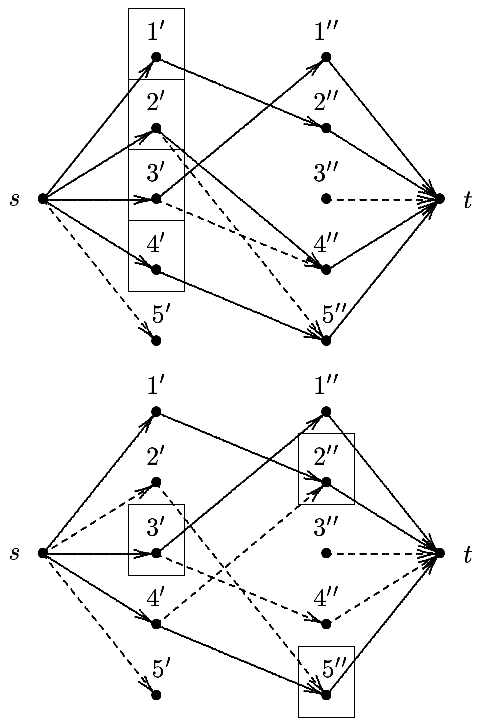

Figure 2: Flow networks associated with the acyclic orientations of Figure 1.

consider the maximum flow from s to t, whose value we assume to be F. This flow is necessarily integral, as illustrated in Figure 2, where solid lines have been used to draw edges carrying unit flow and dashed lines those carrying zero flow. It also establishes a chain decomposition of G according to ω into $n - F$ chains. Take, for example, the topmost network of Figure 2, where $F = 4$. If we only follow edges carrying unit flow, then we can easily trace the single chain $3' \to 1''$, $1' \to 2''$, $2' \to 4''$, $4' \to 5''$. For the bottommost network, $F = 3$ and we get the two chains $3' \to 1''$, $1' \to 2''$ and $4' \to 5''$. These, readily, are the minimum chain decompositions according to the two acyclic orientations.

What is left to note is that only edges leaving s or arriving at t can be saturated by the maximum flow, so the corresponding minimum cut is necessarily given by a group of such edges. In Figure 2, the minimum cut in each network is indicated by boxes enclosing the nodes other than s or t that are involved in the cut. Such nodes do necessarily constitute a minimum node cover of the edges that do not involve s or t in $D(G, \omega)$, and consequently induce a node cover in G as well ($\{1, 2, 3, 4\}$ for the topmost network, $\{2, 3, 5\}$ for the other) with a corresponding independent set ($\{5\}$ and $\{1, 4\}$, respectively).

In general, then, we have the following [2]. If F is the value of the maximum flow from s to t in $D(G,\omega)$, then $n-F$ is the number of chains in the minimum chain decomposition of G according to ω and F is the size of a node cover in G. Consequently, $n-F$ is also the size of an independent set in G. In order to determine the actual nodes that constitute this independent set, it suffices to look at the minimum cut in $D(G,\omega)$ and at the node cover it induces on the edges of $D(G,\omega)$ that do not touch s or t. This node cover corresponds to a node cover in G as well, whose complement with respect to N is then the desired independent set.

We now turn to the three key elements of our formulation, namely how to assess an individual's fitness and how crossover and mutation operate. We have designed these elements in such a way that they can be used directly in most standard templates of fitness-maximization genetic algorithms [30, 45]. What is novel in our formulation is the adoption of $\Omega(G)$ as the search space out of which populations are formed. Not only do the acyclic orientations of G relate cleanly, as we have discussed, to the independent sets of G, but also they allow for evolutionary operators that are simple and yet effective in several aspects of the evolutionary search. We will come to them shortly.

Fitness evaluation

For $\omega \in \Omega(G)$, let $f(\omega)$ denote the fitness to be maximized over $\Omega(G)$ while searching for a maximum independent set of G. By (1), we have

$$f(\omega) = \min_{d \in D_\omega} |d|, \tag{2}$$

that is, the fitness of individual ω is the number of chains in the minimum chain decomposition according to that orientation.

It follows from our preceding discussion that $f(\omega)$ can be assessed along the following steps:

1. Construct the flow network $D(G,\omega)$.
2. Compute the value F of the maximum flow from s to t in $D(G,\omega)$.
3. Let $f(\omega) = n - F$.

We note, with regard to Step 2, that only the value F of the maximum flow is needed for fitness evaluation, not the actual flow. In other words, what is needed is the maximum total flow incoming to t, not the particular assignment of flows to all edges. When a push-relabel method is used to compute the maximum flow, it is a simple matter to separate the computation into two phases [29]: the first phase computes F but may leave excess flow at some nodes; the second phase corrects this by returning flow in order to eliminate excesses. Conveniently, one of the most successful implementations currently available of a maximum-flow algorithm does precisely this [22], and as such allows the computation to stop at the end of the first phase, right after F has been found. What is also convenient is that, when push-relabel methods are thus implemented, at the end of the first phase the minimum cut is also known, which is useful for determining the independent set that corresponds to the best individual found during the evolutionary search, in the manner we indicated earlier in this section. We return to this in Section 4.

