Alexei Barbosa de Aguiar

Tackling the Problem of Dynamic Coverage and Connectivity in Wireless Sensor Networks with an Extended Version of the Generate and Solve Methodology

Fortaleza – CE

December / 2009

Alexei Barbosa de Aguiar

Tackling the Problem of Dynamic Coverage and Connectivity in Wireless Sensor Networks with an Extended Version of the Generate and Solve Methodology

Advisor: Prof. Dr. Plácido Rogério Pinheiro

Coadvisor: Prof. Dr. André Luís Vasconcelos Coelho

Fortaleza – CE

December / 2009

A282t Aguiar, Alexei Barbosa.

Tackling the problem of dynamic coverage and connectivity in wireless sensor networks with an extended version of the generate and solve methodology. / Alexei Barbosa Aguiar. - 2009.

81 f.

Dissertação (mestrado) – Universidade de Fortaleza, 2009.

"Orientação: Prof. Dr. Plácido Rogério Pinheiro."

"Co-Orientação: Prof. Dr. André Luís Vasconcelos Coelho."

1. Sistema de comunicação sem fio. 2. Algoritmos genéticos. 3. Programação linear. 4. Otimização matemática. I. Título.

CDU 621.391

Master's dissertation under the title *"Tackling the Problem of Dynamic Coverage and Connectivity in Wireless Sensor Networks with an Extended Version of the Generate and Solve Methodology"*, defended by Alexei Barbosa de Aguiar and approved on December 28, 2009, in Fortaleza, Ceará, by the dissertation examination committee constituted by:

Prof. Dr. Plácido Rogério Pinheiro
Universidade de Fortaleza - Unifor
Advisor

Prof. Dr. André Luís Vasconcelos Coelho
Universidade de Fortaleza - Unifor
Coadvisor

Dr. Tibérius de Oliveira e Bonates
Universidade de Fortaleza - Unifor

Prof. Dr. Gerardo Valdisio Rodrigues Viana
Universidade Federal do Ceará

I dedicate this dissertation to my parents
for all they did to me in my whole life,
to my wife who loves and completes me
and to my friends Édson, Henrique and Nivaldo
that encouraged me to embrace this journey.

Acknowledgments

First of all, I would like to thank God for showing the path, even though I am not always following this path. My life is very blessed so I thank Him every day for everything.

I am thankful to my father and mother. My father, among many other important things for life, for teaching me enjoying learning, how to investigate and solve problems and how important is a good theoretical base. These lessons were essential to this work. My mother for her dedication and for all she done to me in different aspects. But I would list some that contributed to this accomplishment: Persistence, practical view and how to obtain better results with less wastes.

Without my wife, I would probably never be here writing this work. She always encouraged me to go further than where I was. She also gave me support during this long journey. I also thank her and our two kids, Lara and Davi, for our mutual happiness and love.

Professor Plácido has advised me since the undergraduation course. He not only gave me the necessary knowledge but developed my skills to be a better researcher and professional. Professor André has made very important considerations on this work and aggregated different points of view as coadvisor. In fact, the success of our team work comes from the combination of their two areas and professional qualities. For these years of hard but very constructive work, I express my gratitude.

Napoleão deserves my thanks too. He was always very helpful when I was in doubt and gave me good advices which contributed to our work, even when he was already in France.

I thank Álvaro, Rebecca and Rudy for working on our research group, programming, running computational tests, writting and aiding in many different tasks. Álvaro has proved to be very competent and dedicated, for example, writting highly complex programming implementations or helping me to finish our paper late at night.

I am grateful for the financial support of Capes during this postgraduate course as well as Universidade de Fortaleza for accepting me and having the necessary infrastructure.

"Start doing what is necessary,
then what is possible,
and suddenly you will be doing
what is impossible."
Saint Francis of Assisi

Abstract

The integrative collaboration of Genetic Algorithms and Linear Integer Programming, methodology called Generate and Solve, has offered significant results when applied to cutting and packing domains. However, the methodology shown some deficiencies which limits its performance when applied to other domains, being one of them the remarkable density explosion. Meanwhile, we emphasize the difficulties when applied to Problems of Dynamic Coverage and Connectivity in Wireless Sensor Networks (WSN).

In this work, the methodology has been adapted and enhanced aiming its application to Problems of Dynamic Coverage and Connectivity in Wireless Sensor Networks (WSN) in order to obtain notable flexibility, highlighting the good computational results like the lifetime extension of the wireless sensor network planned by the methodology in 150% and with problem instances with a growth of the number of sensor nodes in 125%. Significant computational results for cutting problems has been also noticed on this new methodological version.

Keywords: Wireless Sensor Networks, Optimization, Hybrid Algorithms, Genetic Algorithms, Linear Integer Programming

Resumo

A colaboração interativa entre os algoritmos genéticos e programação linear inteira, metodologia denominada Gerar e Resolver, obteve significativos resultados quando aplicada a domínios de cortes e empacotamento. Entretanto, a metodologia apresentou algumas deficiências, as quais limitam seu desempenho quando aplicada aos outros domínios, sendo uma delas o insigne crescimento da densidade. Neste ínterim, ressaltam-se as dificuldades quando aplicada a problemas de alocação e planejamento dinâmico de cobertura e conectividade em Redes de Sensores Sem Fio (RSSF).

Neste trabalho, a metodologia foi adaptada e aperfeiçoada objetivando sua aplicação a problemas de alocação e planejamento dinâmico de cobertura e conectividade em redes de sensores sem fio (RSSF) de forma a obter notória flexibilidade, destacando-se os bons resultados computacionais como o aumento de tempo da vida da rede de sensores sem fio planejada pela metodologia em 150% e com instâncias de problemas com um aumento de número de nós sensores em 125%. Nesta nova versão metodológica observou-se também significativos resultados computacionais aos problemas de cutting.

Palavras-chave: Redes de Sensores sem Fio, Otimização, Algoritmos Hibridos, Algoritmos Genéticos, Programação Linear Inteira.

Contents

List of Figures

List of Tables

List of Acronyms

WSN - Wireless Sensor Network

GA - Genetic Algorithm

ILP - Integer Linear Programming

NP-hard - Non-deterministic polynomial-time hard

NP-Complete - Non-deterministic polynomial-time complete

UML - Unified Modeling Language

SRI - Solver of Reduced Instances

GRI - Generator of Reduced Instances

JVM - Java Virtual Machine

XML - Extensible Markup Language

BSC - Base Station Controller

List of Publications

AGUIAR, A. B. de et al. A hybrid methodology for coverage and conectivity in wireless sensor network dynamic planning. In: ACHI, L. S.; SANT'ANNA, A. P. (Ed.). XLI Simpósio Brasileiro de Pesquisa Operacional, 2009. Porto Seguro: Instituto Doris Aragon, 2010. (Anais do XLI Simpósio Brasileiro de Pesquisa Operacional). To appear.

AGUIAR, A. B. de; PINHEIRO, P. R.; COELHO, A. L. V. Optimizing energy consumption in heterogeneous wireless sensor networks: A novel integer programming model. In: VI International Conference on Operational Research for Development - ICORD2007. Fortaleza, 2007. (Proceedings of the VI International Conference on Operational Research for Development - ICORD2007), p. 496-505

AGUIAR, A. B. de et al. Scalability analysis of a novel integer programming model to deal with energy consumption in heterogeneous wireless sensor network. Communications in Computer and Information Science, Springer Berlin Heidelberg, Luxembourg, France, v. 14, p. 11-20, 2008.

AGUIAR, A. B. de et al. Applicability of a novel integer programming model for wireless sensor networks. International Journal of Computer Science and Information Security, v. 3, n. 1, p. 7-13, August 2009.

Introduction

For a long time sensoring a large observation area with rich and accurate data was not feasible. The devices involved were not cheap enough to be deployed in an expressive quantity that would cover the entire region with a reasonable cost. The power supply was another big issue since these devices had a high energy consumption.

With the advent of microprocessor large scale production this scenario started to change. They are very small, cheap, with low energy consumption and capable of executing relatively complex algorithms.

Microprocessed sensor nodes can be equipped with sensor boards capable of measuring many phenomena such as temperature, light, ambient sound, electromagnetic field and humidity. These phenomena relate to the perception of a certain coverage area around the sensor node. The variance of these phenomena along the time determines how often the measures have to be sampled. The total coverage area is the union of all individual coverage areas, so the positioning of each active sensor node is very relevant to a good design.

Radio-frequency transceivers allow these sensor nodes to exchange messages directly or via routing nodes and transmit the sensoring data to processing machines through sink gateways. These links create a cooperative wireless sensor network (WSN). Often there is a dynamic routing algorithm that reconfigures the logical network topology after failures. However, there must be feasible solutions for the routing problem. Otherwise, the network segments.

And even though the energy consumption of each device is considerably low the batteries have a limited capacity. When this lifetime implies the definitive starvation of the network, all resources are important to preserve as long as possible the investment done in the WSN.

Motivation

The dynamic planning for coverage and connectivity in wireless sensor network presented by Nakamura et al. [1] offers a good tool for dealing with this energy optimization need respecting the coverage and connectivity constraints. However, the use of Integer Linear Programming (ILP) by itself executed by a solver library is not capable of dealing with larger scale problem instances as desired.

A recently proposed hybrid methodology called Generate and Solve was sucessful in extending the boundaries of classical benchmarking problems such as cutting & packing [2, 3]. It combines Genetic Algorithm (GA) and ILP cooperatively to handle larger problem instances.

This wireless sensor network problem domain is a good opportunity to extend and adapt this methodology so that it can be used in different problem domains. It is much independent from the original domain and is full of particularities. The process of adaptation drives the need of improvements as a natural evolution and maturation.

Objective

The main objective is to adapt the Generate and Solve methodology to deal with the coverage and connectivity in wireless sensor network dynamic planning, which is a new and different problem domain, in order to give it flexibility, amplitude of application, enhancements, corrections and improving its high effectiveness.

Results

This work has achieved to the following results:

An evidence that this methodology can be used in domain problems other than cutting and packing;

Better vision on how to adapt to new problem domains;

Improvements of some drawbacks;

New complementary techniques;

Implementation as a flexible and extensible framework.

Document Structure

This document is divided in 4 chapters. Chapter 1 introduces the two main optimization approaches that are used in the Generate and Solve methodology. The core of this methodology is presented in Chapter 2. The density explosion and how it was controlled is discussed in Chapter 3. Finally, the major contribution of this work is described in Chapter 4, where the methodology is adapted and applied to a Wireless Sensor Network problem. The conclusion and future works are in the last chapter.

1 Optimization Methods

1.1 Linear Integer Programming

Linear Programming is an exact mathematical methodology of optimization. It can be used to solve problems in a broad variety of areas. Business, economics and engineering are the most extensive uses of linear programming. Planning, routing, scheduling, assignment and design problems are good examples of use in transportation, energy, telecommunications and manufacturing sectors, among others.

The problem is written as a mathematical model which has linear objective function and linear inequalities that represents the constraints. The objective function will be minimized or maximized, subject to the constraints.

The canonical form of a linear model is:

$$\begin{aligned} \text{maximize } & c^T x \\ \text{subject to } & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1.1}$$

where x is the vector of decision variables, c and b are vectors of coefficients and A is a matrix of coefficients.

To solve linear problems, some algorithms have been developed. The first one was the Simplex Method, created by George B. Dantzig in 1947. Afterwards, in 1974, Leonid Khachiyan presented another algorithm for solving thsss models. It is the Interior Point Method. The state-of-the-art of linear programming algorithms and can solve the problems in a polynomial time.

One simple classical linear programming problem is the diet problem. It can be formulated as following. Consider a set of foods F and a set of nutrients N. Each food $i \in F$ has a cost c_i and a minimum a_i and maximum b_i desired amount per day. Each nutrient $j \in N$ has a minimum amount required d_j and a maximum amount allowed per

day e_j. f_{ij} represents the amount of nutrient j present in food i. The decision variables x_i determine how much each food i has to be eaten in order to satisfy the nutrition requirements. The objective here is to minimize the total feeding cost.

$$\begin{aligned} \text{minimize } z &= \sum_{i \in F} c_i x_i \\ d_j \leq \sum_{i \in F} f_{ij} x_i &\leq e_j, \forall j \in N \\ 0 \leq a_i \leq x_i &\leq b_i, x_i \in \mathbb{R}^+, \forall i \in F \end{aligned} \tag{1.2}$$

When the decision variables have to be integers the problem is said to be an Integer Linear Programming problem [4]. In this case the algorithms used to work with linear programming cannot be used in their pure form. To deal with the integrality constraints some new algorithms have born like Branch-and-Bound, Branch-and-Cut, Branch-and-Price and Cutting-plane method. Integer programming problems are often NP-hard (non-deterministic polynomial-time hard) problems in practical cases.

A particular case of Integer Programming problems is the one where the decision variables are required to be 0 or 1. These problems are called 0-1 Integer Programming or Binary Programming problems. Another common case mixes integer and linear decision variables. This is called Mixed Integer Programming. All these problems require Integer Programming algorithms to satisfy the most restrictive integrality constraints.

As an example of an Integer Programming problem, the classical knapsack problem [5], is presented here.

Consider A as the set of types of items to be possibly carried in a knapsack. Each type of item i has its weight w_i, value p_i and its associated decision variable x_i that represents the integer amount of items of that kind that will be in the knapsack. The objective is to carry the items that gather the highest value, subject to the knapsack weight capacity limit W. The problem can be formulated as:

$$\begin{aligned} \text{maximize } z &= \sum_{i \in A} p_i x_i \\ \text{subject to } \sum_{i \in A} w_i x_i &\leq W \\ x_i &\in \mathbb{Z}^+, \forall i \in A \end{aligned} \tag{1.3}$$

1.2 Genetic Algorithms

Genetic algorithms (GA) are search and optimization techniques inspired in the metaphor of biological evolution [6, 7]. Although a GA is an heuristic method which does not guarantee finding the optimal solutions, they can be found in many cases.

Like its biological counterpart, a GA is based on a population that continuously evolve through the operations of selection, crossover and mutation of individuals. These individuals represent possible solutions (phenotypes) through encoding of their characteristics in genes (genotypes). The gene encoding is not restricted to a small set of possibilities. However, binary encoding is often used to represent the solution parts. Each individual has an associated fitness value calculated in the evaluation operation, which guides the evolution process.

Figure 1 shows the basic Genetic Algorithm flow chart. As first step, the initial population is generated, usually randomly. Some kind of heuristic can be used to create good starting points, although in many situations, the gain incurred may be not expressive. The number of individuals affects both exploration and exploitation aspects of the search. After that, every individual is evaluated.

The population is evolved in generations. In each generation some individuals are selected for creation of the new generation. This selection is based upon the fitness of individuals, but stochastically. It means that the fittest individual has a larger probability of selection, although its selection is not guaranteed. On the other hand, the individual with the smallest fitness can be selected, but it is probabilistically difficult. There is a variation on the canonical algorithm that always preserves the best (fittest) individual of each generation. This mechanism is called elitism. This resource is frequently used and avoid loosing good solutions that were already found.

The selected individuals are then recombined using crossover to create their offspring. Usually two parents are mate to create two offspring. This operation exchanges parts of the genetic code and hopefully good blocks will be justaposed influencing positively the individual's fitness. This movement guides the evolution and frequently increases the average population fitness.

Usually the individuals have a low probability of being affected by mutation. When it happens, some sort of perturbation is applied to the generic material. This operation allows exploration of new different areas of the search space.

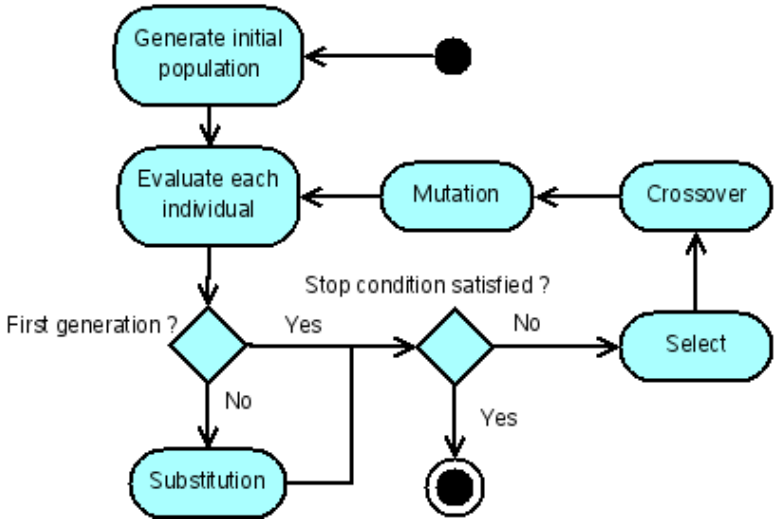

Figure 1: Basic Genetic Algorithm.

This cycle repeats until one stop condition is found. GAs can be guided by many types of conditions such as the number of generations, the total time, an absolute fitness limit, the lack of evolution during some timeframe, etc.

At the end, the individual with the best fitness represents the best solution found so far. Since GAs are heuristical methods, there is no way to know if this solution is an optimum one. However, good solutions can be found in problems where exact methods fail.

There are many possibilities for customizing a genetic algorithm to a specific problem. One of the strong characteristics of genetic algorithms is their high flexibility.

2 The Generate and Solve methodology

2.1 Outline of Some Hybrid Metaheuristic Approaches

Over the last years, interest in hybrid metaheuristics has risen considerably among researchers in combinatorial optimization. Combinations of methods such as simulated annealing, tabu search and evolutionary algorithms have yielded very powerful search methods [8]. This can be evidenced by the diversity of works about this topic found in the literature. Martin et al. [9], for instance, introduced a hybrid approach that combines simulated annealing with local search heuristics to solve the traveling salesman problem. Mahfoud and Goldberg [10] presented a hybrid method that applies simulated annealing to improve the population obtained by a genetic algorithm. Chu [11] uses a local search algorithm, which utilizes problem-specific knowledge, that is incorporated into the genetic operators of a GA instance to solve the multi-constraint knapsack problem.

It is only rather recently that hybrid algorithms which take ideas from both exact and heuristic local search techniques have been proposed. These techniques are traditionally seen as pertaining to two distinct branches of research toward the effective solution of combinatorial optimization problems, each one having particular advantages and disadvantages. Therefore, it appears to be a straightforward idea to try to combine these two distinct techniques into more powerful algorithms [12].

There have been very different attempts to combine strategies and methods from these two scientific streams. Some of these hybrids mainly aim at providing optimal solutions in shorter time, while others primarily focus on getting better heuristic solutions [13]. For instance, in order to reduce the search space, Vasquez and Hao [14] combined tabu search with an exact method for solving the 0-1 multidimensional knapsack problem. By other means, Cook and Seymour [15] proposed a two-phase hybrid method where high quality tours for the traveling salesman problem are generated, and the subproblem induced by

the set of previous tours is solved exactly on the restricted graph.

