Discrete Applied Mathematics 158 (2010) 349–354

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Solving a multicoloring problem with overlaps using integer programming[☆]

Isabel Méndez-Díaz[*], Paula Zabala

Departamento de Computación, FCEyN - Universidad de Buenos Aires, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

ARTICLE INFO

Article history:
Received 29 October 2007
Received in revised form 12 March 2009
Accepted 29 May 2009
Available online 30 June 2009

Keywords:
Graph multicoloring
Integer programming
Branch-and-Cut

ABSTRACT

This paper presents a new generalization of the graph multicoloring problem. We propose a Branch-and-Cut algorithm based on a new integer programming formulation. The cuts used are valid inequalities that we could identify to the polytope associated with the model. The Branch-and-Cut system includes separation heuristics for the valid inequalities, specific initial and primal heuristics, branching and pruning rules. We report on computational experience with random instances.

1. Introduction

In many real-life situations, resources need to be shared among users with conflicting requirements. Graph coloring and its generalizations [6] are useful tools in modelling a wide variety of these problems [1,11].

Generally, resources must be assigned in such a way that a resource cannot be shared by two conflicting users and the optimization goal is to minimize the number of resources needed to satisfy the demands. When all the requests of the users are equal to 1, this is the well known coloring problem [5]. If a positive request greater than 1 is defined for each user, it corresponds to the multicoloring problem [3].

However, the number of available resources may not be enough and it could become necessary to relax the requirements and allow to share resources among conflicting users.

Consider a set V of n projects (papers) and a set R of referees. Each project v has to be evaluated by k_v referees chosen from R. Associated with each project v is a set $r_v \subseteq R$ of possible referees. If a pair of projects u, v have participants (authors) in common, it will be reasonable to restrict the referees' assignment to the ones not having referees in common. But if the number of available referees is not enough, we relax this constraint and allow to share at most c_{uv} referees. For organizational or economical reasons, we want to manage the minimum number of referees for the evaluation process.

The problem can be modeled by a conflict graph $G = (V, E)$, in which the vertices represent the projects, and an edge between two vertices u, v means that the set of referees assigned to the endpoints must have intersection size less than or equal to c_{uv}. We refer to this problem as a multicoloring problem with overlaps.

More formally, let there be given an undirected graph $G = (V, E)$, R a set of colors, $ncol = |R|$, k_v the demanded number of colors of node v, $r_v \subseteq R$ the set of feasible colors for v and c_{uv} the maximum number of colors that can be shared by adjacent vertices u, v.

We define a (k, c, r)-*coloring* of G as a mapping $f : V \to 2^R$ such that $f(v) \subseteq r_v$, $|f(v)| = k_v$ and $|f(u) \cap f(v)| \leq c_{uv}$. The graph (k, c, r)-*coloring* problem is to find a (k, c, r)-*coloring* using as few colors as possible.

☆ This research was partially supported by UBACYT Grant X143, PICT 920, PICT 1600 and PICT Raices 01070.

* Corresponding author. Tel.: +54 11 45763359; fax: +54 11 45763359.
E-mail addresses: imendez@dc.uba.ar (I. Méndez-Díaz), pzabala@dc.uba.ar (P. Zabala).

In case that we impose $c_{uv} = 0$ for all $uv \in E$ and $r_v = R$, for all $v \in V$, this problem can be formalized as the well known multicoloring problem and $c_{uv} = 0$ for all $uv \in E$ and $k_v = 1$, for all $v \in V$ corresponds to a list coloring problem.

Like most optimization problems on graphs, the (k, c, r)-*coloring* problem can be formulated as a linear integer programming problem. Approaches in this way were proposed for the coloring problem [1,4,7–9], multicoloring problem [10] and list coloring [2].

LP-based Branch-and-Cut algorithms are currently the most important tools for dealing with linear integer programming problems computationally. The basic structure is a Branch-and-Bound algorithm which may call a cutting plane algorithm at each node of the search tree. Therefore, the main elements of a Branch-and-Cut algorithm are the formulation, the separation procedures and the branching strategies, which we will describe in the next sections. We also include some additional features, such as heuristics based on rounding LP-solutions.