Crossover

The crossover of the two individuals $\omega_1, \omega_2 \in \Omega(G)$ to yield the two offspring $\omega_1', \omega_2' \in \Omega(G)$ is best described in terms of a linear representations of the individuals involved. For individual $\omega \in \Omega(G)$, the representation we adopt, denoted by $L(\omega)$, is the sequence $L(\omega) = \langle i_1, \ldots, i_n \rangle$, where $i_1, \ldots, i_n$ are the nodes of G. In this sequence, and for $1 \le x, y \le n$, node i_x appears to the left of i_y (i.e., $x < y$) if (i_x, i_y) is an edge of G and is oriented by ω from i_x to i_y. For example, both $\langle 3,1,4,2,5 \rangle$ and $\langle 3,4,1,2,5 \rangle$ are valid linear representations of the bottommost acyclic orientation of Figure 1. Clearly, $L(\omega)$ represents ω unambiguously, though not uniquely.

Now let $L(\omega_1) = \langle i_1, \ldots, i_n \rangle$ and $L(\omega_2) = \langle j_1, \ldots, j_n \rangle$. Let also z such that $1 \le z < n$ be the crossover point. Then $L(\omega_1') = \langle i_1', \ldots, i_n' \rangle$ and $L(\omega_2') = \langle j_1', \ldots, j_n' \rangle$, where

- $\langle i_1', \ldots, i_z' \rangle = \langle i_1, \ldots, i_z \rangle$;
- $\langle i_{z+1}', \ldots, i_n' \rangle$ is the subsequence of $\langle j_1, \ldots, j_n \rangle$ comprising all nodes that are not in $\langle i_1, \ldots, i_z \rangle$;
- $\langle j_1', \ldots, j_z' \rangle = \langle j_1, \ldots, j_z \rangle$;
- $\langle j_{z+1}', \ldots, j_n' \rangle$ is the subsequence of $\langle i_1, \ldots, i_n \rangle$ comprising all nodes that are not in $\langle j_1, \ldots, j_z \rangle$.

Notice that the $L(\omega_1')$ and $L(\omega_2')$ thus determined are valid linear representations of acyclic orientations, since, by construction, both sequences contain all nodes from G. Furthermore, ω_1' and ω_2' inherit edge orientations from ω_1 and ω_2 as follows:

- Edges joining nodes in the set $\{i_1', \ldots, i_z'\}$ to any other nodes are oriented by ω_1' exactly as by ω_1.
- Edges joining nodes in the set $\{i_{z+1}', \ldots, i_n'\}$ exclusively are oriented by ω_1' exactly as by ω_2.
- Edges joining nodes in the set $\{j_1', \ldots, j_z'\}$ to any other nodes are oriented by ω_2' exactly as by ω_2.
- Edges joining nodes in the set $\{j_{z+1}', \ldots, j_n'\}$ exclusively are oriented by ω_2' exactly as by ω_1.

Mutation

Like crossover, our mutation operator is defined in terms of the linear representations of individuals. We use single-locus mutation at the mutation point z with $1 \leq z \leq n$. For individual $\omega \in \Omega(G)$, the mutation operator turns node i_z into a source, thus yielding another acyclic orientation ω'. That ω' is indeed acyclic has been argued elsewhere [10]: in essence, a directed cycle through i_z would be required for ω' not to be acyclic, which is impossible, i_z being a source according to ω'.

We also refer the reader elsewhere ([11], Section 4) for an argumentation as to why this type of mutation does more for the evolutionary search than simply to allow occasional random jumps loosely intended to escape local optima. Specifically, what is shown is that, for any two acyclic orientations $\omega, \omega' \in \Omega(G)$, there necessarily exists a finite sequence of mutations that turns ω into ω'. So this operator can be regarded as providing the search space $\Omega(G)$ with an underlying fundamental connectedness that allows, at least in principle, every acyclic orientation to be reached regardless of where the evolutionary search is started.

3 The experimental test set

In this section we give a brief description of the benchmark graphs used in the experiments reported in Section 4. They have all been extracted from the DIMACS challenge suite [49]. That suite is structured from the perspective of finding maximum cliques, so the graph types listed next characterize $\bar{G}$.

`c-fat`n-c [34]. This graph comes from fault-diagnosis problems [13] and its set of n nodes is partitioned into $n/c \log n$ sets of approximately equal sizes. Edges are deployed so that every node is connected to every other node in its own set and in the two sets that are neighbors to its own (according to an arrangement of the sets into a ring).

`johnson`W-w-d [34]. This graph arises in problems from coding theory. It has $n = \binom{W}{w}$ nodes, each node labeled with a W-digit binary number having exactly w 1's. Two nodes are joined by an edge if the Hamming distance between their labels (the number of digits at which they differ) is at least d.

