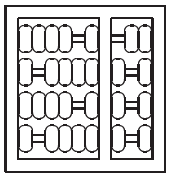

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Optimal Rectangular Partitions

Felipe C. Calheiros
Abilio Lucena
Cid C. de Souza

Technical Report - IC-01-016 - Relatório Técnico

December - 2001 - Dezembro

Optimal Rectangular Partitions

Felipe C. Calheiros
Institute of Computing, Universidade Estadual de Campinas
Caixa Postal 6176, 13084-971, Campinas-SP, Brazil
felipe.calheiros@ic.unicamp.br *

Abilio Lucena
Departamento de Administração, Universidade Federal do Rio de Janeiro
Av. Pasteur 250, Rio de Janeiro-RJ, 22290-240, Brazil
lucena@openlink.com.br †

Cid C. de Souza
Institute of Computing, Universidade Estadual de Campinas
Caixa Postal 6176, 13084-971, Campinas-SP, Brazil
cid@ic.unicamp.br ‡§

26th December 2001

Abstract

Assume that a rectangle R is given on the Euclidean plane together with a finite set P of points that are interior to R. A rectangular partition of R is a partition of the surface of R into smaller rectangles. The length of such a partition equals the sum of the lengths for the line segments that defined it. The partition is said to be feasible if no point of P is interior to a partition rectangle. The Rectangular Partitioning Problem (RGP) seeks a feasible rectangular partition of R with the least length. Computational evidence from the literature indicates that RGPs with non corectilinear points in P, denoted RGNLPs, are the hardest to solve to proven optimality. In this paper, some structural properties of optimal feasible RGNLP partitions are presented. These properties allow for substantial reductions in problem input size to be carried out. Additionally, a stronger formulation of the problem is also made possible. Based on these ingredients, a hybrid Lagrangian Relaxation - Linear Programming Relaxation exact solution algorithm is proposed. Such an algorithm has proved capable of solving RGNLP instances more than twice as large as those found in the literature.

Keywords: Rectangular partitions, optimality properties, Lagrangian relaxation, cutting planes.

*Research partially supported by CNPq.
†Research supported by FAPERJ (grant E26/71.906/00).
‡Research supported by CNPq (grant 300883/94-3) and FINEP (ProNEx-107/97).
§Corresponding author

1 Introduction

A rectangle R on the Euclidean plane is given, together with a finite set P of points that are interior to R. A partition of the surface of R into smaller rectangles is denoted a *rectangular partition*. Any such partition is said to be *feasible*, if no point of P lies in the strict interior of a partition rectangle. Given a rectangular partition, consider the sum of all straight line segments that define partition rectangles. Denote this sum the *length* of the rectangular partition. The Rectangular Partitioning Problem (RGP) is to find a feasible rectangular partition of minimum length. In what follows, unless specified otherwise, *feasible rectangular partitions* are to be referred, simply, as *partitions*.

The RGP belongs to a large class of problems where one seeks an *optimal* partitioning of a rectilinear polygon with *holes* inside. A hole, here, is meant to denote either a rectilinear polygon or else a point. The objective function involved may vary among the different problems in the class. A major motivation for the study of these problems stems from their practical applications in VLSI design. As a result, they have attracted a fair amount of interest from researchers in Computational Geometry (see, among others, [14, 2]). Most of these problems have shown to be quite challenging and many, including the RGP itself, are known to be $\mathcal{NP}$-hard (see [14]).

Several RGP approximation algorithms have been proposed in the literature ([8, 3, 13, 9, 10, 7, 6]). Exact solution algorithms, based on Integer Programming techniques, are discussed in [2] where extensive computational testing has been conducted. Tests involved approximate as well as exact solution algorithms and suggest that RGP instances where no two points of P are corectilinear are, computationally speaking, the hardest ones to solve. This special case of RGP is denoted the Rectangular Partition Problem with Non Corectilinear Points (RGNLP).

It is still an open question as to whether or not RGNLP is $\mathcal{NP}$-hard. Nevertheless, either way, a better understanding of the combinatorial structure of optimal RGNLP solutions should certainly be useful when tackling the general case (i.e. RGP). On the same vein, exact solutions for the small RGNLP instances tested in [2] have shed some light on properties associated with optimal RGNLP solutions. In this paper, some of these properties are introduced. They are used to identify a family of valid inequalities that must be satisfied by any RGNLP optimal solution. They are also used, in a geometric preprocessing test, to eliminate suboptimal variables in a Set Partitioning (SP) formulation of the problem. Typically, over 60% of the initial SP variables are eliminated in this way.

After preprocessing, further substantial reductions have been attained with the use of Lagrangian variable fixation tests. These tests combine upper bounds (generated by a Lagrangian heuristic) with Lagrangian relaxation lower bounds in an attempt to prove that a variable must either be in or out of an optimal solution. Lagrangian lower bounds have been computed in two different ways. The first one uses the Subgradient Method (SM) of [11] as usually done in the literature (see [5], for instance). The second one consists of a Relax and Cut algorithm [4], as introduced in [15, 16]. For such an algorithm, one has exponentially many inequalities as true candidates to Lagrangian dualization. An advantage of the Relax and Cut algorithm over the previous one is that it is capable of generating sharper lower bounds. Very few of the test instances considered could not be solved to

proven optimality exclusively through Lagrangian bounds. For these instances, an Integer Linear Programming (ILP) solver, CPLEX (versions 6.5 and 7.0) [1], was then used to accomplish the task. The ILP solver greatly benefited from previous reductions of problem input size.

The combination of geometric preprocessing, Lagrangian variable fixation tests and CPLEX into a *hybrid* solution algorithm proved very successful. RGNLP instances more than twice as large as those found in the literature have been solved to proven optimality. Indeed, for many of the instances tested, the dimensions involved were such that even after geometric preprocessing, CPLEX alone, under 384 Mb of RAM memory, could not run due to *insufficiency of memory*. Furthermore, for those larger size instances where, after preprocessing, CPLEX could be directly applied, the hybrid approach has proved a better alternative. After geometric reductions, SP instances with as many as $15,876$ rows and almost $2,044,619$ variables ($15,876$ rows and over $7,063,529$ variables, previously) were solved to proven optimality by the hybrid algorithm. These certainly rank amongst the largest SP instances ever solved to proven optimality.

This paper is organized as follows. Section 2 presents some basic definitions and results for RGP. A Set Partitioning formulation of the problem, as introduced in [2], is also described in that section. In Section 3 some geometrical properties that must be satisfied by any optimal RGNLP solution are introduced. Still in section 3, one shows how these properties can be used to eliminate suboptimal variables. Finally, section 3 is closed with the description of a family of strong valid inequalities for (the SP formulation of) RGNLP. In Section 4, Relax and Cut algorithms are described in some detail. In Section 5, a description of Lagrangian based lower and upper bounding algorithms for RGNLP are presented. Tests for eliminating suboptimal variables, together with details of the hybrid algorithm, close the section. Computational experiments are the subject of Section 6. Finally, in Section 7 some conclusions are reached and suggestions for future work are offered.

2 Basic results and an ILP formulation

In this section some definitions and basic results associated with RGP are presented. They highlight some discrete optimization aspects of the problem and motivate the description of a SP model of RGP as introduced in [2].

Given an instance $I = (R, P)$ of a RGP, let *GI(P)* be a *grid* induced by P, in R. Allow, without loss of generality, the left hand side of R to lay on the vertical axis. Thus *GI(P)* can then be defined as the set of straight line segments associated with the vertical and horizontal lines intersecting the points of P and those lines on which the sides of R lie. An intersection point between two such lines is denoted a *grid point*. Grid points are partitioned into two sets. The set of *terminal* points (i.e. points that belong to P) and the set of *Steiner* points (i.e. the remaining ones). Figure 1 illustrates the definitions. Black circles represent terminal points while white ones represent Steiner points. It should be noticed that, contrary to previously published works, grid points that lie on the boundary of R are also included here into the set of Steiner points. That allows for simplifications on some of the forthcoming proofs.

The result that follows, by Lingas *et al.* ([14]), relates optimal RGP solutions with *GI(P)*.

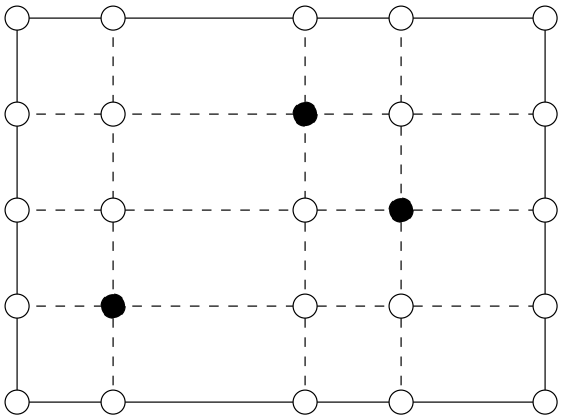

Figure 1: An instance $I = (R, P)$ of the RGP with 3 terminals and 22 Steiner points. The dotted segments are those in the grid *GI(P)*.