The structure of the remainder of the paper is as follows. In the next section, we give an integer programming formulation and present some valid inequalities. In Section 3 we describe the ingredients of our Branch-and-Cut algorithm. Computational results are given in Section 4. The paper closes with final remarks.

2. Integer programming formulation and valid inequalities

We consider binary variables x_{vj}, with $v \in V$ and $j \in r_v$, where $x_{vj} = 1$ if color j is assigned to vertex v and $x_{vj} = 0$ otherwise. We also define *ncol* binary variables w_j $(1 \le j \le ncol)$, that indicate whether color j is used in some vertex, i.e. $w_j = 1$ if $x_{vj} = 1$ for some vertex v. Finally, binary variables y_{uvj}, with $uv \in E, j \in r_u \cap r_v$, where $y_{uvj} = 1$ if color j is assigned to both endpoints u, v. The (k, c, r)-*coloring* problem can be formulated as:

$$\text{Min} \sum_{j=1}^{ncol} w_j$$

subject to

$$\sum_{j \in r_v} x_{vj} = k_v \qquad \text{for all } v \in V \tag{1}$$

$$\sum_{j \in r_u \cap r_v} y_{uvj} \le c_{uv} \qquad \text{for all } uv \in E \tag{2}$$

$$x_{uj} + x_{vj} - y_{uvj} \le 1 \qquad \text{for all } uv \in E, j \in r_u \cap r_v \tag{3}$$

$$x_{vj} \le w_j \qquad \text{for all } v \in V, j \in r_v \tag{4}$$

$$x_{vj} \in \{0, 1\} \qquad \text{for all } v \in V, j \in r_v$$

$$y_{uvj} \in \{0, 1\} \qquad \text{for all } uv \in E, j \in r_u \cap r_v$$

$$w_j \in \{0, 1\} \qquad 1 \le j \le ncol$$

Constraints (1) assert that each vertex must receive exactly k_v colors, and constraints (2) say that every pair of adjacent vertices u, v may not share more than c_{uv} colors. Constraints (3) force $y_{uvj} = 1$ if color j is shared by the adjacent vertices u, v. If color j is assigned to some vertex, constraints (4) impose $w_j = 1$.

Our main focus is on developing a Branch-and-Cut algorithm that takes advantage of the particular structure and exploits the properties of the problem.

From now on, we restrict our attention to the particular case where $k_v = k$, $r_v = R$ for all $v \in V$ and $c_{uv} = c$ for all $uv \in E$. The complexity of the general case makes it very difficult to study the polytope. Despite this restriction, we find the problem interesting on its own and it is still hard.

In order to break some symmetry, we also add the constraints $w_j \ge w_{j+1}$ for all $1 \le j \le ncol - 1$ to impose that color $j + 1$ can be assigned to a vertex provided color j has already been assigned. These constraints imply that any feasible (k, c, r)-*coloring* uses the first colors.

As a result of our polyhedral investigations of the underlying polytope associated with this formulation, the inequalities that follow are proven to be valid for that polytope. The cuts are based on the idea of how many of the vertices of a subset $V' \subset V$ with a particular structure, such as cliques, can share $c + 1$ colors. This idea leads to the following inequalities.

Proposition 2.1. *Let $C = (V', E')$ be a clique of size $p \ge 2$ of G and $j_1 < j_2 < \cdots < j_{c+1} \in R$, then*

$$\sum_{i=1}^{c+1} \sum_{v \in V'} x_{vj_i} \le p \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}$$

is a valid inequality.

Proof. The proof is performed by induction on p.

The case $p = 2$ is trivial.

Assume that this is a valid inequality for $2 \le p \le s$. We have to prove that it is also valid for $p = s + 1$.

Let $C = (V', E')$, $|V'| = s + 1$ and $u \in V'$. We have to show that

$$\sum_{i=1}^{c+1}\sum_{v\in V'} x_{vj_i} = \sum_{i=1}^{c+1}\sum_{v\in V'\setminus\{u\}} x_{vj_i} + \sum_{i=1}^{c+1} x_{uj_i} \le s\sum_{i=1}^{c} w_{j_i} + \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}.$$

If $\sum_{i=1}^{c+1} x_{uj_i} \le c$, then $\sum_{i=1}^{c+1} x_{uj_i} \le \sum_{i=1}^{c} w_{j_i}$ and by the inductive hypothesis $\sum_{i=1}^{c+1}\sum_{v\in V'\setminus u} x_{vj_i} \le s\sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}$. The result follows.