`keller`d [41]. This graph is derived from another with 4^d nodes which arises in connection with proving Keller's 1930 conjecture false for high dimensions. The conjecture is that a "tiling" of Euclidean d-dimensional space by unit cubes necessarily contains two cubes meeting in a full $(d-1)$-dimensional face.

`hamming`W-d [34]. This graph comes from coding-theory problems also and has $n = 2^W$ nodes, each node labeled with a W-digit binary number. Two nodes are connected if the Hamming distance between their labels is at least d.

`san`n`_`f`_x` [36]. This graph is artificially constructed on n nodes to have a maximum clique whose size is determined beforehand. Its number of edges is $fn(n-1)/2$; `x` is only used to differentiate among instances.

`sanr`n`_`p [36]. This is a random graph on n nodes, generated by adding an edge with constant probability p between any two distinct nodes [15]. It is expected to have dimensions close to those of `san`n`_`f`_x` for $f = p$.

`brock`n`_x` [18]. This is a random graph on n nodes, designed to have a maximum clique much larger than would be expected from the nodes' degrees. The number `x` is used for instance differentiation.

`p_hat`n`-x` [46]. This is a random graph on n nodes whose density is based on two parameters. This contrasts with the usual random graphs with fixed edge probability [15], leading to node degrees that are more spread and to larger cliques also. The number `x` differentiates among instances.

`MANN_ax` [44]. This graph gives the clique formulation of the instance A`x` of the Steiner triple problem. This formulation is obtained by a conversion from the set-covering formulation of that problem [43].

Details on the graphs we used in our experiments are given in Tables 1 and 2. For each $\bar{G}$, the tables give the values of n and m (the number of edges in G), as well as $\alpha(G)$, when known from design characteristics.

4 Experimental results

Henceforth, we refer as WAO to the algorithm that results from the formulation of Section 2. This denomination is an acronym after "Widest Acyclic Orientation," as by (1) what is sought during the evolutionary search is an acyclic orientation whose minimum chain decomposition has the most chains over $\Omega(G)$.

WAO iterates for g generations, each one characterized by a population of fixed size s. After generating the last population, it outputs the best individual found during the entire evolutionary search. For $k > 1$, the kth population is obtained from the $k-1$st population as follows. First an elitist step is performed, resulting in the transfer of the fs fittest individuals from the current population to the new, with $0 \leq f < 1$. Then WAO performs the following iteration until the new population is full: with probability p_c, two individuals are selected from the current population and the crossover operator is applied to them, the resulting two individuals being then added to the new population; with probability $1 - p_c$, one single individual is selected from the current population and then is subjected to the mutation operator before being added to the new population.

In order to decide on an appropriate selection method, we ran several initial experiments on reasonably-sized graphs. From these experiments emerged not

Table 1: Benchmark graphs.

$\bar{G}$	n	m	$\alpha(G)$
c-fat200-1	200	18,366	
c-fat200-2	200	16,665	
c-fat200-5	200	11,427	
c-fat500-1	500	120,291	
c-fat500-2	500	115,611	
c-fat500-5	500	101,559	
c-fat500-10	500	78,123	
johnson8-2-4	28	168	
johnson8-4-4	70	560	
johnson16-2-4	120	1,680	
johnson32-2-4	496	14,880	
keller4	171	5,100	
hamming6-2	64	192	
hamming6-4	64	1,312	
hamming8-2	256	1,024	
hamming8-4	256	11,776	
san200_0.7_1	200	5,970	30
san200_0.7_2	200	5,970	18
san200_0.9_1	200	1,990	70
san200_0.9_2	200	1,990	60
san200_0.9_3	200	1,990	44
san400_0.5_1	400	39,900	13
san400_0.7_1	400	23,940	40
san400_0.7_2	400	23,940	30
san400_0.7_3	400	23,940	22
san400_0.9_1	400	7,980	100

only the selection method of our choice but also the suite of parameters we would adopt in all further experiments (we discuss these later). The selection method we used in our experiments picks individuals proportionally to their linearly normalized fitness in the current population. For $1 \le k \le s$, this means that the kth fittest individual—that is, ω such that $f(\omega)$ is the kth greatest—is selected with probability proportional to

$$g(\omega) = L - \left(\frac{L-1}{s-1}\right)(k-1). \tag{3}$$

Ties between two individuals are broken by taking the individual that was added to the current population first as the fitter one. In (3), L is the factor by which the linearly normalized fitness of the fittest individual in the current population (the $k = 1$ case) is greater than that of the least fit individual (the $k = s$ case); that is, $L = g(\omega_1)/g(\omega_s)$, where ω_1 and ω_s are those two individuals, respectively.

Table 2: Benchmark graphs (continued from Table 1).