Theorem 2.1 *[14] Given an instance $I = (R, P)$ of RGP, all straight line segments inducing an optimal partition lie on grid GI(P).*

Given a set S of segments of *GI(P)*, the degree of any point in *GI(P)* with respect to S is defined as the number of segments of S that are incident on the point. A point of degree one is denoted an *island*. A point of degree two, with defining segments of S being orthogonal, is denoted a *knee*. The result that follows is used in [2] to introduce a Set Partitioning formulation of RGP.

Theorem 2.2 *[2] Given an instance $I = (R, P)$ of RGP and a set S of segments of* GI(P), *if S contains no islands nor knees (apart from the vertices of R) and the degree of every point in P is exactly two, then S defines a partition of R.*

For the remaining of this text, one shall use the term *grid segment* to denote a line segment containing exactly two grid points. In this case, the grid points must be located on the extremities of the line segment.

Prior to casting RGP as a Set Partitioning Problem (SPP), a definition for that problem is briefly recalled and a classical ILP formulation of it is given.

Consider a set $H = \{H_i : i \in I\}$, where $I = \{1, \ldots, m\}$. In association, assume that a set of subsets of H is given. Denote these subsets $K = \{K_j : j \in J\}$, where $J = \{1, \ldots, n\}$. A set $J' \subseteq J$ defines a partition of H if $\bigcup_{j \in J'} K_j = H$ and $K_j \cap K_\ell = \emptyset$, for any pair of indices $j, \ell \in J'$ with $j \neq \ell$. If costs $\{c_j : j \in J\}$ are associated with the elements of K, the cost of the partition induced by J' is computed as $\sum_{j \in J'} c_j$. For the SPP one seeks a partition of H with a minimum cost.

Denote by $J_i \subseteq J$, $i \in I$, the set of indices for those K_j, $j \in J$, that contain H_i. Conversely, denote by $I_j \subseteq I$, $j \in J$, the set of indices for those elements of H that are contained in K_j. Finally, let variable $x_j, j \in J$, control the inclusion or not of K_j into a partition of H. A $\{0-1\}$ ILP formulation of SPP is given by

$$\min \left\{ \sum_{j \in J} c_j x_j : x \in \mathcal{R}_0 \right\} \tag{1}$$

where $\mathcal{R}_0$ is a feasibility region described as

$$\sum_{j \in J_i} x_j = 1, i \in I \tag{2}$$

$$x_j \in \{0, 1\}, j \in J. \tag{3}$$

In order to cast RGP as SPP, corresponding sets H and K must be properly defined. Furthermore, appropriate costs for the elements of K must also be introduced. In that respect, define a *canonical rectangle* as a rectangle with sides lying on two consecutive vertical and two consecutive horizontal lines of *GI(P)*. Figure 2 illustrates the definition.

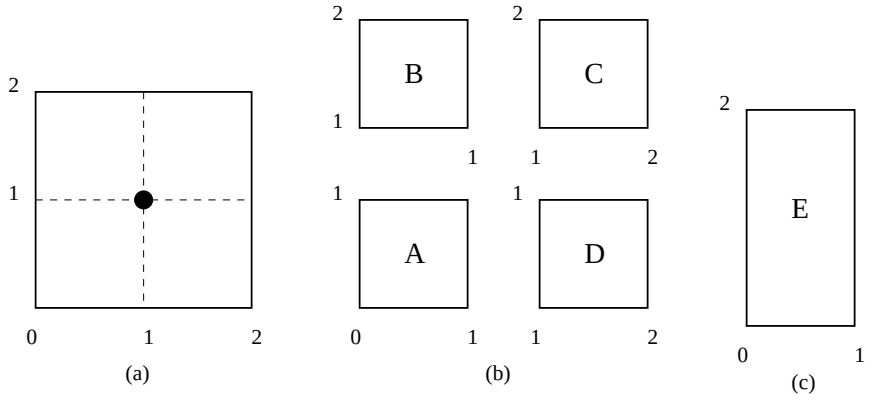

Figure 2: (a) Instance of the RGP. (b) Canonical rectangles in *GI(P)*. (c) A feasible rectangle.

A correspondence now becomes straightforward with H representing the set of all canonical rectangles in *GI(P)* and K representing the set of all those rectangles with sides lying on the defining lines of *GI(P)* and having no terminal points in their interior. Denote these rectangles *feasible*.

It is clear that every canonical rectangle must be feasible. Remaining feasible rectangles can be formed through the union of canonical rectangles. For instance, in Figure 2(c) rectangle E is the union of the canonical rectangles A and B. In order to highlight the fact that the elements of K represent feasible rectangles, they will be denoted, from this

point on, $\{R_j : j \in J\}$, instead of $\{K_j : j \in J\}$. As defined above, one should notice that rectangles in an optimal rectangular partition of R must be contained in the set of feasible rectangles. Furthermore, it should also be clear that those optimal rectangles must *cover* all canonical rectangles.

Posed as above, any rectangular partition of R with respect to P corresponds to a partition of R into feasible rectangles. Accordingly, for every feasible rectangle R_j in K, a cost c_j is computed as the sum of the perimeter of R_j plus the sum of the length of those sides of R_j which belong to the boundary of R. Under these costs, an optimal SPP solution will sum twice as much as the length of an optimal rectangular partition of R. A SPP formulation of the RGP thus follows naturally. The formulation has one equality constraint for every canonical rectangle and one variable for every feasible rectangle and thus involves $O(|P|^2)$ constraints and $O(|P|^4)$ variables.

For the remaining of the text, being somewhat lax with the notation, a canonical rectangle indexed by $i \in I$ may be referred, simply, as canonical rectangle i. Accordingly, the feasible rectangle indexed by $j \in J$ may simply be referred as feasible rectangle j.

3 Some properties of optimal RGNLP solutions

Given an instance of RGNLP, let S^* denote an arbitrary optimal solution for that instance. As explained before, such a solution can be viewed as a collection of grid segments and their intersecting points. Furthermore, as implied by Theorem 2.2, for any feasible partition, the degree of a grid point cannot be one and, if it equals two, then the two segments incident on it cannot be orthogonal (exceptions being the vertices of R). The results that follow establish necessary conditions for the degrees of the grid points in an optimal RGNLP partition, S^*.

Proposition 3.1 *No Steiner point in S^* has degree 4 and every terminal point has degree 2.*

Proof: Assume by contradiction that p is a Steiner point in S^* with degree 4. Suppose, without loss of generality, that the terminal point corresponding to the vertical grid line passing through p is located below p (this situation is depicted in Figure 3). Let p' be the first Steiner point in S^* encountered when one moves upwards form p along that line and such that it has two horizontal grid segments of S^* incident on it (these segments may, eventually, belong to the boundary of R). If one slides as far rightwards as possible all the grid segments of S^* between p and p', one obtains a new solution whose length does not exceed that of S^*. In this process, the degree of p is reduced to 3. Furthermore, some vertical grid segments in S^* (boundary of R segments may be included) will end up coinciding. That, in turn, leads to a shorter new feasible partition, thus contradicting the optimality of S^*. A proof for terminal points follows analogous steps. □

Proof of Proposition 3.1 indicates that every vertical line segment in S^* must be delimited by two Steiner points of degree 3 (forming two "T-junctions"; one of which is upside-down). Moreover, it is straightforward to see that these line segments must contain a

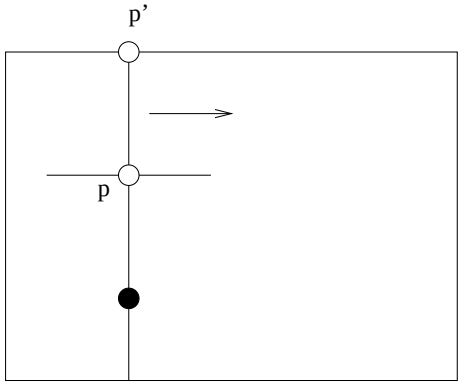

Figure 3:

terminal point. That holds since, otherwise, horizontally sliding the segment would produce a better solution. Notice that symmetrical arguments hold for terminal points traversed by horizontal lines. One then concludes that in any optimal solution and for each terminal point there must be precisely two Steiner points of degree 3 which play the role of *delimiters* for the segment containing the terminal point.

Proposition 3.2 *The number of Steiner points of degree 3 in* S^* *is* $2|P|$.

Proof: Assume that p is a terminal point which belongs to a vertical segment of S^* (a proof for the horizontal case follows analogously). Since the degree of p is 2, no other segment of S^* is incident on it. Moreover, as pointed out before, the vertical line segment of S which contains p must be delimited by exactly two Steiner points of degree 3. Therefore, since all points in P are assumed to be non corectilinear, the number of Steiner points of degree 3 must add to $2|P|$, i.e. two for every terminal point. □

Proposition 3.3 *The number of feasible rectangles in* S^* *is* $|P| + 1$.