If $\sum_{i=1}^{c+1} x_{uj_i} = c + 1$, then $\sum_{i=1}^{c+1} x_{uj_i} \le \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}$. Since $uv \in E'$ for all $v \in V' \setminus \{u\}$, this also implies that $\sum_{i=1}^{c+1} x_{vj_i} \le c = \sum_{i=1}^{c} w_{j_i}$. We conclude that the inequality is valid. ■

Proposition 2.2. *Let $C = (V', E')$ be a clique of size $p \ge 2$ of G and $j_1 < j_2 < \cdots < j_{c+1} \in R$, then*

$$p\sum_{v\in V'} x_{vj_1} - \sum_{uv\in E'} y_{uvj_1} + \sum_{i=2}^{c+1}\sum_{uv\in E'} y_{uvj_i} \le \left(\frac{cp(p-1)}{2} + p\right) w_{j_1}$$

is a valid inequality.

Proof. If $w_{j_1} = 0$ the inequality is trivial. Let us assume that $w_{j_1} = 1$. The proof is performed by induction on p.

Assume $p = 2$ and $V' = \{v_1, v_2\}$.

If $2(x_{v_1j_1} + x_{v_2j_1}) \le 2$ then $y_{v_1v_2j_1} = 0$.

Since $\sum_{i=2}^{c+1} y_{v_1v_2j_i} \le c$ and $(\frac{cp(p-1)}{2} + p)w_{j_1} = c + 2$, the inequality is valid.

If $2(x_{v_1j_1} + x_{v_2j_1}) = 4$ then $y_{v_1v_2j_1} = 1$. Since v_1, v_2 can share at most c colors, $\sum_{i=2}^{c+1} y_{v_1v_2j_i} \le c-1$. Then $4-1+c-1 = c+2$ and the inequality follows.

Assume that this is a valid inequality for $2 \le p \le s$. We have to prove that it is also valid for $p = s + 1$

Let $C = (V', E')$, $|V'| = s + 1$ and $w \in V'$. We have to prove that

$$\begin{aligned}(s+1)\sum_{v\in V'} x_{vj_1} - \sum_{uv\in E'} y_{uvj_1} + \sum_{i=2}^{c+1}\sum_{uv\in E'} y_{uvj_i} &= s\sum_{v\in V'\setminus\{w\}} x_{vj_1} - \sum_{\substack{uv\in E\\ u,v\ne w}} y_{uvj_1} + \sum_{i=2}^{c+1}\sum_{\substack{uv\in E'\\ u,v\ne w}} y_{uvj_i} + \sum_{v\in V'\setminus\{w\}} x_{vj_1}\\ &\quad + (s+1)x_{wj_1} - \sum_{v\in V'\setminus\{w\}} y_{vwj_1} + \sum_{i=2}^{c+1}\sum_{v\in V'\setminus\{w\}} y_{vwj_i}\\ &\le \left(\frac{c(s+1)s}{2} + (s+1)\right) w_{j_1} = \left(\frac{cs(s-1)}{2} + s\right) w_{j_1} + (cs+1)w_{j_1}.\end{aligned}$$

If $x_{wj_1} = 0$ then $\sum_{v\in V'\setminus\{w\}} y_{vwj_1} = 0$. If $\sum_{v\in V'\setminus\{w\}} x_{vj_1} = q$, then $\sum_{i=2}^{c+1}\sum_{v\in V'} y_{vwj_i} \le (c-1)q + c(s-q) = cs - q$. Now, applying the inductive hypothesis, the result follows.

