Joint Routing and Link Scheduling for Wireless Mesh Networks through Genetic Algorithms

Leonardo Badia, Alessio Botta
IMT Lucca Institute for Advanced Studies
piazza S. Ponziano 6, 55100, Lucca, Italy
{leonardo.badia,alessio.botta}@imtlucca.it

Luciano Lenzini
Dept. of Information Engineering, University of Pisa
via Diotisalvi 2, 56122 Pisa, Italy
l.lenzini@iet.unipi.it

Abstract— **Wireless Mesh Networks (WMN) are emerging as an attractive technology for providing broadband connectivity to mobile clients who are just on the edge of wired networks, and also for building self-organized networks in places where wired infrastructures are not available or not deemed to be worth deploying. This paper investigates the joint link scheduling and routing issues involved in the delivery of a given backlog from any node of a WMN towards a specific node (which acts as a gateway), within a given deadline. As in a real WMN, scheduling and routing are assumed to be aware of the physical interference among nodes, which is modeled in the paper by means of a Signal-to-Interference Ratio (SIR). Firstly, using a theoretical model of a WMN we formulate the problem as an Integer Linear Programming (ILP) problem. Secondly, since the problem cannot be dealt with using exact methods, we propose and use a technique based on Genetic Algorithms (GAs). To the best of our knowledge, GAs have never been used before for working out these kinds of optimization problems in a WMN environment. We show that our technique is suitable for this purpose as it provides a good trade-off between fast computation and the overall goodness of the solution found. Our experience has in fact shown that GAs would seem to be quite promising for solving more complex WMN models than the one dealt with in this paper, such as those including multiple flows and multi-radio multi-channels.**

Keywords- **Wireless Mesh Networks, Routing, Link Scheduling, Integer Linear Programming, Genetic Algorithms.**

I. INTRODUCTION

WMNs are an emerging class of networks, usually built on fixed nodes that are inter-connected via wireless links to form a multi-hop network [1]. Their main goal is to provide broadband access to mobile clients who are just on the edge of wired networks. WMNs can be used where cable deployment is not feasible or is too expensive, such as in remote valleys or rural areas, but also in offices and home environments. End-users are served by nodes called *mesh routers*, which are generally assumed to be stationary. Mesh routers are in turn wirelessly interconnected so as to form a network *backhaul*, where radio resource management challenges come into play. Moreover, some mesh routers are generally provided with access (e.g. through wires) to the Internet and therefore can act as gateways for the entire WMN. Communication between any two mesh routers as well as from any router to gateways is multi-hop.

Many of the WMN issues are thus common to those of multi-hop wireless networks, such as determining link scheduling in order to obtain high throughput efficiency [2], [3] or selecting appropriate routes between source and destination [4], [5]. However, the fact that mesh routers are fixed makes the backhaul of a WMN inherently different from distributed wireless networks (e.g. ad hoc networks), where the nodes may be portable devices. For example, problems such as energy consumption are no longer an issue. Also, the uncertainty about positioning terminals due to mobility or difficulty to communicate, as well as their computational capability, are mitigated. This makes it sensible to opt for a centralized network management, as opposed to the distributed approaches used for ad hoc wireless networks. In this case, nodes act in a coordinated fashion under the supervision of a network entity which determines the management based on global knowledge of the network topology and additional conditions.

A cross-layer approach where the routing and link scheduling functionalities are jointly addressed has been extensively studied in multi-hop wireless networks [4]-[7]. However, the problem of determining, for example, the shortest deadline within which a specified backlog vector can be jointly routed and scheduled between WMN nodes and a gateway, does not seem to have been analyzed in the literature. This is the primary objective of the paper, which focuses on two major innovations.

Firstly, we formulate our problem through an ILP framework [8] by capturing the characteristics both of the WMN topology and of the radio channel, which allows us to determine the feasibility conditions for our problem. In the design of our framework we give particular emphasis to interference related aspects. In particular, we employ the so-called *physical interference model*, which computes the Signal-to-Interference Ratio (SIR) at each active node and compares it with an appropriate threshold [9]. ILP formulations generally use another approach, named *protocol interference model*, which is simpler to apply but in our case may actually lead to oversimplifications. To the best of our knowledge, our ILP formulation is the only one available that explicitly addresses the physical interference model in its original version with linear constraints and binary variables. However, we believe that one more merit of our ILP framework is to leave room for possible extensions to specific cases of interest, in which a given objective function is proposed.

Secondly, we use Genetic Algorithms (GAs) to solve the cross-layer problem, and this technique copes reasonably well with our framework. It is known from the literature [2] that finding link activation patterns that satisfy traffic requirements and keeping interference under control typically causes *NP*-complete problems. This means that the solution to the problem

cannot be guaranteed to be found in polynomial time. Exact approaches fail to solve the problem in a reasonable time, even with not very large topologies, e.g. with 8-10 nodes.

GAs are an optimization technique which imitates evolutionary processes existing in nature [10]. They do not guarantee to find the best possible solution within a given time: even if they are customized appropriately, they only solve the problem optimally with unlimited time at their disposal. However, GAs often work in practical cases as they are able to provide a "good enough" solution in a reasonable time. Moreover, they appear to be ideal for handling discrete values, multiple constraints and also multiple objectives, as happens in problems of network planning [11], [12], as well as with the problem discussed in this paper. We would like to stress that although GAs are often seen as a standard technique that can be used within any optimization framework, our problem requires an original solution to implement in the GA, which will be examined in detail in the paper.

The rest of this document is organized as follows: in Section II we discuss the literature. In Section III we outline the basic assumptions of the model by describing the variables and the notation utilized. These are employed in Section IV to formulate the ILP model. In Section V we describe GAs and discuss their application to our case study. Finally, in Section VI we present numerical results, and we draw conclusions in Section VII.

II. Related Work

There are many papers in the literature [2]-[7], [13], [14] that can be related to the present work, as they investigate routing, scheduling, or both, through what can be seen as a Linear Programming framework. For example, [4] discusses routing optimization for wireless networks, but the main focus is on sensor networks, and, as is common for such systems, energy efficiency is considered as the objective. Also, there is no consideration about mutual interference of the nodes, which is important in WMNs.