Proof: From Proposition 3.1 one knows that no partition rectangle exists in S^* having a terminal point as a vertex. Therefore all partition rectangle vertices in S^* are Steiner points (vertices of R, which all have degree zero, included). Notice, from Proposition 3.1, that, excluding the vertices of R, each of these Steiner points must have a degree of 3. Therefore, they must be vertices for exactly two partition rectangles. Denote by r the number of partition rectangles in S^*. Accordingly, let s represent the number of Steiner points in S^*. Then $r = \frac{2.s+4}{4} = \frac{2.2|P|+4}{4} = |P| + 1$ follows from Proposition 3.2. □

The previous result allows RGNLP to be cast as the Cardinality Constrained SPP

$$\min \left\{ \sum_{j \in J} c_j x_j : x \in \mathcal{R}_1 \right\} \tag{4}$$

where $\mathcal{R}_1$ is the feasibility region defined by

$$\sum_{j \in J_i} x_j = 1, i \in I \tag{5}$$

$$\sum_{j \in J} x_j = |P| + 1. \tag{6}$$

$$x_j \in \{0, 1\}, j \in J. \tag{7}$$

It should be noticed that although some feasible solutions in $\mathcal{R}_0$ might be cut off by (6) an optimal solution remains.

3.1 Reducing the ILP model and valid inequalities

The first observation reducing the number of variables in the ILP formulation follows immediately from Proposition 3.1. Since in any optimal solution S^* of a RGNLP instance the degree of a terminal is two and both segments incident to it belong to the same straight line, no feasible rectangle in S^* can have this terminal as one of its vertices. Thus, all variables corresponding to feasible rectangles with at least one vertex on a terminal point can be dropped from the formulation.

Before discussing further reductions in the ILP formulation, it is necessary to introduce the definition below.

Definition 3.1 *Let R_j be a feasible rectangle of a RGNLP instance and let S be a segment representing one side of R_j. The* extended side *relative to S, denoted by ext(S), can be obtained as follows: if S contains a terminal point then ext(S) = S; otherwise, ext(S) is the segment of the straight line r that supports S obtained by extending S in the direction of the unique terminal point p lying on r until the first Steiner point past p is reached.*

To illustrate the above definition, consider the rectangle R_j given in Figure 4. The *extended sides* of R_j are: $\{ \overline{p_{b1}p'_3}, \overline{p_{b2}p'_2}, \overline{p_{b3}p'_2}, \overline{p'_4p'_3} \}$.

The result below can be easily derived from the discussion following Proposition 3.1.

Lemma 3.1 *If a feasible rectangle R_j belongs to an optimal solution S^* of a RGNLP instance then its extended sides are in S^*.*

The result below allows for further reductions on the number of variables in the ILP formulation of RGNLP.

Proposition 3.4 *Let R_j be a feasible rectangle of a RGNLP instance. If ℓ_1 and ℓ_2 are two extended sides of R_j whose intersection is non empty and occurs at a point which is not at the extremities of ℓ_1 or ℓ_2, then R_j is not in an optimal solution.*

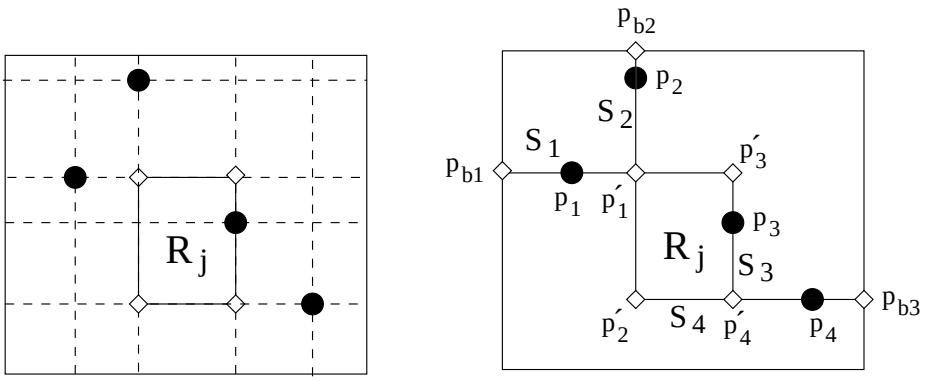

Figure 4: (a) RGNLP instance. (b) Extended sides of R_j.

Proof: Consider again the example given in Figure 4. Suppose that R_j belongs to an optimal solution S^*. Since no *knees* are allowed in S^* and each Steiner point must have degree at most 3, then S^* must contain either one of the extended sides $\overline{p_{b1}p'_3}$ or $\overline{p_{b2}p'_2}$, but not both. Still, irrespective of which of the two extended sides is in S^*, p'_1 would be no delimiter of a Steiner point (see discussion preceding Proposition 3.2). This is a contradiction. □

Extended sides can be used to generalize the definition of intersection between feasible rectangles. This generalization is convenient since it allows for an easy way to describe valid inequalities for the set of optimal solutions of the ILP formulation. These inequalities are used later as cutting planes in a Lagrangian relaxation algorithm. These steps are done below.

Definition 3.2 *Let R_i and R_j be two feasible rectangles of a RGNLP instance. These rectangles are said to have an* extended intersection *if there are extended sides ℓ_i of R_i and ℓ_j of R_j such that the intersection of ℓ_i and ℓ_j is non empty (see Figure 5). The existence of the* extended intersection *between R_i and R_j is denoted by $R_i \wedge R_j$.*

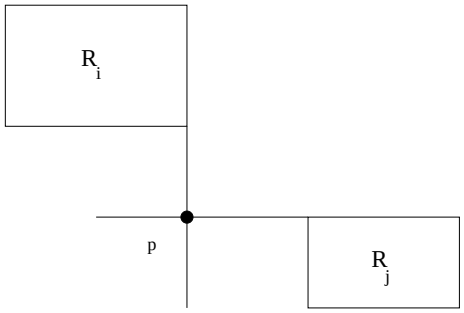

Figure 5: Extended intersection between feasible rectangles.

Proposition 3.5 *Let R_i and R_j be two feasible rectangles of a RGNLP instance and y_i and y_j be the corresponding variables in the ILP formulation. Then, if $R_i \wedge R_j$, the inequality $y_i + y_j \leq 1$ is valid for all optimal solutions.*

Proof: Assume that both y_i and y_j are set to one, meaning that both R_i and R_j are in the optimal solution. From Lemma 3.1, this implies that p (see Figure 5) has degree four which contradicts Proposition 3.1. □

The following observations apply. First, notice that if the usual intersection between two feasible rectangles is non empty then they have an extended intersection. Moreover, one could build the *extended intersection graph* of the set of all feasible rectangles of an RGNLP instance. Such a graph is defined in a natural way with the set of vertices in one-to-one correspondence with the set of feasible rectangles and, by including edges for all pairs of vertices that are associated with two feasible rectangles having an extended intersection. It is easy to see that any optimal solution of RGNLP is represented by a stable set in the *extended intersection graph*. Therefore, valid inequalities for the *stable set polytope*, such as the *clique* and *odd hole* inequalities (cf. [19]), can be used in a cutting plane algorithm for RGNLP.

In particular, the inequalities given in Proposition 3.5 are *clique inequalities*. However, in this simple form, they are unlikely to produce good computational results. To strengthen them, one has to extend the support graph of the inequalities so as to represent maximal cliques. Thus, if C denotes the indices for variables (i.e. rectangles) in one such inequality, clique inequalities can be written as

$$\sum_{j \in C} x_j \leq 1, C \in \mathcal{C} \tag{8}$$

where $\mathcal{C}$ is the set of all subsets of vertices of the extended intersection graph which form maximal cliques.

For a feasibility region $\mathcal{R}_2$, defined by (5)–(6) and (8), a valid formulation for RGNLP is

$$\min \left\{ \sum_{j \in J} c_j x_j : x \in \mathcal{R}_2 \right\}. \tag{9}$$

Though inequalities (8) are redundant for formulation (4), that is not necessarily the case for the continuous relaxation of it. To use them in a cutting plane algorithm, separation can be carried out in a greedy fashion by first building an associated support graph (i.e., a clique). Then, a lifting procedure takes place for as long as a maximal clique is not reached.

4 Relax and Cut algorithms

Assume that a formulation for a $\mathcal{NP}$-hard combinatorial optimization problem (such as the SPP) is given. Assume as well that exponentially many inequalities may be included in it

(as, for instance, when the clique inequalities in (8) are appended to the SPP formulation in (1)). Such a formulation can be generically described as:

$$\min\{cx: \quad Ax \leq b, \quad x \in X\}, \tag{10}$$

where $x \in \mathbb{B}^n$ is an array of variables, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and, finally, $X \subseteq \mathbb{B}^n$. Assume, as it is customary in Lagrangian relaxation, that

$$\min\{cx: \quad x \in X\} \tag{11}$$

is an easy (polynomial time) problem to solve. On the other hand, in what is unusual for the application of Lagrangian relaxation, let m be exponential in n. In spite of that, assume one wishes to dualize

$$\{a_i x \leq b_i: \quad i = 1, 2, \ldots, m\} \tag{12}$$

in a Lagrangian fashion and let $\lambda \in \mathbb{R}^m_+$ be the corresponding array of Lagrangian multipliers. Subgradient Optimization (SO) could then be used to solve

$$\max_{\lambda \geq 0} \big\{ \min\{(c + \lambda A)x - \lambda b: \quad x \in X\} \big\}. \tag{13}$$

Optimization is typically conducted here in an interactive way with multipliers being updated so that the optimal value of (13) is attained. For the sake of completeness, let us briefly review the Subgradient Method (SM) of [11] as implemented in [5].