$\bar{G}$	n	m	$\alpha(G)$
`sanr200_0.7`	200	6,032	
`sanr200_0.9`	200	2,037	
`sanr400_0.5`	400	39,816	
`sanr400_0.7`	400	23,931	
`brock200_1`	200	5,066	21
`brock200_2`	200	10,024	12
`brock200_3`	200	7,852	15
`brock200_4`	200	6,811	17
`brock400_1`	400	20,077	27
`brock400_2`	400	20,014	29
`brock400_3`	400	20,119	31
`brock400_4`	400	20,035	33
`p_hat300-1`	300	33,917	
`p_hat300-2`	300	22,922	
`p_hat300-3`	300	11,460	
`p_hat500-1`	500	93,181	
`p_hat500-2`	500	61,804	
`p_hat500-3`	500	30,950	
`MANN_a9`	45	72	
`MANN_a27`	378	702	

We present our results in comparison to those obtained by the heuristics of the DIMACS challenge [38]. This is not to say that the best results known to date are necessarily the ones obtained by those heuristics, since several new methods have appeared in the meantime for the maximum independent set problem under one of its guises (e.g., [19, 16] and their references). However, all those more recent methods invariably go back to the DIMACS challenge heuristics as references for comparison, so those heuristics serve as an indirect basis for other comparisons as well. One exception to this comparison rule we have adopted is the genetic-algorithm approach of [1], which, like WAO, employs a nontrivial crossover operator. We present next a brief description of all the eleven heuristics to which we compare WAO directly.

B&C [7]. This is a branch-and-cut method for which cutting planes are generated based on the more general technique of [6]. It starts with an integer programming formulation of the maximum clique problem, and proceeds from the initial relaxation by generating new cutting planes and incorporating them into the current linear program.

CLIQMERGE [8]. This heuristic is based on a procedure that finds a maximum clique in the subgraph induced by the nodes of two cliques when they are merged together. The essence of the method is to find a maximum bipartite matching in the complement of this subgraph.

SQUEEZE [17]. This is a branch-and-bound algorithm for the maximum independent set problem. Its lower bounds are obtained through a reduction to the problem of minimizing a general quadratic 0-1 function.

CBH [27]. This is an interior-point approach (cf. [32]) to the determination of maximum independent sets. Following a continuous formulation of the problem, a relaxation of it is solved and the resulting solution is rounded by a heuristic based on [20, 5].

RB-CLIQUE [31]. This method uses backtracking "coordinates" as the entities on which restricted backtracking is to be applied while seeking a maximum clique. The restrictions to which the backtracking coordinates are subject are given as input.

ATA [33]. This strategy employs recurrent neural networks (cf. [9]) to find maximum cliques. The crux of the approach is an adaptive procedure for the determination of appropriate threshold parameters and initial state for the neural network.

SA&GH [35]. This is a blend of heuristics to find maximum cliques. In most cases it employs simply simulated annealing [40], but for very dense graphs the greedy heuristic of [37] is used.

XSD [36]. This is a family of heuristic methods to find maximum cliques. The methods are all related to neural-network models and include deterministic and stochastic descent approaches, with or without an intervening learning step between restarts.

B&B [44]. This is a branch-and-bound approach to the maximum independent set problem. Upper bounds are obtained through a procedure derived from edge projection, a specialization of the clique projection of [42].

XT [47]. This is a family of three variants of tabu search [28], two deterministic and one probabilistic.

OCH [1]. This is a genetic algorithm for the maximum independent set problem. Each individual is an n-digit binary number, each digit indicating whether the corresponding node is in the independent set or not. The centerpiece of the method is the so-called optimized crossover, which generates one optimal offspring based on the same merging procedure of CLIQMERGE and one other having a random character.

All the experiments we report on were conducted with $g = 10n$, $s = 1.5n$, $f = 0.05$, $p_c = 0.2$, and $L \in \{15, 30\}$. As we indicated earlier, these reflect policies and values that emerged from early experiments on reasonably-sized graphs. They by no means represent optimal decisions of any sort, since the number of possible choices is, naturally, far too large.

In all experiments, we also made use of the maximum-flow code of [21], which implements the algorithm of [22], to compute the value of $f(\omega)$ as explained in Section 2. As we also indicated in that section, it suffices for the maximum-flow computation to stop right after completing its first phase, since the value of the maximum flow, and hence the size of the independent set that the individual contributes, is already known at this point. We also recall that running the maximum-flow code through its first phase only is sufficient even for the best individual found during the whole evolutionary search. In this case, what we need is to enumerate the members of the independent set contributed by that individual, not simply to know its size, but this can be obtained by examining the minimum cut that is also already known as the first phase ends.

Our results are shown in Figures 3 and 4, and in Tables 3 through 8. The two figures show, for ten graphs selected from Tables 1 and 2, the evolution of the best fitness (as given by (2)) ever found for an individual as the generations elapse during the best of twenty independent runs (the one that eventually yielded the largest independent set for that graph). In other words, they show the size of the largest independent set yet identified. These ten graphs were selected because, in terms of what is shown in Figures 3 and 4, they have led the evolutionary search to behave either in a way that we found to be somewhat typical or a way that yields interesting insight. We return to this shortly.

