The Convex Hull Relaxation for Nonlinear Integer Programs With Linear Constraints

By

Aykut Ahlatcioglu
and
Monique Guignard[1]

OPIM Department
The Wharton School
University of Pennsylvania

Draft
20 September 2007

Do not quote without the authors' permission

[1] Research partially supported under NSF Grant DMI-0400155.

1. INTRODUCTION

In this paper we introduce a relaxation method for computing both a lower bound on the optimal value of a nonlinear integer minimization program (NLIP), and good integer feasible solutions. For a *linear* integer program (LIP), an optimal integer solution is also optimal over the convex hull of all integer feasible solutions, but this is not usually the case for NLIPs. Rather, the minimization over this convex hull yields a *relaxation* of the NLIP, which we will call the Convex Hull (CH) Relaxation. While we define this relaxation for arbitrary NLIPs, for computational reasons, we restrict our attention to *convex* minimization problems with *linear* constraints, and we show that the lower bound can then be computed using any version of simplicial decomposition, with sub-problems that have the same constraints as the NLIP, but with linear objective functions. If these are easier to solve than their nonlinear counterpart, as would be the case for instance for nonlinear 0-1 knapsack problems, the bound may be tight and relatively inexpensive to compute.

A byproduct of this procedure is the generation of feasible integer points, which provide a tight upper bound to the optimal value of the problem. Indeed since no constraints are relaxed, any integer solution generated while solving a linear integer problem is also an integer feasible solution to the original nonlinear problem. Thus, unlike Lagrangean or primal relaxations, CHR provides integer feasible solutions at no extra cost. From our numerical experiments, these solutions tend to be of excellent quality.

What makes this relaxation very special is that contrary to Lagrangean relaxation (Held and Karp, 1970, 1971) or to primal relaxation (Guignard, 1994, 2003 (p. 183-185)), of which it is an extreme case, this relaxation does not dualize or treat separately any constraints. While for nonlinear integer problems, it cannot be directly compared with Lagrangean relaxation, it always provides a bound at least as good as any primal relaxation. However, if (1) the linear subproblems are difficult to solve, (2) the objective function is nonconvex, and/or (3) there are nonlinear constraints, then one should consider using a primal relaxation instead. In other cases, this new approach appears very attractive.

In the paper, we first define the CH relaxation (CHR) in section 2, analyze the application of simplicial decomposition to the CHR problem in section 3, give details on the algorithm in section 4, discuss the computation of upper and lower bounds on the optimal value in section 5, and describe the application of the approach to convex quadratic knapsack problems in section 6. The remaining sections present the computational results.

2. PRELIMINARIES AND NOTATION

The problem which will be investigated will be the following nonlinear integer program (NLIP)

$$\begin{gathered}(\mathrm{NLIP})\quad \min f(x)\\ \text{s.t.}\\ Ax \le b\\ x \in Y\end{gathered}$$

where

$f(x)$ is a nonlinear convex function of x, a vector pf R^n,

A is an mxn constraint matrix,

b is a resource vector in R^m,

Y is a subset of R^n specifying integrality restrictions on x.

Definition 1

We define the Convex Hull Relaxation of (NLIP) to be

$$\begin{aligned} &(CHR) \quad \min f(x) \\ &s.t \quad x \in Co\{Ax \le b, x \in Y\} \end{aligned}$$

The problem (CHR) is not in general equivalent to (NLIP) when $f(x)$ is nonlinear, because an optimal solution of (CHR) may not be integer, and therefore not feasible for (NLIP). However, it is easy to see that (CHR) is indeed a relaxation to (NLIP), as

(1) $\{x \in Y \mid Ax \le b\} \subseteq Co\{x \in Y \mid Ax \le b\}$

(11) $z_{CHR}(x) = z_{NLIP}(x) = f(x), \ \forall x \in \{Ax \le b, x \in Y\}$

This relaxation is a primal relaxation, in the x-space, and it is an extreme case of the primal relaxation for nonlinear integer problems introduced in Guignard (1994). It is actually a primal relaxation that does not "relax" any constraint.

The difficulty in solving (CHR) comes from the implicit formulation of the convex hull. However the idea of decomposing the problem into a sub-problem and a master-problem, first introduced by Frank & Wolfe (1956), and furthered by Von Hohenbalken with Simplicial Decomposition (1973), and Hearn et al. with Restricted Simplicial Decomposition (1987), provides an efficient way of solving (CHR) to optimality, by solving a sequence of linear integer problems and of nonlinear problems over a simplex. Primal relaxations that also relax constraints require a more complicated scheme, such as that described in Contesse and Guignard (1995, 2007) and Ahn, Contesse and Guignard (1998, 2006), which use an augmented Lagrangean approach, with simplicial decomposition used at each iteration. By contrast, here, due to the absence of relaxed constraints, only one call to simplicial decomposition is needed.

3. APPLYING SIMPLICIAL DECOMPOSITION TO THE CHR PROBLEM

3.1 Assumptions

There are several assumptions that must be imposed on the (NIP) formulation in order for the simplicial decomposition technique to effectively solve (CHR) to optimality. These are:

(i) Compactness and convexity of the feasible region

Compactness and convexity of the feasible region allows writing any element in the feasible space as a convex combination of its extreme points. However, it is not hard to relax the bounded assumption using extreme rays[2].