At any given iteration of the SM, for given feasible values of Lagrangian multipliers λ, let $\overline{x}$ be an optimal solution to (13). Denote by z_{lb} the value of this solution and let z_{ub} be a known upper bound on (10). Additionally, let $g \in \mathbb{R}^m$ be an array of subgradients associated with the relaxed constraints. For the current solution, $\overline{x}$, g is evaluated as

$$g_i = (b_i - a_i \overline{x}), \quad i = 1, 2, \ldots, m. \tag{14}$$

In the literature (see [5], for instance) Lagrangian multipliers are usually updated by firstly determining a *step size* θ,

$$\theta = \frac{\alpha(z_{ub} - z_{lb})}{\sum\limits_{i=1,\ldots,m} g_i^2}, \tag{15}$$

where α is a real number assuming values in $(0, 2]$. One would then proceed to computing

$$\lambda_i \equiv \max\{0; \lambda_i - \theta g_i\}, \quad i = 1, \ldots, m, \tag{16}$$

and then move on to the following iteration of the SM.

Under the conditions imposed here, the straightforward use of updating formulas (14)–(16) is not as simple as it might appear. The reason being the exceedingly large number of inequalities that would, typically, be dualized.

Inequalities in (12), for every SM iteration, may be divided into three groups. The first one contains inequalities that are violtaed by $\overline{x}$. The second group is for those inequalities that have nonzero multipliers currently associated with them. Notice that an inequality may be, simultaneously, in the two groups just defined. Finally, the third group consists of the remaining inequalitites and evaluating their subgradients would account for most of the computational burden at a SM iteration.

One should notice that, under the classification proposed above, inequalities may change groups from one SM iteration to another. It should also be noticed that the only multipliers that may directly contribute to Lagrangian costs ($c+\lambda A$), at any given SM iteration, are the ones associated with inequalities in groups one and two. These inequalities are thus denoted *active inequalitites*. Conversely, group three inequalities are denoted *inactive inequalitites*. Finally, a point to be made is that, from (16), multipliers for inactive inequalitites will not change their present null values at the end of the current SM iteration.

Clearly, inactive inequalitites do not directly contribute to Lagrangian costs (at a current SM iteration). On the other hand, they do play a decisive role in determining the value of θ. Typically, for the application being described, the number of strictly positive subgradients, amongst inequalities in group three, tends to be huge. Consequently, the value of θ would result extremelly small, leaving multiplier values virtually unchanged from iteration to iteration. Bearing this in mind, one may choose to apply (14)–(16) exclusively to subgradients and multipliers associated with active inequalities. That results in a dynamic scheme where the set of active multipliers may continuously change. Notice, in association, that a multiplier may become active at one given SM iteration, then become inactive at a subsequent one and become, yet again, active at a later iteration.

The scheme above is, in a sense, very much akin to cutting planes generation. It has been firstly proposed and successively used for the Steiner Problem in Graphs in [15, 16]. Later on, it has been used for the Edge-Weighted Clique Problem in [12], for the Quadratic Knapsack Problem in [18] and for the Vehicle Routing Problem in [17].

The term *Relax and Cut* was coined in [4] for an algorithm that, although different from the one in [15, 16], bears a number of points in common with it. In this paper, Relax and Cut is used to denote the whole class of Lagrangian Relaxation algorithms where inequalities are dualized *on the fly* (as they become violated at the solution to a Lagrangian Relaxation subproblem). This class therefore encompasses the algorithm in [15, 16] as well as the one in [4].

5 RGNLP bounds and variable fixing

In this Section, at the light of the discussions in Sections 3 and 4, lower bounds are proposed for SPP formulations (4) and (9) of RGNLP. Additionally, it is also described how Lagrangian modified costs are used as an input to a heuristic that attempts to generate feasible RGNLP SP solutions. The proposed lower and upper bounds are then combined in order to try to fix variables to their optimal values. Finally, a hybrid exact solution approach, which uses Lagrangian relaxation as a preprocessor to ILP solver CPLEX 6.5 is introduced.

5.1 Lower bounds for RGNLP cast as a SPP

Two different Lagrangian RGNLP lower bounds are proposed in this study. The first one is obtained by applying the SM method as usually done in the literature (see [5], for instance). The second one is generated through a Relax and Cut algorithm, as described in Section 4. For both algorithms, basically the same Lagrangian subproblem must be solved at every iteration of the SM. In order to introduce that problem, generically denote by λ the set of Lagrangian multipliers involved in either case. For the first Lagrangian approach, multipliers λ are associated with constraints (5) and are therefore unrestricted in sign. For the Relax and Cut algorithm, $\lambda = (\gamma, \psi)$ and the multipliers are associated with constraints (5) and (8). It thus follows that multipliers in γ are unrestricted in sign while those in ψ must be non negative. Under multipliers λ, RGNLP Lagrangian bounds are obtained by solving

$$\min \left\{ \sum_{j \in J} \overline{c}_j x_j + \xi(\lambda) : \sum_{j \in J} \overline{x}_j = |P| + 1, x_j \in \{0,1\} \forall j \in J \right\} \tag{17}$$

where $\overline{c}$ are the Lagrangian modified costs and $\xi(\lambda)$ is a constant that depends on the value of dualized constraint coefficients as well as those of λ.

Irrespective of the type of Lagrangian algorithm that is associated with lower bounds in (17), the SM is allowed to run for up to a given fixed number of iterations. The value of parameter α in (15) is to be halved after a given fixed number of consecutive SM iterations without an overall improvement on the best lower bound so far attained.

Let $\overline{x}$ denote an optimal solution to (17). Such a solution is easily obtained by setting to one the $|P|+1$ variables with the least Lagrangian costs while setting to zero the remaining ones. For the Relax and Cut algorithm, a separation problem must be solved in order to identify clique inequalities (see Proposition 3.5) that violate $\overline{x}$. For the computational results in Section 6 this separation problem has been solved as follows. First, the subgraph induced by the vertices of the extended intersection graph corresponding to the rectangles indexed by the variables in $\overline{x}$ is built. Then, a greedy algorithm is applied that looks for maximal cliques in this subgraph. The details of this algorithm are omitted here since the implementation is very similar to that proposed in [19].

5.2 Primal bounds

Lagrangian modified costs $\{\overline{c}_j : j \in J\}$ are used as an input to a greedy SPP heuristic. The procedure is initiated by fixing to one that variable, say x_{j_1}, with the least Lagrangian cost. As a result, variables $x_j, j \in J \setminus \{j_1\}$, such that either $I_{j_1} \cap I_j \neq \emptyset$ or else Proposition 3.5 applies, can then be disregarded from further consideration. This basic step of choosing a variable to be fixed to one and then eliminating those variables that can not share a solution with it, is to be sequentially repeated (for those variables that are not eliminated at the end of the previous iteration). In the process two possible outcomes are to be expected: either a feasible SPP solution will eventually result or else infeasibility of the solution being extended is detected. Throughout the application of the SM, this simple greedy heuristic is called a number of times, for different values of λ.

5.3 Attempting to fix variables

Consider the impact of forcing $x_{j_1} = 1, j_1 \in J$, in a feasible SPP solution. For given values of the Lagrangian multipliers, a valid lower bound on the least cost SPP solution where $x_{j_1} = 1$ can easily be computed. That is attained by further restricting (17) with $x_{j_1} = 1$ and $x_j = 0$ for $j \in J \setminus \{j_1\}$, such that either $I_{j_1} \cap I_j \neq \emptyset$ or else Proposition 3.5 applies. Let $z_{lb}(j_1)$ be the corresponding lower bound. In association, let z_{ub} be a known RGNLP upper bound. Recalling that the objective function in (4) relates with partition rectangle perimeters, whenever $z_{lb}(j_1) + 2.0 > z_{ub}$, variable x_{j_1} can be eliminated. One should notice that by adding 2.0 to $z_{lb}(j_1)$, above, one may end up eliminating variables that are part of an optimal solution. That, in turn, may force the underlying problem to become infeasible. In that case, optimality of the upper bound would be attested. An $O(|J|)$ implementation of this test has been used in the computational experiments of Section 6.

The lower bound proposed above, for a least cost RGNLP partition which contains rectangle R_{j_1}, is quite expensive in computational terms. Therefore, they have only been used for the very last iteration of the SM, whenever optimality could not be proven. A cheaper alternative, which can be used for every iteration of the SM, can be computed as follows. Firstly, let z_{lb} be the value of an optimal solution $\overline{x}$ to (17) and compute $\overline{c}_l = \max_{j \in J}\{\overline{c}_j : \overline{x}_j = 1\}$. Then, whenever $z_{lb} - \overline{c}_l + \overline{c}_{j_1} + 2.0 > z_{ub}$ variable x_{j_1} can be eliminated.

A similar style test (to the one above) can be devised in an attempt to prove that a given variable must be part of an optimal solution. Nevertheless, for the test instances in Section 6 this option has proved very ineffective.