Talbi [8] has recently presented a taxonomy of hybrid metaheuristic components, which distinguishes the hybridization into two levels. In the low-level scheme, the result is a functional composition of a single optimization method. In this hybrid class, a given function of a metaheuristic is replaced by another metaheuristic. Conversely, in the high-level scheme, the different metaheuristics are self-contained, and there is no direct relationship to the internal workings of the others. By other means, Puchinger and Raidl [13] proposed an alternative classification of existing approaches combining exact and metaheuristic algorithms for combinatorial optimization, which distinguishes the following two main categories: (i) collaborative combinations; and (ii) integrative combinations. By collaboration, it means that the constituent algorithms exchange information to each other, but none is part of the other. The algorithms may be executed sequentially, intertwined, or in parallel. By integration, it means that one technique is a subordinate component of the other. Thus, there is the distinction between a master algorithm – which can be either an exact or a metaheuristic algorithm – and a slave algorithm. An alternative classification of hybridization methods can be found in the work written by Raidl [16].

Few applications of hybrid algorithms have been made yet to the Container Loading Problem. In one of these works [17], Interior Point Algorithms and metaheuristics were mixed to solve the unidimensional Knapsack Problem. In the first moment the Interior Point Method is executed with an early stop to obtain an initial solution. By applying different heuristic modifications many feasible solutions are obtained from that solution to generate an initial population. Then an evolutionary algorithm is applied. This is a typical sequential collaboration.

Puchinger et al. [18] created a mixed method of Genetic Algorithm and Branch-and-Bound to solve a bi-dimensional Cutting Problem. The Genetic Algorithm proposed in that work uses a representation based on order that is decoded using a Greedy Heuristic. Eventually, according to a determined probability, the exact algorithm is applied inside the greedy heuristic to find local optimal solutions of cutting patterns. In this case an integrative combination is used where the exact algorithm incorporated into the metaheuristic works as a slave procedure.

Terno et al. [19] merged an exact method and a greedy heuristic proposing an efficient hybrid method to the Container Loading Problem. First a feasible solution is obtained by a greedy heuristic. The Branch-and-Bound is applied afterwards to enhance this solution.

A time threshold is utilized to avoid infinite execution of the exact algorithm.In this approach local search techniques are incorporated into the Branch-and-Bound that acts as a master algorithm to the hybrid mechanism.

Quintão et al. [20] use a hybrid method to deal with coverage and connectivity in wireless sensor networks problem. It consists on decomposing the problem into two parts. A GA works with the coverage part but neglects the connectivity. To fix the connectivity failure occurrences, the Prim and Dijkstra's algorithm is used inside the GA. Like Menezes et al. [21] and other works, it does not deal with time schedules, which limits the WSN lifetime extension. For a better discussion about dealing with temporal dimension in these problems, see Subsection 4.3.3.1.

2.2 The Original Generate and Solve Methodology

Although distinct, both the exact and metaheuristic approaches have pros and cons when dealing with hard combinatorial optimization problems. But their hybridization, when properly done, may allow the merging of their strong points in a complementary manner. For instance, it is well-known that the direct application of exact methods is only possible for limited-sized instances. However, the size and complexity of the optimization problems faced nowadays have increased a lot, demanding for the development of new methods and solutions that can find acceptable results within a reasonable amount of time.

In this regard, it has become ever more evident that a skilled combination of concepts stemming from different metaheuristics can be a very promising strategy one should resort to when having to deal with complicated optimization tasks. The hybridization of metaheuristics with other Operations Research techniques has been shown great appeal as well, as they typically represent complementary perspectives over the problem solving process as a whole. In general, combinations of components coming from different metaheuristics and/or from more conventional exact methods into a unique optimization framework have been referred to by the label of "hybrid metaheuristics" [8, 12, 16, 22].

In this context, a new hybrid methodology called Generate and Solve has been recently introduced in the literature [2, 3], trying to push forward the boundaries that limit the application of an exact method through the decomposition of the original problem into two conceptual levels. According to the framework underlying this heuristic methodology (see Figure 2), the exact method (encapsulated in the Solver of Reduced Instances (SRI)

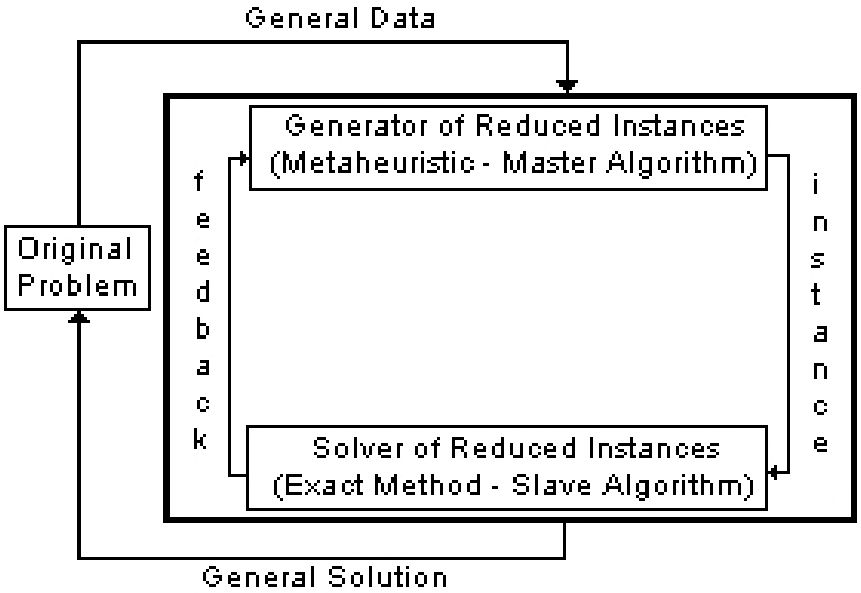

Figure 2: The generate and solve framework under investigation.

component) works no more with the original problem but with reduced instances (i.e. subproblems) of it that still preserve its conceptual structure. By this means, an optimal solution to a given subproblem will also be a feasible solution to the original problem. On the other hand, the metaheuristic component of the framework works on a complementary optimization problem, that is, the design of reduced instances of the original problem formulated as mathematical programming (viz., ILP) models. It is referred to as the Generator of Reduced Instances (GRI), whose goal is to determine the subset of points of the reducible structure that could derive the best subproblem instance; that is, the subproblem which, when submitted to the SRI, would bring about the feasible solution with the highest possible objective function value. In this scenario, the objective function values of the solutions that could be realized by the solver are used as figure of merit (fitness) of their associated subproblems, thus guiding the metaheuristic search process. The interaction between GRI and SRI is iterative and repeats until a given stopping condition is satisfied. If the first execution of SRI returns an infeasible problem status, the whole algorithm is aborted and a proper message is prompted to the operator.

So far, the metaheuristic chosen to implement the Generator of Reduced Instances has been a GA [6]. This option is due mainly to the good levels of flexibility and adaptability exhibited by the class of evolutionary algorithms when dealing with a wide range of optimization problems [23]. The genetic representation of the individuals (chromosomes) follows a binary encoding that indicates which decision variables belonging to the reducible

structure will be kept in the new subproblem to be generated. That is, those genes having '1' as alleles define the subset of variables that generates the reduced instance. Conversely, the exact method is assumed to be any state-of-the-art algorithm used to solve Mixed Integer problems, such as Branch-and-bound or Branch-and-cut [4]. Usually, the solver libraries available incorporate sets of strategies, heuristics, and problem reduction techniques that complement the main exact method and enhance its performance.

According to the classification proposed by Puchinger and Raidl [13], the methodology falls into the category of integrative combinations. The quality of the solutions to the instances generated by the metaheuristic is determined when the sub-problems are solved by the exact method, and the best solution obtained throughout the whole metaheuristic process is deemed to be the final solution to the original problem.

Although showing remarkable levels of performance for some case problems studied in the realm of cutting & packing problems [2, 3], the original version of the aforementioned methodology has drawbacks, some of which are circumvented with the adoption of the mechanisms discussed in this thesis. Another impactant factor that must be noticed is that the original version addressed only the cutting and packing problem class. One consequence of this particularity is that it requires some changes in order to be adapted to new optimization problem classes, as described in Chapter 4

2.3 Parameters and Calibration

As we incorporate more algorithms and heuristics into this methodology the need of parametrization becomes more clear. Parameters allow the variation of many aspects of the execution behavior. Some of them represent tradeoffs between opposite goals. Others only maximize the efficacy and/or efficiency. Problem instances can demand particular values for a given parameter but it is not rare to find parameter values that work well for all problem instances. This is the list of the parameters with some information like its influence, correlation with other parameters and tuning procedure.

2.3.1 Common Parameters

These parameters are used in all applications of this methodology. All of then are related to GA but ST (Solver Timeout), which was created to add more control of the SRI execution.

2.3.1.1 Elitism - EL

It is a GA control parameter [6]. It enables or disables the standard elitism behavior. When enabled the best individual is always preserved through the generations. It increases the evolving speed of the evolutionary process by avoiding the eventual loss of good individuals on the recombination operator. In the empirical experiments it often presented gains in evolving speed without any side-effects.

2.3.1.2 Population Size - PS

This parameter controls how many individuals the GA population [6] will have since there is no population size variation mechanism in this implementation. This parameter is first used in the generation of the initial population. After that, all operators will be influenced because they have to deal with all individuals of the population. There is a tradeoff for this parameter. When the population size is increased there will be more diversity on the function landscape exploration. This increases the probability of finding good local maxima. The negative effect of this population size increase is that the evolution cycle will be slower. It means that in a fixed time interval the larger is the population size the less is the number of generations evolved. This parameter has direct correlation with the parameter MNG (described afterwards) since it affects the total generations in a given time period. Varying this parameter within the range between 10 and 150 in steps of 20 showed that the best results are usually found with the value 50. This parameter value suits well to the whole set of problems and its variation on a narrow range in the pursuit of the best tuning presented no benefits.

2.3.1.3 Crossover Rate - CR

The crossover rate is an important GA control parameter [6]. It determines the probability ($0 < c_r \leq 1$) of each individual to be selected for participating in a crossover and generating new offsprings. The individuals that are not recombined by crossover generate identical offsprings with its original genetic information.

2.3.1.4 Mutation Rate - MR

Mutation rate is another GA control parameter [6]. Each gene of each individual can have its value modified with a probability given by m_r ($0 < m_r < 1$). In the binary encoding used here the value is toggled if this gene is affected by the mutation. The

mutation operator has an important role in GA exploration creating new opportunities for discovering good regions in the function landscape. While small values reduces the exploration, large values may lose good individuals reducing the algorithm's efficacy. In the case of Generate and Solve, values in the range of 0.02 to 0.1 have led to very similar results. Values above 0.1 tends to slow evolution and 0.2 or greater turn GA ineffective. The value 0.05 has given good results in all problem instances considered so far.

2.3.1.5 Total Time to Stop - TTS

This parameter determines the maximum amount of time to be considered when executing the GA. If the time counter reaches the limit imposed by this parameter the execution is aborted considering the best solution found until this time as the final one. It is determined by the user accordingly to his time availability.

2.3.1.6 Max Number of Generations - MNG

This is a stopping condition commonly used in GA. Every time a generation cycle is finished a generation counter is incremented. When this counter reaches MNG the GA stops working.

2.3.1.7 Max Number of Non-evolved Generations - MNNEG

This is another classical stopping condition. Its meaning is that if the best individual remains the same after MNNEG generations then it is considered that it is not worth continuing the GA execution. To adjust this parameter we must increase TTS to a higher value. Usually, twice the original value is enough. In 10 executions it is possible to infer a good value for MNNEG that would reduce the execution time to a value lesser than increased value of TTS without losing any best individual enhancement.

2.3.1.8 Solver Timeout - ST

The original version of Generate and Solve has shown a problem that happens eventually but used to affect adversely. Some individuals which represents subproblems with high complexity could cause the algorithm to spend much time solving and evoluting then than the others. The cost-benefit of this individuals are prohibitive. To avoid this time waste a solver timeout mechanism has been implemented. The solver has an amount of ST seconds to finish its execution. If the execution time exceeds ST a timeout occurs and

the solver execution is aborted. In this situation the individual has a null fitness value assigned because its computation is considered infeasible in the tolerable time interval. Due to the implementation of the solver library that was chosen, the precision of this control is low for small values. The solver uses a callback function to give the application an oportunity for evaluating the execution scenario. This is where the algorithm can abort its execution if the elapsed time overcomes ST. But the intervals between the callback function executions are not as short as desired and it varies in a wide range. But this behavior is not critical. The solver executions subject to intervention are the few ones that are outliners. Its execution times have often an order of magnitude above the normal ones. An average delay of about 20 seconds is not significant compared to tens of minutes of this outliers. It is easy to find good values to ST since the outliers cause the algorithm to spend much more time than the normal individuals. It is just a matter of observation of the solver execution times and the rule is to choose a value slightly higher than the larger ones.

2.3.2 Cutting and Packing Problem Specific Parameters

Since a new density control operator (Section 3.1) has been included in this methodology, new parameters had to be adjusted to better suit the instance variations. These parameters are described in Section 3.2.

2.3.3 WSN Problem Specific Parameters

Section 4.2 describes these parameters after the explanation about the new scenario full of changes which impacts directly on the density control concept.

2.4 Final Considerations

There are many possibilities in the area of hybrid optimization approaches. Integer Linear Programming and Genetic Algorithms can be combined to complement each other and produce good results. The original Generate and Solve methodology as presented in [2,3] confirms this potential. However, this was only the initial version of this methodology, designed originally to deal with cutting and packing problems. It has some drawbacks and good opportunities for enhancements.

One problem that used to limit the number of GA generations was the upward trend

of chromosome density. The new density control operator described in Section 3.1 keeps this density near a desired ideal parametrized value and avoid the premature end of GA evolution process.

Substantial changes have been made to add flexibility to this methodology. These enhancements are described in chapter 4.

3 Avoiding the Density Explosion

One drawback that has limited the effectiveness of the original Generate and Solve methodology as presented in Section 2.2 relates to its propensity for bringing about an uncontrolled density explosion over the individuals (i.e. reduced instances of the original problem) produced by the GRI (Generator of Reduced Instances). We define "density of an individual" as the ratio between the number of genes having '1' as allele (referred to as activated) and its total length. The fact is that an increase in density tends to generate subproblems closer to the original problem, thus possibly yielding better solutions. This situation can be better pictured as if having some sort of an "attractor" pushing the overall population density up as the GRI implemented by GA (Genetic Algorithm) evolves. Although expected, this phenomenon may have an undesirable side-effect when it occurs prematurely. This is because, usually, high densities imply higher complexity to be dealt with by the SRI (Solver of Reduced Instances), which indirectly affects the search process conducted by the GRI as the time spent in each generation tends to become progressively higher. This may cause a drastic limitation over the number of search iterations performed by the GRI, hindering both the effectiveness and efficiency of the whole optimization.

3.1 Density Control Operator

In order to circumvent this problematic tendency, we have devised a novel operator, named as "density control operator" (DCO), to be introduced into the GRI and invoked after the application of the recombination operator (crossover). By adopting this new operator, more feedback information from the SRI to the GRI is allowed, not only the objective function values of the solutions as in the original version of the framework. To each activated gene in a given GRI individual that represents a variable effectively used in the solution generated by the SRI, a credit value is assigned which will be taken into account by the new operator. This credit value amounts to the objective function value of the SRI solution. Non-activated genes receive no credit.

One parameter directly related to DCO is the "ideal density" (IDD) – see Subsection 3.2.2 –, which should be calibrated beforehand. During the density control operation, the following analysis is conducted over each GRI individual c_i. If its density is lower than or equal to IDD, nothing is done. Otherwise, some genes of c_i having null credit values are randomly chosen to be deactivated (i.e., zeroed), triggering the application of the density reduction algorithm (Algorithm 1). After this stage, if the current density value of c_i is still greater than IDD, other genes with some associated credit will also suffer deactivation, those with smaller credits being deactivated first. This last procedure is optional and its execution is controlled by another parameter, PNZCG, described in Subsection 3.2.3.

Variables and functions used on the Algorithm 1:

toBeCleared: Number of genes to be cleared;

activeGenes(x): Returns the number of genes that have its allele equals to 1;

noCredit(x): Returns the number of genes that have its allele equals to 1 and no credit;

clearGene(x): Clear an arbitrary gene that has no credit;

Algorithm 1 Density Reduction Algorithm

$toBeCleared \leftarrow activeGenes(x) - IDD * sizeOf(x)$
$clearedGenes \leftarrow 0$
while $noCredit(x) > 0$ and $clearedGenes < toBeCleared$ **do**
  $clearGene(x)$
  $clearedGenes \leftarrow clearedGenes + 1$
end while

One could picture this scenario as if having two "forces" driving the metaheuristic search toward the ideal density: one propelled by the evolutionary process on its own, rising up the density levels of its individuals, and the other performed by the density control mechanism, pushing the levels down. We feel that a proper balance between these two forces may incur better levels of performance to the methodology as a whole.

For the sake of clarity, a simple example follows. Suppose we have two individuals c_1 and c_2 (Tables 1 and 2). The blue (**bold**) and green (***bold and italic***) values indicate the activated genes of c_1 and c_2, respectively. While the fitness value of c_1 is f_1, that of c_2 is f_2, with $f_1 > f_2$. The second line of each chromosome shows the credit assignment to the genes.

Now suppose that after crossover two offspring were generated, c_3 and c_4 (Tables 3 and

4). In this example the single point crossover is used in the midpoint of the chromosomes. In c_3 and c_4, the blue (**bold**) values show the genes that came from c_1 and the green (***bold and italic***) values those that came from c_2. The red (**bold and underlined**) genes have no credit and so are candidates for being cleared by DCO until the individual's density reaches the IDD.

Table 1: Chromosome c_1

Allele	**1**	0	0	1	0	1	**1**	0	**1**	1
Credit	f_1	0	0	0	0	0	f_1	0	f_1	0

Table 2: Chromosome c_2

Allele	0	0	1	***1***	1	0	0	***1***	0	1
Credit	0	0	0	f_2	0	0	0	f_2	0	0

Table 3: Chromosome c_3

Allele	**1**	0	0	**1**	0	0	0	***1***	0	**1**
Credit	f_1	0	0	0	0	0	0	f_2	0	0

Table 4: Chromosome c_4

Allele	0	0	**1**	***1***	**1**	**1**	**1**	0	**1**	**1**
Credit	0	0	0	f_2	0	0	f_1	0	f_1	0

3.2 Cutting and Packing Problem Specific Parameters

Since the density control operator (Section 3.1) was included in this methodology, new parameters had to be adjusted to better suit the instance variations. These parameters are described here.