(ii) Convexity of the objective function

Simplicial decomposition guarantees convergence to a local minimum. However, a local minimum is not necessarily a lower bound for the optimal value of (NLIP). Convergence of the global minimum is ensured if the function is taken to be pseudo-convex.[3] In this paper, we will assume that objective functions are convex, or made convex via convexification.

(iii) Linearity of the constraint set.

Although the objective function can be nonlinear, we cannot at this point allow nonlinear constraints.

3.2 Subproblem

The first part of the decomposition problem is the sub-problem, and can also be viewed as a feasible descent direction finding problem.[4] Assume we are at a feasible point of (CHR), call it x^k. For the k^{th} iteration of simplicial decomposition, we must find a feasible descent direction for $Co\{Ax \leq b, x \in X\}$, a polyhedron, by solving the following problem

$$(CHS)\quad Min_y\ \nabla f(x^k)^T \times (y - x^k)$$

s.t.

$$x \in Co\{Ax \leq b, x \in Y\}$$

We will call this the Convex Hull Sub-problem (CHS). Note that CHS is a linear program. Therefore, unlike in nonlinear programming, (CHS) has an equivalent integer program, which we will call the Integer Programmming Subproblem (IPS)

$$(IPS)\quad Min_y\ \nabla f(x^k)^T \times (y - x^k)$$

[2] See, for instance, Section 6.3 of Hearn et al.(1987)
[3] For further discussion on pseudo-convexity see, for instance, Bazaraa et al.(1979) p. 106 and seq.
[4] For further discussion on feasible directions see, for instance, Bertsekas (2003) p.214 and seq.

$$\text{s.t.}$$
$$Ay \leq b$$
$$y \in Y$$

Solving (IPS), for many types of integer programming problems is considerably easier compared with solving (NLIP). The solution to (IPS) is an extreme point of the convex hull, unless x^k is optimal for the convex hull relaxation (CHR) problem. Therefore, at each iteration we obtain a feasible point to the original (NLIP) problem. Convergence to the optimal solution will be discussed in section IV. If x^k is not optimal, we proceed to the master problem.

3.3 Master Problem

Consider the following nonlinear programming problem with one simple constraint, which we call the Master Problem (MP).

$$(\text{MP}) \ \text{Min.} \, f(X\beta)$$

s. t.

$$\sum_{l=1}^{r} \beta_l = 1$$
$$\beta_l \geq 0, i = 1, \ldots, r$$

X is the $n \times r$ matrix comprised of a subset of extreme points of the convex hull, along with one of the current iterates x^k or a past iterate. There are r such points in X. Note that in the hypothetical case where we know all the extreme points of the convex hull (MP) would have been equivalent to (CHR). Naturally, if the method required such equivalence, there would be no point in using it to solve (CHR). Luckily, any point within a convex hull of a set can be described as a convex combination of at most $n + 1$ points within that set, a result of Caratheodory Theorem[5]. Therefore, the optimal point can be written as a convex combination of a subset of extreme points. Simplicial decomposition takes advantage of this observation, introducing only one extreme point obtained from the subproblem per iteration. Then at the master problem stage, (MP) is solved, which is a minimization problem over a $r - 1$ dimensional simplex. If the optimal solution of (CHR) is within this simplex, then the algorithm terminates. If not, the optimal solution of (MP) will be the next iterate, x^{k+1} which can be found using the following formula:

$$x^{k+1} = \sum_{l=1}^{r} \beta_l^* \times X_l$$

Then we go back to the subproblem, find another extreme point and increase the dimension of the simplex for (MP).It may seem as if a considerable number of extreme points have to be included in X, before finding the optimal solution. Fortunately, this is not the case, as will be shown

[5] For further discussion on Caratheodory Theorem see, for instance, Bazaraa (1979) p.37 and seq.

at section 8. This can be perhaps explained by the way extreme points are introduced to the X matrix. At each sub-problem, the extreme point chosen y^* is the one which yields the steepest descent direction as $\nabla f(x^k)' \times (y^* - x^k)$ is minimal among all such directions. Therefore at each iteration, we are quickly progressing toward the optimal solution, in contrast what would happen if extreme points were to be chosen arbitrarily.

For some pathological cases, putting no restriction on r could potentially pose computational problems. Restricted simplicial decomposition, introduced by Hearn et al. (1987) puts a restriction on the number of extreme points which can be kept. However, even for such pathological cases, there are certain trade-offs between restricted simplicial decomposition and unrestricted simplicial decomposition. Discussing these tradeoffs is beyond the scope of this paper.

3.4 Convergence to the Optimal Solution of CHR

Because the objective function is convex, the necessary and sufficient optimality condition for x^k to be the global minimum is[6],

$$\nabla f(x^k)(y^* - x^k) \geq 0$$

Lemma 2 of Hearn et.al (1987) proves that if x^k is not optimal, then $f(x^{k+1}) < f(x^k)$, so that the sequence $\{x^k\}$ is monotonically decreasing.Finally Lemma 3 of Hearn et al. (1987) shows that any convergent subsequence of $\{x^k\}$ will converge to the global minimum. The result is proved using contradiction that one cannot have a subsequence such that $\nabla f(x^\infty)(y^\infty - x^\infty) \geq 0$ where $x^k \to x^\infty, y^k \to y^\infty$.

