ELSEVIER Discrete Applied Mathematics 51 (1994) 291–305

DISCRETE APPLIED MATHEMATICS

Periodic assignment and graph colouring

Jan Korst*[,a], Emile Aarts[a,b], Jan Karel Lenstra[b,c], Jaap Wessels[b]

[a] *Philips Research Laboratories, P.O. Box 80.000, 5600 JA Eindhoven, The Netherlands*
[b] *Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*
[c] *CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands*

(Received 20 May 1991; revised 29 July 1992)

Abstract

We analyse the problem of executing periodic operations on a minimum number of identical processors under different constraints. The analysis is based on a reformulation of the problem in terms of graph colouring. It is shown that different constraints result in colouring problems defined on different classes of graphs, viz. interval graphs, circular-arc graphs and periodic-interval graphs. We discuss the complexity of these colouring problems in detail.

Key words: Periodic assignment; Graph colouring; Interval graphs; Circular-arc graphs; Periodic-interval graphs

1. Introduction

In this paper we consider periodic assignment, i.e. the problem of assigning periodic operations to processors. Operations are called periodic if they have to be repeatedly executed at a constant rate over an infinite-time horizon. Here, we assume that the executions of periodic operations have fixed start times. The executions have to be assigned to a minimum number of processors. The more general problem of finding start times that minimize the number of processors is discussed in [15]. The periodic assignment problem naturally arises in such diverse areas as real-time processing, vehicle scheduling and compiler design [21, 23, 25].

We analyse the periodic assignment problem under different constraints, resulting in a graph colouring formulation of the problem for three different classes of graphs. The problem of colouring the vertices of a graph with a minimum number of colours such that adjacent vertices are given different colours is NP-hard for arbitrary graphs. Furthermore, no efficient approximation algorithm is known that colours arbitrary

*Corresponding author.

SSDI 0166-218X(92)00036-N

graphs with a number or colours that lies within a constant factor of the optimum. We show that the graphs in some of the classes related to periodic assignment are less difficult to colour.

The organization of the paper is as follows. In Section 2 we introduce some basic concepts and notation, discuss periodic assignment and show its relation to the problem of colouring interval graphs, circular-arc graphs, and periodic-interval graphs. In Section 3 we consider the computational complexity of colouring these graphs and discuss appropriate graph colouring algorithms.

2. Periodic assignment

A periodic operation is an operation that is repeatedly executed at a constant rate over an infinite-time horizon. Its executions are considered to be nonpreemptive. Hence, a periodic operation can be viewed as an infinite sequence of executions of identical length that are equally spaced in time. A periodic operation o has an *execution time* $e(o) \in \mathbb{N}$, denoting the length of each execution, and a *period* $p(o) \in \mathbb{N}$, denoting the time between the start times of two successive executions. We assume that $e(o) \leqslant p(o)$.

The executions of a periodic operation o are all uniquely determined in time[1] by a *reference time* $r(o) \in \{1, \ldots, p(o)\}$ that specifies the start time of the execution of o that starts in the interval $[1, p(o)]$. Note that $r(o)$ is well defined, since exactly one execution is started in $[1, p(o)]$. The executions of operation o are started at times $r(o) + kp(o)$, $k \in \mathbb{Z}$. For a given set of periodic operations $O = \{o_1, \ldots, o_n\}$, a schedule $S = (r(o_1), \ldots, r(o_n))$ determines the start times of all executions. A schedule S is periodic with period $P = \operatorname{lcm}(p(o), \ldots, p(o_n))$, which means that P is the smallest positive number such that for each time $t \in \mathbb{Z}$ and each operation $o \in O$, we have that operation o is executed at t if and only if it is executed at $t + P$. The periodic assignment problem is now defined as follows.

Definition 2.1. Given a set of periodic operations O and a corresponding schedule S, the periodic assignment problem is the problem of assigning the executions of the operations in O to a minimum number of identical processors, where a processor can execute only one operation at a time.

Given a periodic schedule S with period P, we define the *thickness function* $T: [1, P] \to \mathbb{N}$ as the function that assigns to each time $t \in [1, P]$ the number of operations that are executed simultaneously at that time. Since a processor can execute only one operation at a time, the maximum thickness, defined by $T^{\max} = \max_t T(t)$, gives a lower bound on the number of processors that is required to execute a given schedule S.

[1] In this paper time is given in time units. If an operation o with execution time $e(o)$ starts at time t, then it is started at the beginning of time unit t and is completed at the end of time unit $t + e(o) - 1$. Similarly, a time interval $[t_1, t_2]$ denotes a set of consecutive time units, given by $\{t_1, t_1 + 1, \ldots, t_2\}$.

With respect to the assignment of executions to processors we consider two different cases, namely

- *unconstrained periodic assignment*, where different executions of an operation may be assigned to different processors, and
- *constrained periodic assignment*, where all executions of an operation have to be assigned to the same processor.

An assignment is called *periodic with period P'* if P' is the smallest positive integer such that for each time unit $t \in \mathbb{Z}$, each $o \in O$, and each processor m we have

processor m executes operation o at t if and only if m executes o at $t + P'$.

If for a given periodic schedule S with period P an assignment is periodic with period P', then necessarily $P|P'$. For the constrained periodic assignment problem, an assignment is necessarily periodic with a period equal to $\operatorname{lcm}(p(o_1),\dots,p(o_n))$.

In the following sections we consider in more detail unconstrained and constrained periodic assignment.

2.1. Unconstrained periodic assignment

Before discussing unconstrained periodic assignment, let us first consider as a simple example the assignment problem for a finite set of executions. The problem then amounts to assigning the finite set of executions, with given start and execution times, to a minimum number of processors. This problem can directly be formulated as the problem of colouring the vertices of an interval graph with a minimum number of colours, by associating with each execution a vertex in the corresponding interval graph such that two vertices are adjacent if and only if the corresponding executions overlap in time.