The analysis of [13] is, on the other hand, more applicable to our scenario, since it deals with throughput maximization and focuses on interference relationships. As far as the interference model is concerned, using the physical interference model is very rare in the literature. A notable exception to this is [2]. In this very recent paper, the evaluation of the SINR relationships is used to find feasible schedules in a WMN, and computationally efficient solutions are proposed to this end. Unlike our investigation, a link activation pattern is sought in order to meet pre-determined link weights which can correspond to the routes, whereas in our analysis we solve both routing and scheduling jointly.

This places our work in the field of cross-layer solutions, whose investigation includes also channel assignment, as in [14], where a joint channel assignment and routing problem is approached, but is solved through heuristics. Exact solutions are discussed in [5], where a joint channel assignment and routing is proposed through an ILP framework. The scheduling issue is considered as the solution to a preliminary optimization, where only links that can be scheduled together are used. However, this paper only accounts for the protocol interference model, and in the routing phase non-integer link activation values are utilized. Similar considerations hold for [3], where a two-phase algorithm is introduced. First, a routing LP is solved that takes the protocol interference model into account but does not do any scheduling. This solution is then scheduled over time using a different algorithm.

In [6], an optimization approach is proposed to jointly solve link scheduling and routing, as we do in the present paper. However, the meaning of link scheduling is different, as in [6] the feasibility of a vector of rates is simply sought, whereas our aim is to determine a link activation pattern, which delivers the backlog of every node to the gateways within an assigned deadline. This framework is further extended in [7] to include channel assignment as well. However, again the fact that the binary variables of link activation are relaxed to rational values is shown in [7] to be a limiting assumption, which may lead to inaccuracies in the solution. Also, all previous contributions presenting a cross-layer approach take into account the protocol interference model, whereas we use the more suitable physical interference model.

Finally, our paper also presents an original contribution in terms of the technique used to solve the optimization. GAs are in fact very commonly used for a preliminary planning of wireless networks in general [11], and this is true for WMN as well. For example, in [12] there is an overview on how to use GAs to help the deployment of inter-urban mesh networks. Another very recent paper [16] employs GAs for sensor networks. Even though the kind of network is different, the authors present some physical layer considerations about channel activation and the mutual interference of nodes.

However, in all these investigations dealing with network deployment, the usage of GAs is mostly motivated by the high complexity of the problem, which prevents it from being solved with exact techniques. On the other hand, the speed of GA in giving good solutions to the problem is not exploited, since the time for the optimization process to converge is not as relevant as in shorter time-scale problems such as routing or scheduling. Indeed, we believe that in our problem we are able to show this additional advantage offered by the computational efficiency of GAs. To our knowledge, this fact is not very frequently explored in the literature, thus making our work innovative.

III. Basic Assumptions of the Model

We represent the backhaul of a WMN as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, which consists of $N = |\mathcal{N}|$ nodes representing the mesh routers of the WMN, connected by *directed* edges of the set $\mathcal{E}$ representing potential links between terminals. Notation $e = (i, j) \in \mathcal{E}$ means that $i \in \mathcal{N}$ is the transmitter node of link e and $j \in \mathcal{N}$ is the receiver. We denote with $\mathcal{Y} \subset \mathcal{N}$ the set of the gateways, which are any-cast end destinations for the mesh routers. In general, not all pairs of nodes are connected through an edge. We denote the one-hop input and output neighbors set of a node i as $\mathcal{S}_i$ and $\mathcal{R}_i$. In other words, $\mathcal{S}_i$ and $\mathcal{R}_i$ are the set of nodes for which an edge exists in $\mathcal{E}$ to and

from node i, respectively, i.e. $\mathcal{S}_i = \{ j \in \mathcal{N} \mid (j,i) \in \mathcal{E} \}$ and $\mathcal{R}_i = \{ j \in \mathcal{N} \mid (i,j) \in \mathcal{E} \}$.

Note that in a WMN it is reasonable to consider every pair of nodes as being connected through an edge in $\mathcal{E}$, which means $\mathcal{E} = \{(i,j) \in \mathcal{N} \times \mathcal{N} \mid i \neq j\}$ and $\mathcal{R}_i = \mathcal{S}_i = \mathcal{N} \setminus \{i\}$. This is due to the absence, in real WMNs, of transmitting power limitations (mesh routers can be easily attached to a power outlet), and, through appropriate power control, it is therefore virtually possible to reach any other node. Also, the edges in $\mathcal{E}$ only describe a virtual link between nodes of the WMN, which can even be unused if the routing algorithm detects that they are not worth activating. In general, other techniques often consider an a priori network pruning, but this may lead to approximations when formulating the problem, which we want to avoid. In fact, our methodology applies without any restrictions to every scenario, even strongly connected ones, as we leave open the possibility for any route through the set of nodes. Of course, this also leads to a heavier problem in terms of computational complexity.

Hereafter for the sake of simplicity we also assume that nodes can use a single power level. This is not a limiting assumption, as multiple power levels can be taken into account by considering multiple edges for the same pair of nodes, without changing the rationale of the analysis. Similarly, we assume that all nodes are homogeneous in terms of the number and kinds of radio interfaces they own, as well as the frequency bands they are enabled to transmit on. Indeed, the extension of WMN management to the multiple channel case looks promising and several standards are envisioned to explicitly include support for such a case. All these extensions (multiple channels, multiple power levels, etc) can be seen as extensions of the basic framework discussed here and are left for future research on this topic.

We assume that the WMN system operates in synchronous time slotted mode where timeslots are labeled via integer numbers $0,1,...,t,....$. Every edge $(i,j) \in \mathcal{E}$ is also associated with a transmission rate r_{ij} and a path gain g_{ij}. The former describes the number of packets, assumed to be constant, that can be sent during a timeslot over the edge (i,j), whereas the latter is the inverse of the channel attenuation (transmitted power over received power) between nodes i and j and will be used in the following when modeling interference between transmission links. Both r_{ij} and g_{ij} variables can be collected into matrices $\mathbf{R} = (r_{ij})$ and $\mathbf{G} = (g_{ij})$. Another assumption made for analytical tractability is that it is not possible to underutilize an edge below the available rate r_{ij}, unless the transmitter does not have enough packets to send. This generally prevents the sender from splitting the data into parts smaller than the whole rate of an edge However, this would be really beneficial in a negligible number of cases; thus, this assumption is not restrictive at all in practice.