4. ALGORITHM

The algorithm used in this study follows the restricted simplicial decomposition (Hearn et al. 1987). The parameter R denotes the maximum number of extreme points allowed to be used in solving the master problem. In the test runs done for this paper, the number of extreme points stored in the matrix X was manageable, so that we put no limit on R, making the algorithm below equivalent to the unrestricted simplicial decomposition method of von Hohenbalken (1977) [7]. The stopping condition for the algorithm is taken from Contesse & Guignard (2007).

Note that in this notation:

(1) $[W_s]^k$ is the collection of extreme points stored at iteration k,
(2) $[W_x]^k$ stores at most one point. It could be empty, a current iterate or a past iterate.

Then,with this notation the master problem introduced in section 3.3 will be:

[6] For proof see, for instance, Bertsekas(2003) p.194

[7] The original name of the method is 'Simplicial Decomposition', but in this paper I will call it 'Unrestricted Simplicial Decomposition' to differentiate it from the 'Restricted Simplicial Decomposition'

$$\text{Min. } f(X\beta)$$
$$\text{s. t.}$$
$$\sum_{i=1}^{r} \beta_l = 1;$$
$$\beta_l \geq 0, l = 1, \dots, r$$
$$X = [W_x]^k \cup [W_s]^k = [W]^k$$

An important point to note is that we discard those points within $[W_s]^{k+1}$ and $[W_x]^{k+1}$ with $\beta_l = 0$ after solving the master problem. This prevents an excessive increase in the number of extreme points stored in $[W_s]^{k+1}$.

Step 0: Take a feasible point x^0. Set $k = 0$, $[W_s]^0 = \emptyset, [W_x]^0 = \{x^0\}$

Step 1: Solve $\min\{\nabla f(x^k) y : y \in S\}$ and let $y^k = \operatorname{argmin}\{\nabla f(x^k) y : y \in S\}$.

(i) If $|[W_s]^k| < R$, *set* $[W_s]^{k+1} = [W_s]^k \cup \{y^k\}$, and $[W_x]^{k+1} = [W_x]^k$

(ii) If, $|[W_s]^k| = R$, take the element of $[W_s]^k$ with the minimal weight out and put y^k in instead to obtain

$[W_s]^{k+1}$, and let $[W_x]^{k+1} = \{x^k\}$ Set $W^{k+1} = [W_s]^{k+1} \cup [W_x]^{k+1}$, and go to step 2

Step 2: Let $x^{k+1} = \operatorname{argmin}\{f(x) : x \in H(W^{k+1})\}$ and write x^{k+1} as $x^{k+1} = \sum_{l=1}^{r} \beta_l W_l$.
Then take all elements with weight $\beta_l = 0$ out of $[W]^{k+1}$. Set $k = k + 1$ and go to step 1.

Step 3: If $|x^k - x^{k-1}| < \varepsilon_x \times \max\{|x^k|, |x^{k-1}|\}$ or $|f(x^k) - f(x^{k-1})| < \varepsilon_f \times |f(x^{k-1})|$, x^k is a solution , then terminate. Otherwise, go to step 1.

5. CALCULATING LOWER AND UPPER BOUNDS

As stated in Definition 1, (CHR) is a relaxation to the (NLIP). Simplicial Decomposition finds an optimal solution, say, x^*, to (CHR), and this provides a lower bound on v(NLIP):

$$LB_{CHR} = f(x^*)$$

On the other hand, at each iteration k of the subproblem an extreme point of the convex hull is found , which is an integer feasible point of (NLIP). Each point y_k^* yields an Upper Bound (UB) to the optimal value of the (NLIP), and the best upper bound on v(NLIP) can be computed as
$UB_{CHR} = \text{Min } \{f(y_1^*), f(y_1^*), \dots, f(y_k^*)\}$

6. APPLICATION OF CHR METHOD TO CONVEX QUADRATIC ANTI-KNAPSACK PROBLEMS

As an example problem we implemented CHR to find a lower bound on the optimal value of quadratic anti-knapsack problems. The Quadratic Anti-Knapsack Problem (QKP) is as follows:

$$\text{(NLIP)} \qquad \text{Min.} \sum_{i}^{n} d_i \cdot x_i + \sum_{i}^{n} \sum_{i'}^{n} c_{i,i'} \cdot x_i \cdot x_{i'}$$

$$\text{s.t.}$$

$$\sum_{i} a_i \cdot x_i \geq b$$

$$x_i \in \{0,1\} \; \forall i$$

where

d_i is the linear cost incurred by selecting item i.

$c_{i,i'}$ is the quadratic cost incurred by selecting items i and i' concurrently.

a_i is the space filled if item i is selected.

b is the total minimum space needed to be filled in the knapsack.