5.4 A hybrid approach

Given an instance of RGNLP, after geometric reduction tests, assume that optimality could not be proven through Lagrangian lower and upper bounds. One may then move on an apply an ILP solver, such as CPLEX 6.5, to the further reduced instance one has been left with. Lagrangian relaxation could then be seen as one of the steps in a hybrid algorithm for solving RGNLP to proven optimality. Within such an approach, geometric and Lagrangian reduction tests act as preprocessors for the LIP solver.

After a RGNLP instance has been preprocessed, for the feasible rectangles (variables) one is left with, consider a polyhedral region, $\mathcal{R}_3$, given by

$$\sum_{j \in J_i} x_j = 1, \quad i \in I \tag{18}$$

$$\sum_{j \in J} x_j = |P| + 1 \tag{19}$$

$$0 \leq x_j \leq 1, \quad j \in J. \tag{20}$$

A valid LP lower bound for RGNLP is thus given

$$\min \left\{ \sum_{j \in J} c_j x_j : x \in \mathcal{R}_3 \right\} \tag{21}$$

and corresponds to the lower bound generated by the ILP solver at the root node of the enumeration tree.

A hybrid approach, with lower bounds generated, in an initial stage, by a Relax and Cut algorithm, presents an additional benefit. Clique inequalities (8) that are left dualized at the end of the SM run could now be appended to (18)–(20). As a result, initial LP bounds, for the instances in Section 6, tend to be improved (considerably at times).

6 Computational experiments

The two versions of the hybrid algorithm of Section 5 have been implemented and computationally tested. Tests were performed on an Intel based Pentium III computer running at 450 MHz and having 384 Mb of RAM memory. The operating system used was Linux, kernel 3.2. Geometric reduction tests and all Lagrangian algorithms were coded in `C++`. Compilation was carried out under GNU's `gcc` compiler under optimization option -O3. At the end of this section we report on a few tests with much larger instances that were performed on a `DEC` machine with 4 Gb of RAM and powered by an Alpha processor running at 675 MHz.

RGNLP instances were randomly generated as suggested in [2]. Firstly, a square R with ℓ-units sides, on the Euclidean plane, is considered. A number $|P|$ of non correctilinear points with positive integral valued coordinates, located in the strict interior of R, are then drawn from the uniform distribution. Along the generation process, a point was dismissed whenever it turned out to be correctilinear with any of the points previously drawn. A total of 42 test instances have been used in the study. The 25 instances with $15 \leq |P| \leq 39$ are the ones introduced in [2] and are denoted *small size* instances. Among these, the ones with $|P| \leq 19$ have $l = 20$ while the others have $l = 50$. The remaining 14 instances have been generated specifically for this study. They have $l = 100$ and $|P|$ ranging from 40 to 90. Among them, the ones with $|P| \leq 60$ are denoted *medium size* instances. Remaining ones are denoted *large size* instances.

6.1 Geometric reductions

Two different types of problem input size reductions, based on some geometric properties of RGNLPs, have been implemented in this study. The first one excludes feasible rectangles with terminal points as vertices (see Proposition 3.1). The second one is based on Proposition 3.4. The impact of those tests is shown on Tables 1 and 2. Columns on these tables give, respectively, $|P|$, the number of constraints for SP formulation (4), the initial number of SP formulation variables, the number of variables remaining after the first reduction test,

the number of variables remaining after the two reduction tests, the percentage of remaining variables (over initial ones) after the two reduction tests, the number of nonzero SP elements after reductions, and final instance density.

As it can be appreciated from the results, reductions in excess of 70% have been attained even for large instances. Reduction tests impact, in terms of enhancing one's capability of solving larger RGNLP instances, is considerable. That will become more evident for the computational results that follow.

Even after geometric reductions, the resulting SP instances are still very large. They could have as many as $8,281$ rows and $721,210$ columns. Set Partitioning instances of such a magnitude rank amongst the largest ever solved, to proven optimality, in the literature.

6.2 Initial experiments with CPLEX

After geometric reductions are carried out, experiments with CPLEX 6.5, under default settings, have been conducted. Cardinality Constrained SPP model (4) of RGNLP is the one used in the experiments. One should notice that, even after geometric reductions, instances with $|P| > 50$ could not be tackled by CPLEX due to insufficient RAM memory. Table 3 therefore only contains entries for instances with up to 50 points. Columns in that table correspond, respectively, to $|P|$, CPU time (in seconds) for solving the very first LP relaxation, corresponding LP relaxation lower bound, CPU time (in seconds) required for proving optimality, optimal solution value, total number of CPLEX iterations, number of nodes in the enumeration tree and duality gap between optimal RGNLP solution and LP relaxation lower bound.

As it can be appreciated from the results on Table 3 the SP formulation of RGNLP appears to be very strong indeed. Duality gaps exist for only 6 out of 29 instances considered. One should also notice that CPU times required to prove optimality have been quite modest, bearing in mind that SP problems with up to 2601 rows and 118187 columns are involved (369,469 columns before geometric reductions are applied).

6.3 Traditional Lagrangian bounds and the hybrid algorithm

Lagrangian relaxation RGNLP lower and upper bounds, based on the relaxation of (4), as expressed in (17), are shown on Tables 4 and 5. For every test instance, table columns correspond, respectively, to $|P|$, best upper bound generated, best lower bound generated, percentage gap between best upper and lower bounds, CPU times in seconds, number of SM iterations performed, number of variables remaining after the Lagrangian algorithm, percentage of variables fixed by the Lagrangian algorithm with respect to those left after the geometric reductions, percentage of the original variables (i.e. prior to geometric reduction tests) fixed by the Lagrangian algorithm. Whenever "—" appears for lower bound and gap entries, the corresponding instance has been solved to proven optimality.

The SM is initiated with $\alpha = 2.0$ and run for up to 2000 iterations. Parameter α is halved after 80 consecutive iterations of the SM without an overall improvement on the best lower bound so far generated. The Lagrangian heuristic is initially called for every one of first 100 iterations of the SM and, after that, for every other iteration of the SM.

$\|P\|$	Rows	Cols	Cols Red1	Cols Red2	% Red	Nonzero	Dens (%)
15	256	7262	5767	2810	38.69	36407	5.06
16	289	9292	7438	3356	36.12	48336	4.98
17	324	11131	9075	4075	36.61	62648	4.74
18	361	13319	10852	4866	36.53	77150	4.39
19	400	15994	13202	5821	36.39	98917	4.25
20	441	18731	15638	6707	35.81	119963	4.06
21	484	22214	18721	7941	35.75	151921	3.95
22	529	26286	22278	9053	34.44	185805	3.88
23	576	28434	24116	10440	36.72	204933	3.41
24	625	32855	28052	12046	36.66	248394	3.30
25	676	37952	32613	13600	35.83	298578	3.25
26	729	43508	37608	15295	35.15	352241	3.16
27	784	47767	41531	17276	36.17	399972	2.95
28	841	54646	47677	19492	35.67	476564	2.91
29	900	62133	54544	21367	34.39	553183	2.88
30	961	69612	61472	24371	35.01	662948	2.83
31	1024	76269	67442	26866	35.23	731061	2.66
32	1089	85703	76156	28901	33.72	830162	2.64
33	1156	94713	84253	32416	34.23	956873	2.55
34	1225	104149	93425	35260	33.86	1104910	2.56
35	1296	116590	104952	38505	33.03	1271941	2.55
36	1369	123787	111309	42282	34.16	1343419	2.32
37	1444	149378	135534	47007	31.47	1778684	2.62
38	1521	143874	130125	50064	34.80	1648905	2.17
39	1600	157432	142751	54011	34.31	1835457	2.12
45	2116	258502	237508	84890	32.84	3472641	1.93
46	2209	265395	243804	92472	34.84	3638703	1.78
49	2500	351298	325099	111557	31.76	5218077	1.87
50	2601	369469	342116	118187	31.99	5470033	1.78
53	2916	446723	415282	142444	31.89	6994448	1.68
54	3025	508815	476591	147344	28.96	8225872	1.85
57	3364	551419	515173	179106	32.48	9261540	1.54
58	3481	609384	572002	184283	30.24	10338239	1.61
60	3721	652141	610827	208216	31.93	11309285	1.46

Table 1: SPP column reductions. Small and medium sized instances.

$\|P\|$	Rows	Cols	Cols Red1	Cols Red2	% Red	Nonzero	Dens (%)
80	6561	1729832	1648992	514078	29.72	39679724	1.18
83	7056	1851493	1765696	565031	30.52	42384352	1.06
84	7225	1932570	1844715	593793	30.73	45271553	1.06
85	7396	2047803	1955262	619038	30.23	48487932	1.06
86	7569	2045611	1952451	637382	31.16	48377078	1.00
87	7744	2160537	2064750	667367	30.89	52735326	1.02
88	7921	2225525	2128110	682492	30.67	54023955	1.00
89	8100	2208191	2110458	698859	31.65	52676355	0.93
90	8281	2455502	2351651	721210	29.37	59628674	1.00

Table 2: SPP column reductions. Large sized instances.

As one should notice from Table 4, optimality has been proved for 22 out of the 29 instances solved by CPLEX on Table 3. Furthermore, in spite of the fact that optimality could not always be proven, the CPU times quoted in Table 4, for instances with $|P| \geq 29$, have always been less than the corresponding figures for Table 3. It should also be noticed that the number of variables remaining for those instances where optimality could not be proven is typically fairly small. A hybrid approach where CPLEX is used to solve these reduced instances thus seem an attractive proposition. That is indeed the case as it can be appreciated from the results that follow.