Tables 3 through 8 show the results obtained by WAO alongside the results of the competing algorithms we outlined earlier in this section. Tables 3–5 refer to the graphs in Table 1, while Tables 6–8 refer to those in Table 2. For each graph $\bar{G}$, the tables give the value of $\alpha(G)$, when known from design characteristics, and the sizes of the maximum independent sets obtained by the algorithms on G, when available. The result reported for WAO on each graph is the best result found over twenty independent runs. For the other methods, the results reported are the best results they yielded, as published in [38]. The number appearing in parentheses next to "WAO" in Tables 5 and 8 is the value of L that was used to obtain the results listed in the corresponding column.

The several runs of WAO were executed on a relatively wide assortment of machine architectures, so we refrain from providing detailed timing data. Also, comparing running times to those of the other methods—obtained roughly one decade ago—would be cumbersome however we tampered with the numbers seeking to compensate for the technological gap. Given these constraints, all we do is mention that each of our runs tended to complete somewhere between very few seconds and a week, depending on the graph at hand.

WAO is a competitive method, by all that can be inferred from Tables 3–8. We have in the tables used a bold typeface to indicate the instances on which WAO performed at least as well as the best performers. Several entries are thus marked, and for many that are not WAO is seen to have fallen short by a very narrow margin.

Returning to the plots in Figures 3 and 4 may highlight some of the patterns that help explain success or failure at meeting the best performers' figures during our experiments with WAO. Of the ten graphs to which those figures refer, three correspond to cases in which WAO missed by a narrow margin (`san400_0.9_1`, `sanr400_0.7`, and `brock200_4`) and one to a case in which it missed widely (`brock400_4`). What seems to distinguish one group from the other is that in the latter case the plot becomes flat early in the evolution, perhaps signaling an inherent hardness at escaping some particularly difficult local maximum. For the three graphs in the former group, however, and in fact for the six graphs in the figures outside either group, evolution seems to lead to fitness growth more or less steadily along a comparatively larger number of generations, even though for the `san` graphs it nearly stalls for a significant number of generations before it gains momentum again. In the case of `san400_0.9_1`, particularly, it appears quite likely that a few more generations would have bridged the narrow gap between the 98 that WAO achieved and the 100 of its best contenders.

5 Concluding remarks

We have in this paper introduced WAO, a novel evolutionary heuristic for the maximum independent set problem. WAO is based on a view of the problem that relates independent sets to the acyclic orientations of the graph, and seeks to identify an acyclic orientation that is widest (has the decomposition into the fewest number of chains that requires the most chains) over the set of all the acyclic orientations of the graph. It incorporates no additional sophistication into the usual evolutionary-algorithm machinery, but rather into the design of the individuals' representations and the evolutionary operators, all based on complex graph-theoretic notions.

We have found our new heuristic to perform competitively when compared to several others on the DIMACS benchmark graphs. Notwithstanding this, there certainly is room for further investigation and improvements. For example, there may exist a better set of parameters for the evolutionary search, just as there may exist a better functional dependence of g and s on n (and perhaps also on m, unlike what we adopted in our experiments). Likewise, it is also conceivable that the formulation itself may be improved by the incorporation of optimizations into the crossover or mutation operator, or even by the introduction of new operators.

Acknowledgments

The authors acknowledge partial support from CNPq, CAPES, the PRONEX initiative of Brazil's MCT under contract 41.96.0857.00, and a FAPERJ BBP grant.