Following the discussion above, the Convex Hull Relaxation (CHR), its subproblem (IPS) and its master problem (MP) will be as follows:

$$\text{(CHR)} \qquad \text{Min.} \sum_{i}^{n} d_i \cdot x_i + \sum_{i}^{n} \sum_{i'}^{n} c_{i,i'} \cdot x_i \cdot x_{i'}$$

$$\text{s.t}$$

$$x \in \text{Co}\left\{\sum_{i}^{n} a_i \cdot x_i \geq b \,, x \in B_{+}^{n}\right\}$$

$$\text{(IPS)} \qquad \text{Min.} \left(\sum_{i}^{n} d_i + \sum_{i}^{n} \sum_{i'}^{n} c_{i,i'} + c_{i',i}\right) \cdot y_i$$

$$\text{s.t.}$$

$$\sum_{i} a_i \cdot y_i \geq b$$

$$y_i \in \{0,1\} \; \forall i$$

$$\text{(MP)} \;\; \text{Min.} \sum_{i}^{n} d_i \cdot \left(\sum_{l=1}^{r} x_i^l \cdot \beta_l\right) + \sum_{i}^{n} \sum_{i'}^{n} c_{i,i'} \cdot \left(\sum_{l=1}^{r} x_i^l \cdot \beta_l\right) \cdot \left(\sum_{l=1}^{r} x_{i'}^l \cdot \beta_l\right)$$

$$\text{s.t.}$$

$$\sum_{i}^{n} a_i \cdot \left(\sum_{l=1}^{r} x_i^l \cdot \beta_l \right) \geq b$$

$$0 \leq \beta_l \leq 1 \ \forall l$$

We can calculate the next iterate, using the following formula:

$$x_{k+1} = \sum_{l=1}^{r} \beta_l^* \cdot x^l$$

7. GENERATION OF THE DATA

As noted previously, the objective function needs to be taken convex, for the CHR method to produce a lower bound for the (NLIP) problem. To come up with convex objective functions for (QKP), we used the following fact which enables us to produce positive definite matrices.

FACT 1

Assume we have the following matrices at hand:

(i) An $n \times n$ diagonal matrix D with positive entries, hence positive definite
(ii) An $n \times m$ matrix A with rank m, where $n \geq m$

Then $C = A^T \cdot D \cdot A$ is positive definite. (Proof at section 10.2 Appendix B)

Then it is not hard to generate positive definite matrices using the random number generator (RNG) installed in GAMS for the nonlinear part of the objective function represented by $\sum_i^n \sum_{ir}^n c_{i,i'} \cdot x_i \cdot x_{i'}$. Similarly, data for the linear part of the objective function and the constraint vector are generated through RNG. The elements of the matrix C is populated around $[55000{,}90000]$, d_i is uniformly distributed on $[30000{,}100000]$ and a_i is uniformly distributed on $[15000{,}75000]$.

The runs have been made for 100, 200 and 400 decision variables, which are items in this problem. For each of these, the right hand side denoted by b is changed to observe how the capacity of the knapsack influences the performance of the CHR method. Five different values of b are tried and represented by case I through case V. Case I represents the knapsack with minimum capacity, and Case V represents the maximum. One should note that $\sum_i a_i$, the total space filled if all items were taken into the knapsack, increase in proportion to the number of variables. Therefore for each case to represent the equivalent effect, values for b are set in proportion to the number of items. These values can be seen in Table 1 below.[8]

Table 1- b values for different cases

No. of variables	Case I	Case II	Case III	Case IV	Case V

[8] For a more detailed account of the results, see Section 10.1 Appendix A.

100	5000	125,000	500,000	1,250,000	2,500,000
200	10000	250,000	1,000,000	2,500,000	5,000,000
400	20000	500,000	2,000,000	5,000,000	10,000,000

The problem is run using the CHR method, as well as built-in MINLP solvers of GAMS, namely CPLEX and AlphaECP. At the end comparisons are made for their performances. Results below are obtained using GAMS 2.25 on Dell OptiPlex745.

8. DISCUSSION OF THE RESULTS

8.1 Improvement Over the Continuous Bound

The improvement over the continuous bound provided by CHR bound is very high, when introduction of one item is sufficient to satisfy the knapsack constraint. (Case I). This is expected, and therefore not taken into consideration when calculating the percentage improvement values below. Table 1 below shows percentage improvements for all the 15 instances.

Table 1- % Improvement over the continuous bound

No. of Variable		Case I	Case II	Case III	Case IV	Case V
	100	3310	13.6	12.1	24.5	53.2
	200	1392	19.4	0.25	2.00	0.14
	400	537	1.24	0.19	0.30	0.06

One could talk of decrease of percentage improvement as number of variables increase. The only exception to this is at case II. On the other hand, it is hard to extract a similar strong relation between the knapsack capacity and percentage improvement. Excluding case I, average percentage improvement for the remaining 12 runs is:

$$\%\ \text{Improvement} = 10.57\ \%$$

8.2 Gap Between the CHR Lower Bound and the Optimal Value

The CHR bound provides a remarkably tight bound for the optimal value. Table 2 shows that calculating the CHR bound could indeed be useful in providing a tighter lower bound compared to the one obtained through continuous relaxation (Compare with Table 1).