Definition 2.2. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an interval graph if we can associate with each vertex $v_i \in \mathcal{V}$ an interval $[l_i, r_i]$, with $l_i, r_i \in \mathbb{Z}$ and $l_i \leqslant r_i$, such that $\{v_i, v_j\} \in \mathcal{E}$ if and only if the corresponding intervals $[l_i, r_i]$, and $[l_j, r_j]$, overlap.

An example of a set of executions and the associated interval graph is given in Fig. 1. The set of all interval graphs is denoted by $\mathcal{S}_{\text{IG}}$.

The problem of colouring interval graphs is discussed in Section 3.1. Here, we restrict ourselves to showing that the finite set of executions can be assigned to $T^{\max}$

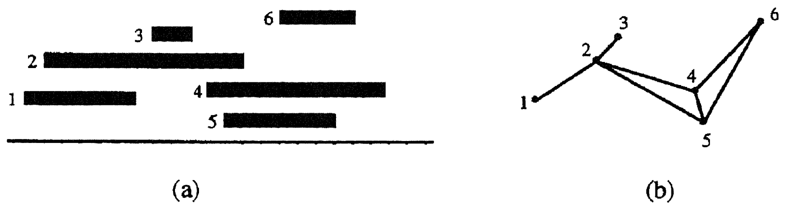

Fig. 1. (a) A set of execution intervals and (b) the associated interval graph. The vertices are adjacent if and only if the associated intervals overlap.

processors, where the maximum thickness $T^{\max}$ is defined in a similar way as for periodic schedules. To that end, we use the *left edge algorithm* [12] and show that this algorithm uses $T^{\max}$ processors to assign the executions. The left edge algorithm first sorts the executions in order of nondecreasing start times and then assigns the executions in this order to the first available processor, i.e. the processor with the smallest index that is idle at the start time of the execution. Now, it is easy to show, by contradiction, that the left edge algorithm uses exactly $T^{\max}$ processors. Suppose that the left edge algorithm uses $T^{\max} + 1$ processors. Then at some point in time an execution is assigned to the $(T^{\max} + 1)$th processor. But this implies that $T^{\max}$ other executions are carried out at that time, which contradicts the assumption that $T^{\max}$ gives the maximum thickness. Consequently, the left edge algorithm assigns a finite set of executions to exactly $T^{\max}$ processors.

Using this result for a finite set of executions, we formulate the following theorem for unconstrained periodic assignment.

Theorem 2.3. *For a given set of periodic operations $O = \{o_1, \ldots, o_n\}$ and a corresponding schedule S, an unconstrained assignment of the executions of O exists that uses $T^{\max}$ processors.*

Proof. Let the left edge algorithm be used to assign the executions, starting at time 0. Clearly, from the above result for a finite set of executions, we deduce that the left edge algorithm uses $T^{\max}$ processors. It remains to be shown that the assignment obtained by the left edge algorithm becomes periodic. The schedule S is periodic with period $P = \mathrm{lcm}(p(o_1), \ldots, p(o_n))$. Now consider the time intervals $[1 + lP, (l + 1)P]$, $l = 0, 1, \ldots$. In each of these intervals the left edge algorithm assigns a finite number of executions to a finite number of processors. Hence, only a finite number of different assignments exist for such intervals. Consequently, the assignment obtained by the left edge algorithm necessarily becomes periodic with a period lP, with $l \in \mathbb{Z}$, after some time t_0. More precisely, for some time $t_0 > 0$ we have that, for each time $t > t_0$, each $o \in O$ and each processor m, processor m executes operation o at time t if and only if m executes o at $t + lP$. The part of this assignment between t_0 and $t_0 + lP$ can clearly be used to construct a periodic assignment with period lP, using only $T^{\max}$ processors. $\square$

Note that the number of executions for which the left edge algorithm has to specify a processor need not be a polynomial in the number of operations. In fact, we can prove the following result.

Theorem 2.4. *The problem of determining the minimum number of processors for an unconstrained periodic assignment is* NP*-hard in the strong sense.*

Proof. This is shown by a reduction from the simultaneous congruences problem, which has been shown NP-hard by Leung and Whitehead [19] and NP-hard in the strong sense by Baruah et al. [1]. The simultaneous congruences problem is defined as follows. Given a set T of n ordered pairs of positive integers $(a_1, b_1), \ldots, (a_n, b_n)$, determine the cardinality of largest subset $T' \subseteq T$ for which there is a positive integer x with the property that $x \equiv a_i (\mathrm{mod}\, b_i)$ for all $(a_i, b_i) \in T'$.

Now, with each instance of the simultaneous congruences problem, as defined above, we can associate an instance of our problem, such that the cardinality of the largest subset T' equals k if and only if the minimum number of processors equals k. With each pair $(a_i, b_i) \in T$ we associate a periodic operation o_i with period $p(o_i) = b_i$ and execution time $e(o_i) = 1$, The corresponding schedule S is given by $S = (a'_1, \ldots, a'_n)$, where $a'_i \in \{1, \ldots, b_i\}$ and $a'_i \equiv a_i (\mathrm{mod}\, b_i)$. It is now easy to see that an integer x with the property that $x \equiv a_i (\mathrm{mod}\, b_i)$ for all $(a_i, b_i) \in T'$ corresponds to a time t where the corresponding operations are executed simultaneously. □

Using the left edge algorithm we can determine the minimum number of processors in a time that is polynomial in P' and n.

2.2. *Constrained periodic assignment*

If all executions of a periodic operation have to be assigned to the same processor, then an assignment is fully determined if for each periodic operation the processor on which it is repeatedly executed is specified. A periodic operation o_i with period $p(o_i)$, execution time $e(o_i)$ and reference time $r(o_i)$, requires an infinite set of time intervals during which it has to be executed. This set is given by $\{[r(o_i) + lp(o_i), r(o_i) + lp(o_i) + e(o_i) - 1] \mid l \in \mathbb{Z}\}$. Such an infinite set of intervals is called a *periodic interval* and is denoted by the 3-tuple $(p(o_i), e(o_i), r(o_i))$, with $0 < e(o_i), r(o_i) \leqslant p(o_i)$.