Suppose that $x_{wj_1} = 1$ and $\sum_{v\in V'\setminus\{w\}} x_{vj_1} = q$ with $q \ge 0$. Then

$$\begin{aligned}& s\sum_{v\in V'\setminus\{w\}} x_{vj_1} - \sum_{\substack{uv\in E'\\ u,v\ne w}} y_{uvj_1} + \sum_{i=2}^{c+1}\sum_{\substack{uv\in E'\\ u,v\ne w}} y_{uvj_i} + \sum_{v\in V'\setminus\{w\}} x_{vj_1} + (s+1)x_{wj_1} - \sum_{v\in V'\setminus\{w\}} y_{vwj_1} + \sum_{i=2}^{c+1}\sum_{v\in V'\setminus\{w\}} y_{vwj_i}\\ &\le sq - \frac{q(q-1)}{2} + (c-1)\frac{q(q-1)}{2} + c\frac{(s-q)(s-q-1)}{2} + q + s + 1 - q + (c-1)q + c(s-q)\\ &= \frac{c(s+1)s}{2} + (s+1) + (c-1)(q^2 - sq).\end{aligned}$$

Since $q \le s$ and $c \ge 1$, then $(c-1)(q^2 - sq) \le 0$ and the result follows. ■

Proposition 2.3. *Let $C = (V', E')$ be a clique of size p of G and $j_1 < j_2 < \cdots < j_{c+1} \in R$, then*

$$p\sum_{v\in V'} x_{vj_{c+1}} - \sum_{uv\in E'} y_{uvj_{c+1}} + \sum_{i=1}^{c}\sum_{uv\in E'} y_{uvj_i} \le \frac{p(p-1)}{2}\sum_{i=1}^{c} w_{j_i} + pw_{j_{c+1}}.$$

Proof. Suppose that $\sum_{v\in V'} x_{vj_{c+1}} = 0$. We know that $\sum_{uv\in E'} y_{uvj_i} \le \frac{p(p-1)}{2} w_{j_i}$, hence $\sum_{i=1}^{c}\sum_{uv\in E'} y_{uvj_i} \le \frac{p(p-1)}{2}\sum_{i=1}^{c} w_{j_i}$, and the inequality is valid.

Suppose that $\sum_{v\in C} x_{vj_{c+1}} = q$. Then

$$p\sum_{v\in C} x_{vj_{c+1}} - \sum_{uv\in E'} y_{uvj_{c+1}} + \sum_{i=1}^{c}\sum_{uv\in E'} y_{uvj_i} = pq - \frac{q(q-1)}{2} + c\frac{(p-q)(p-q-1)}{2} + (c-1)\frac{q(q-1)}{2}$$
$$= \frac{p(p-1)}{2} + q.$$

Since $q \le p$, the inequality follows. ■

Finally, the next inequality is a transitive consequence of using color j_0 among three adjacent vertices.

Proposition 2.4. *Let* $\{u_1, u_2, u_3\}$ *be adjacent vertices and* $j_0 \in R$. *Then*

$$y_{u_1u_2j_0} + y_{u_2u_3j_0} \le x_{u_2j_0} + y_{u_1u_3j_0}$$

is a valid inequality.

3. Branch-and-Cut algorithm

Given an integer programming problem, the idea of a Branch-and-Cut is recursively to partition the solution set into subsets and solving the problem over each subset. This procedure generates an enumeration tree where offspring of a node correspond to the partition of the set associated with the parent node. In each node of the tree, a linear relaxation of the problem is considered by dropping integrality requirements and adding valid inequalities which cut off the fractional solution.

To reduce the number of nodes of the tree, it is important to have good lower and upper bounds, good rules to partition the feasible set, good strategies to search on the tree and a good strengthening of the linear relaxations.

In the following, we describe the different aspects we consider in our implementation.

3.1. Initial upper bound

An upper bound is obtained with a greedy heuristic which is run once at the beginning of the algorithm.

The procedure starts with an ordered color list $L = \{1, \ldots, k\}$. In every step, a vertex is chosen and it is assigned the first k colors not being incompatible with its neighbors from the list L. If there are not enough colors in L, new colors are added to it. After the assignment, the used colors are moved to the end of the list and the procedure goes to the next step.

The procedure runs two times by considering different vertex orderings to perform the assignment:

- The highest number of colors used by its neighbors. Ties break by the highest number of non-colored neighbors.
- The highest number of colors used by its neighbors. Ties break by the lowest number of colored neighbors.

3.2. Initial lower bound

The lower bound is initiated with a simple heuristic to find a clique K, as large as possible. Then, we apply a complete enumeration procedure to find an optimal (k, c, r)-*coloring* for K.