To solve the joint link scheduling and routing problem, we define a 0-1 scheduling variable $x_{ij}(t)$ for every $(i,j) \in \mathcal{E}$, as

$$x_{ij}(t) = \begin{cases} 1 & \text{if } i \to j \text{ is active on time slot } t \\ 0 & \text{otherwise} \end{cases}$$

In other words, $x_{ij}(t)$ denotes whether or not there is a data transmission (i.e. the link is activated) on time slot t. These variables are bound to be integer, varying over a discrete (slotted) time, so as to determine a time-division scheduling pattern for the WMN backhaul [2]. Similarly to the analysis presented by [6], we remark that the derivation of a scheduling pattern of links implicitly determines the routing as well. However, rather than approaching the routes on a per-flow basis, we derive the routes by looking at the dynamics of the link activation over time. Unlike other papers [5], [7], in this work we impose the $x_{ij}(t)$ variables to be strictly binary and varying over discrete time t. In other words, we explicitly avoid relaxing constraints about variables to be integer, which is an approximation that can lead to strongly sub-optimal results in the ILP.

For the sake of analytical tractability, we will focus on periodic scheduling, where a frame of duration T slots is assumed to set the cycle of link activations. This means that links are activated according to the solution found for t between 0 and $T-1$, and this pattern can be repeated identically every T slots. We assume that each node supports a single flow towards a gateway. The amount of traffic per node is known in advance and is already available at the beginning of the frame at the non-gateway nodes. We leave for future work any extension about packet arrivals delayed throughout the whole frame. The goal within a single frame is to deliver the traffic to the gateways. This can be done by sending it directly to a gateway or by relaying to one or more nodes before reaching the destination gateway, depending on the status of WMN backhaul links. In the latter case there is flow traffic aggregation at some intermediate node towards a gateway.

The progress status of the transmission to the gateways is modeled through variables $q_i(t)$, which describe at every time slot t the queue length at each node i. In reality, these are more like *auxiliary* variables, since, as shown in the following, they can be put in relationship through flow constraints with the binary variables $x_{ij}(t)$. We assume that, for every t in 0, 1, …, $T-1$, $q_i(t)$ represents the amount of traffic in queue at node i, that needs to be delivered to one of the gateways before the end of the frame. The connection between $q_i(t)$ and $x_{ij}(t)$ is such that $q_i(t)$ represents the amount of data *before* the application of the transmissions identified by $x_{ij}(t)$, whereas $q_i(t+1)$ describes the outcome of these transmissions. For this reason, $q_i(t)$ varies over time, so that at the beginning of the frame $q_i(0)$ represents the overall amount of data (i.e. the aggregated demand from its associated users) to deliver for node i, and $q_i(T)$ describes the residual backlog at node i after the application of the joint routing and link scheduling pattern.

IV. Problem Formulation and Main Constraints

The problem of assigning meaningful 0-1 values to $x_{ij}(t)$ can be seen as a flow optimization problem subject to three kinds of constraints. The first one is related to the flow conservation and delivery of all traffic to the gateways. Also, two other types of conditions are needed to check the feasibility of the link activation pattern. Both of them are related to the feasibility of

simultaneous activations of links, which is generally beneficial as it improves the transmission parallelism. Only *compatible* transmissions can be scheduled in the same time slot, where "compatibility" means "possibility to be used simultaneously." Modeling this property among *wireless* link transmission is challenging, and several models have been proposed [9]. To check whether two transmissions can coexist, two conditions must be met:

- the radio equipment of a single node cannot be used for too many tasks (i.e., transmission/reception). According to whether the channel is *full duplex* or *half duplex* [7], it is either possible to have at most one transmission and one reception at the same node per each slot, or one single task comprising both reception and transmission.
- interference issues also need to be checked. Several models can be used, and we will refer to the physical interference model [9].

We classify the three kinds of constraints: flow constraints, direct compatibility constraints, and interference constraints. These are discussed in their respective subsections.

A. Flow constraints

The flow constraints include flow conservation for every time slot *t* at each node:

$$q_i(t+1) = \max\left(0, q_i(t) - \sum_{j\in\mathcal{R}_i} x_{ij}(t)\, r_{ij}\right) + \\ + \sum_{j\in S_i} \left(\min(q_j(t), x_{ji}(t)\, r_{ji})\right) \qquad \forall i \in \mathcal{N}, \forall t = 0,\ldots,T-1 \tag{1}$$

The formulation of this constraint in a linear version accounts for the possibility of having transmission and reception simultaneously, i.e. a full duplex case is considered. In fact, the right-hand term sum accounts for both incoming and exiting packets. However, the fact that the active incoming links (in the first term) and the active outgoing links (in the second term) can be at most one is implicitly accounted for. If the channel is half duplex, no modification is needed, since the condition is even more restrictive: at most one among all incoming and outgoing links can be active.

Additionally, at time *T* everything has to be delivered to the gateways:

$$\sum_{i\in\mathcal{N}} q_i(0) = \sum_{i\in\mathcal{Y}} q_i(T) \tag{2}$$

We also assume that the gateways do not generate traffic. The formulation of a related constraint is not strictly necessary, but it is useful to simplify the resulting algorithm.

$$q_i(0) = 0, \qquad \forall i \in \mathcal{Y}, \tag{3}$$

$$\sum_{j\in\mathcal{R}_i} x_{ij}(t) \le 0, \quad \forall i \in \mathcal{Y}, \forall t \in 0,1,\ldots,T-1. \tag{4}$$

B. Direct compatibility constraints

The constraints that we call *direct compatibility constraints* relate to the impossibility of utilizing a transceiver equipment of a node for more purposes than is designed for. For full duplex links, the direct compatibility constraints can be written as

$$\sum_{j\in\mathcal{R}_i} x_{ij}(t) \le 1 \qquad \forall i \in \mathcal{N}, \forall t = 0,\ldots,T-1 \tag{5}$$

$$\sum_{j\in S_i} x_{ji}(t) \le 1 \qquad \forall i \in \mathcal{N}, \forall t = 0,\ldots,T-1 \tag{6}$$