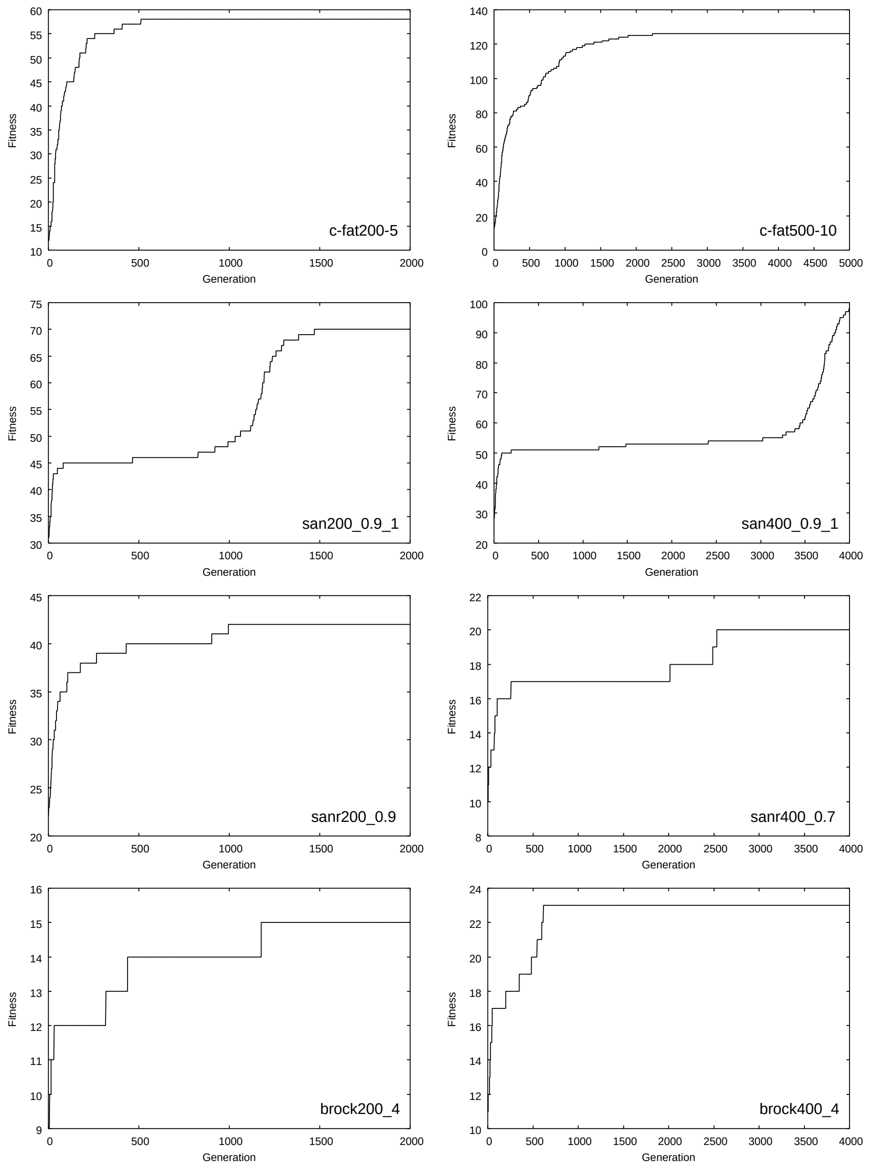

Figure 3: Fitness evolution for selected graphs.

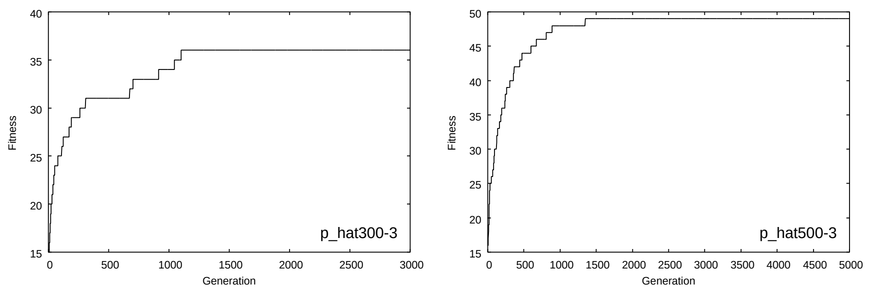

Figure 4: Fitness evolution for selected graphs (continued from Figure 3).

Table 3: Comparative performance on the benchmark graphs of Table 1.

$\bar{G}$	$\alpha(G)$	B&C	Cliqmerge	Squeeze	CBH
c-fat200-1		12		12	12
c-fat200-2		24		24	24
c-fat200-5		58		58	58
c-fat500-1				14	14
c-fat500-2				26	26
c-fat500-5				64	64
c-fat500-10				126	126
johnson8-2-4		4		4	4
johnson8-4-4		14		14	14
johnson16-2-4		8		8	8
johnson32-2-4				16	16
keller4		11	11	11	10
hamming6-2		32		32	32
hamming6-4		4		4	4
hamming8-2		128		128	128
hamming8-4		16	16	16	16
san200_0.7_1	30	30		30	15
san200_0.7_2	18	18		18	12
san200_0.9_1	70	70		70	46
san200_0.9_2	60	60		60	36
san200_0.9_3	44	44		44	30
san400_0.5_1	13	13			8
san400_0.7_1	40				20
san400_0.7_2	30	30			15
san400_0.7_3	22				14
san400_0.9_1	100			100	50

Table 4: Comparative performance on the benchmark graphs of Table 1 (continued from Table 3).

$\bar{G}$	$\alpha(G)$	RB-CLIQUE	ATA	SA&GH	XSD
c-fat200-1					12
c-fat200-2					24
c-fat200-5					58
c-fat500-1					14
c-fat500-2					26
c-fat500-5					64
c-fat500-10					126
johnson8-2-4					4
johnson8-4-4					14
johnson16-2-4					8
johnson32-2-4					16
keller4		11	11	11	11
hamming6-2					32
hamming6-4					4
hamming8-2					128
hamming8-4		16	16	16	16
san200_0.7_1	30				30
san200_0.7_2	18				15
san200_0.9_1	70				70
san200_0.9_2	60				60
san200_0.9_3	44				36
san400_0.5_1	13				9
san400_0.7_1	40				33
san400_0.7_2	30				19
san400_0.7_3	22				16
san400_0.9_1	100				100

Table 5: Comparative performance on the benchmark graphs of Table 1 (continued from Table 4).