3.2.1 Initial Density – IND

This parameter stipulates a yardstick for the density levels of the individuals of the first generation. That is, it is expected that the average population density reaches closely this mark, although some variation is allowed considering each individual alone. This is an important parameter to be set: If IND is too low, the time spent (efficiency) in the first generations of the GRI should be kept small but also will be the average population fitness value (effectiveness); on the other hand, if it assumes a high value, the inverse should be true. So, the tuning of this parameter should take into account the intricacies

of the problem instance at hand in order to strike a good balance between efficiency and effectiveness criteria. This parameter is closely related to IDD, which is described below.

3.2.2 Ideal Density – IDD

As mentioned before, this parameter is a threshold used to trigger the density reduction algorithm (Algorithm 1), exerting a direct influence over the time of optimization effectively spent by the GRI or SRI components. Low values of it should accelerate the GRI's evolutionary process but also restrict the possibilities available to the solver. Increasing the IDD value should approximate the density of the reduced instances to that of the original problem, providing more "opportunities" to the solver for generating high-quality solutions; however, this decision may incur an increase in the time elapsed from one generation to the other in the GRI. One should visualize the IND parameter as a sort of "first-shot version" of IDD, although they do not need to share the same value. The average population density could start at a high level, because of a high IND, but then decrease swiftly or smoothly, due to the influence of a lower IDD value. The opposite may also happen. Since the PNZCG parameter (discussed below) affects the outcome of the application of the density reduction algorithm, the calibration of this parameter should be done in concert with that of IDD. For instance, if high values of IDD are chosen and PNZCG is on, an undesirable problem may occur: An increase in the number of solver timeouts due to an increase in the number of individuals with very high density values. So, if the adoption of PNZCG is in demand, a lower value of IDD will be a better option.

In order to find out a proper value for IDD, we have come up, after some experimentation with different cutting problem instances, with the following rule of thumb: Start with a low IDD value and then increase it until the limit of nearly 10% of the solver's timeout rate is reached. At this point, the solver will work closely to its operational limit but without harming too much the performance of the GRI. We have found that this rule provides an adequate balance between the activities of the two components of the framework.

It is important to note that some less complex problems can be solved by setting the value of IND and IDD equals to 1 (full density). This means that the GRI will not complete even the first evolution cycle, since the first individual evaluated will be the original problem itself and its solution will be the optimal value. The SRI will solve the original problem as an ILP approach, which is a particular case of the Generate and Solve methodology.

3.2.3 Preservation of Non-Zero Credit Genes – PNZCG

As discussed earlier, the DCO has an optional behavior. In this context, PNZCG is a boolean parameter that enables or disables the execution of the last part of the density reduction algorithm. When enabled it turns the density control process more restrictive by allowing no individual to have a density value greater than IDD. This decision should provide a more effective control over the density explosion problem. On the other hand, high-fitness individuals with high-density values may lose possibly-good genetic material if PNZCG is activated. So, we have reached the conclusion that the decision whether enabling this parameter or not directly relates to the complexity of the problem under consideration. For instance, in the experiments reported in the next section, we have kept it disabled when coping with the "FekSch" class of problem instances but enabled when dealing with the "Ngcutfs" class.

3.3 Application

In order to benchmark this new version of the Generate and Solve methodology against the original version and some other approaches, a classical problem was used. Until this version, the concept of this methodology was totally focused on the cutting and packing problem family. This limitation has been overcome due to some other improvements as described in Chapter 4. The Constrained Non-guillotine Cutting Problem is a good problem within this problem class to be used as benchmark because there are many benchmarking problem instances as well as many solution approaches. Besides, there are very hard problem instances which can reach up to 10^{54} cutting patterns.

3.3.1 Constrained Non-guillotine Cutting Problem

The Constrained Non-guillotine Cutting Problem consists of cutting rectangular pieces from a single large rectangular object. Each piece is of fixed orientation and must be cut with its edges parallel to the edges of the object. The number of pieces of each type that are cut must lie within prescribed limits and, in addition, the cuts may not go from one end to another. Each piece has an associated value and the objective is to maximize the total value of the pieces cut. This problem has been shown to be NP-Complete (Non-deterministic polynomial-time complete) [24], meaning that it is impossible to find their optimal solution by resorting to an enumerative, brute-force approach alone, in a reasonable time in real applications, except in trivial cases.

Dowsland and Dowsland [24] use two combinatorial methods that generate constrained cutting patterns by successive horizontal and vertical builds of ordered rectangles. Each of the algorithms uses a parameter to bound the maximum waste they may create. Error bounds measure how close the pattern wastes are to the waste of the optimal solution. These algorithms are fast and can yield efficient solutions when applied to small problems.

Beasley [25] presented a tree search approach based upon the Lagrangean relaxation of a 0-1 integer linear programming formulation of the problem to derive an upper bound on the optimal solution. The formulation makes use of variables that relate to whether or not a piece of a particular type is cut with its bottom-left hand corner at a certain position. Subgradient optimization is used to optimize the bound derived from the Lagrangean relaxation. Problem reduction tests derived from both the original problem and the Lagrangean relaxation are given. This procedure is oriented to the solving of moderately-sized problems.

By other means, Hadjiconstantinou and Christofides [26] introduced a new exact tree-search procedure. The algorithm limits the size of the tree search by using a bound derived from a Lagrangean relaxation of the problem. Subgradient optimization is used to optimize this bound. Reduction tests derived from both the original problem and the Lagrangean relaxation version produce substantial computational gains. The computational performance of the algorithm indicates that it is an effective procedure capable of optimally solving practical cutting problems of medium size.

Fekete and Shepers [27] combined the use of a data structure for characterizing feasible packings with new classes of lower bounds, as well as other heuristics, in order to develop a two-level tree search algorithm for solving high-dimensional packing problems to optimality. In that approach, projections of cut pieces are made onto both the horizontal and vertical edges of the stock rectangle. Each such projection is translated into a graph, where the nodes in the graph are the cut pieces and an edge joins two nodes if the projections of the corresponding cut pieces overlap. They show that a cutting pattern is feasible if and only if the projection graphs have certain properties. The authors have shown that problems of considerable size can be solved to optimality in reasonable time with this approach.

In a recent paper, Beasley [28] developed a new non-linear formulation of the constrained non-guillotine cutting problem. Based upon this formulation, a population heuristic is conceived, which explicitly works with a population of solutions and combine them together in some way to generate new solutions. Computational results for that

heuristic approach on typical benchmarking test problems taken from the literature [29] are reported by the author, and are used in this work for the purpose of comparison.

Polyakovsky and M'Hallah [30] used an agent-based (A-B) implementation of a new guillotine botton left (GBL) constructive heuristic. The A-B is pseudo-parallel and its agents dynamically interacts to fill the bins. Each agent has its own parameters, decision process and fitness assessment. The GBL is sequential. It packs items into a bin and creates new bins when the current one can no longer have an item inserted.

Mi et al. [31] designed a model of GA and its operators for optimizing production costs on guillotine cutting in rail-vehicle manufacturing. The implementation was applied in a practical case. GA was used combined with the lowest-horizontal-line search algorithm.

Mahanty et al. [32] tackle a different kind of problem that belongs to the same problem family. It is the two-dimensional irregular-shaped polygonal elements packing. They propose two solutions. One that uses only a real-encoded GA and other that hybridizes this real-encoded GA with a new local optimization algorithm. This second algorithm uses the Coulomb potential technique. GA is responsible for generating the order of the polygons while the Coulomb potential algorithm chooses among the fixed combinations which one has the layout that minimizes the scrap.

Tiwari and Chakraborti [33] uses GA with binary chromosome encoding in both guillotine and non-guillotine cutting problems. However, their focus is on a multi-objective approach which minimizes the length of the mother sheet and the total number of cuts required to produce the smaller pieces. Vidyakiran et al. [34] followed a similar approach adapted to a three-dimensional problem. But there is a concept of best orientation between two cuboids, and two evolutionary algorithms were used and compared.

Mohanty et al. [35] explore the GA mechanics applied to the optimization of a hot holled coil. Six selection schemes are evaluated as well as the elitism selection scheme. The problem can be summarized as: For each given set of forecasted customer widths, an attempt is made to find a single width of the mother coil to be manufactured and kept in stock.

3.3.2 Problem Formulation

To formulate the Constrained Non-guillotine Cutting Problem, we resort to the Integer Linear Programming model proposed in [25]. Consider a set of items grouped into m types. For each item type i, characterized by its length, width (l_i, w_i), and value v_i, there is an

associated number of items b_i. Consider also a large object that has (L, W) as its length and width dimensions respectively. The items should be cut orthogonally from the object. Each 0-1 variable u_{ide} alludes to the decision of whether to cut or not an item of type i at the coordinate (d, e).

$$u_{ide} = \begin{cases} 1, \text{if an item with type } i \text{ is cut at the position } (d,\ e) \\ 0, \text{otherwise.} \end{cases} \tag{3.1}$$

d and e belongs respectively to the following sets of achievable points:

$$X = \{x \mid x = \sum_{i=1}^{n} \alpha_i l_i, x \leq L - \min(l_j), 1 \leq j \leq m, \alpha_i \in \mathbb{Z}^+\}; \tag{3.2}$$

$$Y = \{y \mid y = \sum_{i=1}^{m} \beta_i w_i, y \leq W - \min(w_i), 1 \leq j \leq m, \beta_i \in \mathbb{Z}^+\}. \tag{3.3}$$

To avoid box overlapping the incidence matrix g_{idepq} is defined as:

$$g_{idepq} = \begin{cases} 1, \text{if } d \leq p \leq d + l_i - 1 \text{ and } e \leq q \leq e + w_i - 1 \\ 0, \text{otherwise.} \end{cases} \tag{3.4}$$

that has to be computed a priori for each box type i $(1 \leq i \leq m)$, for each coordinate (d, e), and for each coordinate (p, q).

The Constrained Non-guillotine Cutting Problem can be formulated as:

$$\max \quad \sum_{i=1}^{m} \sum_{d \in X} \sum_{e \in Y} v_i u_{ide} \tag{3.5}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \sum_{d \in X} \sum_{e \in Y} g_{idepq} u_{ide} \leq 1, \forall p \in X, \forall q \in Y \tag{3.6}$$

$$\sum_{d \in X} \sum_{e \in Y} u_{ide} \leq b_i, 1 \leq i \leq m \tag{3.7}$$

$$u_{ide} \in \{0, 1\}, 1 \leq i \leq m, \forall d \in X, \forall e \in Y \tag{3.8}$$

3.3.3 Binary Chromosome Encoding

The reducible structure chosen was composed of the discretization sets X and Y. They are directly responsible for the number of decision variables and the size of the incidence matrices. Another important trait that a good reducible structure must have is the ability of generating subproblems that are feasible solutions to the original one and maintain all its characteristics. The third desirable behavior of a good reducible structure is the scalability of the complexity. The subproblem matrix sizes and, indirectly, the problem instance complexity, must have a wide range from near zero up to the size of the original problem.

So, in order to generate subproblems, the GRI must put a "mask" over the elements of these discretization sets, selecting only a subset of each. To represent this choice of including or not an element of the original discretization set in the generated reduced set, we encode genes of the chromosome that is associated to this subproblem. Each gene that has an allele equal to "1" causes the inclusion of the respective discretization set (X or Y) element to the new subset. If the allele is equal to "0", the equivalent discretization set element is not included in the new subset. This way, the chromosome has the number of its genes equal to the product of the discretization set cardinalities, $|X||Y|$.

3.3.4 Computational Results

To evaluate the potential of the new version of the Generate and Solve methodology, a series of experiments have been made and some of the results achieved so far are discussed here. We have performed experiments over several test problems taken from the literature for which the optimal solutions are already known (Beasley [28, 29]). However, we have decided to concentrate our analysis here only on those problem instances that appeared to be more complicated to be dealt with, referred to as the "FekSch" (after the work of Fekete and Schepers [27]) and "Ngcutfs" classes of problem instances.

The new version of the hybrid framework has been implemented in Java. The different classes of problems have their particularities modeled separately but afterwards integrated into a single Java project that makes use of the framework. For the computational tests reported here we have used as SRI the Integer Programming Solver library that comes with the Ilog Cplex version 10.1 64 bits [36] running atop OpenSuse Linux 11.0 64 bits version. The Java Virtual Machine (JVM) adopted was the Sun HotSpot 64 bits on Server Mode, version 1.6.0-b105, whereas the machine used for the computational tests

was equipped with an Intel Pentium Core 2 Quad 2.4GHz processor with 8 GBytes of RAM memory.

For collecting the results obtained for each problem instance, we ran the framework ten times, after which some measures related to effectiveness (best, average, and standard deviation of the objective function values) and efficiency (time elapsed) were calculated.

The parameters introduced in Section 3.1 were calibrated as follows: $IND = 0.05$, $IDD = 0.065$, and $PNZCG = false$ for the "FekSch" class of problem instances; and $IND = 0.04$, $IDD = 0.05$, and $PNZCG = true$ for the "Ngcutfs1" class. For all runs, a maximum limit of four hours on the execution time of the framework was used.

The results achieved by the novel version of the methodology are presented in Table 5, whereas Table 6 provides a contrast with previous work and Figure 3 displays an optimal cut configuration achieved for a particular problem instance, namely "FekSch04". In these tables, "Optimal" and "Best Sol." mean the objective function values of the known optimal solution (if this is the case) and of the best solution produced by all runs of the respective technique. Likewise, "Avg. Sol." denotes the average performance, whereas "Percent" indicates how close is the best or average solution to the known optimal one.

Instance	Optimal	Best Sol.	Percent	Time(s)	$Avg.Sol. \pm Std.Dev.$	Percent	$Avg.Time \pm Std.Dev.$
FekSch01	27,718	27,539	99.35%	8,027.20	$27,506.25 \pm 025.89$	99.24%	$7,433.97 \pm 1,288.27$
FekSch02	22,502	22,502	100.00%	494.20	$22,235.75 \pm 332.62$	98.82%	$9,663.89 \pm 5,618.55$
FekSch03	24,019	24,019	100.00%	178.13	$23,932.75 \pm 119.04$	99.64%	$4,464.51 \pm 4,369.70$
FekSch04	32,893	32,893	100.00%	1,832.50	$32,441.33 \pm 526.36$	98.63%	$10,368.86 \pm 4,963.37$
FekSch05	27,923	27,923	100.00%	9,461.13	$26,601.67 \pm 687.29$	95.27%	$13,623.48 \pm 2,039.45$
Ngcutfs1-1	29,955	28,032	93.58%	3,035.99	$27,918.40 \pm 186.18$	93.20%	$6,792.47 \pm 4,026.09$
Ngcutfs1-2	30,000	28,946	96.49%	2,413.59	$28,268.67 \pm 388.68$	94.23%	$6,176.59 \pm 2,458.72$
Ngcutfs1-3	30,000	27,966	93.22%	10,506.87	$27,932.25 \pm 067.50$	93.11%	$10,452.96 \pm 2,135.74$
Ngcutfs1-4	30,000	28,494	94.98%	14,401.02	$28,117.05 \pm 403.84$	93.73%	$8,289.14 \pm 4,750.73$
Ngcutfs1-5	30,000	28,677	95.59%	3,361.34	$28,677.00 \pm 000.00$	95.59%	$8,616.41 \pm 3,234.19$
Ngcutfs3-1	29,943	28,315	94.56%	14,401.60	$28,271.20 \pm 025.46$	94.42%	$14,425.22 \pm 24.07$
Ngcutfs3-2	29,991	28,046	93.51%	14,456.59	$27,905.00 \pm 062.79$	93.04%	$14,427.51 \pm 33.44$
Ngcutfs3-3	30,000	28,877	96.26%	14,403.58	$28,777.10 \pm 061.72$	95.92%	$14,421.18 \pm 23.71$
Ngcutfs3-4	28,170	26,604	94.44%	14,402.85	$26,506.20 \pm 090.36$	94.09%	$14,420.40 \pm 24.87$
Ngcutfs3-5	30,000	27,787	92.62%	14,401.28	$27,558.40 \pm 190.66$	91.86%	$14,418.32 \pm 17.08$

Table 5: Results for the constrained non-guillotine cutting problem instances

		New version			Original version [3]			Beasley's heuristics [28]		
Instance	Optimal	Best Sol.	Percent	Time(s)	Best Sol.	Percent	Time(s)	Best Sol.	Percent	Time(s)
FekSch1-01	27,718	27,539	99.35%	8,027.20	27,360	98.71%	5,773.08	27,486	99.16%	19.71
FekSch1-02	22,502	22,502	100.00%	494.20	21,888	97.27%	7,200.00	21,976	97.66%	13.19
FekSch1-03	24,019	24,019	100.00%	178.13	23,743	98.85%	5,997.21	23,743	98.85%	11.46
FekSch1-04	32,893	32,893	100.00%	1,832.50	32,018	97.34%	7,200.00	31,269	95.06%	32.08
FekSch1-05	27,923	27,923	100.00%	9,461.13	27,923	100.00%	830.53	26,332	94.30%	83.44
Average	27,011	26,975.2	99.87%	3,998.63	26,586.4	98.43%	5,400.16	26161,2	97.01%	31.98

Table 6: Comparative analysis in terms of best solution achieved

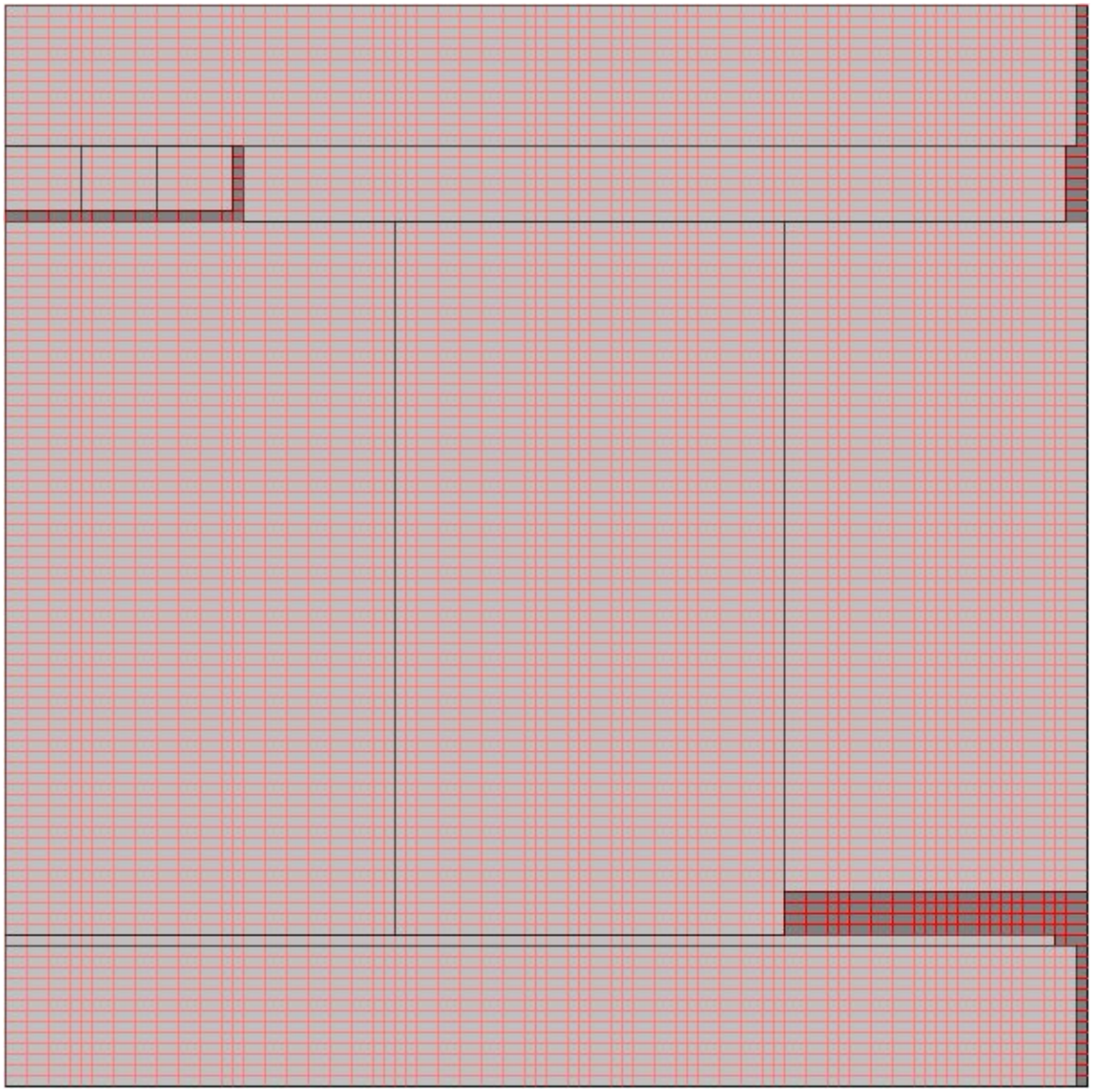

Figure 3: Cutting configuration for the optimal solution ("FekSch04")

Considering the "FekSch" class, except given to the first instance, the application of the new version of the framework could find all optimal solutions, a remarkable performance in terms of effectiveness. Actually, for the first instance, the best quasi-optimal solution achieved is very close to the true one (only 0.65% in deviation). Besides, the average solutions achieved by the new version of the methodology are better than the best solutions found by Fekete and Schepers [27] in all cases and better than those achieved by the original version of the methodology [3] in most of the cases.