Table 2- % Gap between the CHR lower bound and the optimal value

No. of Variable		Case I	Case II	Case III	Case IV	Case V
	100	9.41	4.94	0.70	0.12	~ 0

	200	14.9	2.64	0.05	0.03	~0
	400	13.7	1.29	0.09	~0	~0

Excluding case I, average gap between CHR bound and the optimal value is

$$\% \text{ Gap} = 0.82\%$$

We see that as we require more variables to get into the knapsack (note the decrease from case I to case V), the gap closes up.

8.3 Gap Between the CHR Upper Bound and the Optimal Value

The CHR upperbound also provides a very tight bound for the optimal value. In fact, for most of the instances we tried, it returned the optimal solution. However to verify this remarkable strength of the upper bound, there is need for testing the algorithm for other type of problems, especially those known to be difficult to solve to optimality.

Table 3- % Gap between the CHR Upper Bound and the optimal value

		Case I	**Case II**	**Case III**	**Case IV**	**Case V**
No. of Variable	**100**	0.0	0.0	0.0	0.0	0.0
	200	0.0	0.12	0.0	0.0	0.0
	400	0.0	0.0*	0.0	0.0	0.0

8.4 Run Time and Reliability of the CHR Method

As important as obtaining tight bounds, is the cost of obtaining that bound, as well as the reliability of the bound or the optimal value obtained. The obvious measure of cost is the CPU time it takes for the algorithm to converge and return upper and lower bounds, and how this compares with the existing MINLP solvers in GAMS. Reliability can be measured whether the algorithm is able to turn in a bound or an optimal solution for all instances with precision in a reasonable amount of time. Table 3 below shows CPU time in seconds for CPLEX, AlphaEcp and CHR. Comparing the average run times, one could see that AlphaEcp is not comparable to CPLEX or CHR. Although CPLEX performs better than CHR on average, this is mostly because of one particular instance where CHR performs poorly (400 variable, case III). Excluding the worst performances, CHR indeed performs better than both CPLEX and AlphaEcp. In fact 7 out of the 15 runs, CHR terminates with the shortest run time.

One should also note that CPLEX is only able to solve problems with quadratic objective functions, whereas CHR does not take advantage of the quadracity of the objective function, and will work just as well for a non-quadratic objective function. Table 3 also sheds light on the reliability of these algorithms. For two instances, AlphaEcp fails to terminate in less than 30 minutes. On the other hand, CPLEX terminated with a feasible but not optimal integer solution for four instances due to worsening of the objective function of the nonlinear objective function.

Table 3- Run times in seconds

# of Variables	Case	CPLEX	AlphaEcp	CHR
100	I	1.01	10	1.07
100	II	1.10	9	0.62
100	III	1.26*	251	2.44
100	IV	0.76	32	2.84
100	V	0.83	2	0.57
200	I	2.40	53	2.95
200	II	4.10*	1611	4.28
200	III	2.98	2	1.4
200	IV	2.59	25	6.81
200	V	2.40	3	1.78
400	I	15.16	376	5.65
400	II	19.54*	N/A**	27.79
400	III	15.77*	N/A**	90.68
400	IV	10.25	4	1.00
400	V	11.04	7	8.12
Average (all)		6.08	183.46	10.53
Average (excluding the worst)		5.12	64.5	4.8

*CPLEX Warning: The search was stopped because the objective function of the NLP subproblem started to deteriorate
**Unable to terminate in less than 30 minutes

Especially, in two instances CHR found an upper bound with a lower objective function value, compared to the value returned by CPLEX as the optimal value, while AlphaECP could not solve the problem. Table 4 focuses on these two instances.

# of Variables	Case	CHR	CPLEX	AlphaEcp
400	II	3,358,685	3,359,690	N/A
400	III	49,337,624	49,344,302	N/A

As seen, CHR performs better than CPLEX and AlphaEcp in terms of returning a feasible solution with the lowest objective function. However, one should note that this feasible solution is still a upper bound, but can not be proven optimal.

9. CONCLUSIONS AND FUTURE DIRECTION OF RESEARCH

The Convex Hull Relaxation (CHR) provides tight lower and upper bounds by (1) transforming a nonlinear integer optimization problem in one over the convex hull of all feasible solutions, and (2) replacing this problem by a sequence of linear programs and simple nonlinear programs. The potential strength of the proposed algorithm is that the difficulty of the problems solved at each iteration stays relatively unchanged from iteration to iteration. It will be most suitable for those nonlinear integer problem types that would be much easier to solve with a linear objective function. One should expect that CHR will have a robust performance for large-scale problems if one has access to solvers able to handle large linear programs and simple nonlinear programs efficiently. Further testing is needed for larger problem sizes and other problem types.
The most important restriction of the method seems to be the convexity requirement on the objective function, and linear constraints. But there is no further structural requirement on the objective function, in contrast to available MINLP solvers such as CPLEX, which require a quadratic objective function.

10. APPENDIX

10.1 Appendix A: Computational Results in Detail

Note thatfor the data sets used we have
$\sum_i a_i = 4315000$ for 100 variable data set
$\sum_i a_i = 8930700$ for 200 variable data set
$\sum_i a_i = 17424000$ for 400 variable data set
Therefore, capacity requirements corresponding to each case has been chosen accordingly.

A. # of variables =100

Table 1

	Case I (b=5000)	Case II (b=125,000)	Case III (b=500,000)	Case IV (b=1250,000)	Case V (b=2500,000)
Cont. Bound	2649.7	263,678	3,479,767	19,156,699	74,612,738
Convex Hull Lower Bound	90377.4	299,457	3,899,468	23,857,310	114,289,200
Convex Hull Upper Bound	99766.8	315,028	3,927,058	23,885,029	114,289,580
% Improvement	3310.9	13.57	12.06	24.5	53.2
Optimal value (AlphaECP)	99766.8	315,028	3,927,058	23,885,029	114,289,580
Optimal value (CPLEX)	99766.8	315,028	3,937,286**	23,885,029	114,289,580
% lower bound gap	9.41	4.94	0.70	0.12	3.3249e-004
% upper bound gap	0.0	0.0	0.0	0.0	0.0

** CPLEX Warning: The search was stopped because the objective function of the NLP subproblem started to deteriorate.

Table 2- Run times in seconds (b=5,000)

	NLP	MIP	ECP	TOTAL
CPLEX	0.88	0.13	-	1.01
AlphaECP	-	9.00	1.00	10.00
CHR	0.78	0.29	-	1.07

Table 3- Run times in seconds (b=125,000)

	NLP	MIP	ECP	TOTAL
CPLEX	0.97	0.13	-	1.10
AlphaECP	-	8.00	1.00	9.00
CHR	0.43	0.19	-	0.62

Table 4- Run times in seconds (b=500,000)

	NIP	MIP	ECP	TOTAL
CPLEX	1.02	0.24	-	1.26
AlphaECP	-	246.98	4.02	251.00
CHR	1.66	0.780	-	2.440

Table 5- Run times in seconds (b=1,250,000)

	NIP	MIP	ECP	TOTAL
CPLEX	0.65	0.11	-	0.76
AlphaECP	-	28.00	4.00	32.00
CHR	2.10	0.74	-	2.84

Table 6- Run times in seconds (b=2,500,000)

	NIP	MIP	ECP	TOTAL
CPLEX	0.66	0.17	-	0.83
AlphaECP	-	2.00	0.00	2.00
CHR	0.44	0.13	-	0.57

Table 8- Analysis of Simplicial algorithm (# of variables =100)

b	**No. of Simplicial Iterations**	**Max. no of extreme points at a given iteration**
5000	12	12
125000	7	7
500000	15	15
1250000	11	11
2500000	3	3

B. # of variables =200

Table 9

	Case I (b=10000)	**Case II (b=250000)**	**Case III (b=1000000)**	**Case IV (b=2500000)**	**Case V (b=5000000)**
Cont. Bound	6181.1	915,114	12,782,780	83,464,883	394253860
Convex Hull Lower bound	92261.1	1,091,412	12,814,670	85,137,220	394,794,400
Convex Hull Upper Bound	105,980.6	1,123,458.7	12,820,782	85,158,703	394,794,740
% Improvement	1392.6	19.37	0.25	2.00	0.14
Optimal value (AlphaECP)	105,980.6	1,122,090	12,820,782	85,158,703	394,794,740
Optimal value (CPLEX)	105,980.6	1,137,485**	12,820,782	85,158,703	394,794,740
% lower bound gap	14.87	2.64	0.05	0.03	8.6121e-005
% upper bound gap	0.0	0.12	0.0	0.0	0.0

**CPLEX Warning: The search was stopped because the objective function of the NLP subproblem started to deteriorate.

Table 10- Run times in seconds (b=10,000)

	NLP	**MIP**	**ECP**	**TOTAL**
CPLEX*	2.31	0.09	-	2.40
AlphaECP	-	46.00	7.00	53.00
CHR	2.56	0.39	-	2.95

Table 11- Run times in seconds (b=250,000)

	NLP	MIP	ECP	TOTAL
CPLEX	3.73	0.32	-	4.05
AlphaECP		1600	11	1611
CHR	3.42	0.860	-	4.28

Table 12- Run times in seconds (b=1,000,000)

	NIP	MIP	ECP	TOTAL
CPLEX	2.79	0.19	-	2.98
AlphaECP		1.00	1.00	2.00
CHR	0.89	0.51	-	1.40

Table 13- Run times in seconds (b=2,500,000)

	NIP	MIP	ECP	TOTAL
CPLEX	2.25	0.34	-	2.59
AlphaECP	-	24.00	1.00	25.00
CHR	5.88	0.93	-	6.81

Table 14- Run times in seconds (b=5,000,000)

	NIP	MIP	ECP	TOTAL
CPLEX	2.19	0.21	-	2.40
AlphaECP	-	3.00	0.0	3.00
CHR	1.57	0.21	-	1.78

Table 15- Analysis of Simplicial algorithm (# of variables =200)

b	No. of Simplicial Iterations	Max. no of extreme points at a given iteration

10000	15	15
250000	16	16
1000000	5	5
2500000	11	11
5000000	3	3

C. # of variables =400

Table 16

	Case I (b=20000)	**Case II (b=500000)**	**Case III (b=2000000)**	**Case IV (b=5000000)**	**Case V (b=10000000)**
Cont. Bound	14383	3,275,799	49,203,510	328,628,230	1,661,081,100
Convex Hull Lower Bound	91684	3,316,422	49,298,830	329,626,900	1,662,065,000
Convex Hull Upper Bound	106,260	3,358,685	49,337,624	329,626,924	1,662,065,750
% Improvement	537.4%	1.24%	0.19%	0.3%	0.06%
Optimal value (AlphaECP)	106,260	N/A*	N/A*	329,626,924	1,662,065,750
Optimal value (CPLEX)	106,260	3,359,690**	49,344,302**	329,626,924	1,662,065,750

% lower bound gap	13.7	1.29	0.09	6.0675e-006	4.5125e-005
% upper bound gap	0.0	0.0	0.0	0.0	0.0

*Fail to terminate in less than 30 minutes.

** CPLEX Warning: The search was stopped because the objective function of the NLP subproblem started to deteriorate.

Table 17- Run times in seconds (b=20,000)

	NLP	**MIP**	**ECP**	**TOTAL**
CPLEX*	15.00	0.16	-	15.16
AlphaECP	-	332.01	43.99	376
CHR	5.18	0.47	-	5.65

Table 18- Run times in seconds (b=500,000)

	NLP	**MIP**	**ECP**	**TOTAL**
CPLEX	18.03	1.51	-	19.54
AlphaECP	FAIL TO TERMINATE			
CHR	17.59	10.20	-	27.79

Table 19- Run times in seconds (b=2,000,000)

	NIP	**MIP**	**ECP**	**TOTAL**
CPLEX	15.16	0.61	-	15.77
AlphaECP	FAIL TO TERMINATE			
CHR	14.54	126.34	-	140.88

Table 20- Run times in seconds (b=5,000,000)

	NIP	**MIP**	**ECP**	**TOTAL**
CPLEX	9.95	0.30	-	10.25
AlphaECP	-	1.00	3.00	4.00

CHR	0.81	0.19	-	1.00

Table 21- Run times in seconds (b=10,000,000)

	NIP	**MIP**	**ECP**	**TOTAL**
CPLEX	10.55	0.49	-	11.04
AlphaECP	-	4.00	3.0	7.00
CHR	7.59	0.53	-	8.12

Table 22- Analysis of Simplicial algorithm (# of variables =400)

b	**No. of Simplicial Iterations**	**Max. no of extreme points at a given iteration**
20000	16	16
500000	30	30
2000000	19	10
5000000	2	2
10000000	4	4

10.2 Appendix B: Proof of Fact I

FACT I :

Assume we have the following matrices at hand:

- A $n \times n$ diagonal matrix D with positive entries, hence positive definite
- A $n \times m$ matrix A with rank m, where $n \geq m$

Then $C = A^T \cdot D \cdot A$ is positive definite.

PROOF :

Because D is positive definite,

$$(AX)^T \cdot D \cdot (AX) \geq 0 \rightarrow X^T \cdot (A^T DA) \cdot X \geq 0$$

Hence, the matrix C is positive semi-definite. To show that it is also positive definite, it suffices to show that,
$\forall x \in R^m,\ AX \neq 0$. This immediately follows from the assumption that A is a matrix of rank m.

10.3 Appendix C: GAMS CODE OF THE CHR ALGORITHM

```
$title Gams Code for QKP problem of 200 variables
$eolcom!
option limrow=0,limcol=0,solprint=off,sysout=off;
$offsymlist offsymxref offlisting
option NLP=minos;
option optcr=0;
sets i jobs / i1*i200/
   k this is for set of ext  points generated for  rest simp decomp/k1*k1000/
   iter set for simplicial iterations      /it1*it5000/
   ind(k)'indicator yes when the extreme point remains'
alias(i,ir);
scalars ro
     flag
     flagS
     r
     iteration
     ex
     ef
     fnew
     fold
     mini
     maxix
     normold
     normnew
     foptimal
     counter
```

```
       cccmaster
       cccsub
       ccc;
variables y(i)
         beta(k)
         z1
         z2;
positive variables beta(k);
binary variables y(i);

parameters xA(i,iter)
          u(i)
          x(i)/i1*i150 1/
          xold(i)
          x0(i)
          t(k)
          w(i,k)
          ayk(i)
          opt;

equations objgrad
         obj
         beqn
         capacity
         MC;
*****Important Parameters to set*****

r=200; !'Number of allowable extreme points for restricted simplicial decomposition. 100 seems to
be fairly safe, and makes it act like unrestricted SD.'
 !'These are needed for stopping condition. Decreasing them may naturally cause more iterations,
but improve accuracy.'
ef= 0.000000000000000000001;
ex=0.0000000000000000000001;
*==================================================================================
*                         DATA INPUT                              =
*==================================================================================
scalar b /5000000/;
sets j /j1*j500/
alias (j,jr);
parameters m(i,j)
          d(jr,j)
          q(i,j)
          c(i,ir)
          install(i)
          a(i);
```

```
d(jr,j)=0;
m(i,jr)=uniform(0,10);
d(jr,jr)=uniform(0,10);
q(i,j)=sum(jr, m(i,jr)*d(jr,j));
c(i,ir)=sum(jr,q(i,jr)*m(ir,jr));
install(i)=uniform(30000,100000);
a(i)=uniform(15000,75000);
;
*==============================================================================
*                        EQUATIONS                         =
*==============================================================================
obj.. sum(i,install(i)* sum(k$ind(k),beta(k)*w(i,k)))
+sum((i,ir), c(i,ir)*sum(k$ind(k),beta(k)*w(i,k))*sum(k$ind(k),beta(k)*w(ir,k)))=e=z2;
objgrad.. sum(i,[install(i)+sum(ir,[c(i,ir)+c(ir,i)]*x(ir))]*(y(i)-x(i)))=e=z1;
capacity.. sum(i,a(i)*y(i))=g=b;
beqn.. sum(k$ind(k),beta(k))=e=1;
*==============================================================================
*                          MODELS                        =
*==============================================================================
model sub/objgrad,capacity/;
model master/obj,beqn/;
*==============================================================================
*                  Rest of the Initialization                =
*==============================================================================
*These initializations are not meant to change under normal conditions.
cccmaster=0;
cccsub=0;
flag=1;
w(i,k)=0;
t(k)=0;
ind(k)=no;
ind('k1')=yes;
fnew=100;
fold=0;
maxix=0;
flagS=1;
iteration=1;
normnew=100;
counter=0;
w(i,'k1')=x(i);
*==============================================================================
*                       ALGORITHM                          =
*==============================================================================
file count/count.dat/;
file upperbounds/upperbound.dat/;
```

```
    while(((flagS=1) or ((normnew>ex*maxix) and (abs(fnew-fold)>ef*abs(fold)))),
        xA(i,iter)$(iteration=ord(iter))=x(i);
        flagS=0;
         solve sub using mip minimizing z1;
         cccsub=cccsub+sub.resusd;
                   if((counter<r),
            loop(k$((flag=1) and (ord(k)>=2)),
            if((t(k)=0),
                   w(i,k)=y.l(i);
                     opt=sum(i,install(i)*y.l(i))+sum((i,ir), c(i,ir)*y.l(i)*y.l(ir));
                     put upperbounds;
                     put opt///;
                   ind(k)=yes;
                   counter=counter+1;
                                  flag=0;
                   );    ! end if t
               );   ! end loop k
           );  ! end if counter
          flag=1;
         if((counter=r),
            mini=smin(k$(ord(k)>=2 and ord(k)<r+1),t(k));
            loop(k$(ord(k)>=2 and ord(k)<=r+1),
                if((mini=t(k)),
                  w(i,k)=y.l(i);
                  );     ! end if mini
                );   ! end loop k
             w(i,'k1')=x(i);
         );   ! end while flagS=1
    put count;
    put counter///;
    solve master using nlp minimizing z2;
    cccmaster=cccmaster+master.resusd;
    t(k)=beta.l(k);
    loop(k$(ord(k)>=2 and ind(k)),
        if((t(k)=0),

           ind(k)=no;
          );    ! end if k
        ); ! end loop k

   xold(i)=x(i);
   x(i)=sum(k$ind(k),t(k)*w(i,k));
fold=sum(i,install(i)*xold(i))+sum((i,ir), c(i,ir)*xold(i)*xold(ir));
fnew=sum(i,install(i)*x(i))+sum((i,ir), c(i,ir)*x(i)*x(ir));
normold=sqrt(sum(i,sqr(xold(i))));
normnew=sqrt(sum(i,sqr(x(i))));
maxix=max(normnew,normold);
iteration=iteration+1;
 );  ! end while counterA=1
foptimal=sum(i,install(i)*xold(i))+sum((i,ir), c(i,ir)*xold(i)*x(ir));
ccc=cccmaster+cccsub;
display foptimal,cccmaster,cccsub,ccc;
```

11. REFERENCES

Ahn, S., L. Contesse and M. Guignard, "An Augmented Lagrangean Relaxation for Nonlinear Integer Programming Solved by the Method ofMultipliers, Part II: Application to Nonlinear Facility Location," Working Paper, latest revision 2007.

Bazaraa, M.S. and C.M Shetty., "Nonlinear Programming Theory and Algorithms", John Wiley & Sons, Inc., 1979.

Bertsekas D., "Nonlinear Programming", Athena Scientific, 2d printing, 2003.

Contesse L. and M. Guignard, "An Augmented Lagrangean Relaxation for Nonlinear Integer Programming Solved by the Method of Multipliers, Part I: Theory and Algorithm," Working Paper, OPIM Department, Univ. of Pennsylvania, latest revision 2007.

Guignard, M., 'Primal Relaxation in Integer Programming,' VII CLAIO Meeting, Santiago, Chile, 1994, also Operations and Information Management Working Paper 94-02-01, University of Pennsylvania, 1994.

Guignard, M., "A New, Solvable, Primal Relaxation For Nonlinear Integer Programming Problems with Linear Constraints," Operations and Information Management Working Paper, University of Pennsylvania, 2007.

Hearn, D.W., Lawphongpanich S. and Ventura J.A, "Restricted Simplicial Decomposition: Computation and Extensions", Mathematical Programming Study 31, 99-118, 1987.