Let us first consider the special case where $p(o_i) = p$ for all $o_i \in O$. Clearly, in this case an assignment is periodic with period p. The periodic assignment problem can then be formulated as the problem of colouring a circular-arc graph with a minimum number of colours.

Definition 2.5. A graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is a circular-arc graph if it can be associated with a circle that is divided into a number of segments, numbered clockwise as $1, \ldots, n$, in such a way that each vertex $v_i \in \mathscr{V}$ can be associated with a circular arc $A_i = [l_i, r_i]$, with $l_i, r_i \in [1, \ldots, n]$, i.e. an arc on the circle that stretches clockwise from segment l_i to segment r_i, containing both l_i and r_i, and such that $\{v_i, v_j\} \in \mathscr{E}$ if and only if the corresponding arcs $[l_i, r_i]$ and $[l_j, r_j]$ overlap.

Fig. 2 gives an example of a set of periodic operations and an associated circular-arc graph. The set of all circular-arc graphs is denoted by $\mathscr{S}_{\mathrm{CAG}}$. In Section 3.2 we examine the problem of colouring circular-arc graphs in more detail.

If the operations in O can have arbitrary integral periods, then we can reformulate the periodic assignment problem as the problem of colouring a periodic-interval graph with a minimum number of colours.

Definition 2.6. A graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is a periodic-interval graph if one can associate with each vertex $v_i \in \mathscr{V}$, a periodic interval (p_i, e_i, r_i), with $p_i, e_i, r_i \in \mathbb{N}$ and $0 < e_i, r_i \leqslant p_i$, such that $\{v_i, v_j\} \in \mathscr{E}$ if and only if the corresponding periodic intervals

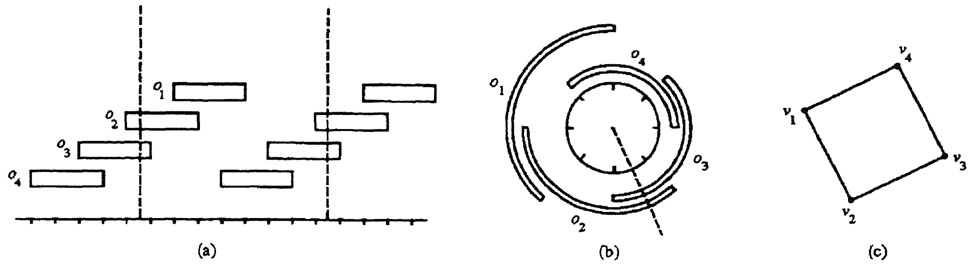

Fig. 2. (a) The executions of a set of operations with identical periods, (b) the associated set of circular arcs and (c) the associated circular-arc graph. Note that the graph is not an interval graph.

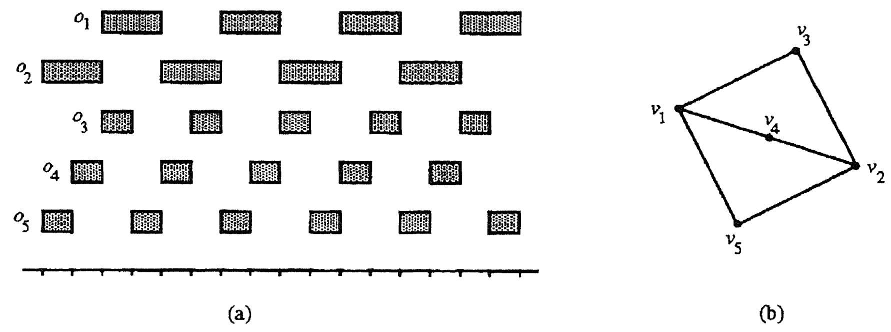

Fig. 3. (a) The executions of a set of periodic operations and (b) the associated periodic-interval graph. One can verify that this graph is not a circular-arc graph.

(p_i, e_i, r_i) and (p_j, e_j, r_j) overlap, i.e. if and only if there exist integers l, m for which

$$[r_i + lp_i, r_i + lp_i + e_i - 1] \cap [r_j + mp_j, r_j + mp_j + e_j - 1] \neq \emptyset .$$

Fig. 3 gives an example of a periodic-interval graph. The set of all periodic-interval graphs is denoted by $\mathscr{S}_{\text{PIG}}$. The following theorem gives a necessary and sufficient condition for the overlap of two periodic intervals.

Theorem 2.7. *Two periodic intervals (p_i, e_i, r_i) and (p_j, e_j, r_j), with $r_i \leqslant r_j$, do not overlap if and only if*

$$e_i \leqslant (r_j - r_i) \bmod g_{ij} \leqslant g_{ij} - e_j, \tag{1}$$

where $g_{ij} = \gcd(p_i, p_j)$.

Proof. Without loss of generality, we may assume that $r_i = 0$. The sufficiency of (1) is shown as follows. Let us consider time intervals $[kg_{ij}, (k+1)g_{ij} - 1]$, with $k \in \mathbb{Z}$. The first e_i time units of each of these intervals can be allocated to (p_i, e_i, r_i), and the remaining $g_{ij} - e_i$ time units to (p_j, e_j, r_j). Now, if (1) holds, then the allocated time units surely suffice to avoid overlap. The first e_i time units of the intervals are only used by (p_i, e_i, r_i) once every p_i/g_{ij} intervals. The remaining $g_{ij} - e_i$ time units are only (partly) used by (p_j, e_j, r_j) once every p_j/g_{ij} intervals.