The largest number of generations evolved by the GRI in one execution was 68, much higher than what could be achieved previously [2, 3]. This is another indication that the density control operation was indeed instrumental to leverage the performance of the framework. It is also worth mentioning that we have set the limit of four hours for one

execution while conducting the experiments, but we have perceived that, by eliminating this limit, the search process conducted by the GRI could have run for days.

To visualize how the search and density control processes are interrelated, refer to Figure 4, which shows how the fitness of the best individual, the average fitness of the population, the density of the best individual and the average density of the population values vary along the generations. The curves, from top to bottom, denote, respectively: the best fitness value, the average fitness value, the density of the best individual, and the average population density. The parameter IDD in this particular execution was set as 0.05. Since parameter PNZCG was kept disabled for this problem instance ("FekSch04"), the density of some fitter individuals could exceed the IDD while the majority of density values remain lower. The important point to be grasped here is that the average population density could be kept under control, oscillating between 0.2 and 0.3. In the original version of the methodology the average population density grows rapidly [2, 3].

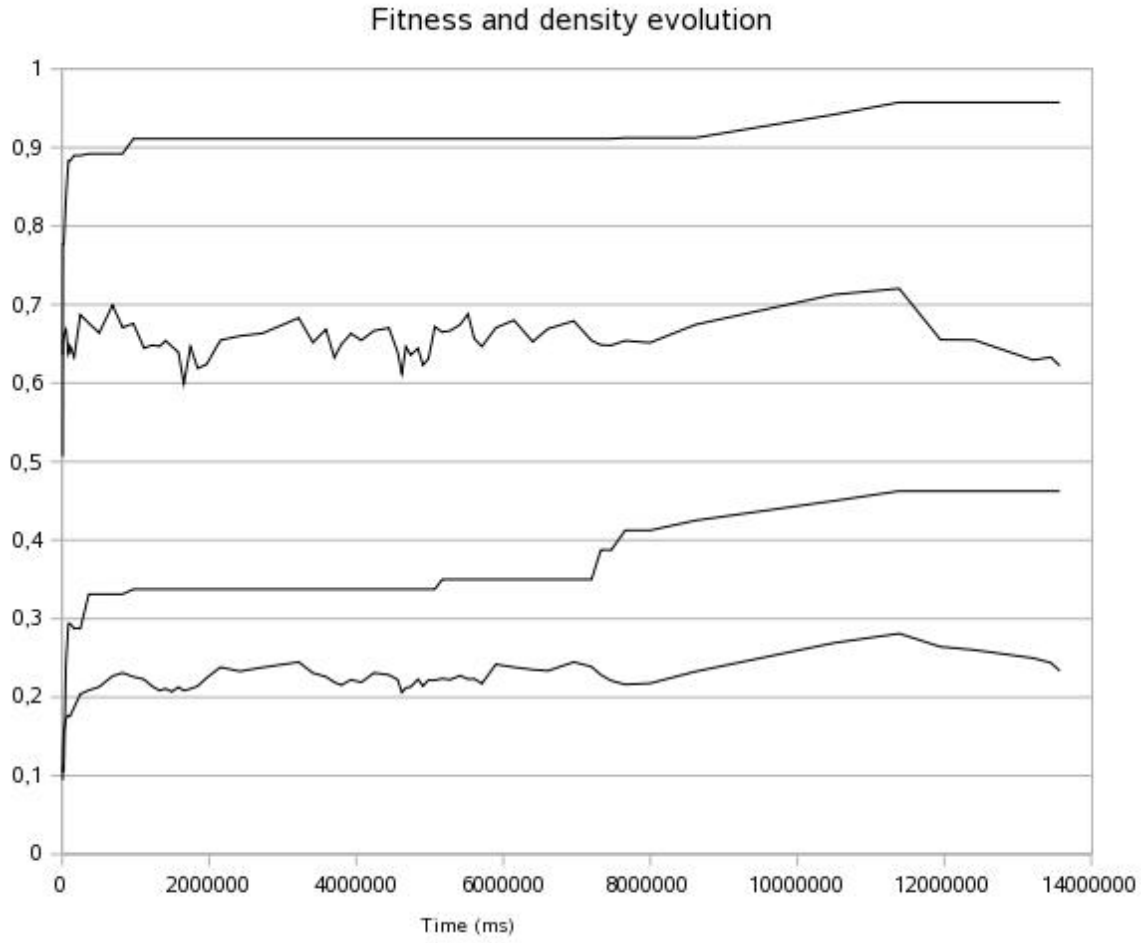

Figure 4: Fitness and density evolution for one FekeSch04 problem instance.

3.4 Final Considerations

The same problem domain used for assessing the original Generate and Solve methodology was used here to measure the benefits of the new implementation. The classical problem instances set of OR-Library [29] were chosen for benchmarking. The solutions found by this new version of the methodology proves that it can reach better results than its predecessor and Beasley's approach. The execution time could be extended or parallelism could be used to keep the running time while increasing the number of GA generations. Besides the results, the better comprehension about the particularities of Generate and Solves were the base of the improvements discussed in Chapter 4.

4 Improvements for the Problem of Dynamic Coverage and Connectivity in Wireless Sensor Networks

4.1 The Improvements

As presented by Aguiar et al. [37], adapting the original Generate and Solve methodology to be suitable for a totally different class of problem is a challenge. A very undesirable characteristic of the original version of this methodology is that its binary chromosome encoding can be prohibitively long, depending on the chosen reducible matrix. Long chromosomes can lead to convergence problems.

4.1.1 Compact Chromosome Encoding

Considering the problem presented in section 4.3, the largest structure in the model is the decision variables x. A structure like this is a good candidate for reducible structure. The GA can handle the election of which sensors in each time slot will be present in the subproblem reducing the matrix size and thus the problem complexity. These subproblem solutions remain valid for the original problem. Other important characteristic of a good reducible structure is the ability of scaling the subproblem sizes in a wide range from near zero until the original problem itself. However, a drawback with this reducible structure is its size, which is not well suited to the original chromosome encoding.

According to [6], the correct representation of the individuals is one of the most difficult steps while designing a good evolutionary algorithm.

The binary chromosome encoding was used in the original version of the Generate and Solve methodology. Each gene represents the inclusion of the equivalent element of

the reducible structure that will be considered in the generation of the new subproblem. It is well suited for the cutting and packing problem class for which the methodology was designed. This type of chromosome encoding however is not appropriate for other problem domains like the one treated in this work. It would generate too large chromosomes (i.e. 10 time intervals × 36 sensors). Table 7 shows a possible chromosome with the binary encoding. Each color represents a set of 16 genes associated with its respective sensors of each time interval.

The proposed new encoding (Table 8) represents the integer indices of the sensors that must be taken in the subproblem generation. So there is no need of representing all sensors. Only a small amount of sensors has to be considered and the length of this chromosome can be reduced to 17% of the binary encoding one.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	...

Table 7: Part of a chromosome with binary encoding

3	5	11	16	2	5	10	13	4	8	9	15	1	7	...

Table 8: Part of a chromosome with the new compact encoding

The compact chromosome encoding maintains the density control as explained in Chapter 3, but in a different manner. Indeed, the individuals now have a hard density limit which is determined by a parameter called Maximum Density (MD) as described in Subsection 4.2.1.

The important change is that there is a fix number of integer alleles to represent the sensor nodes which would be encoded as value '1' on binary alleles of the original binary encoding scheme. So there is no way of encoding more than the number of this integer alleles. Thus, the maximum density is the number of genes for this new encoding scheme divided by the total number of sensor nodes, considering all time intervals. However, the density is not fixed. Some alleles can be set to '-1' which means that it does not represent and sensor node for that time interval. This mechanism gives some flexibility to individuals for varying its density instead of suppressing this characteristic.

4.1.2 Demand Points Sampling Technique

Although practical, density parameter adjustments can lead to a reduction of the matrix X to only 16.67% of its original size; this still implies an intense computational burden for the solver and for the system as a whole. In order to scale down the subproblem complexity to a more efficient operational range, a complementary technique was developed.

The main point here is to deal with the largest of the X matrix dimensions, which is the set of demand points D. The density of those demand points in an observation area with same size is related to the accuracy of the coverage rate measure. So the smaller are the demand points density and quantity, the smaller is the problem complexity. However, the smaller will also the coverage rate accuracy be.

But there is a way of reducing the demand point set size while maintaining the original accuracy. It consists in using a sampled subset of the original set of demand points in the subproblem but recalculating the objective value a posteriori with the original set. In a $m \times n$ grid demand point set D, the subset S sampled in a ratio of 0.25 can be created as follows:

$$\begin{aligned} S &= \{d_{ij} \in D | i \in A, j \in B\} \\ A &= \{i | 0 \leq i \leq m, i = 2k + 1, k \in \mathbb{N}\} \\ B &= \{j | 0 \leq j \leq n, j = 2k + 1, k \in \mathbb{N}\} \end{aligned} \tag{4.1}$$

This way the size of the matrix X can be shrunk to only 25% of the original size.

The objective function value found by the solver will be inaccurate regarding the original problem, but it does not mean that this technique losses its original characteristics. To maintain the original accuracy, a subsequent recalculation is done. Having the solution found by the SRI, only the active sensor nodes are taken into account in each time slot. This time the original set of demand points will be considered. Each of these points is checked against the active sensor nodes whether it is inside or outside the coverage radius. Now the coverage rate can be determined as the ratio between the total number of covered demand points and the total demand points, considering all time slots.

4.1.3 Spread Sensor Heuristic in Population Initialization

The first initial population generation strategy is a randomized one. Due to the evolutionary nature of GA, the start point should not necessarily be a good one. And this was the case.

However, there is a trend that sensor nodes that are selected to participate in a sub-problem are too close to each other by comparing their coverage radius. Overlapping of coverage is desirable until a certain point because it provides diversity of choices and network reliability. But keeping sensor nodes too close can lead to a concentration of coverage in the most central part of the observation area and possibly leaving some peripheral areas uncovered. The configuration that maximizes the coverage area ratio (covered area divided by total area) with a minimum number of sensors is that where the distances between two neighbor sensor nodes are equal to two times the coverage radius (Figure 5).

The key parameter to provide good balance between reliability and the number of sensor nodes is the density. This density can be determined by the network designer based upon the requirements of the specific application. But due to irregular activation and deactivation of sensor nodes this density tends to be not so homogeneous along the observation area. As a side effect this model tries to distribute this density more homogeneously because this is the configuration where the uncovered points penalize less and the number of active sensors spending energy is reduced, minimizing the objective function.

The idea of the spread sensors heuristic is to provide a good starting point to the GA in order to reduce the time spent in the initial phase of the evolution process. It is based on the selection of sensor nodes spread along the observation area which are encoded in the chromosomes of the initial population. This heuristic criterion of sensor node selection causes the density to be more regular.

There is a risk of decreasing the genetic diversity and even of causing some genetic drift when a deterministic algorithm is used to create individuals within an initial population. To avoid this possible drawback, some sort of stochastic behavior was maintained. The algorithm is described as follows.

The population is divided into two classes of individuals: the random and the heuristic. A parameter called SSHR (see Subsection 4.2.2) controls the probability of each individual being generated by the heuristic approach. The others are created by the standard random procedure. The heuristic starts picking a random sensor node for the first gene. Each

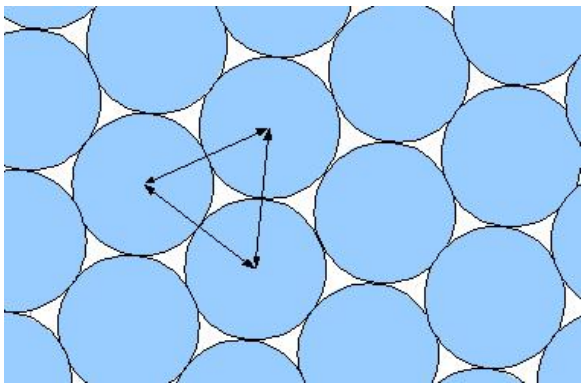

Figure 5: Maximizing coverage.

other sensor node is selected as having the largest of the smallest distance from other previously selected sensor nodes. This procedure is repeated for each group of genes that represents the set of sensor nodes of each time interval.

4.2 Wireless Sensor Network Problem Specific Parameters

4.2.1 Maximum Density - MD

Density plays an important role on efficacy since it provides the proper balance between GA and solver. Subsection 4.1.1 explains how to generate subproblems by encoding a sensor node subset for each time slot. This chromosome is composed by groups of genes. Each group stores a set of sensor node references (indices) for a specific time slot. So, the number of genes for each group determines how many sensor nodes could be present at most on this subproblem for each timeslot. This means that the maximum density is limited by the size of these groups. This is exactly what the parameter MD does: It indicates how many genes each of these groups contains. Another definition for MD is the ratio between chromosome size and the number of time slots.

This parameter has a behavior related to IND and IDD parameters described in Subsections 3.2.1 and 3.2.2.

4.2.2 Spread Sensor Heuristic Selection Rate - SSHSR

SSHSR is the probability of selection ($0 \leq sshr \leq 1$) for each GA individual to be created by spread sensor heuristic (Subsection 4.1.3). If the individual is selected, it will

be generated by this heuristic; otherwise, it will follow the standard random approach generation. This parameter is used in the population initialization phase to control the tradeoff between the benefit of this heuristic and the risk of genetic drift.

4.3 Application

The dynamic coverage and connectivity in wireless sensor networks problem (DCCP) is a very different problem class than the cutting and packing class used in the original version. This means a good opportunity to motivate changes on the Generate and Solve towards flexibility. Moreover, the publication of some papers related to this subject [38–40] brought a good understanding about how to give new contributions to this area and present better solutions.

4.3.1 The Problem of Dynamic Coverage and Connectivity in Wireless Sensor Networks

A wireless sensor network typically consists of a large number of small, low-power, and limited-bandwidth computational devices, named sensor nodes. These nodes can frequently interact with each other, in a wireless manner, in order to relay the sensed data towards one or more processing machines (also known as sinks) residing outside the network. For such a purpose, special devices, called gateways, are also employed, in order to interface the WSN with a wired, transport network. To avoid bottleneck and reliability problems, it is pertinent to make one or more of these gateways available in the same network setting, a strategy that can also reduce the length of the traffic routes across the network and consequently lower the overall energy consumption.

A typical sensor node is composed of four modules, namely the processing module, the battery, the transceiver module and the sensor module [41]. Besides the packet building processing, a dynamic routing algorithm runs over the sensor nodes in order to discover and configure in runtime the “best” network topology in terms of number of retransmissions and waste of energy. Due to the limited resources available to the microprocessor, most devices make use of a small operating system that supplies basic features to the application program.

To supply the power necessary to the whole unit, there is a battery, whose lifetime duration depends on several aspects, among which, its storage capacity and the levels of electrical current employed in the device. The transceiver module, conversely, is a

device that transmits and receives data using radio-frequency propagation as media, and typically involves two circuits, viz. the transmitter and the receiver. Due to the use of public-frequency bands, other devices in the neighborhood can cause interference during sensor communication. Likewise, the operation/interaction among other sensor nodes of the same network can cause this sort of interference. So, the lower the number of active sensors in the network, the more reliable tends to be the radio-frequency communication among these sensors. The last component, the sensor module, is responsible to gauge the phenomena of interest; the ability of concurrently collecting data pertaining to different phenomena is a property already available in some models of sensor nodes.

For each application scenario, the network designer has to consider the rate of variation for each sensed phenomenon in order to choose the best sampling rate of each sensor device. Such decision is very important to be pursued with precision as it surely has a great impact on the amount of data to be sensed and delivered, and, consequently, on the levels of energy consumed prematurely by the sensor nodes. This is the temporal aspect to be considered in the network design.

Another aspect to be considered is the spatial one. Megerian et al. [42] define coverage as a measure of the ability to detect objects within a sensor field. The lower the variation of the physical variable being measured across the area, the shorter has to be the radius of coverage for each sensor while measuring the phenomenon. This will have an influence in the number of active sensors to be employed to cover all demand points related to the given phenomenon. The fact is: the more sensors are active in a given moment, the higher is the overall energy consumed across the network. WSNs are sometimes deployed in hostile environments, with many restrictions of access. In such cases, the network would be very unreliable and unstable if the minimum number of sensor nodes was effectively used to cover the whole area of observation. If some sensor node fails to operate, its area of coverage would be out of monitoring, preventing the correlation of data coming from this area with others coming from other areas. The localization of each sensor node is assumed to be known a priori by an embedded GPS circuit or another method [43].