Table 6 contains CPLEX statistics for those instances with $|P| \leq 80$ which remained unsolved (to proven optimality) in Table 4. Columns on Table 6 follow an identical pattern to that on Tables 3. Instances with over 80 points, even after the geometric and Lagrangian reductions, still could not be tackled by CPLEX 6.5 due, yet again, to insufficiency of RAM memory. One should notice that all of these instances with $|P| \leq 50$ which remained unsolved in Table 4, could now be solved to proven optimality for an additional increase in CPU times of less than 10 seconds. In summary, for the 29 test instances with $|P| \leq 50$, the hybrid algorithm managed to beat CPLEX 6.5, run as a stand alone procedure, 19 times. Overall, CPLEX only managed to beat the performance of the hybrid algorithm for 10 of the rather small 14 instances where $|P| < 29$.

Another experiment has been conducted in an attempt to improve the quality of the Lagrangian bounds. That is justified by noticing that, for many of the larger instances with $|P| \leq 80$, Lagrangian lower bounds in Table 4 are bellow their theoretical best possible values (i.e. the corresponding LP relaxation values in Tables 3 and 6). Allowing the SM to run for longer has the potential of improving those bounds, albeit at an increase in CPU times.

At this point, one should notice that only one of the 10 test instances with $|P|$ ranging between 80 and 90 has been, so far, solved to proven optimality (either by CPLEX alone or the hybrid algorithm). For those instances, the SM was allowed to run for up to 8000 iterations with α being halved after 300 consecutive iterations without an overall improvement on the value of the best lower bound so far attained. The results obtained are shown

$\|P\|$	Time Relax	Opt Relax	Total Time	Integer Opt	Iter	B&B Nodes	GAP (%)
15	0.17	380	0.66	380	226	—	—
16	0.29	386	0.96	386	291	—	—
17	0.45	395.33	4.14	398	408	2	0.67
18	0.88	408	2.21	408	424	—	—
19	0.85	414	2.89	414	415	—	—
20	0.69	1022	3.32	1022	383	—	—
21	1.08	1046	4.96	1046	397	—	—
22	2.11	1062	16.47	1062	569	1	—
23	1.34	1052	7.06	1052	464	—	—
24	3.58	1134	11.18	1134	724	—	—
25	4.30	1112	14.75	1112	801	—	—
26	3.87	1142	17.05	1142	716	—	—
27	8.68	1140	24.37	1140	1010	—	—
28	6.60	1156	28.50	1156	797	—	—
29	11.02	1174	67.98	1174	1002	—	—
30	12.51	1174	50.30	1174	1076	—	—
31	21.86	1220	64.06	1220	1463	—	—
32	14.44	1220	64.65	1220	1199	—	—
33	18.47	1246	80.03	1246	1327	—	—
34	16.81	1236	94.78	1236	1326	—	—
35	52.58	1269	217.80	1270	2366	2	0.08
36	39.37	1279	176.73	1280	1821	2	0.08
37	95.06	1278	286.49	1278	2791	—	—
38	104.50	1314	329.60	1314	3022	—	—
39	89.42	1324	260.22	1324	2715	—	—
45	225.60	2749	711.09	2752	4302	2	0.11
46	195.48	2794	643.29	2794	3570	—	—
49	171.30	2802	871.57	2802	3468	—	—
50	468.76	2833	1258.27	2834	5760	9	0.04

Table 3: Results for CPLEX 6.5

$\|P\|$	UB	LB	GAP (%)	Time	Iter	Var Rem	% Var Fix	% Var Fix2
15	380	—	—	1.12	399	119	95.77	98.36
16	386	—	—	1.67	511	125	96.26	98.65
17	398	395.2869	0.68	3.94	2000	198	95.14	98.22
18	408	—	—	4.15	795	110	97.74	99.17
19	414	—	—	4.16	543	363	93.76	97.73
20	1022	—	—	5.80	690	258	96.15%	98.62
21	1046	—	—	7.39	742	495	93.77	97.77
22	1062	—	—	8.24	1157	60	99.34	99.77
23	1052	—	—	5.53	436	182	98.26	99.36
24	1134	—	—	19.04	1134	123	98.98	99.63
25	1112	—	—	15.61	719	450	96.69	98.81
26	1142	—	—	19.08	827	163	98.93	99.63
27	1140	—	—	23.83	770	164	99.05	99.66
28	1156	—	—	31.29	833	151	99.23	99.72
29	1174	—	—	41.01	1103	113	99.47	99.82
30	1174	—	—	42.40	1057	231	99.05	99.67
31	1220	—	—	48.91	1278	109	99.59	99.86
32	1220	—	—	54.40	1012	167	99.42	99.80
33	1246	—	—	71.48	1041	503	98.45	99.47
34	1236	—	—	59.31	1236	1761	95.01	98.31
35	1270	1266.7919	0.25	89.07	2000	698	98.19	99.40
36	1280	—	—	84.71	1633	140	99.67	99.89
37	1282	1275.5851	0.50	156.65	2000	4438	90.56	97.03
38	1314	—	—	88.49	1305	185	99.63	99.87
39	1326	1321.2452	0.36	141.12	2000	2015	96.27	98.73
45	2752	2746.9338	0.18	215.76	2000	1338	98.42	99.48
46	2794	—	—	209.90	1226	255	99.72	99.90
49	2810	2801.2864	0.31	514.40	2000	3609	96.76	98.97
50	2834	2828.6609	0.19	450.98	2000	2046	98.27	99.45
53	2960	2936.4180	0.80	902.39	2000	39881	72.00	91.07
54	2896	2859.5322	1.26	1119.57	2000	58639	60.20	88.48
57	3076	3051.8574	0.78	1130.67	2000	50433	71.84	90.85
58	3040	3019.2156	0.68	1352.76	2000	44354	75.93	92.72
60	3092	3082.9722	0.29	1022.41	2000	11031	94.70	98.31

Table 4: Results for traditional Lagrangian relaxation: small and medium sized instances.

$\|P\|$	UB	LB	GAP (%)	Time	Iter	Var Rem	% Var Fix	% Var Fix2
80	3462	3447.0022	0.43	2935.88	2000	63246	87.70	96.34
83	3548	3524.7271	0.66	3737.07	2000	147292	73.93	92.04
84	3584	3544.6189	1.10	4068.00	2000	265365	55.31	86.27
85	3610	3548.0371	1.72	4234.63	2000	400159	35.36	80.46
86	3650	3600.7646	1.35	4623.12	2000	351888	44.79	81.79
87	3678	3637.7058	1.10	4198.42	2000	320253	52.01	85.18
88	3652	3631.9907	0.55	4686.37	2000	151949	77.74	93.17
89	3736	3699.9316	0.97	4551.24	2000	286710	58.97	87.02
90	3678	3645.5200	0.88	4439.60	2000	270570	62.48	88.98

Table 5: Results for traditional Lagrangian relaxation: large sized instances.

$\|P\|$	Time Relax	Opt Relax	Total Time	Integer Opt	Iter	B&B Nodes	GAP (%)
35	0.25	1269	1.72	1270	434	2	0.08
37	5.29	1278	8.20	1278	1283	—	—
39	1.65	1324	2.20	1324	870	—	—
45	0.71	2749	2.86	2752	635	2	0.11
49	2.69	2082	3.81	2082	1065	—	—
50	1.49	2833	7.84	2834	835	—	0.04
53	417.70	2947	922.69	2948	8946	2	0.03
54	243.27	2868.66	802.84	2870	5689	2	0.05
57	454.55	3066	664.94	3066	9454	—	—
58	505.99	3027.33	1837.14	3036	21304	65	0.29%
60	65.76	3091	142.16	3092	4603	2	0.03
80	1373.35	3456	1718.16	3456	19911	—	—

Table 6: CPLEX 6.5 results after traditional Lagrangian Relaxation.

$\|P\|$	UB	LB	GAP (%)	Time	Iter	Var Rem	% Var Fix	% Var Fix2
80	3456	—	—	5209.28	5184	1003	99.80	99.94
83	3532	—	—	5490.18	4816	2891	99.49	99.84
84	3562	3556.7437	0.15	6360.91	8000	9009	98.48	99.52
85	3572	3561.3384	0.30	6601.48	8000	44534	92.81	97.83
86	3628	3612.4097	0.43	8107.73	8000	97238	84.74	95.35
87	3666	3654.5254	0.31	7668.26	8000	61280	90.82	97.16
88	3644	—	—	5612.90	8000	2082	99.69	99.91
89	3722	—	—	7318.70	8000	2755	99.61	99.88
90	3676	3657.8540	0.49	10139.98	8000	129759	82.01	94.72

Table 7: Results for traditional Lagrangian relaxation (under different parameter settings): large instances.