We do not preassign colors to the vertices of K, as it is usually done on classic coloring, because it could be not a feasible assignment for an optimal (k, c, r)-*coloring* of G.

The enumeration of feasible solutions for instances associated with cliques up to 10 vertices is very fast to carry out. However, it takes a prohibitive time for greater cliques if we consider that it is an initial phase of a Branch-and-Cut algorithm. Then, we impose this limit on the size of the clique.

3.3. Primal heuristic

The availability of good feasible solutions may reduce the size of the Branch-and-Cut tree significantly. During the procedure, we solve many linear problems and obtain fractional solutions. Every time we have a fractional solution, we try to get an integer solution from it by using a rounding procedure. Many times, this process improves the best feasible solution known so far. A simple procedure consists of choosing a vertex v and ordering the fractional variables x_{vk} in a nonincreasing order of values. We take the first variable on the list and round it up to 1 if the corresponding color is feasible and repeat the step while there are still colors to be assigned to v. The criteria we mentioned for the initial heuristic are used for choosing the vertices.

3.4. Variable selection and enumeration strategy

The choice of branching variable has a large effect on the performance of the algorithm. We tried several strategies and finally the pseudo-cost branching rule was shown to be the most successful.

Table 1
Computational results.

k-c		Low		Medium		High	
		Time	Nodes	Time	Nodes	Time	Nodes
2-1	B&C	0.06	1.26	1.81	9.11	2.1	5.74
	CPLEX	0.31	26.03	31.04	1098.89	2156.76	28878.8
3-1	B&C	1.01	7.13	16.71	38.86	559.06 (20)	519.31
	CPLEX	9.8 (29)	305.8	1258.8 (14)	15678.4	*****	*****
3-2	B&C	0.1	0.06	0.52	0.9	5.59	18.3
	CPLEX	0.45	32.7	13.53	185.5	107.35	4239.63
4-2	B&C	3.99	15.26	412.69 (27)	475.03	116.85 (13)	116.63
	CPLEX	65.46 (29)	1120.9	884.3 (26)	1744.1	1487.8 (1)	2934

The idea underlying the pseudo-cost branching rule is to determine a priority of the variables in terms of the change in the optimal objective value. We apply this strategy by prioritizing the w variables. On the left branch, the variable is fixed to 1, on the right branch the variable is fixed to 0. We use a best first search strategy in choosing the node to evaluate.

3.5. Cutting plane generation

In order to make the inequality separation, we developed an heuristic procedure for the first three inequalities.

For each color k_1, we consider the list of non-zero variables in decreasing order of the $x^*_{vk_1}$ values, where x^* denotes the current fractional solution.

We initialize a clique with vertex v for each $x^*_{vk_1}$ in the list. For each v, we do several trials bounded by an input parameter. In trial j, we choose the fractional variable $x_{v'k_1}$ such that vertex v' is the *j-th* adjacent vertex to v in the list. We add this vertex to the clique. Then, the clique is grown in size trying to add other adjacent vertices following the ordered list.

Once this is done, it remains to determine the other colors. We attempt all possibilities to find violated inequalities which are added to the LP relaxation.

The last inequalities are handled by brute-force.

4. Computational experiments

We report in this section on computational experiences with our Branch-and-Cut algorithm. The code was implemented in C++ using the CPLEX 10.1 LP solver.

We have performed the experiments on a SUN UltraSparc III workstation with a CPU running at 1 GHz and 2 GB of RAM memory. CPU times are reported in seconds. We impose a CPU time limit of 1 h. The algorithm is tested on random graphs $G(20, p)$ of 20 vertices and an edge between each pair of vertices with independent probability p. We use random graphs with low ($p \leq 30\%$), medium ($40\% \leq p \leq 60\%$) and high ($p \geq 70\%$) density.

Experiments were carried out over 30 instances per density considered and three different k, c values. We report the average CPU time and the average number of tree nodes explored. Asterisks indicate that the time limit was exceeded. When the average is taken over less than 30 instances because the others could not be solved within the time limits, we indicate the number of instances solved in brackets. For the root node of the enumeration tree 10 cutting plane rounds were implemented. For tree nodes other than the root, a limit of 2 cutting plane rounds were considered.