Another even worst-case scenario occurs when we have sensor nodes as network bottlenecks, being responsible for routing all data coming from the sensor nodes in the neighborhood. In this case, a failure in such nodes could jeopardize the whole network deployment. To avoid these problems and make a robust design of the WSN, extra sensor nodes are usually employed in order to introduce some sort of redundancy. By this means, the routing topology needs to be dynamic and adaptive: When a sensor node that is routing data

from other nodes fails, the routing algorithm discovers all its neighbor nodes and then the network reconfigures its own topology dynamically. One problem with this approach is that it entails unnecessary energy consumption. This is because the coverage areas of the redundant sensor nodes overlap too much, giving birth to redundant data. And these redundant data bring about extra energy consumption in retransmission nodes. The radio-frequency interference is also stronger, which can cause unnecessary retransmissions of data, increasing the levels of energy expenditure.

Megerian and Potkonjak [44] present many integer linear programming models to maximize energy consumption but do not consider the dynamic time scheduling. Nakamura et al. [1] and Quintão et al. [45] use a model that consider the temporal dimension as worked here but do not treat the phenomena individually, dimensioning by the worse of their characteristics. They use pure integer linear programming which limits the matrix sizes while the Generate and Solve methodology presented here surpasses this barrier. Quintão et al. [46] use a genetic algorithm to solve the coverage problem, but does not address the connectivity problem, neither deal with time schedules. Menezes et al. [21] use Lagrangean Relaxation to improve the results of previous pure integer linear programming approaches. However, this work does not consider the time either. Subsection 4.3.3.1 explores the points of view about using or not the temporal aspect.

4.3.2 Problem Formulation

The solution proposed by Nakamura et al. [1] is to create different schedules, each one associated with a given time interval, that activate only the minimum set of sensor nodes necessary to satisfy the coverage and connectivity constraints. The employment of different schedules prevents the premature starvation of some of the nodes, bringing about a more homogeneous level of consumption of battery across the whole network. This is because the alternation of active nodes among the schedules is often an outcome of the model, as it optimizes the energy consumption of the whole network taking into account all time intervals and coverage and connectivity constraints.

In order to properly model the WSN setting, some preliminary remarks are necessary:

1. A demand point is a geographical point in the region of monitoring where one or more phenomena are sensed. The distribution of such points across the area of monitoring can be regular, like a grid, but can also be random in nature. The density of such points varies according to the spatial variation of the phenomenon under observation. At least one sensor must be active in a given moment to sense each demand point. Such constraint

is implemented in the model;

2. Usually, the sensors are associated with coverage areas that cannot be estimated with accuracy. To simplify the modeling, we assume plain areas without obstacles. Moreover, we assume a circular coverage area with a radius determined by the spatial variation of the sensed phenomenon. Within this area, it is assumed that all demand points can be sensed. The radio-frequency propagation in real WSNs is also irregular in nature. In the same way, we can assume a circular communication area. The radius of this circle is the maximum distance at which two sensor nodes can interact;

3. A route is a path from one sensor node to a sink possibly relaying data through one or more sensor nodes by retransmission. Gateways are regarded as special sensor nodes whose role is only to interface with the sinks. Each phenomenon sensed in a node has its data associated with a route leading to a given sink, which is independent from the routes followed by the data related to other phenomena sensed in the same sensor node;

4. The energy consumption is actually the electric current drawn by a circuit in a given time period.

These are the input constants:

$$a_{ij} = \begin{cases} 1, \text{if exists an arc } ij \text{ that interconnects sensors}, i \in S, j \in S \cup M \\ 0, \text{otherwise.} \end{cases} \tag{4.2}$$

EH Penalty applied when a demand point in any time interval is not covered by any sensor

These are the sets of the problem:

S	Set of sensors
D	Set of demand points
M	Set of sinks
$T = \{x \mid 1 \leq x \leq n, x \in \mathbb{N}\}$	Set of n scheduling periods
AD_{ij}	Set of arcs $ij, i \in S, j \in D$ that link sensors to demand points
EB_i	Accumulated battery energy for sensor $i \in S$
EA_i	Energy dissipated while activating sensor $i \in S$
EM_i	Energy dissipated while sensor $i \in S$ is activated (effectively sensing)
ET_{ij}	Energy dissipated when transmitting data from sensor $i \in S$ to sensor $j \in S$. Such values can be different for each arc ij if a sensor can have its transmitter power adjusted based on the distance to the destination sensor
ER_i	Energy expended in the reception of data for sensor $i \in S$

These are the decision variables:

$x_{ij}^t \in \mathbb{R}^+$	If sensor $i \in S$ covers demand point $j \in D$ in period $t \in T$
$z_{lij}^t \in \{0, 1\}$	If arc ij belongs to the route from sensor $l \in S$ to a sink in period $t \in T$
$w_i^t \in \mathbb{R}^+$	If sensor i was activated in period t for at least one phenomenon
$y_i^t \in \{0, 1\}$	If sensor i is activated in period t
$h_j^t \in \{0, 1\}$	If demand point j is not covered by any sensor in period t
$e_i \in \mathbb{R}^+$	Energy consumed by sensor i considering all time periods

The objective function (4.3) minimizes the total energy consumption through all time periods and penalizes the existence some uncovered demand points, but the solution continues feasible.

$$\min \sum_{i \in S} e_i + \sum_{t \in T} \sum_{j \in D} EH h_j^t \tag{4.3}$$

These are the constraints of the model:

$$\sum_{i \in S} \sum_{j \in S} AD_{ij} x_{ij}^t + h_j^t \geq 1, \forall j \in D, \forall t \in T \tag{4.4}$$

Constraint (4.4) enforces the activation of at least one sensor node i to cover the demand

point j in period t. Otherwise, the penalty variable h is set to one. When the penalty EH is high, this last condition will occur only in those cases when no sensor node can cover the demand point.

$$x_{ij}^t \leq y_i^t, \forall i \in S, \forall j \in S, \forall t \in T \tag{4.5}$$

Constraint (4.5) turns on variable y (which means that a sensor node is actively sensing in period t) if its associated sensor node is indeed allocated to cover any demand point.

$$\sum_{i \in (S-\{j\})} a_{ij} z_{lij}^t - \sum_{k \in (S \cup M - \{j\})} a_{jk} z_{ljk}^t = 0, \forall j, l \in S, \forall t \in T \tag{4.6}$$

Constraint (4.6) relates to the connectivity issue using the flow conservation principle. This constraint enforces that an outgoing route exists from sensor node j to sensor node k if there is already an incoming route from sensor node i to sensor node j.

$$\sum_{k \in (S \cup M - \{l\})} a_{jk} z_{llk}^t = y_l^t, \forall l \in S, \forall t \in T \tag{4.7}$$

Constraint (4.7) enforces that a route is created if a sensor node is already active.

$$\sum_{i \in S} \sum_{j \in M} a_{ij} z_{lij}^t = y_l^t, \forall t \in T, \forall l \in S \tag{4.8}$$

Constraint (4.8) is necessary to create a route that reaches a sink if a sensor node is active.

$$a_{ij} z_{lij}^t \leq y_j^t, \forall j \in S, \forall l, i \in (S - \{j\}), \forall t \in T \tag{4.9}$$

In Constraint (4.9), if there is an outgoing route passing through sensor node i, then this sensor node has to be necessarily active.

$$a_{ij} z_{lij}^t \leq y_i^t, \forall j \in S, \forall l, i \in (S - \{j\}), \forall t \in T \tag{4.10}$$

In the same way, with Constraint (4.10), if there is an incoming route passing through sensor i, then this sensor has to be active.

$$\sum_{t \in T}(EM_i y_i^t + EA_i w_i^t + \sum_{l \in (S-\{i\})} \sum_{k \in S} ER_i z_{lki}^t + \sum_{l \in S} \sum_{j \in (S \cup M)} ET_{ij} z_{lij}^t) \leq e_i, \forall i \in S \tag{4.11}$$

The total energy consumed by a sensor node is the sum of the parcels given in Constraint (4.11).

$$0 \leq e_i \leq EB_i, \forall i \in S \tag{4.12}$$

Constraint (4.12) enforces that each sensor node should consume at most the energy capacity limit of its battery.

$$w_i^0 - y_i^0 \geq 0, \forall i \in S \tag{4.13}$$

Constraint (4.13) determines when the sensor node should start to sense (decision variable w). If a sensor is active in the first period, its corresponding w should be set to 1.

$$w_i^t - y_i^t + y_i^{t-1} \geq 0, \forall i \in S, \forall t \in T \tag{4.14}$$

In Constraint (4.14), the past and current activation states of a sensor node are compared. If the sensor node was active from period $t-1$ to period t, then w is set to 1.

4.3.3 Computational Results

4.3.3.1 Energy Consumption Reduction Versus WSN Lifetime Extension Discussion

Nakamura et al. [1] conceived their computational experiments to produce WSN's with the minimum energy consumption as possible, while maintaining the coverage and connectivity constraints. The gain obtained in their paper is calculated by the comparison of the minimum set of sensors that have to be active and the waste of energy caused by the activation of all sensors with high coverage overlapping in a high density configuration.

This idea seems coherent at first glance since energy waste reduction is often desirable. However, this point of view distorts the actual need of a WSN that stands in a place where is infeasible to change the batteries: Extend its lifetime as far as possible. All scenarios proposed by Nakamura et al. [1] have only 4 timeslots. In the first scenario, timeslots

have 72 hours. Thus, each sensor node can keep on working for up to 3 timeslots. So the solution consists in spending 2/3 of the battery capacity of 8 sensors. This way, when the WSN finishes the plan, there will be yet other 8 sensor nodes that were not used at all.

One could argument that these nodes would be used as backup since the higher density is used for reliability enhancement. But this lack of use is arguable. The dynamic plan can be used to distribute the energy expenditure of all 16 sensor nodes, alternating through timeslots in a wider range of time. If a failure occurs in some sensor nodes during the WSN operation, another planning will be created to adapt effectively to the new WSN configuration. Of course the WSN lifetime can be reduced in case of failures, but only if it is required. This procedure is better than cutting unnecessarily the lifetime of the WSN and keeping half of its sensors useless.

Another possible argument for justifying the energy consumption reduction point of view is that these 8 sensor nodes will be used afterwards to extend the WSN lifetime. This is also a weak argument. If a set of smaller plans are made considering each one only a part of the total time, it tends to be less efficient than the global plan that covers the entire lifetime. This happens because the first plans will be greedy and take the best configurations. The remaining plans will inherit network topologies each time less favorable for attending the constraints.

The energy consumption reduction point of view is taken to its limit by Menezes et al. [21]. This time up to 87% of the sensor nodes are kept turned off and only one timeslot is considered. Actually, time is not even mentioned in that paper. The time dimension was removed from the decision variable matrices, which simplifies the Lagrangean Relaxation application but creates an unreal problem context.

In this work, the lifetime extension point of view is considered. The number of timeslots is increased while the solution is achieved in a reasonable time limit of 4 hours and the coverage rate of 100% is kept. Notice that if we permit some amount of uncovered demand point rate this number of timeslots can be even larger. The result is that the same WSN that was handled by only 4 timeslots is now capable of working by 10 timeslots in the same conditions, as explained in subsection 4.3.4. It represents a gain of 150%.

4.3.4 Results

Experiments were made for the dynamic coverage and connectivity in wireless sensor networks problem using the new version of the Generate and Solve methodology.

By following one of the scenarios used by Nakamura et al. [1], we defined 400 demand points distributed regularly as a grid in a square area of 20 per 20 meters. This leads to a density of one demand point per square meter. Also following this scenario, 16 sensor nodes were placed in a regular 4 × 4 grid. The network is composed of only one type of node, with the same hardware characteristics. The coverage radius adopted was 8.8 meters, while the radius that allows communication between two neighbor sensors was 11 meters. The sampling rate was set as two samples per minute.

There is only one sink placed in the center of the square area. The model has matrices used to determine if the distance from sensor-demand point (coverage) and sensor-sensor (communication) is under the respective radius of reach. In order to have them filled in properly, geographic coordinates were generated to each demand point, sensor and the sink. The 0-1 value used in each element of the matrices corresponds to the Boolean condition of being bellow the radius threshold or not. The sensor node manufacturer spreadsheet [47] was used for the calculation of the energy parameters for the model.

The portion of time spent sensing data and its sampling rate determine the data flow to be carried over. This flow combined to the data transmission rate used by the nodes will result in the energy values for transmission and reception. In order to minimize the uncovered demand points percentage, high values of penalty were used.

The machine used for this experiment was an Intel Core 2 Quad 64 bits with 8 GB of RAM machine with OpenSuse Linux 11.0 64 bits. As ILP solver, Ilog Cplex 10.1 dynamic library [36] was used attached to the Java program implementation of the methodology. The pure ILP approach is a particular case of the methodology and is obtained by a proper parameter set in a XML (Extensible Markup Language) execution batch script. Table 9 and 10 present the comparison betweed ILP and Generate and Solve methodology (GS) approaches. In these experiments, the demand points are disposed in a grid. Due to the stochastic nature of the Generate and Solve methodology, the average and standard deviation of results found in a batch of 10 problem instances is presented. For the same reason, the random instances were processed in the same manner. The notation used here is the average value followed by the ± sign and the standard deviation value. The objective function is composed of two parts: The summation of electrical charge consumption in all sensor nodes and the penalties. The penalties are an artifact that allows uncovered demand points giving flexibility to the model, but at the same time, avoid the unnecessary use of this expedient. Thus, the objective is calculated by subtracting the artificial coverage penalties or just calculating the first part (summation)

of the objective expression. In Table 9 the spaces filled with dashes (-) represent the instances where the solver could not finish its execution in a time limit of 24 hours.

As described in Sections 3.2.1, 3.2.2 and 4.2.1, the ILP approach is a particular case of the Generate and Solve methodology. It happens when the problem instance is less complex in such a manner that the ideal adjust of the density parameters IND and IDD or MD reaches the maximum value of one (full density). When it happens, the subproblem is, in fact, the original problem and the Generate and Solve corresponds simply to the ILP approach.

Due to this nature, it is useless to make comparisons between ILP and Generate and Solve methodology because in this cases it would be, actually, a comparison between the Generate and Solve methodology and itself. The effort of comparing the two approaches can lead to the following situations: Higher complexity problem instances which cannot be solved by the ILP pure approach; Lower complexity problem instances which can be correctly parameterised with full density and solved exactly with the ILP pure approach.

Another situation that could be experienced is the miscalibration of the density parameters IND and IDD or MD with a density below one for solving problem instances that would be ideally solved with the full density value of one. Although this is not designed to happen, it was intentionally done in this experiments in order to show the loss of both efficiency (time) and efficacy (objective value) caused by the heuristic nature of the methodology when GRI starts to evolve the individuals and the subproblem are incomplete versions of the original problem. This can be viewed in the second half of Table 9 which represents the problem instances with 36 sensor nodes. The only cases which were solved were the smallest with 2 intervals. In Table 10, the problem instances with 2 intervals where miscalibrated with a density value of 0.17, which is suitable for larger instances (4, 6, 8 and 10 intervals) but far away from the ideal value of one.

Type	Intervals	Points	Sensors	Time (min.)	Uncovered points (%)	Objective
Grid	2	400	16	0.02	0.00	5,376.22
Grid	4	400	16	0.15	0.00	10,752.43
Grid	6	400	16	0.34	0.00	15,698.30
Grid	8	400	16	44.60	0.00	22,418.57
Grid	10	400	16	298.55	0.33	26,665.91
Random	2	400	16	0.02 ± 0.00	1.05 ± 1.69	5,779.43 ± 649.24
Random	4	400	16	0.13 ± 0.04	0.92 ± 1.44	11,521.06 ± 216.69
Random	6	400	16	0.34 ± 0.24	0.56 ± 0.49	17,464.36 ± 1,718.17
Random	8	400	16	80.06 ± 148.74	2.39 ± 1.44	21,181.07 ± 1,236.94
Random	10	400	16	212.48 ± 322.46	4.32 ± 2.29	26,483.01 ± 571.77
Grid	2	400	36	0.29	0.00	5,376.22
Grid	4	400	36	-	-	-
Grid	6	400	36	-	-	-
Grid	8	400	36	-	-	-
Grid	10	400	36	-	-	-
Random	2	400	36	0.52 ± 0.15	0.00	5,376.22 ± 0.00
Random	4	400	36	-	-	-
Random	6	400	36	-	-	-
Random	8	400	36	-	-	-
Random	10	400	36	-	-	-

Table 9: Simulation results for demand point in grid with ILP

Type	Intervals	Points	Sensors	Time (min.)	Time (first final solution)	Uncovered points (%)	Objective
Grid	2	400	36	7.35 ± 0.37	3.24 ± 2.39	0.56 ± 0.62	7,930.52 ± 1,757.30
Grid	4	400	36	20.93 ± 2.95	5.67 ± 6.93	1.44 ± 0.70	16,429.72 ± 2,204.05
Grid	6	400	36	151.78 ± 14.61	79.84 ± 54.65	0.93 ± 0.39	22,223.27 ± 2,614.63
Grid	8	400	36	63.46 ± 2.9	28.77 ± 18.65	2.87 ± 0.53	30,170.00 ± 1,406.76
Grid	10	400	36	189.80 ± 61.01	104.14 ± 75.06	2.35 ± 0.54	33,374.04 ± 2,389.76
Random	2	400	36	7.27 ± 0.43	2.67 ± 2.59	0.48 ± 0.53	8,000.56 ± 1,830.75
Random	4	400	36	23.82 ± 16.11	6.74 ± 6.33	2.24 ± 0.90	14,371.32 ± 3,649.27
Random	6	400	36	146.77 ± 32.45	91.55 ± 47.14	1.56 ± 0.61	20,472.51 ± 4,135.35
Random	8	400	36	55.15 ± 6.91	14.54 ± 10.12	2.56 ± 0.79	27,956.55 ± 3,513.41
Random	10	400	36	366.28 ± 13.54	196.05 ± 78.30	2.10 ± 0.53	32,522.51 ± 2,628.33

Table 10: Simulation results with Generate and Solve

The real purpose of the model described in section 4.3 is to extend the WSN lifetime as far as possible, preserving the WSN cost. So, lower electrical charge consumption is not necessarily an important issue if it does not reflect in more time slots. The number of time slots multiplied by the duration of each time slot represents this WSN lifetime.

Given this explanation it is reasonable to assert that both solutions found by Generate and Solve methodology and ILP are equivalent in effectiveness. However, the GS approach can handle an amount of sensors 125% times larger, extending the working range of this application.

The only drawback here is the uncovered demand point rate which is worse than that achieved by ILP. Despite this loss of accuracy, many real applications tolerate some lack of coverage by the nature of the observed phenomenon and other aspects. Even though this uncovered demand points are often situated at the periphery of the observed area. The coverage radius does not reflect necessarily a sharp threshold of sensing.