The necessity of (1) is shown as follows. Let us again consider the time intervals $[kg_{ij}, (k+1)g_{ij} - 1]$, with $k \in \mathbb{Z}$. If (1) does not hold then (p_j, e_j, r_j) overlaps the first e_i time units once every p_j/g_{ij} time intervals. We have already seen that the first e_i time units of the intervals are used by (p_i, e_i, r_i) once every p_i/g_{ij} time units. Now, by definition, $\gcd(p_i/g_{ij}, p_j/g_{ij}) = 1$. Hence, if (1) does not hold, then (p_i, e_i, r_i) and (p_j, e_j, r_j) necessarily overlap. This completes the proof of the theorem. □

From Theorem 2.7 we immediately derive the following result.

Corollary 2.8. *Two periodic intervals (p_i, e_i, r_i) and (p_j, e_j, r_j), with $e_i = e_j = 1$, overlap if and only if*

$$r_j \equiv r_i (\operatorname{mod} g_{ij}).$$

We end this section with a few remarks. From the definitions of interval, circular-arc and periodic-interval graphs it is obvious that

$$\mathcal{S}_{\mathrm{IG}} \subset \mathcal{S}_{\mathrm{CAG}} \subset \mathcal{S}_{\mathrm{PIG}}.$$

Furthermore, the examples given in Figs. 2 and 3 show that the inclusions are strict.

Finally, we observe that these classes of graphs can all be considered as intersection graphs, i.e. for each of these graphs we can associate objects with the vertices such that vertices are adjacent if and only if the associated objects intersect or overlap. Intersection graphs can thus be represented in two different ways, either as a graph (i.e. as sets of vertices and edges) or as a collection of associated objects (intervals, circular arcs, periodic intervals). The latter representation is called the *intersection model*. In the following sections we will use both representations interchangeably, since both representations apply to periodic assignment. Furthermore, for the intersection graphs considered in this paper, a graph representation can be constructed from an intersection model in polynomial time. Using Theorem 2.7, it is easily seen that this holds for periodic-interval graphs and hence also for interval and circular-arc graphs. With respect to the inverse transformation, often called the recognition problem, we make the following remarks. Early results on characterizing interval graphs are given by Lekkerkerker and Boland [18], Gilmore and Hoffman [10] and Fulkerson and Gross [8]. Based on these characterizations, $O(n^3)$ recognition alogrithms can be constructed, with n the number of vertices. An $O(n + m)$ recognition algorithm is given by Booth and Lueker [4], with n the number of vertices and m the number of edges. A simpler $O(n + m)$ algorithm is given by Korte and Möhring [16]. Tucker [30] proved that also circular-arc graphs can be recognized in polynomial time. Periodic interval graphs can also be recognized in polynomical time, as is shown in Section 3.3.

3. Graph colouring

In this section we discuss colouring interval graphs, circular-arc graphs and periodic-interval graphs. Let us first summarize some results known for colouring arbitrary graphs. Graph colouring is defined as the problem of colouring the vertices of a graph with a minimum number of colours, such that adjacent vertices receive different colours [3]. The minimum number of colours necessary for colouring a graph G is called the chromatic number of G, which is denoted by $\chi(G)$. Graph colouring has been shown to be NP-hard [14], which implies that it is unlikely that there exists a polynomial-time algorithm that colours every graph with $\chi(G)$ colours. Furthermore, Garey and Johnson [7] showed that if a polynomial-time algorithm exists that colours any graph G with at most $a\chi(G) + b$ colours, with $a < 2$, then there also exists a polynomial-time algorithm that colours each graph G with $\chi(G)$ colours. Consequently, unless $\mathscr{P} = \mathscr{NP}$, no polynomial-time approximation algorithm exists that is guaranteed to use $a\chi(G) + b$ or less colours, with $a < 2$. Furthermore, no polynomial-time approximation algorithm is known that guarantees to colour each graph G with at most $a\chi(G) + b$ colours, for any fixed a and b, and there is evidence that such an algorithm does not exist [20]. The best-known performance ratio for a polynomial-time approximation algorithm is $O(n(\log\log n)^3/(\log n)^3)$, where n denotes the number of vertices [2]. Hence, graph colouring is not only difficult to solve to optimality, but also seems equally hard to solve to proximity within a constant factor of the optimum.

3.1. Colouring interval graphs

As we have already showed in Section 2.1, interval graphs can be optimally coloured in $O(n \log n)$ time by the left edge algorithm of Hashimoto and Stevens [12]. We showed that this algorithm uses $T^{\max}$ colours to colour the vertices of an interval graph.

Given a set of intervals $\{[l_i, r_i] \mid l_i \leqslant r_i, i = 1,\dots,n\}$ and a set of colours $\{c_1,\dots,c_n\}$, the algorithm can be restated as follows.

Left edge

(1) Sort the intervals in order of nondecreasing left end-point.

(2) Colour the intervals in this order by assigning to each interval $[l_i, r_i]$ the colour with the smallest index that has not been assigned to an interval overlapping $[l_i, r_i]$.

Gupta et al. [11] show that obtaining a minimum number of colours for interval graphs requires $O(n \log n)$ time, by relating it to the problem of determining whether n intervals are pairwise disjoint, for which an $\Omega(n \log n)$ lower bound is shown by Shamos and Hoey [26] and Fredman and Weide [5]. Hence, the time complexity of the left edge algorithm is optimal to within a constant factor.

3.2. Colouring circular-arc graphs

Garey et al. [8] showed that colouring circular-arc graphs is NP-hard. Furthermore, they showed that k-colourability, i.e. the problem of determining whether a circular-arc graph can be coloured with k or less colours, can be solved in $O(nk!k \log k)$ time. Thus, for fixed k this problem can be solved in polynomial time.

A circular-arc graph is said to be *proper* if none of the corresponding arcs is completely contained in another arc. Proper circular-arc graphs can be coloured with a minimum number of colours in polynomial time. Orlin et al. [24] gave an $O(n^2 \log n)$ algorithm which is based on the following observation. For proper circular-arc graphs, k-colourability can be transformed into a shortest path problem which can be solved in $O(n^2)$ time. Combining this with a binary search procedure results in an $O(n^2 \log n)$ algorithm. Successive improvements of this result are presented by Teng and Tucker [28] and Shih and Hsu [27], having $O(n^{3/2} \log n)$ and $O(n^{3/2})$ time complexities, respectively.