However, wireless links are intrinsically half-duplex, unless special techniques are employed, which implement full duplexing, such as directional antennas [17] or multiple channels [7]. If there is no frequency or spatial separation between transmitter and receiver, if a node is transmitting, any simultaneous reception will be destroyed by the self-interfering transmitted power. Thus, the right constraint for a WMN is a half duplex one. To account for a half duplex channel, the constraints above are simply merged so as to form

$$\sum_{j\in\mathcal{R}_i} x_{ij}(t) + \sum_{j\in S_i} x_{ji}(t) \le 1 \quad \forall i \in \mathcal{N}, \forall t = 0,..,T-1 \tag{7}$$

C. Interference compatibility constraints

The physical interference model evaluates the Signal-to-Interference Ratio (SIR) of every transmission and assumes that, in order to be successful, the received power at every active receiver *i* has to overcome a SIR threshold called γ_i. Even though γ_i can be a different value for every node *i*, if the traffic flows are homogeneous and the modulation techniques are the same, it is sensible to use the same threshold for all the nodes. Also, for the sake of simplicity and without loss of generality, we omit ambient noise terms, which could be included by considering the SINR (Signal-to-Interference-plus-Noise Ratio) instead of the SIR. This does not lead to any significant changes in the mathematical formulation.

The interference compatibility constraint can be written as

$$\gamma x_{ij}(t) \le \frac{g_{ij} x_{ij}(t)}{\sum_{k\in S_j\setminus\{i\}} g_{kj} \sum_{\ell\in\mathcal{R}_k\setminus\{j\}} x_{k\ell}(t)} \\ \forall (i,j) \in \mathcal{E}, \forall t = 0,\ldots,T-1 \ , \tag{8}$$

which is in accordance with the most commonly used definition of SIR [9]. The physical meaning of this expression is as follows. Assuming all links use the same power, the activation of link from *i* to *j* at time *t*, corresponding to having $x_{ij}(t)$ equal to 1, is subject to having an SIR on this link greater than or equal to γ, which is obtained by checking whether the ratio between the useful power (numerator term) over the interfering power plus noise (denominator) is greater than or equal to γ. Note that, to be meaningful, the interference constraint must be applied to active links only. This is the reason behind the mathematical formulation of (8), where if $x_{ij}(t) = 1$, the above inequality holds and the term $x_{ij}(t)$ can be removed from both left-hand and right-hand terms, whereas if $x_{ij}(t) = 0$, the above inequality is trivially always verified.

In order to have a linear constraint rather than a quadratic one, the following artifice is employed. Rearrange (8) as:

$$g_{ij} x_{ij}(t) \ge \gamma x_{ij}(t) \sum_{k\in S_j\setminus\{i\}} g_{kj} \sum_{\ell\in\mathcal{R}_k\setminus\{j\}} x_{k\ell}(t) \\ \forall (i,j) \in \mathcal{E}, \forall t = 0,\ldots,T-1 \ . \tag{9}$$

This is still a quadratic constraint. However, after some manipulations we can derive the following linear relationship as the interference constraint:

$$g_{ij} \geq \gamma_j \sum_{k \in S_j \setminus \{i\}} g_{kj} \left(\left(\sum_{\ell \in \mathcal{R}_k \setminus \{j\}} x_{k\ell}(t) \right) + x_{ij}(t) - 1 \right) \quad \forall (i,j) \in \mathcal{E}, \forall t = 0, \ldots, T-1 \tag{10}$$

This formulation can be shown to be equivalent by considering each possibility of $x_{ij}(t)$ being either 0 or 1 and relying on constraint (7), which bounds the inner-most summation to be always less than or equal to 1. Note that this would also hold true in the full duplex case, since the role of constraint (7) could be played by (5) and (6) together.

Even though the problem can be entirely formulated within an ILP framework, the solution is hard to find with exact methods. This happens since the problem can be shown to be *NP*-complete [2]. As it will be shown in section VI, the problem becomes untreatable even with a limited number of nodes, i.e. more than 5 mesh routers including a gateway. The computational complexity is also strongly dependent on *T*. Heuristic solutions [14] might work in certain cases, but they fail to adapt to different network scenarios.

For these reasons, we propose in this paper a self-configurable and efficient methodology based on Genetic Algorithms, which will be explained in detail in the next section.

V. A Genetic Approach

Genetic Algorithms (GAs) are a meta-heuristic technique employed to solve optimization problems, which imitate *Natural Selection*, i.e. the process of adaptation to the environment performed by living beings [10], [18]. GAs are an appealing approach to solve the complex problem stated in the previous sections. Among their most interesting features, GAs

- are able to find "good solutions" to an unconstrained problem in a reasonable time, and they *always* find at least one "good" suboptimal solution,
- does not require a differentiable objective functions and can be tailored to handle any sort of constraint,
- can easily handle discrete problems by choosing a discrete alphabet of symbols (e.g., integer numbers) for the chromosome,
- can scale with the problem by changing the setup of some parameters (e.g., number of individuals in the population),
- can be customized to include some heuristics and experts' knowledge in the generation of the initial population and in the design of the genetic operators.

For these reasons, we chose GAs as the heuristic approach to solve the problem formulated in Section IV but without relaxing any of the constraints, including the integer constraint of variables $x_{ij}(t)$.

A. Genetic Algorithms: Background

A GA determines, rather than a single solution, a whole *population* consisting of *individuals*, which are all candidate solutions to the problem. The distinctive features of each individual are coded into a structure called *chromosome*. The chromosome is a string of *genes*, whose values can be chosen in a set of symbols. An application-dependant process generates the individual by decoding its chromosome. The symbols used as values of the genes are usually binary, integer or real numbers, depending on the nature of the problem. Once an individual is generated, a *fitness function* is used to evaluate its goodness as a solution to the problem. Usually, low values of fitness function are given to the best individuals (minimization problem). For the sake of simplicity, in the following we will blur the definitions of individual and chromosome.