Figure 6 shows the evolution of the best individual fitness with plain line and population fitness average with a dotted line. As an example, figures 7 to 16 in Appendix shows the graphical representations of one particular solution with 10 time slots. Each figure represents one time slot. Their active sensor nodes have its coverage area drawn as circles. The routes from sensor nodes to the sink (the central point) are drawn as lines.

4.4 Final Considerations

One gain obtained with the new version of the Generate ad Solve is extending the WSN lifetime from 4 to 10 timeslots comparing to literature [1]. It is a gain of 150%. Other important gain is the number of sensor nodes which has increased from 16 to 36. It represents a gain of 125%.

More than just presenting an alternative tool for solving this class of problem, this work shows that this methodology can be adapted to work with different problem classes as well as it can aggregate auxiliary heuristics and techniques.

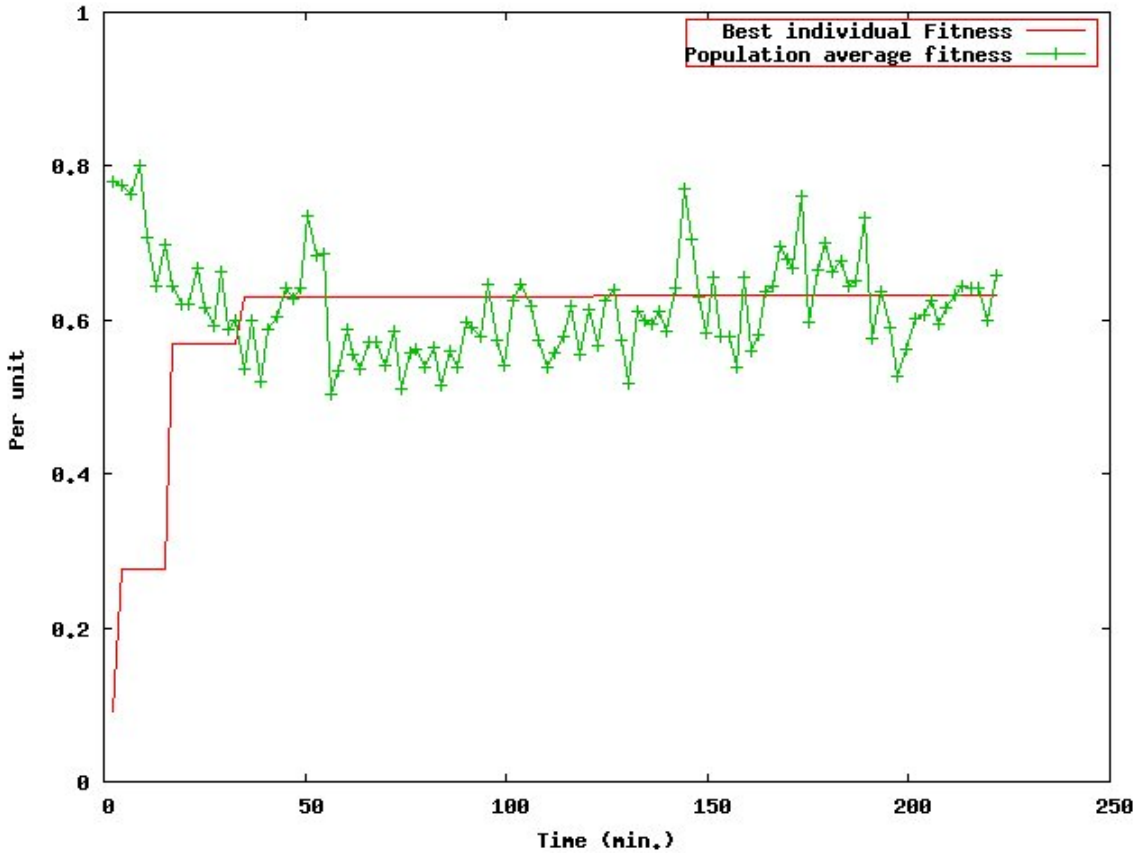

Figure 6: Evolution chart

Conclusion and Future Works

The Generate and Solve methodology is not only suitable for solving complex instances in the domain of cutting and packing problems. It can be adapted to tackle other problem classes like WSN design as shown here.

The key point in this adaptation is finding the best or at least a good reducible structure. This analysis is very linked to the chromosome encoding choice as it represents a tradeoff between subproblem complexity range width and chromosome size. A good reducible structure allows a wide range of subproblem complexities from very light and fast subproblems to the actual real problem. On the other hand the reducible matrix size affects the chromosome size and a large chromosome size reduces the GA (Genetic Algorithm) effectiveness.

In the problem of dynamic coverage and connectivity in wireless sensor networks, a good reducible structure was found but it is much larger than the ones found in the cutting and packing problem instance. That is the reason why a new chromosome encoding was developed. This new encoding makes the matrix choice viable.

The results found are better than reference literature and leaves opportunities of future enhancements as new supplementary algorithms and heuristics are aggregated to this methodology.

As future work, the application of the new version of the Generate and Solve methodology to the container loading problem as presented by Nepomuceno el al. [2] sounds reasonably easy to do, as it has the same model base structure than Constrained Non-guillotine Cutting Problem. Maybe the layer heuristic is no more necessary and the solutions could be more effective due to the global view of the container. Although it has one more dimension than the previous one, actually the size of the matrices and other particularities of each problem instance will determine which problem will be harder to solve.

Another natural path is to replace the homogeneous model by the heterogeneous one as described by Aguiar et al. [38–40]. This might require some sort of adaptation but possibly it will be straightforward. The heterogeneous model offers harder problem

instances which represents a promissing challenge.

The original methodology was designed to work with cutting and packing problems domain. The adaptation to this class of problem presented here gave it a broad flexibility and revealed the possibility of working with many other problem domains.

In principle, once a good reducible structure is determined and subproblems could be generated, the methodology could be readily deployed. This choice is the key for a well successful application of the methodology. The good candidates to reducible structures are those matrices from which possible to create subsets of their elements to generate subproblems and this subproblems have feasible solutions for the original problem. Small matrices are better for being encoded in chromosomes but its reduction must cause a significant reduction in the problem instance complexity and computational burden. These characteristics can be antagonistic in some candidate matrices so its important to find a good balance.

The proposed compact chromosome encoding (subsection 4.1.1) is now a flexible alternative to original binary encoding but many other encoding schemes can be discovered. This encoding depends strongly on the chosen reducible structure and helps to make possible to work with them.

The geographical nature of the problem treated here is linked to the network design problems found in Afonso et al. [48], Aguiar et al. [49–51], Aguiar and Pinheiro [52] and Neto et al. [53]. Sensor nodes matrix was a good choice for reducible structure. In a similar way the matrices that represent the geographical allocation of hubs and BSC (Base Station Controller) nodes can be good reducible structures candidates as well. Because of this similarity in this aspect the use of this methodology on cellular networks design might possibly be proper.

Another promising line of investigation involves the design and implementation of parallel/distributed versions of the framework as presented by Viana [7], by means of which several GRI (Generator of Reduced Instances) and SRI (Solver of Reduced Instances) instances could run concurrently, each one configured to explore different aspects of the optimization problem at hand. The use of insular GA can bring more diversity and possibilities, resulting in effectiveness enhancement. Also, other metaheuristics such as particle swarm could be experimented replacing or working cooperatively with GA.