To the best of our knowledge, Tucker [29] is the only author that considered approximation algorithms for colouring circular-arc graphs. Here, we consider two approximation algorithms, viz.

(i) *Sequential Colouring*, a generally applicable graph colouring algorithm that was first proposed by Welsh and Powell [31], and

(ii) *Sort&Match*, an extension of an algorithm that was proposed by Tucker.

In the following sections we formulate both algorithms and examine their worst-case behaviour. To discuss the worst-case behaviour of an approximation algorithm A, we introduce the following notation. For a colouring algorithm A, let $A(\mathscr{G})$ be the maximum number of colours A might use when applied to graph $\mathscr{G}$. Then, the performance ratio $R_A(\mathscr{G}) = A(\mathscr{G})/\chi(\mathscr{G})$ gives an upper bound on the relative deviation from the optimum for $\mathscr{G}$.

Sequential colouring

Let a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ and a set of colours $\{c_1, \ldots, c_n\}$ be given. Then sequential colouring can be described as follows.

Sequential colouring (SC)

(1) Sort the vertices in $\mathscr{V}$ in order of nonincreasing degree. The degree $d(v_i)$ of a vertex $v_i \in \mathscr{V}$ gives the number of vertices to which v_i is adjacent.

(2) Colour the vertices in this order by assigning to each vertex v_i the colour with the smallest index that has not yet been assigned to a vertex that is adjacent to v_i.

For arbitrary graphs, SC can give results that are arbitrarily far from optimal, i.e., $R_{SC}(\mathscr{G})$ has no finite upper bound. Moreover, graphs exist for which the performance ratio $R_{SC}(\mathscr{G})$ increases linearly with $|\mathscr{V}|$. This can be seen from the following subset of instances. Let $\mathscr{G}_m = (\mathscr{V}_m, \mathscr{E}_m)$ with $\mathscr{V}_m = \{a_i, b_i | 1 \leqslant i \leqslant m\}$ and $\mathscr{E}_m = \{\{a_i, b_i\} | i \neq j\}$. Since all vertices have equal degree, they can be ordered arbitrarily in the first step. If the order of the vertices is $a_1, b_1, a_2, b_2, \ldots, a_m, b_m$, then $SC(\mathscr{G}_m) = m$, while $\chi(\mathscr{G}_m) = 2$. Fortunately, $\mathscr{G}_m$ is not a circular-arc graph, if $m > 3$.

Colouring circular-arc graphs with SC requires less than twice the minimum number of colours, as is shown by the following theorem.

Theorem 3.1. *For any circular-arc graph* $\mathscr{G}$, $R_{SC}(\mathscr{G}) < 2$.

Proof. The proof is by contradiction. For reasons of convenience, we use 'vertices' and 'circular arcs' interchangeably in this proof. Suppose that for some circular-arc graph

$\mathscr{G}$ sequential colouring requires m colours, with $m \geqslant 2\chi(\mathscr{G})$. Clearly, in that case some arc a_i receives colour C_m and must consequently be adjacent to at least $m - 1$ other arcs, which receive a colour from $\{c_1, \ldots, c_{m-1}\}$ prior to arc a_i. Let this subset of neighbours of a_i be denoted by $N(a_i)$. Clearly, $d(a_j) \geqslant d(a_i)$ for each $a_j \in N(a_i)$. Now we consider the following two cases.

– *None of the arcs in $N(a_i)$ are completely contained in a_i.* Then each of the arcs in $N(a_i)$ covers at least one of the end-points of a_i. Hence, one of the end-points is covered by at least $\lceil (m-1)/2 \rceil$ arcs. Since $\lceil (m-1)/2 \rceil \geqslant \chi(\mathscr{G})$, this results in a thickness of at least $\chi(\mathscr{G}) + 1$. However, this contradicts the fact that $\chi(\mathscr{G})$ at least equals the maximum thickness.

– *One or more arcs in $N(a_i)$ are completely contained in a_i.* Then there is at least one of these arcs, say arc a_j, such that none of the other arcs in $N(a_i)$ are completely contained in a_j. Since a_j is completely contained in a_i and $d(a_j) \geqslant d(a_i)$, we conclude that $d(a_j) = d(a_i)$. Consequently, this implies that all arcs that overlap with a_i also overlap with a_j, and vice versa. Hence, one of the end-points of a_j is covered by at least $\lceil (m-1)/2 \rceil$ arcs. Again, this leads to a contradiction with the fact that $\chi(\mathscr{G})$ at least equals the maximum thickness.

Hence, for both cases we have derived a contradiction, which completes the proof of the theorem. □

We next show that the worst-case performance bound given by Theorem 3.1 is tight. To that end, we first give the following lemma.

Lemma 3.2. *For all $m \in \mathbb{N}$, $\gcd(m^2, 2m - 1) = 1$.*

Proof. Let $a = \gcd(m^2, 2m - 1)$. Suppose $a > 1$. Now, if $a \mid m^2$ and $a \mid (2m - 1)$, then also for any prime factor π of a, $\pi \mid m^2$ and $\pi \mid (2m - 1)$. However, for any prime number π, if $\pi \mid m^2$ then $\pi \mid m$ and if $\pi \mid m$ then $\pi \nmid (2m - 1)$. Consequently, a cannot be greater than 1. □

Using Lemma 3.2 we can now prove the following theorem.