$\|P\|$	Time Relax	Opt Relax	Total Time	Integer Opt	Iter	B&B Nodes	GAP (%)
84	45.02	3558	337.11	3562	5522	19	0.11
85	2299.52	3565.1429	6354.58	3568	40561	4	0.08
86	—	3614.3333	—	3618	—	—	0.10
87	3068.74	3660.2588	23490.31	3666	154106	51	0.16
90	—	3660	—	3664	—	—	0.11

Table 8: Results for CPLEX 6.5 after traditional Lagrangian relaxation (under different parameter settings): large instances.

on Table 7, where columns follow an identical pattern to those in Table 4.

From Table 7 one should notice that optimal solutions could now be proved (exclusively through Lagrangian bounds) for 4 out of the 9 instances tested. Furthermore, in relation with the results on Table 5, for the remaining 5 instances which are left unsolved, gaps between best upper and lower bounds have been closed by at least 44% and at most 86%. Under such substantial gap reductions, a large number of suboptimal variables could now be eliminated. That, in turn, allowed 3 out of the 5 instances remaining unsolved to be processed by CPLEX, under the 384 Mb of RAM memory available (the instances with $|P| = 86$ and $|P| = 90$ remained far too large, even after reductions have been carried out). As it can be appreciated from Table 8, these 3 instances could now be solved to proven optimality by CPLEX (i.e. the second stage of the hybrid algorithm).

In summary, the hybrid algorithm managed to solve larger RGNLP instances than CPLEX 6.5 (as a stand alone procedure). Furthermore, the hybrid algorithm compares favorably with CPLEX 6.5 for all those instances with $|P| \geq 29$.

6.4 Relax and Cut bounds and the hybrid algorithm

Relax and cut based RGNLP lower and upper bounds are shown on Tables 9 and 10. These bounds are obtained from the Lagrangian relaxation of (9), as detailed in Section 4. For such a bounding algorithm, as mentioned before, at every iteration of the SM, a separation problem must be solved to identify inequalities in (8) that are violated by the current Lagrangian relaxation solution. This separation problem is solved by inspection, in such a way that maximal violated clique inequalities are identified.

Columns on tables Tables 9 and 10 are akin to those on Tables 4 and 5. Previous computational experience with Relax and Cut type algorithms (see [15, 16, 18, 17]) indicate that a considerable number of SM iterations are required for the algorithm to attain the good quality dual bounds it is capable of. In spite of that, results on Tables 9 and 10 are obtained for the same SM parameter settings as those associated with the results on Tables 4 and 5.

A direct comparison of the results on Tables 9 and 10 with those on Tables 4 and 5 does not appear very conclusive. CPU times for Relax and Cut did increase, though only marginally. On the other hand, no particular Lagrangian algorithm dominated the other in terms of lower bounds. Upper bounds, as well, varied either way. One conclusion, nevertheless, is clear from this experiment. Bearing in mind the LP relaxation values on Table 3, it is evident that the Relax and Cut lower bounds are not as sharp as they could possibly be. That applies since, in principle, higher lower bounds than those on Table 3 are possible to be attained.

Once again, as before, even after the substantial reductions in problem input size, CPLEX 6.5 could not be used for instances with $|P| \geq 80$. As one would expect, in view of the comments above, results for this particular version of the hybrid algorithm, do not differ much from their previous counterparts on Table 6. Nevertheless, an important point to make is that, by bringing into the LP relaxations those clique inequalities that are left dualized at the very last iteration of the SM appears to pay off. For the 7 instances on Table 6 for which LP relaxations are not naturally integral, only one remains so on Table 11.

Experiments with the Relax and Cut algorithm, under up to 8000 iterations of SM, have also been conducted. Only those test instances with $80 \leq |P| \leq 90$ took part on the experiment. Parameter α, initially set at 2.0, was halved after 300 consecutive iterations of the SM without an overall improvement on the best lower bound so far attained. Corresponding results are shown on Table 12.

For the SM parameter settings introduced above, Lagrangian bounds improved considerably in relation with those quoted on Table 10. Overall, lower bounds also turned out a lot sharper than their counterparts on Table 7. That includes an instance with $|P| = 84$. From the corresponding entry on Table 8, it can be attested that the LP relaxation of (1), for that instance, is at least 4 units away from an optimal integral solution. Clearly the use of clique inequalities (8), within Lagrangian relaxation, must be responsible for closing that gap. Finally, in comparison with Table 10, duality gaps for the remaining instances have been closed by at least 74% and as much as 89%.

For those 4 remaining instances on Table 12 where optimality could not yet be proved,

$\|P\|$	UB	LB	GAP (%)	Time	Iter	Var Rem	% Var Fix	% Var Fix2
15	380	—	—	1.99	555	189	93.27	97.40
16	386	—	—	2.27	442	200	94.04	97.85
17	398	—	—	5.25	1254	93	97.72	99.16
18	408	—	—	4.55	705	110	97.74	99.17
19	414	—	—	6.11	562	216	96.29	98.65
20	1022	—	—	8.02	804	222	96.69	98.81
21	1046	—	—	6.76	614	219	97.24	99.01
22	1062	—	—	11.08	749	84	99.07	99.68
23	1052	—	—	9.11	521	260	97.51	99.09
24	1134	—	—	23.53	1345	89	99.26	99.73
25	1112	—	—	28.07	1043	217	98.40	99.43
26	1142	—	—	28.09	997	267	98.25	99.39
27	1140	—	—	33.78	1084	180	98.96	99.62
28	1156	—	—	25.35	983	255	98.69	99.53
29	1174	—	—	60.19	1388	246	98.85	99.17
30	1174	—	—	30.38	972	218	99.11	99.69
31	1220	—	—	75.03	1277	219	99.18	99.71
32	1220	—	—	65.76	1041	679	97.65	99.21
33	1246	—	—	99.16	1432	657	97.97	99.31
34	1236	—	—	60.43	819	1229	96.51	98.82
35	1270	—	—	134.76	2000	487	98.74	99.58
36	1280	1277.4324	0.20	102.39	2000	690	98.37	99.44
37	1282	1273.9564	0.63	198.21	2000	7653	83.72	94.88
38	1314	—	—	187.34	1491	438	99.13	99.70
39	1326	1321.6467	0.33	215.84	2000	486	99.10	99.69
45	2752	2748.7854	0.12	291.77	2000	912	98.93	99.65
46	2794	—	—	272.01	1426	386	99.58	99.85
49	2814	2799.9705	0.50	517.21	2000	14252	87.22	95.94
50	2834	2827.897	0.22	481.24	2000	4411	96.27	98.81
53	2954	2936.8979	0.58	1003.89	2000	26935	81.09	93.97
54	2872	2861.3911	0.37	913.67	2000	11723	92.04	97.70
57	3096	3053.147	1.38	1397.04	2000	90813	49.30	83.53
58	3050	3016.894	1.09	1424.95	2000	79548	56.83	86.95
60	3092	3082.876	0.30	1224.48	2000	13465	93.53	97.94

Table 9: Results for Realx and Cut algorithm: small and medium sized instances.

$\|P\|$	UB	LB	GAP (%)	Time	Iter	Var Rem	% Var Fix	% Var Fix2
80	3456	3447.5120	0.25	2511.12	2000	23525	95.42	98.64
83	3544	3526.1497	0.50	3267.08	2000	112156	80.15	93.94
84	3578	3542.8313	0.98	4484.13	2000	254435	57.15	86.83
85	3600	3550.5642	1.37	4176.16	2000	347066	43.93	83.05
86	3644	3601.5862	1.16	4478.02	2000	314660	50.63	84.62
87	3678	3638.4788	1.07	4653.46	2000	319859	52.07	85.20
88	3654	3632.6797	0.58	4545.54	2000	178815	73.80	91.97
89	3742	3700.6572	1.10	4311.14	2000	332736	52.39	84.53
90	3678	3646.2471	0.86	5119.90	2000	276103	61.72	88.76

Table 10: Results for Relax and Cut algorithm: large sized instances.

$\|P\|$	Time Relax	Opt Relax	Total Time	Integer Opt	Iter	B&B Nodes	GAP (%)
36	0.89	1280	1.07	1280	492	—	—
37	35.25	1278	43.65	1278	2867	—	—
39	0.39	1324	0.58	1324	312	—	—
45	1.57	2752	1.86	2752	665	—	—
49	43.69	2802	58.75	2802	2938	—	—
50	16.75	2834	18.68	2834	2030	—	—
53	287.55	2948	351.47	2948	7165	—	—
54	106.00	2870	119.16	2870	4985	—	—
57	999.40	3066	1619.05	3066	8458	—	—
58	1221.95	3029.32	5106.07	3036	20859	31	0.22
60	177.10	3092	293.46	3092	6751	—	—
80	355.50	3456	401.29	3456	8471	—	—

Table 11: CPLEX 6.5 results after Relax and Cut.

$\|P\|$	UB	LB	GAP (%)	Time	Iter	Var Rem	% Var Fix	% Var Fix2
80	3456	—	—	4569.36	4617	1106	99.78	99.94
83	3532	—	—	5208.84	5015	767	99.86	99.96
84	3562	—	—	6622.52	8000	623	99.90	99.96
85	3568	3562.7854	0.15	6483.72	8000	11395	98.16	99.44
86	3620	3613.8882	0.17	7942.72	8000	17357	97.28	99.15
87	3666	3656.2256	0.27	8425.67	8000	49784	92.54	97.70
88	3644	—	—	6089.28	5592	646	99.91	99.97
89	3722	—	—	7606.61	8000	2002	99.71	99.91
90	3668	3661.0769	0.19	9178.65	8000	26879	96.27	98.91

Table 12: Results for Relax and Cut algorithm (under different parameter settings): large instances.