Appendix

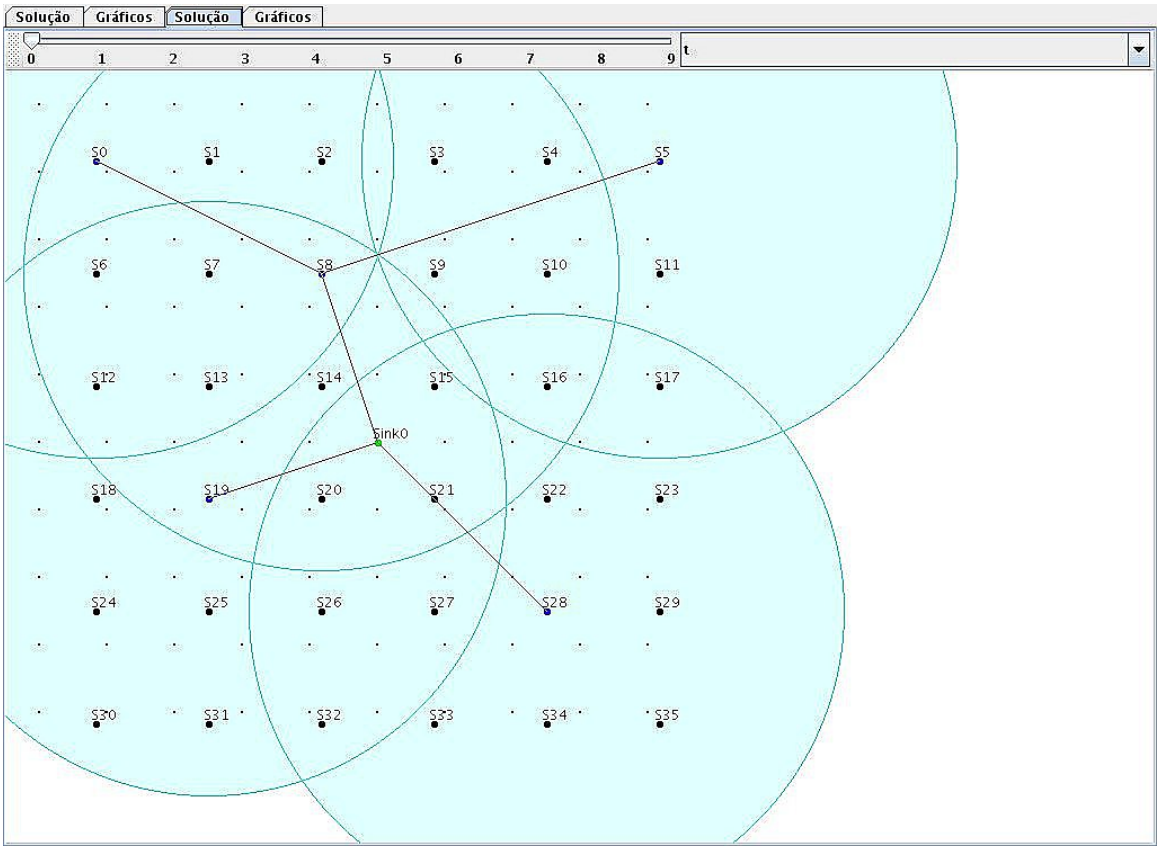

Figure 7: Interval 1

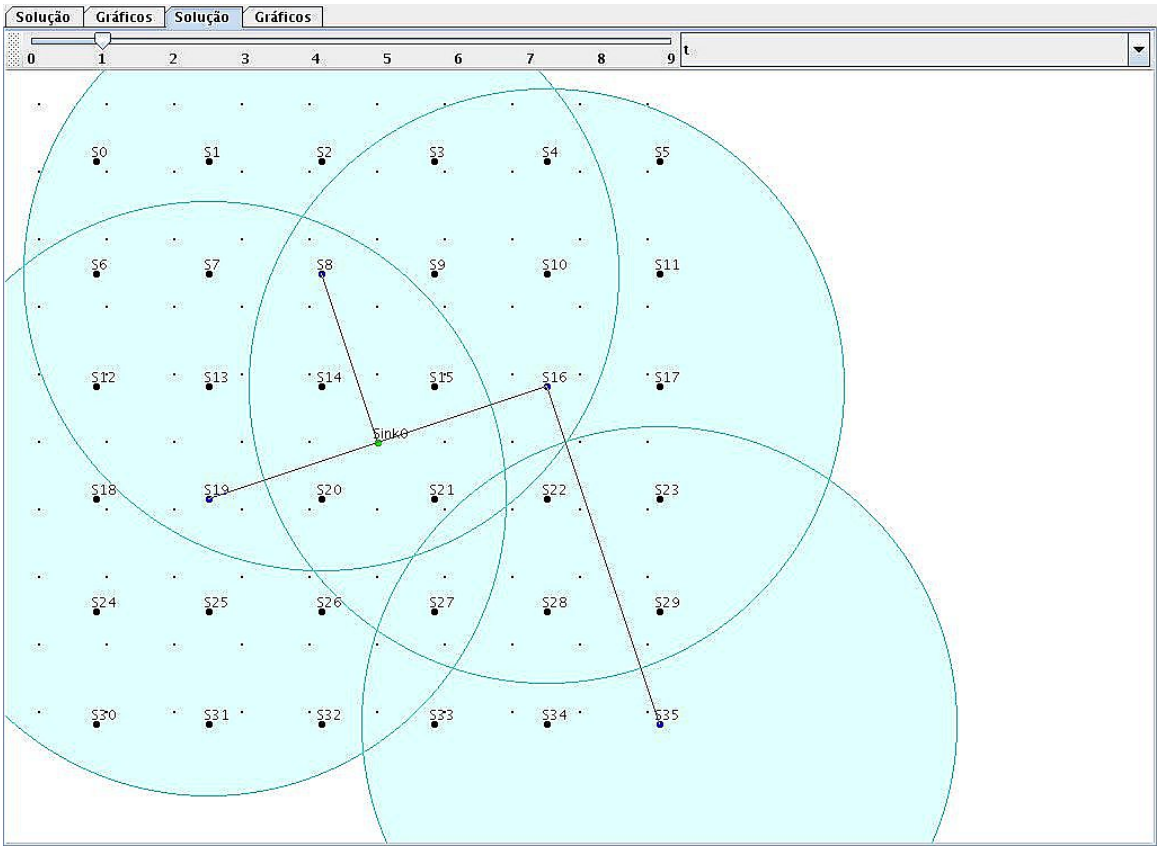

Figure 8: Interval 2

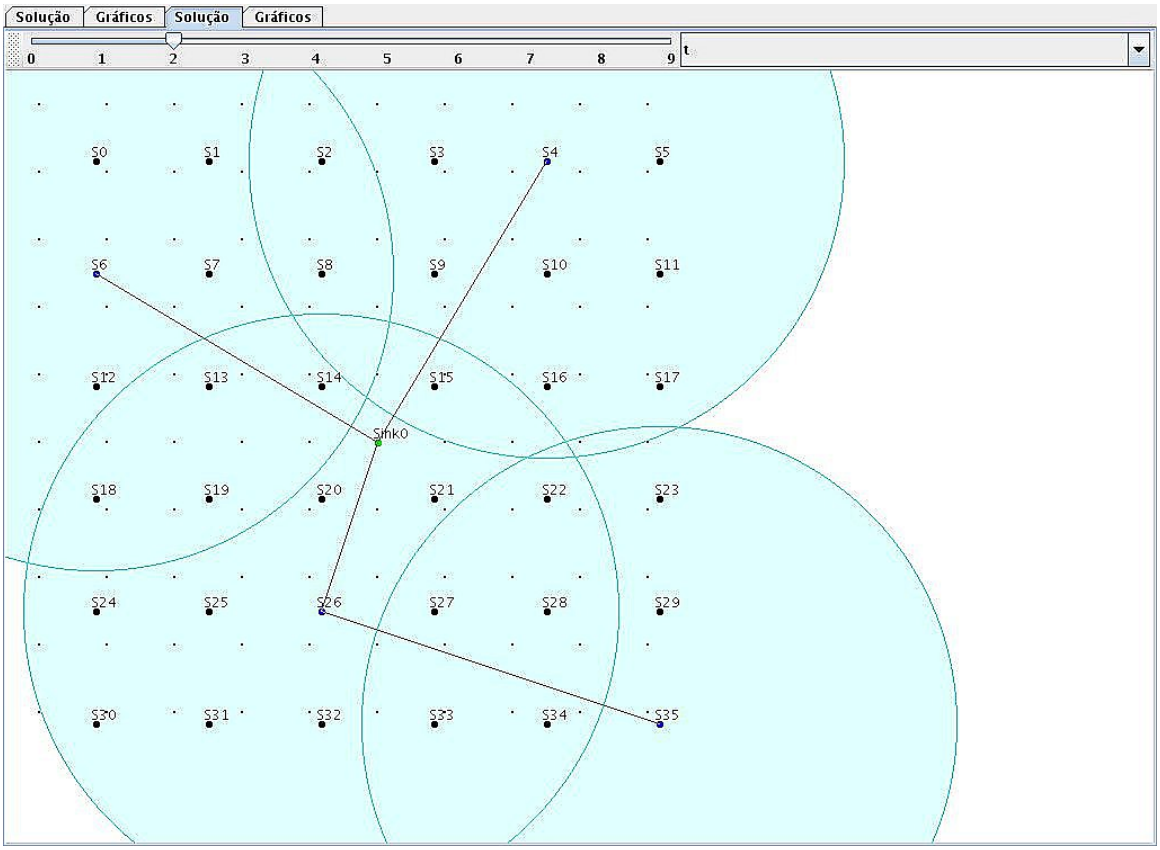

Figure 9: Interval 3

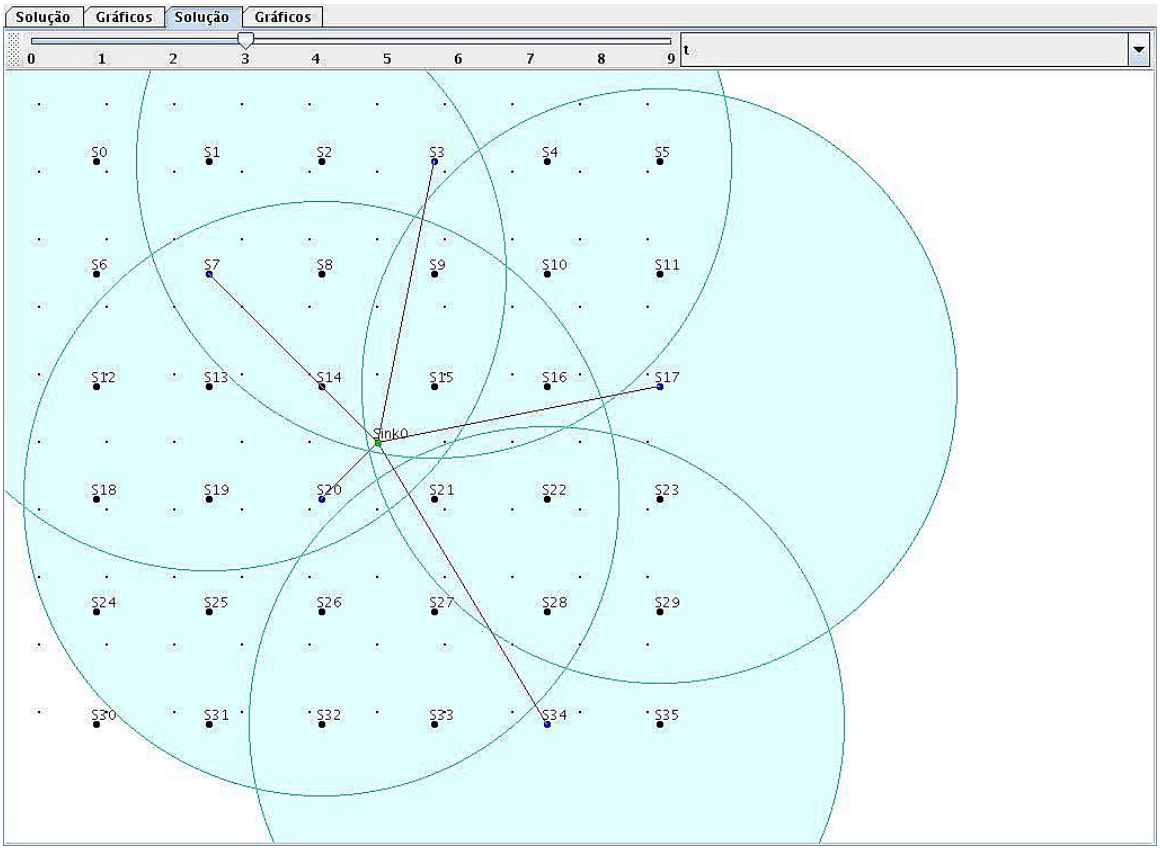

Figure 10: Interval 4

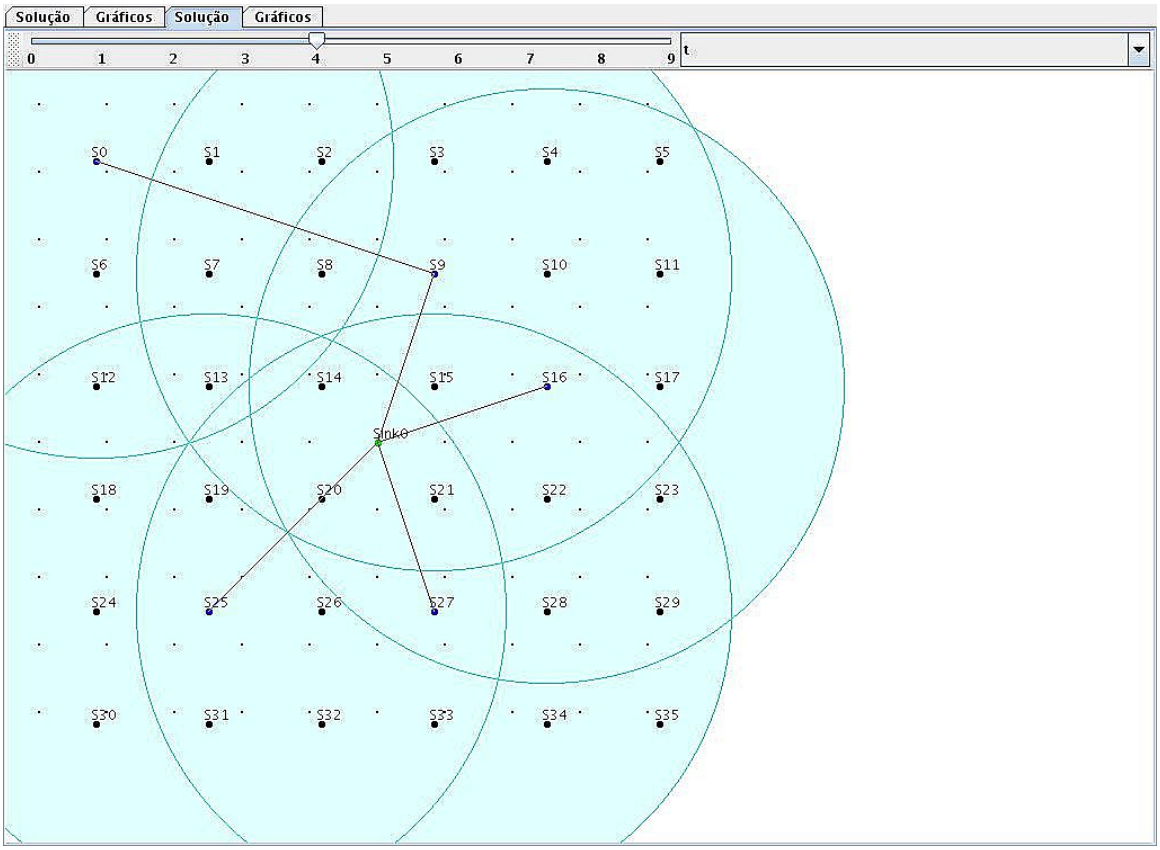

Figure 11: Interval 5

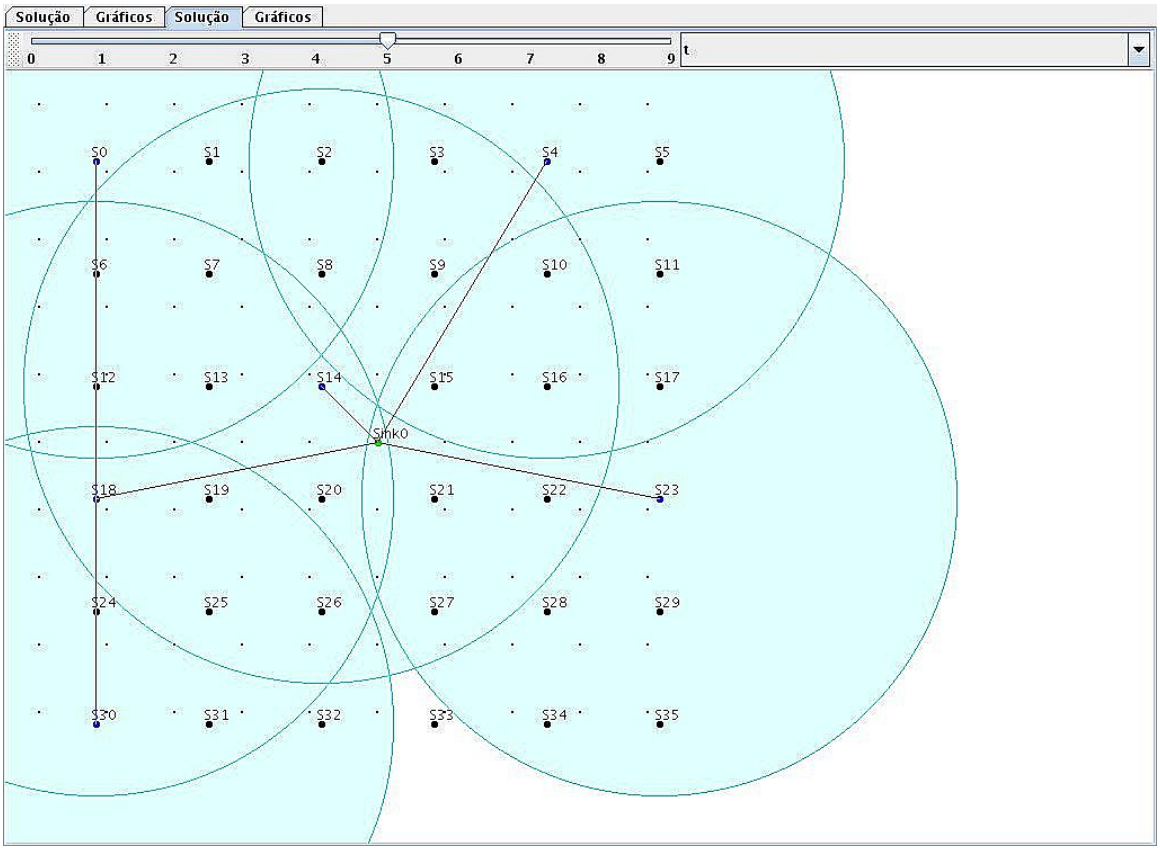

Figure 12: Interval 6

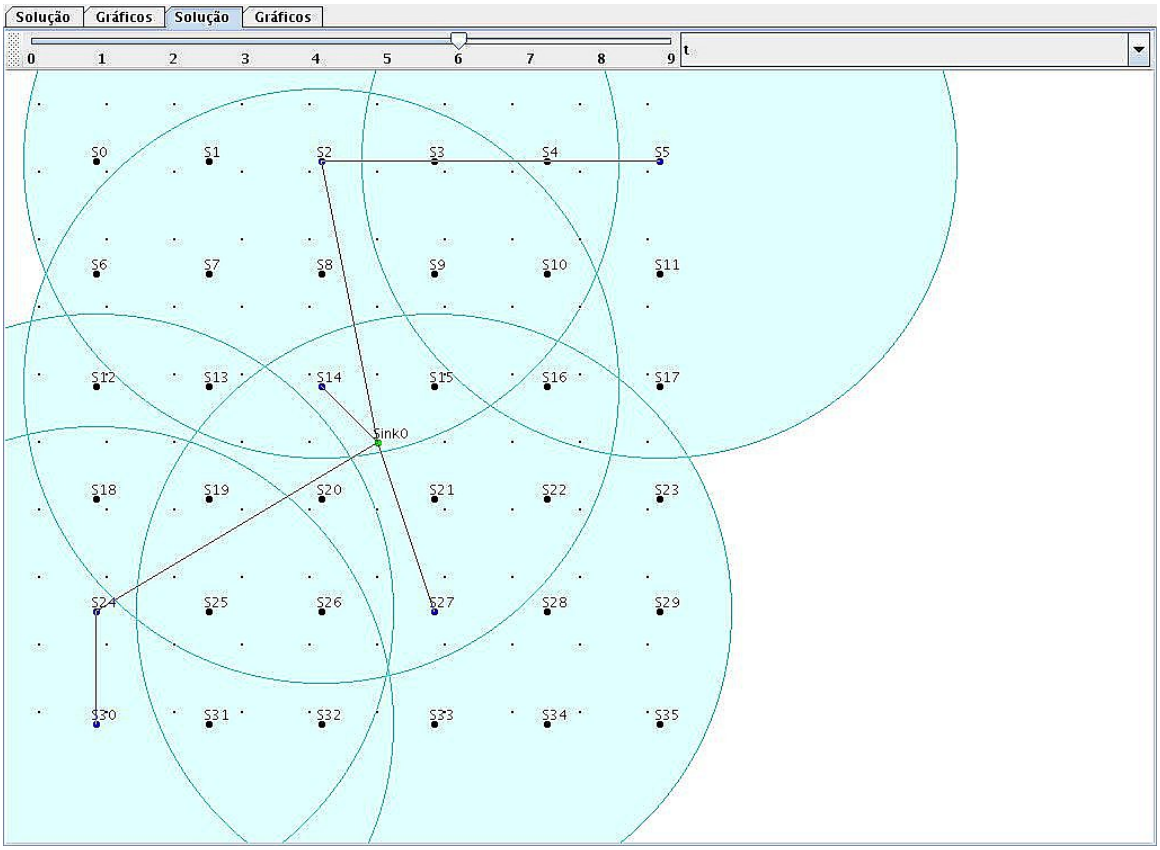

Figure 13: Interval 7

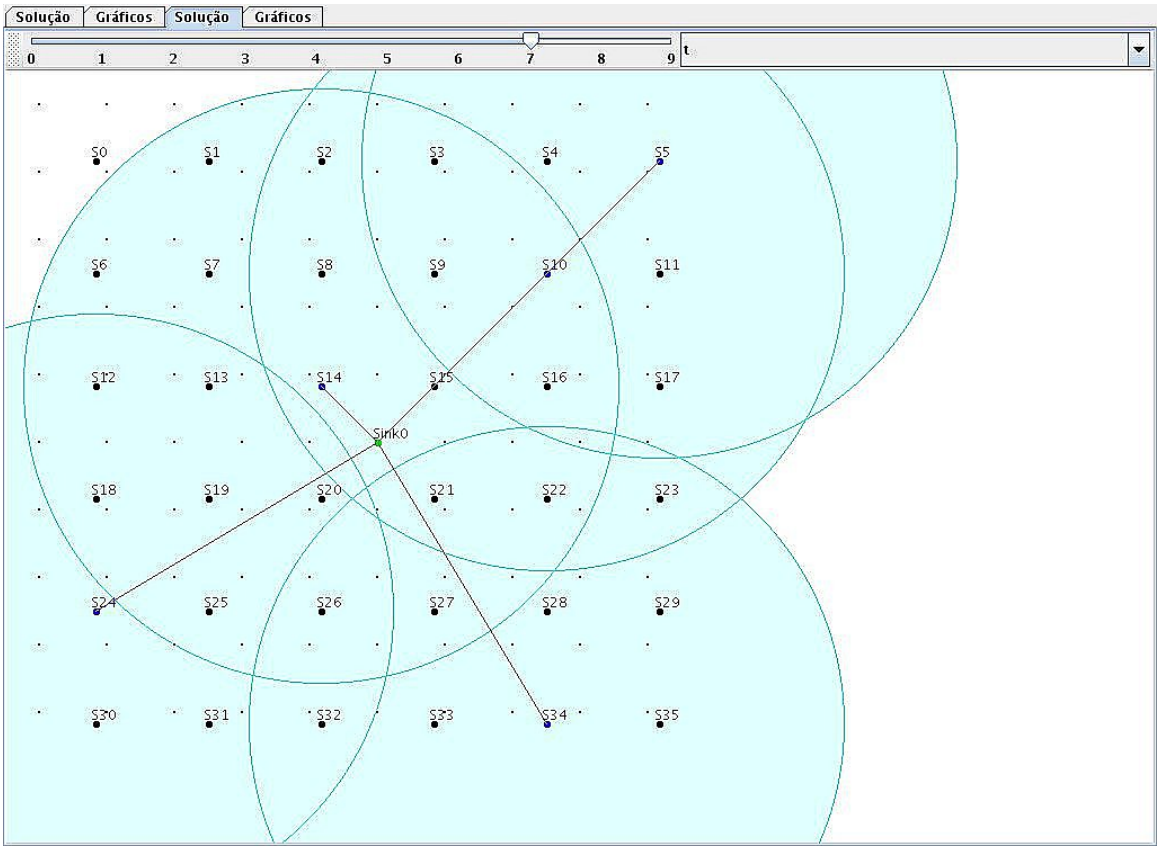

Figure 14: Interval 8

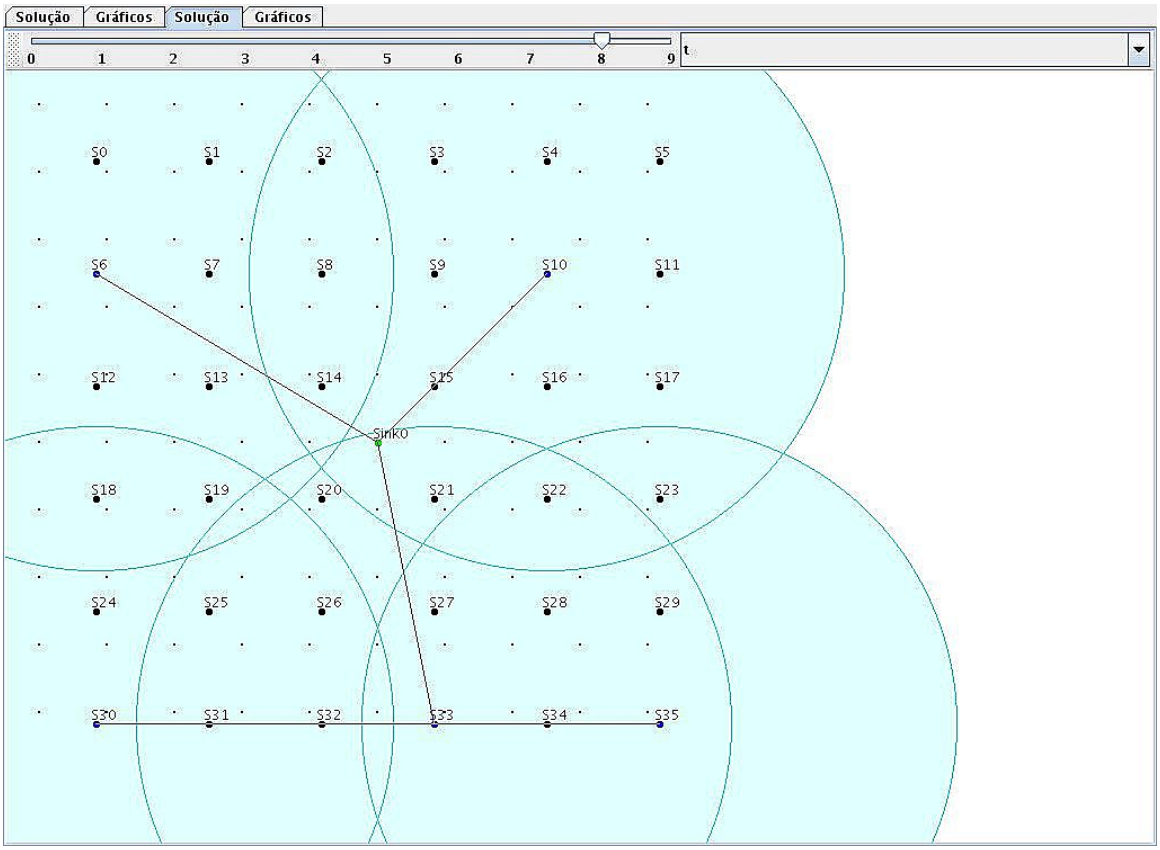

Figure 15: Interval 9

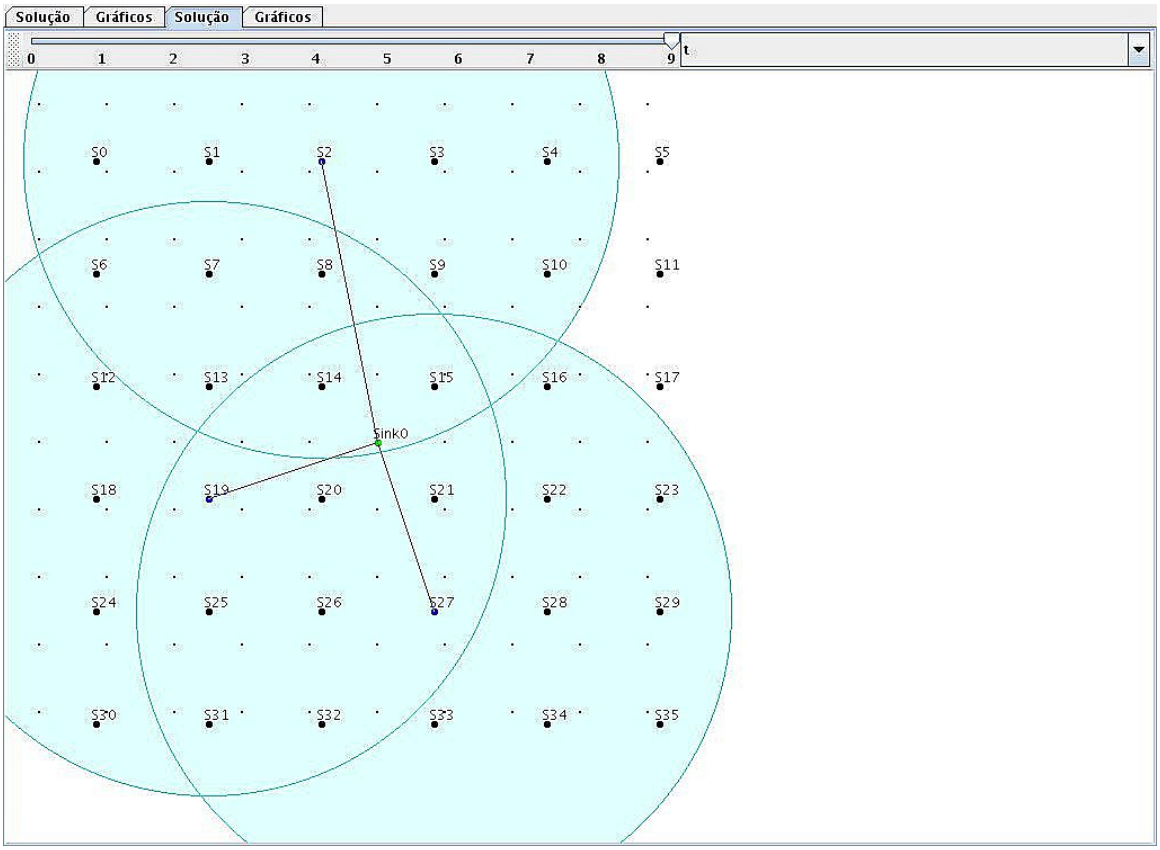

Figure 16: Interval 10

References

1 NAKAMURA, F. G. et al. Planejamento dinâmico para controle de cobertura e conectividade em redes de sensores sem fio. *Workshop de Comunicação sem Fio e Computação Móvel*, v. 1, p. 182–191, 2004.