Theorem 3.3. *For any $\varepsilon > 0$, a circular-arc graph $\mathscr{G}$ exists such that $R_{SC}(\mathscr{G}) > 2 - \varepsilon$.*

Proof. This follows directly from the set of instances defined below. Let $\mathscr{S}_m$ be a set of m^2 arcs on a circle with $2m^2$ segments numbered $0, \ldots, 2m^2 - 1$, with m odd and $m \geqslant 3$. The arcs are defined by

$$[(4lm - 2l) \bmod 2m^2, (4lm - 2l + 2m - 1) \bmod 2m^2], \quad l = 0, 1, \ldots, m^2 - 1.$$

All of these arcs have different left end-points, since $\gcd(m^2, 2m - 1) = 1$, as stated in Lemma 3.2. All arcs overlap with $2m - 2$ other arcs. Consequently, all vertices in the corresponding circular-arc graph have the same degree and the arcs are thus coloured in an arbitrary order by SC. If the arcs are coloured in the order as given above, then SC requires $2m - 1$ colours. With each new colour SC colours at most $\frac{1}{2}(m + 1)$ arcs. However, an optimal colouring requires only m colours. Hence, choosing $m > 1/\varepsilon$

results in a graph that has the required property. This completes the proof of the theorem. □

Sort&Match

Elaborating on the work of Tucker [29], we present a two-step approximation algorithm for colouring circular-arc graphs, called *Sort&Match*. For reasons of simplicity, the algorithm is formulated in terms of colouring circular arcs instead of vertices.

Sort&Match (S&M)

(1) Determine on the circle a point t with minimum thickness $T_{\min}$. Partition the set of arcs into two subsets $\mathscr{A}$ and $\mathscr{B}$, where $\mathscr{A}$ is the set of arcs that cover point t. Hence, $|\mathscr{A}| = T^{\min}$. Now, the arcs in $\mathscr{B}$ define an interval graph. Consequently, the arcs in $\mathscr{B}$ can be coloured, using the left edge algorithm, with $T^{\max}$ colours.

(2) Determine a maximum subset $\mathscr{A}' \subseteq \mathscr{A}$, whose arcs can be coloured with a colour that has already been used in step 1. This problem can be formulated as a maximum-cardinality matching problem in a bipartite graph $\mathscr{G} = (\mathscr{V}_1, \mathscr{V}_2, \mathscr{E})$. Each vertex $v \in \mathscr{V}_1$ is associated with an arc in $\mathscr{A}$ and each vertex $u \in \mathscr{V}_2$ is associated with a colour that is used in step 1. An edge $\{v, u\}$ is in $\mathscr{E}$ if the arc associated with v can be given the colour associated with u. This matching problem can be solved efficiently using an augmenting path algorithm [17, 13]. Finally, each remaining arc in $\mathscr{A} - \mathscr{A}'$ is given a different free colour.

The following theorem states the worst-case performance of S&M.

Theorem 3.4. *For any circular-arc graph* $\mathscr{G}$, $R_{\mathrm{S\&M}}(\mathscr{G}) \leqslant 2$.

Proof. Since the arcs in subset $\mathscr{B}$ are coloured with $T^{\max}$ colours and the arcs in subset $\mathscr{A}$ are coloured using at most $T^{\min}$ colours, we obtain that $\mathrm{S\&M}(\mathscr{G}) \leqslant T^{\max} + T^{\min} \leqslant 2T^{\max}$. Combining this result with the fact that $\chi(\mathscr{G}) \geqslant T^{\max}$, we obtain the theorem. □

Tucker [29] only considers the first step of the algorithm presented above, but essentially proves the same worst-case performance bound. We can again prove that this bound is tight.

Theorem 3.5. *For any* $\varepsilon > 0$, *a circular-arc graph* $\mathscr{G}$ *exists such that* $R_{S\&M}(\mathscr{G}) > 2 - \varepsilon$.

Proof. This is directly derived from the following subset of instances. Let C_m be a set of $3m - 3$ arcs on a circle with $6m$ segments, $m \geqslant 4$, defined by

$$[2l, 2m + 2l - 1], \qquad l = 0, \ldots, m - 1,$$

$$[2m + 2l + 2, 4m + 2l + 1], \quad l = 0, \ldots, m - 1,$$

$$[4m + 2l + 2, 2l + 1], \qquad l = 0, \ldots, m - 4.$$

Applying S&M to C_m results in a colouring with $2m - 3$ colours, while the minimum number of colours is m. Hence, by choosing $m > 3/\varepsilon$, we obtain a graph with the required property. This completes the proof of the theorem. □

Note that applying SC to the above set of arcs may also result in a colouring with $2m - 3$ colours. Hence, choosing the best result of both S&M and SC does not improve the worst-case performance ratio of 2.

Experimental results

In this section we present some experimental results that give an indication of the performance of S&M, SC and min(S&M, SC). The results are obtained by applying the algorithms to randomly generated instances. Each instance contains 100 arcs on a circle with a circumference equal to 1. The left end-point of an arc is chosen uniformly from the interval $[0, 1)$ and the length of an arc is chosen uniformly from the interval $[l_{\min}, l_{\max})$, with $0 \leqslant l_{\min} \leqslant l_{\max} \leqslant 1$. For different choices of $l_{\min}$ and $l_{\max}$, Table 1 gives the average relative difference and corresponding standard deviation for S&M, SC and min(S&M, SC). The average relative difference is defined as the average difference between the number of colours found by the algorithms and the lower bound given by the cardinality of a maximum clique. The differences are expressed as a percentage of this lower bound. The results are compared with this

Table 1
Results obtained by applying S&M, SC and min(S&M, SC) to randomly generated circular-arc graphs