$\bar{G}$	$\alpha(G)$	B&B	XT	OCH	WAO (15)	WAO (30)
c-fat200-1				12	**12**	**12**
c-fat200-2				24	**24**	**24**
c-fat200-5				58	**58**	**58**
c-fat500-1				14	**14**	**14**
c-fat500-2				26	**26**	**26**
c-fat500-5				64	**64**	**64**
c-fat500-10				126	**126**	**126**
johnson8-2-4				4	**4**	**4**
johnson8-4-4				14	**14**	**14**
johnson16-2-4				8	**8**	**8**
johnson32-2-4				16	**16**	**16**
keller4		11	11	11	**11**	**11**
hamming6-2				32	**32**	**32**
hamming6-4				4	**4**	**4**
hamming8-2				128	**128**	**128**
hamming8-4		16	16	16	**16**	**16**
san200_0.7_1	30			30	16	16
san200_0.7_2	18			15	14	14
san200_0.9_1	70			70	**70**	**70**
san200_0.9_2	60			60	**60**	58
san200_0.9_3	44			36	37	**44**
san400_0.5_1	13			13	8	8
san400_0.7_1	40			40	20	20
san400_0.7_2	30			30	17	17
san400_0.7_3	22			16	16	16
san400_0.9_1	100			100	54	98

Table 6: Comparative performance on the benchmark graphs of Table 2.

$\bar{G}$	$\alpha(G)$	B&C	Cliqmerge	Squeeze	CBH
sanr200_0.7				18	18
sanr200_0.9				41	41
sanr400_0.5					12
sanr400_0.7				20	20
brock200_1	21			21	20
brock200_2	12	12	11	12	12
brock200_3	15			15	14
brock200_4	17		16	17	16
brock400_1	27				23
brock400_2	29		25		24
brock400_3	31				23
brock400_4	33		25		24
p_hat300-1		8	8	8	8
p_hat300-2			25	25	25
p_hat300-3			36	36	36
p_hat500-1				9	9
p_hat500-2				36	35
p_hat500-3					49
MANN_a9				16	16
MANN_a27			126	126	121

Table 7: Comparative performance on the benchmark graphs of Table 2 (continued from Table 6).

$\bar{G}$	$\alpha(G)$	RB-CLIQUE	ATA	SA&GH	XSD
sanr200_0.7					18
sanr200_0.9					41
sanr400_0.5					12
sanr400_0.7					21
brock200_1	21				20
brock200_2	12	12	11	11	10
brock200_3	15				15
brock200_4	17	17	16	16	16
brock400_1	27				24
brock400_2	29	25	25	25	24
brock400_3	31				24
brock400_4	33	33	25	25	24
p_hat300-1		8	8	8	8
p_hat300-2		25	25	25	25
p_hat300-3		35	36	36	34
p_hat500-1					
p_hat500-2					
p_hat500-3					
MANN_a9					
MANN_a27		126	125	126	126

Table 8: Comparative performance on the benchmark graphs of Table 2 (continued from Table 7).

$\bar{G}$	$\alpha(G)$	B&B	XT	OCH	WAO (15)	WAO (30)
`sanr200_0.7`				18	17	**18**
`sanr200_0.9`				42	**42**	41
`sanr400_0.5`				12	11	11
`sanr400_0.7`				20	20	19
`brock200_1`	21			21	19	19
`brock200_2`	12	12	11	11	10	9
`brock200_3`	15			14	13	13
`brock200_4`	17	17	16	16	15	15
`brock400_1`	27			24	21	22
`brock400_2`	29		25	24	21	22
`brock400_3`	31			24	22	22
`brock400_4`	33	33	25	24	23	22
`p_hat300-1`		8	8	8	**8**	7
`p_hat300-2`		25	25	25	24	**25**
`p_hat300-3`		36	36	36	34	**36**
`p_hat500-1`				9	**9**	**9**
`p_hat500-2`				36	34	35
`p_hat500-3`				49	**49**	48
`MANN_a9`				16	**16**	**16**
`MANN_a27`		126	125	126	**126**	**126**

References

[1] C. C. Aggarwal, J. B. Orlin, and R. P. Tai. Optimized crossover for the independent set problem. *Operations Research*, 45:226–234, 1997.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] S. Arora and C. Lund. Hardness of approximations. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 399–446. PWS Publishing Company, Boston, MA, 1997.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer-Verlag, Berlin, Germany, 1999.

[5] L. Babel. Finding maximum cliques in arbitrary and in special graphs. *Computing*, 46:321–341, 1991.

[6] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming A*, 58:295–324, 1993.