A GA starts with an initial population generated either randomly, or with some heuristic approach that exploits the knowledge of an expert in the problem domain. The algorithm then proceeds in steps called *generations*. At each generation *t*, a new population $P(t+1)$ is evolved from $P(t)$. As generations pass, the population should improve globally thanks to the application of *genetic operators* that mimic the natural evolution mechanisms. To this aim, the best individuals are chosen from $P(t)$ (*selection*) to be mated (*crossover*) and slightly modified (*mutation*), so as to create the new population $P(t+1)$.

The selection operator is used to decide which individuals in $P(t)$ should be chosen to generate $P(t+1)$. Optionally, an *elite* of the selected individuals (i.e. a small number of the best performing individuals) survives and is moved from $P(t)$ to $P(t+1)$ without any change. The crossover operator consists in choosing some of the individuals and *mating* them, that is, substituting them with their *children*, i.e., individuals generated by mixing the genetic material in the parents' chromosomes. The actual implementation of a crossover operation very much depends on the coding schema of the chromosome. Finally, the mutation operator introduces some new genetic material in the population by randomly modifying the values of some genes. Again, different kinds of mutation operations can be defined to handle different sets of symbols. The population continues to evolve until a stopping criterion is fulfilled, the simplest being a maximum number of generations. The overall basic GA algorithm is shown in pseudo-code in Figure 1.

If crossover and mutation are general enough, GAs can be shown to allow the exploration of the whole solution space. If an optimization goal is set, they are bound to find the optimal solution, even though it cannot be guaranteed that it will be the optimal one, nor can the time to find it be predicted. However, since the execution time is generally rapid, GAs are also interesting for practical purposes as they can be seen as fast procedures to find a "good enough" solution to the problem. This gives them an advantage with respect to exact techniques such as Branch and Cut used in commercial solvers, since any solution produced by a GA is directly applicable. Therefore, GA could be used to operate online WMN management, where the solution may be iteratively updated. This could also be an interesting development of the present analysis for future work.

B. A GA-based approach for the ILP problem

While some classes of problems can be solved by directly applying a basic version of a GA, more often the development of such an algorithm for a specific problem is an engineering process that involves a good amount of design and tailoring.

Indeed, the design of a GA includes finding suitable representation schemas, coding strategies, genetic operators, values of parameters, etc. Furthermore, if the problem is constrained, like the ILP formulated in Section IV, we are forced to select and adapt appropriate constraint-handling methods from the ones available in the literature [19], [20].

The first step when designing the GA is to identify how to mathematically represent a solution as an individual, in order to create a population for the algorithm. In our case, given the natural binary formulation of the problem, we consider genes to be binary digits. Thus, the chromosomes are coded as sequences of bits representing the variables $x_{ij}(t)$, sorted first internally to each frame by any ordering of the edges, then frame-by-frame in an increasing order. Formally, the genetic map of any individual is: $(x_1(0), x_2(0), \ldots, x_E(0), \cdots x_e(t) \cdots, x_1(T-1), x_2(T-1), \ldots, x_E(T-1))$, where indices $1, 2, \ldots, E$ refers to a suitable ordering of the set $\mathcal{E}$.

To generate an initial population, composed of 200 individuals, we investigated the use of several heuristics for routing and scheduling problems, but currently none of them seems to significantly improve the solutions with respect to a completely random initial population. Nevertheless, we observed that good candidate solutions show overall a count of active link that is much less than the number of inactive links. Thus, to speed the convergence of the GA, we non-uniformly generate the random initial population with a ratio of active links equal to 0.2 over all links. However, this point may deserve further investigation in future research.

The GA proceeds by iteratively modifying the population, that is, by cyclically applying the selection, the crossover, and the mutation operators as described in Section V.A. As the selection operator, we use the robust and well-known *stochastic universal sampling* [18]. As regards the other operators (crossover and mutation), we developed our customized versions. Our coding schema has two granularity levels: the link level, represented by a single gene, and the frame level, coded by the overall configuration of the network for one time slot. We designed our operators so as to work on both levels of granularity.

The crossover operator is the *0.5-uniform crossover* [18]. The standard version of this operator chooses the value of each gene in the chromosome of a child between the two values of the parents, with a uniform probability. This is a link-level granularity. Nevertheless, it can be useful to apply the same approach on the frame level, so as to exploit the knowledge already discovered about whole time slots. Thus, our modified uniform crossover may act, with a uniform probability, on one of the two different granularities, mixing single bits or whole time slots from the two parents to generate the child.

A similar approach was used to develop the mutation operator. We recall that the aim of this operator is to introduce some new previously unexplored solutions in the population by slightly modifying the current ones. Thus, our mutation operator can perform, with uniform probability, one of the following operations:

```
initialize P(0)
repeat
   evaluate P(t) via fitness_function;
   apply selection to choose parents;
   apply crossover to generate offspring
   apply mutation to offspring
   generate P(t+1)
   increase t by 1
until a termination condition is verified
```

Fig. 1. The pseudo-code of a basic GA.

- mutate the chromosome on a link-level granularity by the *uniform random mutation* [18],
- scramble some of the time slots of the chromosome (e.g. switch slot number 1 and number 5),
- replace some time slots with a duplicate of other slots of the same chromosome (e.g., replace slot number 5 with a copy of slot number 1),
- replace some slots with empty slots.

To satisfy the constraints, we used two different techniques. We divided constraints into two classes: constraints that have to be satisfied by each individual generated during the algorithm, and constraints that can be unsatisfied by some individuals. The first class includes the direct compatibility constraints of (7). The second class includes the flow constraints of (1) and (2), and the interference compatibility constraints of (10).

Constraints in the first class are always satisfied by means of a *repair* process, which is performed after the application of any genetic operator that might produce an infeasible individual. For instance, suppose that the mutation operator generated an individual in which, at some time, a node activates two output links in the same time slot. In this case, the repair randomly deactivates one of the links, and fixes the individual. Another case handled by repair is the activation of an output link by a node that has no more packets to send. Since repair is performed each time a genetic operator is applied, it must be designed to be an extremely fast and efficient routine. Thus, we decided to deactivate conflicting links in a random fashion, and to repair only "easy" constraints that are the basis to derive all the ILP problems in Section IV. Nevertheless, further research could lead to a more effective repair process based on a pre-evaluation of all the possible fixed individuals generated by an infeasible one.