To appreciate the benefits of using our cutting planes and other specific aspects we have implemented, we compare our Branch-and-Cut algorithm with the general purpose IP-solver CPLEX. Tests were performed using all the advantages of preprocessing, cutting, etc. that CPLEX offers.

As one may appreciate from the results reported in Table 1, our strategies attain a better performance than CPLEX. CPLEX could not solve instances that could be managed by our Branch-and-Cut within the CPU time limit imposed.

In many cases, both algorithms find the optimal solution very early in the enumeration process, but CPLEX requires too much time to obtain the optimal certification. However, our cutting planes improve the lower bounds and this improvement allows one to fathom a large percentage of nodes earlier in the tree. The use of cutting planes seems to be essential for solving most of the instances.

For low density graphs, the gap between the initial lower and the initial upper bounds is very small. The increase of the lower bound applying cutting planes is achieved on the first levels of the tree, and it is not possible to find violated inequalities on depth nodes.

For medium and high density graphs, the initial lower bound increases during the first levels, and then the objective function does not change from one round to the next. However, the cutting planes are useful to fathom depth nodes for infeasibility. For all instances considered, the cutting plane phase finds violated cuts.

We can also see from the table that the results do not show a clear pattern. Some instances could be solved to proven optimality in a few CPU seconds, while others could not be solved within the time limit imposed. No obvious explanation seems to exist for that behavior.

For any density, instances with $c = k - 1$ are the easiest to solve.

5. Final remarks

We have introduced a new generalization of the graph coloring problem. We propose an integer programming approach and a Branch-and-Cut algorithm. We have characterized several new valid inequalities. Although no polyhedral analysis was carried out, the usefulness of these inequalities has been demonstrated through computational experiments. A comparison with version 10.1 of CPLEX shows that the Branch-and-Cut algorithm proposed here reduces both the CPU time and the number of nodes explored in the Branch-and-Bound tree, and that it is capable of solving instances that are out of the reach of CPLEX.

As a result of this work, we could solve relatively small instances. Experimental tests show that this new generalization of the graph coloring problem is computationally harder than the classical version. For the graph coloring problem, instances with 20 vertices and any density can be tackled in few seconds with a Branch-and-Cut technique, and instances of up to 80 or 100 nodes can be solved within 1 h (see [8]).

Further work is needed to improve the algorithm. There is still a place for new cutting plane generation, methods for new valid inequalities and other schemes to prune the search tree.

References

[1] K. Aardal, S. van Hoesel, A. Koster, C. Mannino, A. Sassano, Models and solution techniques for frequency assignment problems, 4OR 1 (4) (2003) 261–317.
[2] A. Billionnet, Using integer programming to solve the train-platforming problem, Transportation Science 37 (2) (2003) 213–222.
[3] M. Halldórsson, G. Kortsarz, Multicoloring: Problems and Techniques, in: Lecture Notes in Computer Science, vol. 315, 2004, pp. 25–41.
[4] P. Hansen, M. Labbé, D. Schindl, Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results, Tech. Rep. No. G-2005-76, Montreal, Canada: GERAD, 2005.
[5] T. Jensen, B. Toft, Graph Coloring Problems, Wiley-Interscience, New York, 1995.
[6] D.S. Johnson, A. Mehrotra, M.A. Trick, Special issue on computational methods for graph coloring and its generalizations, Discrete Applied Mathematics 156 (2008).
[7] J. Lee, F. Margot, On a binary-encoded ILP coloring formulation, INFORMS Journal on Computing 19 (3) (2007) 406–415.
[8] I. Méndez-Díaz, P. Zabala, A Branch-and-Cut algorithm for graph coloring, Discrete Applied Mathematics 154 (5) (2006) 826–847.
[9] A. Mehrotra, M. Trick, A column generation approach for Graph Coloring, INFORMS Journal on Computing 8 (1996) 344–354.
[10] A. Mehrotra, M. Trick, A branch and price approach for graph multicoloring, in: Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, in: Springer Operations Research/Computer Science Interfaces Series, vol. 37, 2007, pp. 15–30.
[11] L. Narayan, Channel assignment and graph multicoloring, in: Handbook of Wireless Networks and Mobile Computing, in: Wiley Series On Parallel And Distributed Computing, 2002, pp. 71–94.