[7] E. Balas, S. Ceria, G. Cornuéjols, and G. Pataki. Polyhedral methods for the maximum clique problem. In Johnson and Trick [38], pages 11–28.

[8] E. Balas and W. Niehaus. Finding large cliques in arbitrary graphs by bipartite matching. In Johnson and Trick [38], pages 29–51.

[9] V. C. Barbosa. *Massively Parallel Models of Computation.* Ellis Horwood, Chichester, UK, 1993.

[10] V. C. Barbosa. *An Atlas of Edge-Reversal Dynamics.* Chapman & Hall/CRC, London, UK, 2000.

[11] V. C. Barbosa, C. A. G. Assis, and J. O. do Nascimento. Two novel evolutionary formulations of the graph coloring problem. *Journal of Combinatorial Optimization.* To appear.

[12] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and non-approximability—towards tight results. *SIAM Journal on Computing*, 27:804–915, 1998.

[13] P. Berman and A. Pelc. Distributed fault diagnosis for multiprocessor systems. In *Proceedings of the Twentieth Annual International Symposium on Fault-Tolerant Computing*, pages 340–346, 1990.

[14] B. Bollobás. *Modern Graph Theory.* Springer-Verlag, New York, NY, 1998.

[15] B. Bollobás. *Random Graphs.* Cambridge University Press, Cambridge, UK, second edition, 2001.

[16] I. M. Bomze, M. Budinich, M. Pelillo, and C. Rossi. Annealed replication: a new heuristic for the maximum clique problem. *Discrete Applied Mathematics*, 121:27–49, 2002.

[17] J.-M. Bourjolly, P. Gill, G. Laporte, and H. Mercure. An exact quadratic 0-1 algorithm for the stable set problem. In Johnson and Trick [38], pages 53–73.

[18] M. Brockington and J. C. Culberson. Camouflaging independent sets in quasi-random graphs. In Johnson and Trick [38], pages 75–88.

[19] S. Busygin, S. Butenko, and P. M. Pardalos. A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *Journal of Combinatorial Optimization*, 6:287–297, 2002.

[20] R. Carraghan and P. M. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.

[21] B. V. Cherkassky and A. V. Goldberg. `http://www.avglab.com/andrew/soft/hipr.tar`.

[22] B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

[23] R. W. Deming. Acyclic orientations of a graph and chromatic and independence numbers. *Journal of Combinatorial Theory B*, 26:101–110, 1979.

[24] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–165, 1950.

[25] T. Etzion and P. R. J. Östergård. Greedy and heuristic algorithms for codes and colorings. *IEEE Transactions on Information Theory*, 44:382–388, 1998.

[26] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[27] L. E. Gibbons, D. H. Hearn, and P. M. Pardalos. A continuous based heuristic for the maximum clique problem. In Johnson and Trick [38], pages 103–124.

[28] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[29] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35:921–940, 1988.

[30] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, MA, 1989.

[31] M. K. Goldberg and R. D. Rivenburgh. Constructing cliques using restricted backtracking. In Johnson and Trick [38], pages 89–101.

[32] C. C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34:167–224, 1992.

[33] T. Grossman. Applying the INN model to the Maximum Clique problem. In Johnson and Trick [38], pages 125–145.

[34] J. Hasselberg, P. M. Pardalos, and G. Vairaktarakis. Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization*, 3:463–482, 1993.

[35] S. Homer and M. Peinado. Experiments with polynomial-time CLIQUE approximation algorithms on very large graphs. In Johnson and Trick [38], pages 147–167.

[36] A. Jagota, L. Sanchis, and R. Ganesan. Approximately solving Maximum Clique using neural network and related heuristics. In Johnson and Trick [38], pages 169–204.

[37] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[38] D. S. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability.* American Mathematical Society, Providence, RI, 1996.

[39] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, 1972.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[41] J. C. Lagarias and P. W. Shor. Keller's cube-tiling conjecture is false in high dimensions. *Bulletin of the American Mathematical Society*, 27:279–283, 1992.

[42] L. Lovász and M. D. Plummer. *Matching Theory.* North-Holland, Amsterdam, The Netherlands, 1986.

[43] C. Mannino and A. Sassano. Solving hard set covering problems. *Operations Research Letters*, 18:1–5, 1995.

[44] C. Mannino and A. Sassano. Edge projection and the maximum cardinality stable set problem. In Johnson and Trick [38], pages 205–219.

[45] M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, MA, 1996.

[46] P. Soriano and M. Gendreau. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41:385–403, 1993.

[47] P. Soriano and M. Gendreau. Tabu search algorithms for the maximum clique problem. In Johnson and Trick [38], pages 221–241.

[48] R. P. Stanley. Acyclic orientations of graphs. *Discrete Mathematics*, 5:171–178, 1973.

[49] M. A. Trick. Appendix: Second DIMACS Challenge test problems. In Johnson and Trick [38], pages 653–657.