Constraints in the second class are handled by allowing infeasible individuals to survive in the population. Those individuals are given higher values in the fitness function by means of *penalty functions*. The penalty is computed in the following way:

$$\rho = \sum_{i \in \mathcal{N}} q_i(0) - \sum_{i \in \mathcal{Y}} q_i(T) + \sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{E}} p_{ij}(t), \qquad (11)$$

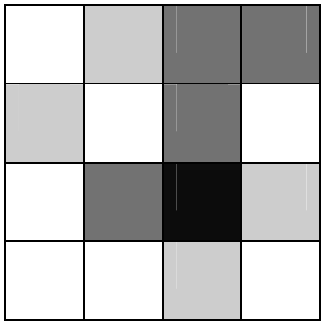

Fig. 2. Grid topology with 5 or 9 nodes, including a gateway.

where $p_{ij}(t)$ describes the interferences violations at time slot *t*, that is

$$p_{ij}(t)=\begin{cases}1 & \text{if } g_{ij}-\gamma_j\sum_{k\in\mathcal{S}_j\setminus\{i\}} g_{kj}\Big(\big(\sum_{\ell\in\mathcal{R}_k\setminus\{j\}} x_{k\ell}(t)\big)+x_{ij}(t)-1\Big)<0\\ 0 & \text{otherwise}\end{cases} \quad (12)$$

The fitness function can also incorporate, by a linear combination, some metrics of the network that we want to optimize. Interestingly, the best results, both in terms of convergence speed and goodness of the solution found, were given by also including a (small) penalty proportional to the number of activated links per slot, i.e. to the sum of $x_{ij}(t)$ over all edges and the whole frame. This is because activating too many links not only causes more interference, but also prevents the GA from trying alternative routes by means of crossover or random mutation, which is due to the reparation. This vanilla approach can be easily modified to incorporate any other network measures, such as global interference, throughput, minimum number of time slots, etc.

As stated above, the typical termination condition of a GA is a fixed maximum number of generations. We used a hybrid stopping condition which still stops the GA after a maximum of 200 generations and tries to perform an early stop in two cases:

- a good feasible solution is found quickly, or
- the problem seems to be infeasible.

The idea is that, if we already are in the feasible region, we are not interested in optimizing the network metric much more, and that if the problem seems infeasible, we should give up early with the best solution found. Thus, in the former case, we perform an early stop if, after a first feasible solution is found (that is, a solution with $\rho > 0$), we do not find any other better solution in 5 generations. In the latter case, we perform an early stop if we have not found any feasible solution and we have noticed no improvements in the last 50 generations.

VI. Numerical Evaluations

To evaluate the performance of our GA, we consider a grid consisting of 35 m × 35 m squares, as reported in Fig. 2. Each square can be occupied by at most one node, according to a pre-determined pattern indicated below. A specific square occupancy identifies a *different scenario instance*. The node is randomly placed (with uniform distribution on both coordinates) within the square. The rational behind this approach is to model channel variations through changes in the network topology, while at the same time keeping constant some parameters, e.g., the number of nodes. For this reason, any instance of the same scenario can also be viewed as a different topology which can be created on the same physical network, where nodes have fixed placements but the channel is time-varying. We consider two scenarios, both with a gateway placed in the black square. In the first one, which consists of 5 nodes, beyond the gateway node, 4 mesh routers are placed in each of the dark grey squares. Four additional routers are placed in the light grey squares so as to form the second scenario, with a 9-node topology. For each scenario we generated 20 different topology instances by varying the node position within the square it belongs to.

The graph resulting from node placement is also determined by considering the path gain of each edge of length *d* as proportional to $d^{-3.5}$ and its rate as a discrete value function of the distance, determined as constantly equal to 8 packets/slot for *d* below 50 meters, 4 packets/slot between 50 and 75 meters, 2 packets/slot between 75 and 100 meters, and 1 packet/slot for larger distances. We assumed half duplex channels and SIR target γ equal to 3.0 (in linear scale) for all the receivers.

We implemented the GA algorithm as discussed in Section V, using the procedures contained in the genetic package of MATLAB Release 2006a [21] as a basis.

The GA was run five times for each topology instance, in order to avoid particularly unfortunate cases where the GA terminates in a dead end of the state space. The plotted results refer to the best solution out of the five trials. Whenever feasible, to test the goodness of the solutions found by the GA, we also implemented an exact ILP solution technique using the LPSOLVE model solver [22]. Both algorithms were tested for the 20 different topology instances mentioned above.

As performance metrics, we considered both the fraction of cases (i.e. topology instances) in which the GA finds a feasible solution (i.e., a solution which allows the delivery of the backlog from any node to the gateway within the frame duration) within the above termination conditions, and the delivery ratio (i.e., the ratio of delivered packets over the total traffic of each node) of the best solution found.

We show detailed results considering GA performance, and also comparing it with exact optimization techniques for the 5-node topology. It is hard to make a detailed comparison of both genetic and exact algorithms on topology with a higher number of nodes due to the computational complexity of exact techniques. In a sense, not only is the Genetic Algorithm more computationally efficient, but it also has the considerable advantage of being more scalable.

In Fig. 3, we show the performance of the GA in the 5-node network, for the case where the load to deliver to the gateway is fixed for each node to 10 packets, and we vary the frame length *T*. For comparison, exact results are also plotted.

As expected, the fraction of feasible solutions found is an increasing function of the frame length, for both GA and the

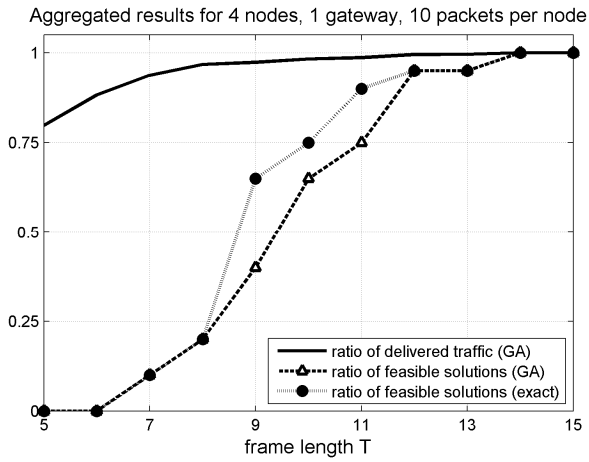

Fig. 3. 5-node scenario, GA performance as a function of the frame length.