2 NEPOMUCENO, N. V.; PINHEIRO, P. R.; COELHO, A. L. V. Tackling the container loading problem: A hybrid approach based on integer linear programming and genetic algorithms. In: *EvoCOP 2007 - Valência, Spain. Lecture Notes in Computer Science*. Springer-Verlag, 2007. v. 4446, p. 154–165. Disponível em: <http://dx.doi.org/10.1007/978-3-540-71615-0_14>.

3 NEPOMUCENO, N. V.; PINHEIRO, P. R.; COELHO, A. L. V. A hybrid optimization framework for cutting and packing problems: Case study on constrained 2d non-guillotine cutting. In: COTTA, C.; HEMERT, J. van (Ed.). *Recent Advances in Evolutionary Computation for Combinatorial Optimization*. Berlin: Springer-Verlag, 2008. v. 153, p. 87–99. Berlin / Heidelberg.

4 WOLSEY, L. A. *Integer Programming*. New York: John Wiley & Sons, 1998.

5 MARTELLO, S.; TOTH, P. *Knapsack Problems: Algorithms and Computer Implementations*. New York: John Wiley & Sons, 1990. (Wiley-Interscience Series in Discrete Mathematics and Optimization).

6 EIBEN, A. E.; SMITH, J. E. *Introduction to Evolutionary Computing*. Amsterdam: Springer-Verlag, 2003. ISBN 3540401849.

7 VIANA, G. R. V. *Meta-heurísticas e programação paralela em otimização combinatória*. Fortaleza: Edições UFC, 1998.

8 TALBI, E.-G. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, v. 8, n. 5, p. 541–564, 2002.

9 MARTIN, O.; OTTO, S. W.; FELTEN, E. W. Large-step markov chains for the tsp: Incorporating local search heuristics. In: *Operations Research Letters*. [S.l.: s.n.], 1992. v. 11, n. 4, p. 219–224.

10 MAHFOUD, S. W.; GOLDBERG, D. E. Parallel recombinative simulated annealing: A genetic algorithm. v. 21, n. 1, p. 1–28, 1995.

11 CHU, P. C. *A Genetic Algorithm Approach for Combinatorial Optimization Problems*. Tese (Doutorado), 1997.

12 DUMITRESCU, I.; STÜTZLE, T. Combinations of local search and exact algorithms. In: RAIDL, G. R. et al. (Ed.). *EvoWorkshops*. United Kingdom: Springer, 2003. p. 211–223.

13 PUCHINGER, J.; RAIDL, G. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In: . Berlin: Springer Verlag, 2005. v. 3562, cap. Proceedings of the 1st International Work-Conference on the Interplay Between Natural and Artificial Computation, Lecture Notes in Computer Science, p. 41–53.

14 VASQUEZ, M.; HAO, J.-K. A hybrid approach for the 0-1 multidimensional knapsack problem. In: *IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. p. 328–333. ISBN 1-55860-812-5, 978-1-558-60812-2.

15 COOK, W.; SEYMOUR, P. Tour merging via branch-decomposition. *INFORMS J. on Computing*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 15, n. 3, p. 233–248, 2003. ISSN 1526-5528.

16 RAIDL, G. R. A unified view on hybrid metaheuristics. In: ALMEIDA, F. et al. (Ed.). *Hybrid Metaheuristics*. Gran Canaria, Spain: Springer, 2006. (Lecture Notes in Computer Science, v. 4030), p. 1–12. ISBN 3-540-46384-4.

17 PLATEAU, A.; TACHAT, D.; TOLLA, P. A hybrid search combining interior point methods and metaheuristics for 0-1 programming. *International Transactions in Operational Research*, Blackwell Publishing, p. 731–746, November 2002. ISSN 0969-6016. Disponível em: <http://dx.doi.org/10.1111/1475-3995.00385>.

18 PUCHINGER, J.; RAIDL, G. R.; KOLLER, G. Solving a real-world glass cutting problem. In: GOTTLIEB, J.; RAIDL, G. R. (Ed.). *EVOLUTIONARY COMPUTATION IN COMBINATORIAL OPTIMIZATION – EVOCOP 2004*. Coimbra, Portugal: Springer Verlag, 2004. (Lecture Notes in Computer Science, v. 3004), p. 162–173.

19 TERNO, J.; SCHEITHAUER, J. R.; SOMMERWEISS, U. An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, v. 123, p. 372–381, 2000.

20 QUINTãO, F. P.; NAKAMURA, F. G.; MATEUS, G. R. A hybrid approach to solve the coverage and connectivity problem in wireless sensor networks. In: *4th European Workshop on Metaheuristics: Design and Evaluation of Advanced Hybrid Metaheuristics*. Nottingham, UK: [s.n.], 2004.

21 MENEZES, G. C. de; NAKAMURA, F. G.; MATEUS, G. R. Uma abordagem lagrangeana para os problemas de densidade, cobertura e conectividade em uma rede de sensores sem fio. *Workshop de Comunicação sem Fio e Computação Móvel*, v. 1, p. 192–201, 2004.

22 BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, 2003. ISSN 0360-0300.

23 BACK, T.; FOGEL, D. B.; MICHALEWICZ, Z. *Handbook of Evolutionary Computation*. Bristol, UK, UK: IOP Publishing Ltd., 1997.

24 DOWSLAND, K. A.; DOWSLAND, W. B. Packing problems. *European Journal of Operational Research*, v. 56, n. 1, p. 2–14, January 1992. Disponível em: <http://ideas.repec.org/a/eee/ejores/v56y1992i1p2-14.html>.

25 BEASLEY, J. E. An exact two-dimensional non-guillotine cutting tree search procedure. v. 33, n. 1, p. 49–64, 1985.

26 HADJICONSTANTINOU, E.; CHRISTOFIDES, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. In: *European Journal of Operational Research*. [S.l.]: Elsevier, 1995. v. 83, cap. European Journal of Operational Research, p. 39–56.

27 FEKETE, S. P.; SCHEPERS, J. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: GOOS, G.; HARTMANIS, J.; LEEUWEN, J. van (Ed.). *European Symposium on Algorithms*. Austria: Springer-Verlag, 1997, (Lecture Notes in Computer Science). cap. Proceedings of the 5th Annual European Symposium on Algorithms, p. 144–156. Disponível em: <citeseer.ist.psu.edu/fekete97new.html>.

28 BEASLEY, J. E. A population heuristic for constrained two-dimensional non-guillotine cutting. v. 156, n. 3, p. 601–627, 2004.

29 BEASLEY. *OR-Library*. June 1990. Internet. Disponível em: <`http://people.brunel.ac.uk/~mastjjb/jeb/info.html`>.

30 POLYAKOVSKY, S.; M'HALLAH, R. An agent-based approach to the two-dimensional guillotine bin packing problem. *European Journal of Operational Research*, v. 192, n. 3, p. 767–781, February 2009. Disponível em: <http://ideas.repec.org/a/eee/ejores/v192y2009i3p767-781.html>.

31 MI, X. et al. Case study on optimization of rectangular object layout by genetic algorithm. Springer-Verlag, Berlin, Heidelberg, p. 608–619, 2008.

32 MAHANTY, B. et al. Hybrid approach to optimal packing using genetic algorithm and coulomb potential algorithm. *Materials and Manufacturing Processes*, v. 22, n. 5, p. 668–677, June 2007.

33 TIWARI, S.; CHAKRABORTI, N. Multi-objective optimization of a two-dimensional cutting problem using genetic algorithms. *Journal of Materials Processing Technology*, v. 173, n. 3, p. 384–393, 2006.

34 VIDYAKIRAN, Y.; MAHANTY, B.; CHAKRABORTI, N. A genetic-algorithms-based multiobjective approach for a three-dimensional guillotine cutting problem. *Materials and Manufacturing Processes*, v. 20, n. 4, p. 697–715, July 2005.

35 MOHANTY, S.; MAHANTY, B.; MOHAPATRA, P. K. J. Optimization of hot rolled coil widths using a genetic algorithm. *Materials and Manufacturing Processes*, v. 18, n. 3, p. 447–462, January 2003.

36 ILOG. *ILOG CPLEX 10.0 User's Manual*. United States of America, January 2006.

37 AGUIAR, A. B. de et al. A hybrid methodology for coverage and conectivity in wireless sensor network dynamic planning. In: ACHI, L. S.; SANT'ANNA, A. P. (Ed.). *XLI Simpósio Brasileiro de Pesquisa Operacional, 2009*. Porto Seguro: Instituto Doris Aragon, 2010. (Anais do XLI Simpósio Brasileiro de Pesquisa Operacional). To appear.

38 AGUIAR, A. B. de; PINHEIRO, P. R.; COELHO, A. L. V. Optimizing energy consumption in heterogeneous wireless sensor networks: A novel integer programming model. In: *VI International Conference on Operational Research for Development - ICORD2007*. Fortaleza: [s.n.], 2007. (Proceedings of the VI International Conference on Operational Research for Development - ICORD2007), p. 496–505.

39 AGUIAR, A. B. de et al. Applicability of a novel integer programming model for wireless sensor networks. *International Journal of Computer Science and Information Security*, v. 3, n. 1, p. 7–13, August 2009.

40 AGUIAR, A. B. de et al. Scalability analysis of a novel integer programming model to deal with energy consumption in heterogeneous wireless sensor network. *Communications in Computer and Information Science*, Springer Berlin Heidelberg, Luxembourg, France, v. 14, p. 11–20, 2008.

41 LOUREIRO, A. et al. Redes de sensores sem fio. *Simpósio Brasileiro de Computação*, Jornada de Atualização de Informática, 2002.

42 MEGERIAN, S. et al. Exposure in wireless sensor networks: Theory and practical solutions. *Wireless Networks*, v. 8, n. 5, p. 443–454, 2002.

43 LI, X.-Y.; WAN, P.-J.; FRIEDER, O. Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, v. 52, n. 6, June 2003.

44 MEGERIAN, S.; POTKONJAK, M. Lower power 0/1 coverage and scheduling techniques in sensor networks. *University of California, Los Angeles, Technical Reports 030001*, January 2003.

45 QUINTãO, F.; NAKAMURA, F. G.; MATEUS, G. R. Evolutionary algorithm for the dynamic coverage problem applied to wireless sensor networks design. In: EDIMBURGO, UK. *IEEE Congress on Evolutionary Computation*. [S.l.], 2005.

46 QUINTãO, F. P.; NAKAMURA, F. G.; MATEUS, G. R. Uma abordagem evolutiva para o problema de cobertura em redes de sensores sem fio. *Revista Eletrônica de Iniciacão Científica (REIC) da Sociedade Brasileira de Computacão (SBC)*, v. 3, September 2004.

47 MOTE Battery Life Calculator. May 2007. Internet. Disponível em: <`http://www.xbow.com/Support/Sypport_pdf_files/PowerManagement.xls`>.

48 AFONSO, L. O. R. et al. Design of cellular network using lagrangean relaxation algorithm. In: CHU, H. W. et al. (Ed.). *International Conference on Computer, Communication and Control Technologies: CCCT'03 and The 9th International Conference on Information Systems Analysis and Synthesis: ISAS'03*. Orlando, Florida, USA: [s.n.], 2003. (Proceedings of International Conference on Computer, Communication and Control Technologies and The 9th International Conference on Information Systems Analysis and Synthesis), p. 154–168.

49 AGUIAR, A. B. de; PINHEIRO, P. R.; RODRIGUES, M. M. Um modelo para telefonia celular. In: SANT'ANNA, A. P. (Ed.). *XXXV Simpósio Brasileiro de Pesquisa Operacional*. Rio de Janeiro: Instituto Doris Aragon, Rio de Janeiro, 2003. (Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional).

50 AGUIAR, A. B. de et al. A multi-layer gsm network design model. In: SOBH, T.; ELLEITHY, K.; MAHMOOD, A. (Ed.). [S.l.]: Springer Netherlands, Dordrecht, 2010. cap. Novel Algorithms and Techniques in Telecommunications and Networking. To appear.

51 AGUIAR, A. B. de; PINHEIRO, P. R.; NETO Álvaro de M. S. Scalability analysis of a model for gsm mobile network design. In: SOBH, T.; ELLEITHY, K.; MAHMOOD, A. (Ed.). [S.l.]: Springer Netherlands, Dordrecht, 2010. cap. Novel Algorithms and Techniques in Telecommunications and Networking. To appear.

52 AGUIAR, A. B. de; PINHEIRO, P. R. Innovative algorithms and techniques in automation, industrial electronics and telecommunications: subtitle. In: SOBH, T. et al. (Ed.). Springer Netherlands, Dordrecht, 2007. XVI, cap. A Model for GSM Mobile Network Design, p. 365–368. CISSE2007. Disponível em: <http://www.springerlink.com/content/n22v2802771thv04>.

53 NETO, A. de M. S.; PINHEIRO, P. R.; AGUIAR, A. B. de. A novel model for multilayer gsm network design. *International journal of computer science and network security*, v. 9, p. 73–77, 2009.

54 AGUIAR, A. B. de et al. A novel model for optimized gsm network design. *International Journal of Computer Science and Information Security*, v. 4, p. 147–152, 2009.

55 BISCHOFF, E.; MARRIOTT, M. A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, v. 44, p. 267–276, 1990.

56 CHEN, C. F.; LEE, S. M.; SHEN, Q. S. An analytical model for the container loading problem. v. 80, p. 68–76, 1995.

57 FISHER, M. L. The lagrangian relaxation method for solving integer programming problems. *Manage. Sci.*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 50, n. 12 Supplement, p. 1861–1871, 2004. ISSN 0025-1909.

58 GEHRING, H.; BORTFELDT, A. A genetic algorithm for solving the container loading problem. *European Journal of Operational Research*, v. 131, p. 143–161, 2001.

59 GEHRING, H.; BORTFELDT, A.; MACK, D. A parallel hybrid local search algorithm for the container loading problem. v. 11, p. 511–524, 2004.

60 GEHRING, H.; MENSCHNER, K.; MEYER, M. A computer-based heuristic for packing pooled shipment containers. v. 44, p. 277–288, 1990.

61 GEORGE, J. A.; ROBINSON, D. F. A heuristic for packing boxes into a container. v. 7, p. 147–156, 1980.

62 HAESSLER, R. W.; TALBOT, F. B. Load planning for shipments of low density products. *European Journal of Operational Research*, v. 44, n. 2, p. 289–299, 1990.

63 KUBAT, P.; SMITH, J. M. A multi-period network design problem for cellular telecommunication systems. *European Journal of Operational Research*, v. 134, n. 2, p. 439–456, October 2001. Disponível em: <`http://ideas.repec.org/a/eee/ejores/v134y2001i2p439-456.html`>.

64 KUBAT, P.; SMITH, J. M.; YUM, C. Design of cellular networks with diversity and capacity constraints. *IEEE Transactions on Reliability*, v. 49, n. 2, p. 165–175, June 2000. Disponível em: <`http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/24/18992/00877334.pdf?arnumber=877334`>.

65 MARTINS, G. H. *Packing in Two and Three Dimensions*. Tese (Doutorado) — Naval Postgraduate School, Monterey, EUA, 2003.

66 MORABITO, R.; ARENALES, S. Abordagens para o problema do carregamento de contêineres. In: *Pesquisa Operacional*. [S.l.: s.n.], 1997. v. 17, n. 1, p. 29–56.

67 MORABITO, R.; ARENALES, S. An and/or-graph approach to the container loading problem. In: *International Transactions in Operations Research*. [S.l.: s.n.], 1994. v. 1, p. 59–73.

68 RIGOLON, A. A. et al. Relaxação lagrangeana com método de subgradiente aplicada no projeto de uma rede de telefonia móvel. In: SOCIEDADE BRASILEIRA DE PESQUISA OPERACIONAL - SOBRAPO. *XXXVII Simpósio Brasileiro de Pesquisa Operacional (SBPO)*. [S.l.]: Sociedade Brasileira de Pesquisa Operacional - Sobrapo, 2005.

69 SAVVIDES, A.; PARK, S.; SRIVASTAVA, M. B. On modeling networks of wireless microsensors. In: . [S.l.: s.n.], 2001, (ACM SIGMETRICS Performance Evaluation Review, 1). p. 318–319.

70 TSAI, R. D.; MALSTROM, E. M.; KUO, W. Three-dimensional palletization of mixed box sizes. v. 25, n. 4, p. 64–75, 1993.

71 ULLAH, F. Security protocols for wireless sensor networks. In: . Shaheed Zulfikar Ali Bhutto Institute of Science and Technology: International Journal of Computer Science and Information Security, 2009. v. 1, n. 1.

72 NOVELL INC. *OpenSuse 11.0 Start-Up*. [S.l.], June 2008.

73 TINYOS Community Forum - An open-source OS for the networked sensor regime. May 2007. Internet. Disponível em: <`http://www.tinyos.net`>.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo