Working Paper

Constraint Aggregation Principle: Application to a Dual Transportation Problem

Rafał Różycki

WP-95-103
September 1995

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Constraint Aggregation Principle: Application to a Dual Transportation Problem

Rafał Różycki

WP-95-103
September 1995

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Abstract

Constraint aggregation technique is a new method for solving convex optimization problems. This paper focuses on the examination of the efficiency of the aggregation technique. Some properties of the basic version of the algorithm are presented for convex optimization problems with linear constraints. Various parameters and advanced versions of this algorithm are examined on the example of the dual transportation problem. The results obtained allow to formulate some interesting conclusions. Special attention is directed to the advantages achieved by implementation of partial aggregation idea.

Key words: Nonsmooth optimization, constraint aggregation, transportation problem

Constraint Aggregation Principle: Application to a Dual Transportation Problem

Rafał Różycki

The paper deals with constraint aggregation technique which is a general method for solving convex optimization problems of the form

$$\min f(x) \tag{1}$$

$$h_j(x) \leq 0, \quad j = 1, \ldots, m, \tag{2}$$

$$x \in X. \tag{3}$$

It is assumed that the functions $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $h_j : \mathbb{R}^n \mapsto \mathbb{R}$, $j = 1, \ldots, m$, are convex and $X \subset \mathbb{R}^n$ is convex and compact. Moreover it is assumed that the feasible set defined by (2)-(3) is non-empty what guarantees that the problem has an optimal solution.

The problems in which both the number of constraints m and/or number of variables n are very large seem to be a particulary fruitful area of application of aggreagtion technique. To such a class of problems one can include various semi-infinite programming problems (e.g. path planning problems in robotics), stochastic programming problems with constraints that have to hold with probability one, etc.

The main idea of aggregation technique is based on the assumption that the structure of X is simple and the main difficulty comes from the large number of constraints (2). In this case, commonly used methods may fail, because of insufficient size of computer memory. To overcome these difficulties, the original problem is replaced by a sequence of problems, in which the complicating constraints (2) are represented by one (or in general more than one) *surrogate inequality*

$$\sum_{j=1}^{m} s_j^k h_j(x) \leq 0, \tag{4}$$

where $s^k \geq 0$ are iteratively modified *aggregation coefficients.* By an aggregation of original constraints a substantial simplification of (1)-(3) is achieved, because (4) inherits linearity or differentiability properties of (2). In [1] theoretical bases, the way how to update the aggregation coefficients and how to use the solution of the simplified problems to arrive at the solution of (1)-(3) is presented. Although the described method is general enough to solve nonsmooth convex optimization problems this paper will concentrate on the linear ones.

In section 1 the basic algorithm for the simplified version of the problem having only linear constraints is presented. In section 2 an extension of full aggregation idea to partial aggregation is showed. Section 3 is devoted to examine efficiency of aggregation technique on the example of known linear optimization problem. Sensibility to the parameters values and modifications of basic algorithm is showed.

1 The basic algorithm

Let us consider the simplified version of the problem:

$$\min f(x) \tag{5}$$

$$Ax = b, \tag{6}$$

$$x \in X. \tag{7}$$

where f is convex and X is bounded, convex and closed. Below the basic algorithm utilizing the aggregation idea for solving (5)-(7) is presented:

Algorithm A

1. : Find $x^0 \in X$ with $f(x^0) \leq f_{min}$, set $k = 0$.

2. : Find u^k which solves

$$\min f(u) \tag{8}$$

$$\langle Ax^k - b, Au - b\rangle \leq 0, \tag{9}$$

$$u \in X. \tag{10}$$

3. : Find

$$\tau_k = 1/(k+1), k = 0, 1, 2, \ldots \tag{11}$$

and define

$$x^{k+1} = x^k + \tau_k(u^k - x^k) \tag{12}$$

Increase k by one and if not stop criterion go to Step 1.

the above algorithm can be viewed as an iterative constraint aggregation method. The initial equality constraints (6) are replaced by a sequence of non-stationary scalar inequalities (9). Obviously, (9) is a relaxation of (6), so u^k exists and $f(u^k) \leq f^*$. As an alternative stepsize τ_k selection one can assume:

$$\tau_k \in [0, 1], \tau_k \to 0, \sum_{k=0}^{\infty} \tau_k = \infty \tag{13}$$

It can be proved that for Algorithm A and τ_k which fulfilled (13), every accumulation point x^k is a solution of problem (5-7). The stepsizes τ_k in the basic procedure can also be generated in a more systematic way:

$$\tau_k = \arg \min_{0 \leq \tau \leq 1} |(1-\tau)(Ax^k - b) + \tau(Au^k - b)|^2. \tag{14}$$

To show how the aggregation technique extends to linear inequality constraints, let us replace equality constraints (6) by a set of inequalities:

$$Ax \leq b, \tag{15}$$

The basic algorithm can be adapted to our modified problem by transforming (9) into

$$\langle r^+(x^k), Au - b\rangle \leq 0, \tag{16}$$

where $r^+(x) = \max(0, Ax - b))$. $r^+(x)$ is simply a vector of constraint residuals for current solution x. Obviously, if linear equality constraints $Ax = b$ are written as inequalities $Ax - b \leq 0$ and $-Ax + b \leq 0$, then the surrogate inequalities (9) and (16) are identical.

2 Partial aggregation

It is not necessary to aggregate exactly to one aggregate constraint. It is possible to aggregate the constraints in groups as follows. Let

$$A_l x = b_l, \quad l = 1, \ldots, L, \tag{17}$$

be subgroups of constraints (6), such that each row of (6) is represented at least once. The overlapping of such subgroups is possible. One full aggregate constraint(9) may be replaced by L aggregates:

$$\langle A_l x^k - b_l, A_l u - b_l\rangle \leq 0, \quad l = 1, \ldots, L. \tag{18}$$

Then the optimizing stepsize (14) should be replaced by

$$\tau_k = \arg \min_{0 \leq \tau \leq 1} \sum_{l=1}^{L} |(1-\tau)(A_l x^k - b_l) + \tau(A_l u^k - b_l)|^2. \tag{19}$$

For such a formulation of aggregated problem, the convergence properties remain the same. In real optimization problems with many constraints, one can distinguish very often some blocks of constraints. Such blocks contain constraints correlated by common properties. Just in these cases it is justified to aggregate them into the subgroups. The subgroup aggregates created in this way often have interesting practical interpretation and may be useful in some applications.

3 An example of application

From the theoretical point of view, the aggregation technique is first of all applicable to the convex problems with very many constraints. As an illustrative example the particular instance of the dual transportation problem, known as Lemarechal's problem TR48 [5], was assumed. Although this is a relatively small ("only" 2304 constraints) linear problem, it is very interesting due to the strong solving difficulties. In original version (without

introducing additional variables) it is a nonsmooth optimization problem with a piecewise linear objective [4, 3].

Problem formulation

Let us formulate the dual of the transpotration problem:

$$\max \left[\sum_{i=1}^{N} s_i w_i - \sum_{j=1}^{N} d_j v_j \right] \tag{20}$$

$$w_i - v_j \leq a_{ij}, \quad i = 1, \ldots, N, \; j = 1, \ldots, N, \tag{21}$$

where w_i, $i = 1, \ldots, N$ are unknown potentials of sources, v_j, $j = 1, \ldots, N$ are unknown potentials of destination nodes, a_{ij} denotes the transportation cost from source i to destination j, s_i and d_j are the amounts available at source i and required at destination j, respectively. In TR48 problem $N = 48$ and $n = 2N$. The particular values of transportation costs and amounts s_i, $i = 1, ..., N$ and d_j, $j = 1, ..., N$ one can find in Appendix.

Optimum value

638565 is the optimum value of the TR48 problem objective function. The primal and dual simplex method from the CPLEX callable library [6] solve the problem in 138 and 157 iterations, respectively.

The computational experiment

The main purpose of the computational experiment was to recognize the behavior of the basic algortihm various modifications and comparision of its both practical and theoretical convergences to the optimum value. Moreover, the aggregation technique has been tested to study its sensitivity to the values of the parameters. A number of succesive experiments were carried out to achieve better and better convergence. The algorithm has been implemented in C++ and the experiment has been carried out on Sun Workstation. The CPLEX ver.2.1 callable library was a tool to solve subproblems. Starting point was the common one for all tests and results from Step 1 of Algorithm A. The given number of iterations was the stop criterion and was fixed at 200. The below description of succeding computational experiment stages results from the succesion of realized tests.

TEST A

The first test deal with the basic method with full aggregation as described in section 1. It means that at each step one aggregate constraint

$$\sum_{i=1}^{N} \sum_{j=1}^{N} h_{ij}^{+}(w^k, v^k)(w_i - v_j) \leq \sum_{i=1}^{N} \sum_{j=1}^{N} h_{ij}^{+}(w^k, v^k) a_{ij},$$

with aggregation coefficients defined as residuals:

$$h_{ij}^{+}(w^k, v^k) = \max(0, w_i^k - v_j^k - a_{ij}).$$

is formed. There are not any bounds for variables in original version of the TR48 problem. However it appears from (7), that aggregation technique requires bounds for all variables. With no loss of generality we may assume that all variables of TR48 problem are not less than zero. Upper bounds were set to the same value M for all variables to simplify calculations. Full formulation of our TR48 problem version must be completed by the bounds:

$$0 \leq w_i \leq M, \quad i = 1, \ldots, N,$$
$$0 \leq v_j \leq M, \quad j = 1, \ldots, N, \tag{22}$$

Such a bounding box must fulfil the assumption that it contains the optimum solution of the problem. This assumption is fulfilled for $M \geq 1656$. In order to examine the influence of a size of bounding box on basic version (the single aggregate, harmonic stepsize rule (11)) of iterative algorithm efficiency three values of M were employed : $M = 3000, 10000, 100000$. The results are presented in Figure 1 and Figure 2. Notice that (20)-(22) is a maximization problem, and therefore the objective value in Figure 1 decreases as a function of the iteration number. One can see that convergence of the basic algorithm is strongly related to the size of bounding box. The starting point of Algorithm A is calculated as a point optimized objective function (5) in a case of constraints (6) absence. However, condition (7) has to be still satisfied. So, it is rather obvious that starting point for linear objective function has to lie on the simplex defined by bounding box. Thus the large size of bounding box, together with relatively fast decreasing of the stepsize, causes weak convergence of the algorithm. The upper bound M was set at 10000 to the further tests.

TEST B

The partial aggregation idea was an objective of this test. It has been tested for various numbers of aggregates and compared with full aggregation idea. In a single test, each aggregate contained the same number of constraints of original problem (20)-(22). Single aggregate constraint was formed by successive constraints (21) independently of existence practical interpretation of such an operation. It was considered that number of aggregates L has fulfilled the equation:

$$(N * N)\mathbf{mod}L = 0$$

If $L \leq N$ then aggregate constraints ($l = 1, \ldots, L$) had the form:

$$\sum_{i=O_1}^{O_2} \sum_{j=1}^{N} h_{ij}^{+}(w^k, v^k)(w_i - v_j) \leq \sum_{i=O_1}^{O_2} \sum_{j=1}^{N} h_{ij}^{+}(w^k, v^k) a_{ij} \quad \text{for } l = 1, \ldots, L \tag{23}$$

where

$$O_1 = (l-1)(N\mathbf{div}L) + 1$$
$$O_2 = l * (N\mathbf{div}L)$$

In case $L > N$ aggregate constraint l was created as follow:

$$\sum_{j=O_1}^{O_2} h_{Pj}^{+}(w^k, v^k)((w_P - v_j) \leq \sum_{j=O_1}^{O_2} h_{Pj}^{+}(w^k, v^k)(a_{Pj} \quad \text{for } l = 1, \ldots, L \tag{24}$$

where

$$O_1 = ((l-1)\mathbf{mod}(L\mathbf{div}N)) * (N * N/L) + 1$$

$$O_2 = ((l-1)\mathbf{mod}(L\mathbf{div}N)) * (N * N/L) + N * N/L$$

$$P = ((l-1)\mathbf{mod}(L\mathbf{div}N)) + 1$$

Comparison of convergence of basic algorithm with upper bound $M = 10000$ for number of aggregates $L = 1, 96, 192$ is presented in the Figure 3. Experiments show that increasing of the number of aggregates distinctly improves the algorithm convergence. Nevertheless, it cannot be forgotten that increasing of aggregates number causes enlargement of the subproblem (8),(18),(10) and as a consequence of it, growth of a subproblem solving time. The solving time needed by the single iteration of Algorithm A for various number of aggregates is presented in the Figure 4.

The number of aggregates $L = 96$ was assumed in the further tests.

TEST C

Interesting results were obtained by testing various ways of partial aggregation with number of aggregates fixed at 96. In previous test the way of constraints aggregation of problem (20)-(22) was dictated only by a comfort of programist. Much better effect gives aggregation supported by an economical interpretation. Let us create the aggregation constraints in more reasonable way:

$$w_i - \sum_{j=1}^{N} \gamma_{ij} v_j \leq \sum_{j=1}^{N} \gamma_{ij} a_{ij} \quad for\ i = 1, \ldots, N \tag{25}$$

where the aggregation coefficients γ_{ij}, $j = 1, \ldots, N$ are normalized residuals:

$$\gamma_{ij} = \frac{h_{ij}^{+}(w^k, v^k)}{\sum_{j=1}^{N} h_{ij}^{+}(w^k, v^k)}.$$

and

$$\sum_{i=1}^{N} \sigma_{ij} w_i - v_j \leq \sum_{i=1}^{N} \sigma_{ij} a_{ij}, \quad for\ j = 1, \ldots, N \tag{26}$$

with

$$\sigma_{ij} = \frac{h_{ij}^{+}(w^k, v^k)}{\sum_{i=1}^{N} h_{ij}^{+}(w^k, v^k)}.$$

Note that such a form of aggregates has the interesting economical interpretation.

Economic interpretation of aggregates

Considering inequalities (25) and (26), variables w_i and v_j have to be counted with the same measure as c_{ij}. c_{ij} is the cost of product unit transport between purveyor i and receiver j. Thus variables w_i and v_j may be interpreted as a sale price of product unit at source node i and a purchase price of product unit at destination node j respectively. In a case of wiseblock aggregation for every source node i one aggregate constraint has

been constructed. The aggregation coefficients $\gamma_{ij}, j = 1, \ldots, N$ are normalized residuals. Such a single aggregation constraint expresses the fact that difference between an average purchase cost at destination nodes seen from i and a sale price at source node i ought not to exceed an average transportation cost from node i (to protect against unjustifiable purchase price growth !). The similar situation applies to destination nodes j. There is again one aggregate for one destination node. This time, however, conditions expressed above are formulated from the destinations nodes points of view. Notice, that in a case of aggregates defined by (25)-(26), the subgroups overlapping appears (every single constraint from original problem occurs in aggregates twice). Figure 5. shows a benefit of a wiseblock aggregation applying. The wiseblock aggregation was applied in the next tests.

TEST D

Although consecutive modifications of the solving method for TR48 problem based on constraints aggregation, gave the improvement of the convergence, the efficiency of the algorithm has been still poor. It must have been caused by a harmonic stepsize calculation (11). This simplest method used in tests until now led to premature saturation state. On the contrary, an attempt of stepsize calculation from (14) constitutes an optimization problem by itself. It is because of nonsmooth character of function (14). Because of it, the following rules for stepsize calculation were employed:

1. Heuristic 1 rule - it uses $\tau_k = 1$, if it decreases the Euclidean norm of the constraint residuals; otherwise $\tau_k = 1/(k+1)$;

2. Heuristic 2 rule - it uses, as above, $\tau_k = 1$, if it decreases the Euclidean norm of the constraint residuals; otherwise if τ_{k-1} decreases this norm then $\tau_k = \tau_{k-1}$ (stepsize is kept); otherwise (if τ_{k-1} does not decrease the norm) $\tau_k = \beta * (\tau_{k-1})$ (where $\beta = 0.95$);

3. Near optimal stepsizes - a kind of estimation of (14) was used, namely a modificated "golden division" rule.

First two methods were implemented in order to slow down the rate of stepsize decreasing. Heuristic 1 rule due to its simplicity is very easy to solve, but the rate of stepsize decreasing remains too fast. Much more flexible is Heuristic 2. The stepsize exchange may be controled by the β coefficient setting. The difficulty is that the setting of such a value needs many experiments and it is not easy to set one, best value for the various solved problems. To make matters worse, this value is strongly interrelated with other parameters of the algorithm and changing one of the parameters may lead to the deterioration of the convergence. In our tests β was set at 0.95 - the value that ensure the slower stepsize decreasing. The purpose of the third method is not to control the stepsize decreasing but to find the stepsize as near as possible to the optimum (14). Unfortunatelly it is difficult to find such an exact value. Thus method using "golden division" rule was adapted to find near optimal stepsize. The modification comprises a way of search interval defining. Of

course primitive (in the first iteration) search interval is [0,1]. In the subsequent iterations of the algorithm this interval depends on the stepsize of the previous iteration. In most of cases the search intervals decrease but it is still possible to take the size [0,1].

Let assume the following order of points of "golden division":

$$0.0 < x1 < x2 < x3$$

For current iteration k of the Algorithm A the procedure of the near optimal stepsize finding seems as follow:

1. : $x1 = \tau_k - 1$; $\alpha = 0.618034$;
2. : $x2 = (\alpha * x1)/(1 - \alpha)$;
3. : if $x2 > 1$ then $x3 = 1.0$; $x2 = \alpha$; go to Step 6
4. : If $\|res(x2)\| > \|res(x1)\|$ then $x3 = x2$; $x2 = x1$; go to Step 6
5. : $x2 = x1$ go to Step 2
6. : Apply traditional "golden division" method for the stepsize search interval $[0.0, x3]$ and the current value of $x2$;
7. : $\tau_k = (x2 + x1)/2$;

The number of iterations of traditional "golden division" method was set at 10, the number sufficient for calculation the near optimal value with the absolute accuracy 4

All above methods were compared with the simplest, harmonic method (11). Results are shown in Figure 6. Surprisingly, the Heuristic 1 stepsize rule efficiency was comparable with the Near optimal one, while Heuristic 2 rule for $\beta = 0.95$ was worse even than Harmonic one (possibly, this time, the stepsize decreasing was to slow).

TEST E

Addition of all active constraints (aggregates) from previous iteration to the current solved subproblem (8)-(10) was the last modification introduced into a method of TR48 problem solving, based on aggregation principle. Obviously, it causes increasing of a total number of constraints in subproblem and growth of a solving time. The number of all constraints in subproblem may fetch the number of $2*L$. Advantages achieved by such a modification fully make up for above weakness. Convergence of the aggregation method with maintaining of active constraints is much better than the previous versions of the algorithm. Comparison of various stepsize calculation methods for this modification is presented in Figure 7 and Figure 8. The advantages of the last modification are most visible when Figure 7 is compared with Figure 6.

4 Conclusions

Although it is not an objective of this paper to compare different methods, some advantages of aggregation technique can be observed. As a result of logical aggregation in related subgroups, that is to say by exploiting the specific characteristics of the model, one can improve computational efficiency. Partial aggregation, which does not take into consideration relations among constraints does not give such good effects. It follows from the tests that harmonical stepsize rule is computationaly inpractical. As it could be expected, the optimizing stepsize rule performed significiantly better than other ones. Surprisingly good results gives, however, adding in the auxiliary subproblem all active aggregates from the previous iteration. Moreover, additional experiments show, that these results are not sensitive with respect to the size of the bounding box. Obviously experiment has been performed for only one specific problem and in other cases results may differ sligthly. Boldly one can say, however, that applying of aggregation technique to huge network problems may bring surprisingly good results.