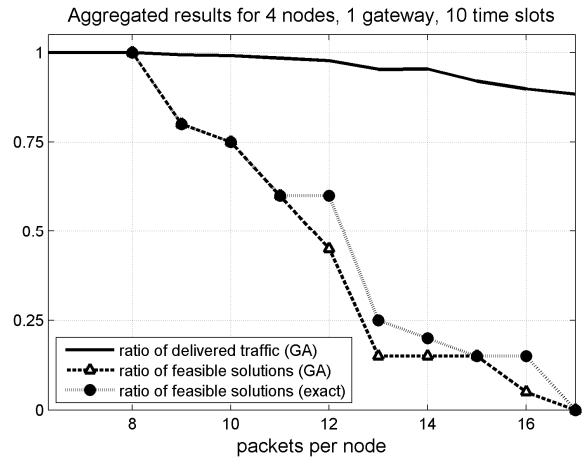

Fig. 4. 5-node scenario, GA performance as a function of the number of packets per node.

exact technique, since a larger T offers a higher degree of freedom in accommodating the packets over the schedule. It is possible to see that for low and high values of T, the performance of the GA matches the exact results perfectly. For intermediate values, the GA slightly underestimates the solvability of some instances, since it may fail to find an existing feasible solution. However, in the worst case the ratio in finding a feasible solution whenever it exists is 61.3% for $T = 9$, whereas it is considerably higher for any other case.

However, even when the GA fails to find an exact solution, either because the optimization stops to a suboptimal value or since it does not actually exist, the delivery ratio achieved by GA is still fairly high. For small values of T, the GA is always able to deliver 80% of the traffic or more, whereas for $T \geq 8$ this ratio is above 96%. This represents a very important advantage of GA in practical implementation, as it gives a solution in any case, and when this is not the optimal one it is still very close to it.

Fig. 4 shows the result of another similar investigation, where T is kept constantly equal to 10 and instead the load per node is changed. The trend is reverted, since the higher the load the more difficult it is to have a solution and also to find it through the GA. However, in this case too the worst performance of the GA is an exact solution of 60% of the cases, in relation to a delivery ratio which is very high in any case. For example, if the load per node equals 14 packets, only 20% of the topologies admit a solution, 75% of which are found by the GA. However, the delivery ratio is larger than 95%. From an information theory point of view [9], these curves may also be used to discuss network capacity. In this case, the "critical" load of the network, i.e., the value around which the fraction of feasible allocations drops significantly, is 12 packets per node, which corresponds to a maximum capacity of around 4.8 packets/slot.

In Fig. 5 the same results as Fig. 3 are reported for the 9-node scenario. The trend is more or less qualitatively similar, though due to the larger number of nodes the time to deliver all packets becomes larger. Here, it was impossible to include exact results as well, since the network is already too large to keep the execution time of any exact algorithm within reasonable bounds. Likewise, Fig. 6 reports the same analysis as Fig. 4 for the 9-node scenario. Due to larger network size, in this case the time frame was fixed to $T = 20$. Again, the range between 8 and 12 packets per node is critical for the network. The solutions found by the GA indicate that the 9-node scenario is able to accommodate 10 packets per node in 50% of the cases, that is, the capacity is approximately 4.0 packets/slot. The slight decrease with respect to the 5-node topology is perfectly in line with the larger network size and also with the fact that the nodes added in this scenario are further from the gateway (so they have both a lower rate for their connections and a higher number of hops).

These results seem to suggest that the GA scales sufficiently well for larger topologies. To understand in more detail the complexity of the algorithm, and in order to have comparison results with the exact techniques, we can refer to Fig. 7, where a complexity analysis is performed. Here, we focus on the only instances of the 5-node scenario where a feasible solution was found by both algorithms, and we measure the complexity through the following performance indices: a) number of evaluations of the fitness function made by GA; b) simplex iterations performed by LPSOLVE. This gives a rough idea of how the algorithms scale when the size of the problem increases. Moreover, we vary the frame size T, since the complexity of the problem strongly depends on it.

As shown in the figure, whereas the exact technique explodes already when T is changed from 7 to 10, the GA complexity stays more or less constant. Indeed, it even slightly decreases when the frame length is very high, since in these cases a solution is found very rapidly, as is reasonable to expect. This proves how good the GA is in finding a quick valid solution to easy problems. In practical cases, it is possible that the network resources are not fully utilized, e.g., the traffic per node is significantly lower than what can be allocated over an entire frame. However, exact techniques can fail to quickly solve the problem, due to its large size In this case, GAs can be seen as a

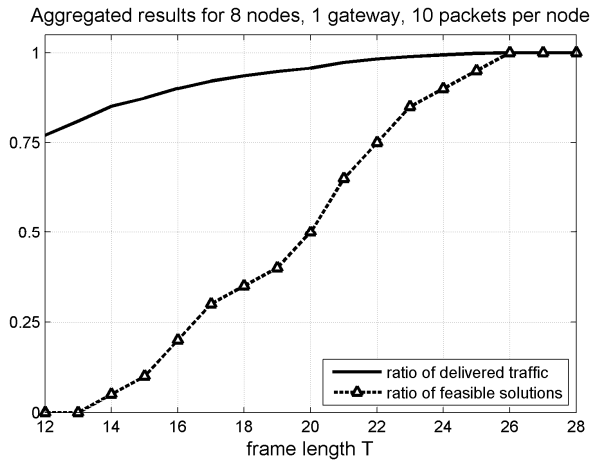

Fig. 5. 9-node scenario, GA performance as a function of the frame length.

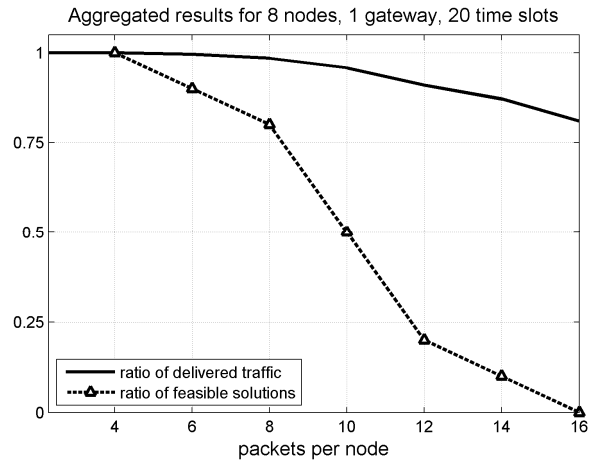

Fig. 6. 5-node scenario, GA performance as a function of the number of packets per node.

very good alternative to heuristics, since by modifying their meta-parameters they are able to adapt themselves to different problem instances.

VII. CONCLUSIONS

In this paper we have investigated joint link scheduling and routing strategies for Wireless Mesh Networks. We have proposed an optimization framework making use of an entirely ILP formulation, where we particularly aimed at keeping the integer constraint of link activation variables and adopting the more realistic physical interference model. This led us to the formulation of an ILP problem whose solution captures both levels of link scheduling and routing in a cross-layer fashion, by supplying a periodic link activation pattern which is able to deliver a given amount of traffic to the network gateways.

The main findings are that the physical interference model is still treatable within the ILP framework. The hard part of the problem is due to the integer constraint, which causes the computational complexity to grow exponentially, both in the number of nodes and in the length of the time frame. Due to the inherent complexity of solving such a problem, we also proposed a fast and efficient solution technique, namely Genetic Algorithms. After having discussed theoretical principles of GAs, we introduced several original implementation parts in order to obtain efficient GAs for the problem under investigation.

Finally, the proposed GA has been tested in two wireless mesh network scenarios. The numerical evaluations show that the GA is able to solve both scenarios reasonably well, and also scales well, whereas exact optimization techniques are unable to solve the larger topologies. The solution found by GA is not always optimal. However, it is always very close to the optimum. Moreover, the GA is a very good approach for realistic cases where feasible solutions are easy to find, since in these cases they converge very rapidly, compared to other techniques, to a solution which is good in practice. For these reasons, we believe that GAs could be very useful tools for a

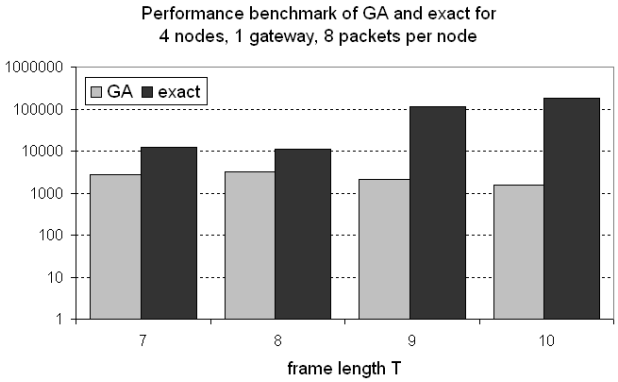

Fig. 7. Computational complexity benchmark.

centralized management of WMNs due to their good level of efficiency in a reasonable computational time.

Future research could be devoted to further optimizing the proposed GA, for example to enable it to deal with non-binary structure in order to better manage larger networks and/or decrease the computational complexity even more. Also, we envision that GAs could be used in more complex problems characterized by multiple flows and multi-radio multi-channels, due to their ability to cope with multi-dimensional constraints and objectives.

REFERENCES

[1] R. Bruno, M. Conti, E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123–131, March 2005.

[2] G. Brar, D. Blough, P. Santi, "Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks," *Proc. ACM Mobicom 2006*, pp. 2–13, 2006.

[3] V. S. Anil Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "Algorithmic aspects of capacity in wireless networks," *Proc. ACM SIGMETRICS*, 2005, 133–144.

[4] J.-H. Chang and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609–619, 2004.

[5] M. Alicherry, R. Bhatia and L. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks", *Proc. ACM Mobicom*, 2005, pp. 58–72.

[6] M. Kodialam, and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem," *Proc. ACM Mobicom*, 2003, pp. 42–54.

[7] ____, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," *Proc. ACM Mobicom*, 2005, pp. 73–87.

[8] A. Wolsey, *Integer Programming*, John Wiley and sons, New York, 1998.

[9] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. on Inf. Theory*, vol. 46, no. 2, 2000.

[10] K. A. De Jong, *Evolutionary Computation: A Unified Approach*, MIT Press, 2006.

[11] B. Al-Bassam, A. Alheraish, and S. H. Bakry, "A tutorial on using genetic algorithms for the design of network topology," *International Journal of Network Management*, vol. 16, no. 4, pp. 253–262, July-August 2006.

[12] K.-T. Ko, K.-S. Tang, C.-Y. Chan, K.-F. Man, and S. Kwong, "Using genetic algorithms to design mesh networks," *IEEE Comput. Mag.*, vol. 30, no. 8, pp. 56–60, Aug. 1997.

[13] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance," *Proc.ACM Mobicom*, 2003, pp. 66–80.

[14] A. Raniwala, K. Gopalan, and T.-C. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM Mobile Computing and Communications Review*, vol. 18, no. 2, pp. 50–65, 2004.

[15] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," *Proc. ACM Mobicom*, pp. 114–128, 2004.

[16] K. Ferentinos, T. Tsiligiridis, "Adaptive design optimization of wireless sensor networks using genetic algorithms," *Elsevier Computer Networks*, vol. 51, pp. 1031–1051, 2007.

[17] J. A. Stine, "Exploiting smart antennas in wireless mesh networks using contention access," *IEEE Wireless Communications*, vol. 13, no. 2, pp. 38–49, April 2006.

[18] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program*, 3rd Edition, Springer-Verlag, Berlin, 1996.

[19] Z. Michalewicz, M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[20] C.A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11–12, pp. 1245–1287, January 2002.

[21] The Mathworks, MATLAB Release 2006a, `www.mathworks.com`.

[22] M. Berkelaar, K. Eikland, and P. Notebaert, lp_solve: Open source (Mixed-Integer) Linear Programming system, available at (website): `lpsolve.sourceforge.net/5.5/`.