$\|P\|$	Time Relax	Opt Relax	Total Time	Integer Opt	Iter	B&B Nodes	GAP (%)
85	339.23	3567.3846	1662.40	3568	8767	—	0.02
86	475.67	3617.2069	1861.57	3618	11087	—	0.02
87	2447.77	3663.6338	7056.78	3666	31367	4	0.06
90	955.69	3664	1010.17	3664	15694	—	—

Table 13: Results for CPLEX 6.5 after Relax and Cut (under different parameter settings): large instances.

CPLEX 6.5 has been applied. Once again, that can be interpreted as being the second phase of a hybrid algorithm where instances are initially preprocessed through geometric and Lagrangian variable reduction tests. It should be stressed that the instances with $|P| = 86$ and $|P| = 90$, contrary to what prevailed before, could now be solved to proven optimality. Furthermore, for most of the instances tested, the overall CPU time spent by this Relax and Cut version of the hybrid algorithm dominated CPU times for the Traditional Lagrangian Relaxation version of the algorithm.

Results from Table 13 indicate that bringing *clique inequalities* over from Relax and Cut into LP relaxation is an attractive proposition. Initial LP relaxation values have improved considerably, in relation with their counterparts on Table 8. That results in very substantial reductions on the CPU times required by CPLEX to prove optimality.

The results above appear to indicate that, for larger RGNLP instances, the Relax and Cut algorithm (when allowed to run for sufficiently long) appears to be a more attractive alternative than traditional Lagrangian relaxation. The same does also seem to apply for those instances, large or small, where a duality gap exists between the LP relaxation of (4) and optimal integral solutions. In order to make this point clearer a number of RGNLP instances with $|P| = 22$ were randomly generated, as before, and solved to optimality by

CPLEX 6.5. Among those instances, 18 have duality gaps ranging from 2 up to 4 units and were selected for further testing. The average duality gap for selected instances is 0.22%.

The two Lagrangian algorithms, running for up to 4000 iterations of the SM have been compared for the 18 instances mentioned above. Parameter α was initially set at 2.0, being halved after 150 consecutive iterations without an overall improvement of the best lower bound so far attained. Results obtained are quoted on Table 14 where the first column indicates the algorithm being used (either Traditional Lagrangian Relaxation or Relax and Cut). The second column gives the average percentage gap between upper and lower bounds obtained for each algorithm. CPU times are quoted on column 3, while average number of SM iterations appear on column 4. Finally, average percentages on the number of variables eliminated by the Lagrangian algorithms with respect to those remaining after geometric reduction tests and to those in the original formulation appear on columns 5 and 6 respectively. The traditional Lagrangian relaxation algorithm managed to prove optimality (due, in this case, to variable fixation tests) for 3 out of the 18 instances tested. On the other hand, the Relax and Cut algorithm succeeded in doing so 17 times. Furthermore, the average number of SM iterations required by the Relax and Cut algorithm has been, on average, significantly less than the maximum number of SM iterations allowed.

Algorithm	GAP (%)	Time	Iter	% Var Fix	% Var Fix2
Traditional	0.207	16.80	3750	98.48	99.46
Relax and Cut	0.016	22.24	1920	98.53	99.48

Table 14: Average results for 18 instances with $|P| = 22$ and nonzero duality gap.

To conclude, it is worth mentioning that further tests have been conducted on a DEC machine using a Alpha processor of 675 MHz and 4 Gb of RAM. In those experiments, 10 instances were tested with $|P|$ chosen between 100 and 145. Three such instances were solved to optimality using the hybrid algorithm combining CPLEX (version 7.0) and the Relax and Cut algorithm. In all cases the gap after the Lagrangian algorithm remains below 2%. For the largest instance solved to optimality P had 125 points, corresponding to a SPP formulation with 15,876 rows and 7,063,529 columns (prior to geometric reductions). The total CPU time to solve this problem was 29.5 hours. Few seconds were spent by CPLEX to close the 0.02% gap left after the Lagrangian phase of the hybrid algorithm which eliminated (together with geometric reductions) 99.96% of the variables from the original SPP formulation.

7 Conclusions and suggestions for future work

In this paper, a hybrid exact solution approach for solving the RGNLP has been proposed and computationally tested. The approach draws heavily on structural properties associated with optimal RGNLP solutions. The investigation of these properties, as described in this paper, has led to a powerful preprocessing test and also to the characterization of a

family of valid inequalities for the optimal RGNLP solution (namely, clique inequalities in (8)). In addition, two different Lagrangian relaxation based lower and upper bounding algorithms have been introduced for RGNLP, cast as a SPP. One of these is a Relax and Cut algorithm which takes advantage of clique inequalities. For a number of randomly generated RGNLP instances, Lagrangian bounds have shown to be very strong. Indeed for most of the RGNLP test instances in this study, Lagrangian bounds alone are enough to prove optimality. Furthermore, when that is not the case, instances would have been substantially further reduced by Lagrangian variable fixation tests. That, in turn, allowed the ILP solver CPLEX to successfully tackle RGNLP instances which otherwise would not have been solved (for the amount of RAM memory available). In the process, RGNLP instances about three times as large as those in the literature could be solved to proven optimality.

In the process of solving RGNLP, very large SPP instances have been solved to proven optimality. The largest of these involved $15,876$ rows and almost $2,044,619$ variables (after geometric reduction tests). SPP instances of such a magnitude rank amongst the largest ever solved to proven optimality in the literature. A relax and cut algorithm for general SPPs thus seem an attractive proposition. Equally attractive should be a hybrid Relax and Cut/Branch and Cut algorithm for these problems.

References

[1] CPLEX ILOG, Inc. CPLEX Division, 1999.

[2] C. N. de Meneses and Cid C. de Souza. Exact solutions of optimal rectangular partitions via integer programming. *International Journal of Computational Geometry and Applications*, 10(5):477–522, 2000.

[3] D. Z Du, L. Q. Pan, and M. T. Shing. Minimum Edge Length Guillotine Rectangular Partition. Technical report, Mathematical Sciences Research Institute, University of California, Jan. 1986.

[4] L. Escudero, M. Guignard, and K. Malik. A lagrangian relax and cut approach for the sequential ordering with precedence constraints. *Annals of Operations Research*, 50:219–237, 1994.

[5] M.L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 1981.

[6] T. F. Gonzalez, M. Razzazi, M. T. Shing, and S. Q. Zheng. On Optimal Guillotine Partitions Approximating Optimal d-Box Partitions. *Computational Geometry*, 4:1–11, 1994.

[7] T. F. Gonzalez, M. Razzazi, and S. Q. Zheng. An Efficient Divide-and-Conquer Approximation Algorithm for Partitioning into d-boxes. *International Journal of Computational Geometry and Applications*, 3(4):281–287, 1993.

[8] T. F. Gonzalez and S. Q. Zheng. Bounds for Partitioning Rectilinear Polygons. *Proc. Symp. Computacional Geometry*, pages 281–287, June 1985.

[9] T. F. Gonzalez and S. Q. Zheng. Improved Bounds for Rectangular and Guillotine Partitions. *J. Symbolic Computation*, 7:591–610, 1989.

[10] T. F. Gonzalez and S. Q. Zheng. Approximation Algorithms for Partitioning a Rectangle with Interior Points. *Algorithmica*, 5:11–42, 1990.

[11] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[12] M. Hunting, U. Faigle, and W. Kern. A Lagrangian relaxation approach to the edge-weighted clique problem. *European Journal of Operational Research*, 131(1):119–131, 2001.

[13] C. Levcopoulos. Fast Heuristics for Minimum Length Rectangular Partitions of Polygons. *Proceedings of the 2nd Computational Geometry Conference*, June 1986.

[14] A. Lingas, R.Y. Pinter, R.L. Rivest, and A. Shamir. Minimum Edge Length Partitioning of Rectlinear Polygons. In *Proceedings of the 20th Annual Allerton Conference Communication, Control, and Computing*. University of Illinois, 1982.

[15] A. Lucena. Steiner problem in graphs: Lagrangean relaxation and cutting-planes. *COAL Bulletin*, 21:2–8, 1992.

[16] A. Lucena. Tight bounds for the Steiner problem in graphs. In *Proceedings of NETFLOW93*, pages 147–154, 1993.

[17] C. Martinhon, A. Lucena, and N. Maculan. A relax and cut algorithm for the vehicle routing problem. relatório técnico, Laboratório de Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, 2000.

[18] M. de Moraes Palmeira, A. Lucena, and O. Porto. A relax and cut algorithm for quadratic knapsack problem. relatório técnico, Laboratório de Métodos Quantitativos, Departamento de Administração, Universidade Federal do Rio de Janeiro, 1999.

[19] G. L. Nemhauser and G. Sigismondi. A Strong cutting plane branch-and-bound algorithm for node packing. *Journal of Operational Research Society*, 43(5):443-457, 1992.