$l_{\min}$	$l_{\max}$									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.0	0.0 0.0	0.6 2.6	1.2 3.2	2.1 3.7	4.3 5.5	6.1 5.7	9.3 4.9	10.3 4.8	8.0 4.0	5.9 3.0
	2.5 4.3	2.8 3.7	2.7 2.9	3.5 2.5	3.5 2.7	3.6 2.5	2.7 2.0	1.9 1.6	1.3 1.4	0.7 1.0
	0.0 0.0	0.3 1.4	0.6 1.7	1.0 1.9	2.0 2.0	2.8 2.6	2.5 1.9	1.8 1.6	1.2 1.3	0.6 1.0
0.1		0.8 2.5	2.5 5.3	4.2 5.6	4.9 5.6	7.3 5.9	10.1 6.0	9.6 4.3	6.4 3.5	5.1 3.2
		4.9 4.0	6.3 3.7	5.6 3.4	5.6 3.3	4.2 2.5	3.0 2.1	1.4 1.6	0.8 1.1	0.6 0.8
		0.6 1.8	1.7 3.3	2.9 4.0	3.4 3.9	3.4 2.7	2.9 2.2	1.4 1.6	0.8 1.1	0.5 0.8
0.2			6.0 7.7	4.7 6.7	8.3 7.4	7.3 5.5	8.7 5.0	6.5 3.6	4.4 2.8	3.5 2.4
			5.4 3.9	7.7 4.9	6.8 3.8	3.6 2.2	1.4 1.4	0.8 1.1	0.5 0.8	0.4 0.7
			3.6 4.1	3.8 5.0	5.4 4.7	3.2 2.4	1.2 1.3	0.8 1.1	0.5 0.8	0.4 0.7
0.3				27.0 13.6	10.2 8.5	5.6 4.3	4.0 3.0	2.4 1.8	1.8 1.6	1.4 1.2
				16.6 7.4	5.6 3.4	1.3 1.4	0.5 0.9	0.2 0.4	0.1 0.4	0.1 0.4
				16.3 7.8	4.5 4.0	1.2 1.4	0.5 0.9	0.1 0.4	0.1 0.4	0.1 0.4
0.4					1.1 2.1	1.2 1.3	0.4 0.7	0.2 0.5	0.2 0.5	0.0 0.4
					1.8 1.8	0.2 0.5	0.0 02	0.0 0.2	0.0 0.1	0.0 0.0
					0.6 1.1	0.1 0.4	0.0 0.1	0.0 0.1	0.0 0.0	0.0 0.0

Each entry in the table gives the average relative difference and corresponding standard deviation for S&M, SC and min(S&M, SC), respectively. The results of each entry are obtained by applying the algorithms to 100 instances.

lower bound, since the minimum number of colours is unknown and determining it is considered too time consuming. The cardinality of a maximum clique clearly gives a lower bound on the minimum number of colours. Although determining the cardinality of a maximum clique is NP-hard for arbitrary graphs, it can be obtained in polynomial time for circular-arc graphs, by iteratively constructing a maximum matching in bipartite graphs [9].

Comparing both algorithms, we see that, on average, S&M outperforms SC if the arc lengths are small. However, for larger arc lengths SC produces better average results than S&M. This motivates the interest in the best result of both algorithms. From Table 1 we observe that the average relative difference of min(S&M, SC) remains almost always within 5% of the optimum.

An important exception is given by instances with arc lengths chosen from [0.3, 0.4). Both S&M and SC seem to perform less well for these instances—they give average relative differences of 27.0% and 16.6%, respectively. One might assume that these large differences are caused by the fact that for these instances the cardinality of a maximum clique is a bad approximation of the chromatic number. However, experimental results contradict this assumption. We have optimally coloured 50 instances from this class each containing 30 arcs. For these instances the chromatic number deviated, on average, only 1% from the cardinality of the maximum clique.

In addition to the information in Table 1, we mention that, except if arc lengths are chosen from [0.3, 0.4), the observed maximum relative difference for S&M, SC and min(S&M, SC) is 33.3%, 21.2% and 18.4%, respectively. If the arc lengths are chosen from [0.3, 0.4), then the observed maximum relative difference for S&M, SC and min(S&M, SC) is 56.4%, 32.5% and 32.5%, respectively.

From the experimental results presented in this section, we conclude that the average-case performance of S&M and SC is usually much better than the worst-case bounds given in the previous sections. Furthermore, we conclude that the both algorithms perform less well if the lengths of the circular arcs are approximately one-third of the circumference of the circle.

3.3 Colouring periodic-interval graphs

Colouring periodic-interval graphs is NP-hard. This follows immediately from the fact that colouring circular-arc graphs is NP-hard and the observation that each circular-arc graph is a periodic-interval graph. The next theorem gives a somewhat surprising result.

Theorem 3.6. *Each graph is a periodic-interval graph.*

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary graph, with loops. We show that we can associate with each $v_i \in \mathcal{V}$ a periodic interval (p_i, e_i, r_i), such that, for each pair of distinct vertices v_i, $v_j \in \mathcal{V}$, $\{v_i, v_j\} \in \mathcal{E}$ if and only if the associated periodic intervals (p_i, e_i, r_i) and (p_j, e_j, r_j) overlap.

Let us first select the smallest $\frac{1}{2}n(n-1)$ distinct prime numbers larger than n, and let these prime numbers be denoted by π_{ij} for $1 \leqslant i < j \leqslant n$. Furthermore, let $\pi_{ji} = \pi_{ij}$ for $i < j$. By the prime number theorem, the magnitude of the largest of these numbers is $O(n^2 \log n)$. We can now associate a periodic interval (p_i, e_i, r_i) with each $v_i \in \mathscr{V}$ according to $e_i = 1$ and $p_i = \Pi_{j \neq i} \pi_{ij}$. *The reference time* r_i is chosen such that for each $j \neq i$, $r_i \equiv 1 \pmod{\pi_{ij}}$ if $\{v_i, v_j\} \in \mathscr{E}$, and $r_i \equiv i \pmod{\pi_{ij}}$ if $\{v_i, v_j\} \notin \mathscr{E}$. Reference times that satisfy these constraints always exist and they can be determined in polynomial time by using the Chinese remainder theorem (see, e.g. [22]).

Now, if $\{v_i, v_j\} \notin \mathscr{E}$ for two distinct vertices $v_i, v_j \in \mathscr{V}$, then (p_i, e_i, r_i) and (p_j, e_j, r_j) do not overlap. This can be seen as follows. Since $r_i \equiv i \pmod{\pi_{ij}}$, $r_j \equiv j \pmod{\pi_{ij}}$, and $0 < |i - j| < n < \pi_{ij}$, we derive that r_i and r_j are not congruent modulo π_{ij}. Furthermore, $g_{ij} = \gcd(p_i, p_j) = \pi_{ij}$. Hence, using Corollary 2.8, we obtain that (p_i, e_i, r_i) and (p_j, e_j, r_j) do not overlap.

Furthermore, if $\{v_i, v_j\} \in \mathscr{E}$, then (p_i, e_i, r_i) and (p_j, e_j, r_j) can be shown to overlap. Since $r_i \equiv 1 \pmod{\pi_{ij}}$ and $r_j \equiv 1 \pmod{\pi_{ij}}$, we derive that r_i and r_j are congruent modulo π_{ij}. Hence, again using Corollary 2.8, we obtain that (p_i, e_i, r_i) and (p_j, e_j, r_j) overlap. This completes the proof of the theorem. □

Note that the above construction is polynomial. In Section 1 we stated that no polynomial-time approximation algorithm is known that colours arbitrary graphs within a constant factor of the optimum and that there is evidence that no such algorithm exists. Theorem 3.6 implies that the same holds for periodic-interval graphs.

Acknowledgement

The authors are grateful to J.B. Orlin for his valuable comments and, in particular, for suggesting the polynomial-time construction in the proof of Theorem 3.6.

References

[1] S.K. Baruah, R.R. Howell and L.E. Rosier, On preemptive scheduling of periodic, real-time tasks on one processor, in: B. Rovan ed., Mathematical Foundations of Computer Science 1990 (Springer, Berlin, 1990) 173–179.
[2] B. Berger and J. Rompel, A better performance guarantee for approximate graph colouring, Algorithmica 5 (1990) 459–466.
[3] J.A. Bondy and U.S.R. Murthy, Graph Theory with Application (Macmillan, New York, 1976).
[4] S. Booth and S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, J. Comput. System Sci. 13 (1976) 335–379.
[5] M.L. Fredman and B. Weide, On the complexity of computing the measure of $U[a_i, b_i]$, Comm. ACM 21 (1978) 540–544.
[6] D.R. Fulkerson and O.A. Gross, Incidence matrices and interval graphs, Pacific J. Math. 15 (1965) 835–855.
[7] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1979).
[8] M.R. Garey, D.S. Johnson, G.L. Miller and C.H. Papadimitriou, The complexity of coloring circular arcs and chords, SIAM J. Algebraic Discrete Methods 1 (1980) 216–227.
[9] F. Gavril, Algorithms on circular-arc graphs, Networks 4 (1974) 357–369.
[10] P.C. Gilmore and A.J. Hoffman, A characterization of comparability graphs and of interval graphs, Canad. J. Math. 16 (1964) 539–548.

[11] U.I. Gupta, D.T. Lee, and J.Y.-T. Leung, An optimal solution for the channel-assignment problem, IEEE Trans. Comput. 28 (1979) 807–810.
[12] A. Hashimoto and J. Stevens, Wire routing by optimizing channel assignment with large apertures, in: Proceedings of the 8th Design Automation Conference (1971) 155–169.
[13] J.E. Hopcroft and R.M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2 (1973) 225–231.
[14] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York, 1972) 85–103.
[15] J.H.M. Korst, E.H.L. Aarts, J.K. Lenstra and J. Wessels, Periodic multiprocessor scheduling, in: Proceedings of the Conference on Parallel Architectures and Languages Europe, PARLE '91, Lecture Notes in Computer Science, Vol. 505 (Springer, Berlin, 1991) 166–178.
[16] N. Korte and R.H. Möhring, An incremental linear-time algorithm for recognizing interval graphs, SIAM J. Comput. 18 (1989) 68–81.
[17] E.L. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart & Winston, New York, 1976).
[18] C.G. Lekkerkerker and J.Ch. Boland, Representation of a finite graph by a set of intervals on the real line, Fund. Math. 51 (1962) 45–64.
[19] J.Y.-T. Leung and J. Whitehead, On the complexity of fixed-priority scheduling of periodic, real-time tasks, Performance Evaluation 2 (1982) 237–250.
[20] L. Linial and U. Vazirani, Graph products and chromatic numbers, in: Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (1989) 124–128.
[21] C.L. Liu and J.W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, J. ACM 20 (1973) 46–61.
[22] I. Niven and H.S. Zuckerman, An Introduction to the Theory of Numbers (Wiley, New York, 1960).
[23] J.B. Orlin, Minimizing the number of vehicles to meet a fixed periodic schedule: an application of periodic posets, Oper. Res. 30 (1982) 760–776.
[24] J.B. Orlin, M. Bonuccelli and D. Bovet, An $O(n^2)$ algorithm for coloring proper circular arc graphs, SIAM J. Algebraic Discrete Methods 2 (1981) 88–93.
[25] K.S. Park and D.K. Yun, Optimal scheduling of periodic activities, Oper. Res. 33 (1985) 690–695.
[26] M.I. Shamos and D. Hoey, Geometric intersections problems, in: Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science (1976) 208–215.
[27] W.-K. Shih and W.-L. Hsu, An $O(n^{1.5})$ algorithm to color proper circular arcs, Discrete Appl. Math. 25 (1989) 321–323.
[28] A. Teng and A. Tucker, An $O(qn)$ algorithm to q-color a proper family of circular arcs, Discrete Math. 55 (1985) 233–243.
[29] A. Tucker, Coloring a family of circular arcs, SIAM J. Appl. Math. 29 (1975) 493–552.
[30] A. Tucker, An efficient test for circular-arc graphs, SIAM J. Comput. 9 (1990) 1–24.
[31] D.J.A. Welsh and M.B. Powell, An upper bound on the chromatic number of a graph and its application to timetabling problems, Comput. J. 10 (1967) 85–87.