5 Appendix

The values of transportation costs c_ij and amounts s_i, $i = 1, ..., N$ and d_j, $j = 1, ..., N$ for dual Transpotration Problem TR48:

$s_i =$
22,53,64,15,66,37,16,23,67,18,52,69,
17,29,50,13,95,34,59,36,22,94,28,34,
36,38,55,77,45,34,32,58,30,88,74,59,
93,54,89,30,79,46,35,41,99,52,76,93

$d_j =$
61,67,24,84,13,86,89,46,48,50,74,75,
88,40,29,45,32,21,61,21,51,14,89,79,
38,20,97,19,10,73,59,92,52,66,89,65,
63,47, 7,61,87,19,36,43, 9,12, 8,67;

Matrix C used to define c_{ij}:

273,1272, 744,1138,1972,1580,1878,1539,1457, 429,1129,1251,1421, 588, 334, 837,
1364, 229, 961, 754,1169,1488, 720,1280, 816, 664,1178, 939,1698, 983,1119,1029,
1815, 721,1753, 330,1499,1107,1576, 942, 484, 617, 896,1184,1030,1718, 604, 999,
809, 866,1722,1338,1640,1266,1185, 440, 894, 992,1173, 334, 358, 626,1124, 358,
847, 533, 915,1219, 481,1009, 543, 937, 915, 667,1441, 812, 848, 776,1560, 526,
1494, 598,1244,1304,1306, 685, 668, 444,1157,1359,1176,1475, 335,1519, 140, 937,
697, 951, 267, 227,1229, 587, 369, 554, 721,1212, 739, 596,1291,1114, 701, 426,
285, 676, 155, 456,1936, 319, 337, 604, 907, 214, 424, 748, 817, 666,1592, 521,
2172, 356, 467,1583, 882,2139,2182,1961, 781, 678,1425,1861,1473,1713,1761,1617,
370,1073,1304,1369,1092, 453, 798,1283, 973, 565,1315,1204,1796, 846,1447,1143,
959,1275,1213,2085, 742,1309,1479,1760, 703,1727, 872,1479, 686,1698,1057, 387,
1252, 904, 668, 443,1600, 930,1052, 776,1049, 402, 361,1119, 578, 406, 618, 581,
1095, 670, 641,1152,1060, 567, 433, 374, 579, 235, 325,1802, 331, 217, 665, 862,
182, 312, 864, 732, 783,1456, 608,2066, 491, 400,1466, 744,2013,2082,1865, 875,
552, 400, 182, 820, 721,1735, 851, 740, 551,1551,1769,1159, 613,2072,1300,1605,
807,1017,1251, 818,1259,2596, 826,1137,1255,1123, 943,1359, 188,1282, 271,2300,
483,2540, 609,1038,2099,1766,2699,2493,2266, 264,1398, 304, 699, 538,1335, 454,
393, 173,1198,1370, 760, 216,1692, 919,1286, 435, 879, 861, 548, 913,2198, 483,
803,1181, 731, 627,1086, 292, 883, 279,1906, 178,2156, 490, 662,1699,1430,2300,
2117,1888, 138,1023, 884, 755,1612, 749, 690, 476,1501,1654,1049, 516,1995,1149,
1580, 739,1079,1161, 815,1214,2485, 780,1100,1347, 985, 916,1361, 260,1171, 328,
2202, 445,2385, 665, 966,1969,1729,2568,2333,2108, 177,1327, 177,1486, 757, 506,
609, 981,1474, 967, 681,1552,1317, 926, 594, 197, 928, 316, 723,2203, 500, 604,

482,1104, 455, 630, 641,1058, 562,1857, 528,2425, 220, 704,1845,1122,2405,2428,
2204, 738, 945,1362, 587, 335, 435, 930,1358, 819, 504,1496,1153, 927, 428, 341,
803, 180, 649,2119, 343, 521, 652, 939, 340, 649, 533, 918, 451,1783, 362,2290,
130, 568,1727,1105,2301,2285,2059, 595, 853, 891,1082,1199, 726, 96, 583,1125,
653, 563, 947, 986,1493, 560,1183, 813, 882,1033, 902,1763, 642,1032,1131,1604,
463,1556, 663,1298, 947,1461, 795, 371, 882, 967, 973, 768,1472, 588, 252, 308,
803, 920, 309, 238,1252, 569, 940, 165, 863, 414, 454, 552,1745, 269, 482,1188,
355, 397, 833, 713, 432, 666,1453, 410,1758, 642, 262,1260,1051,1858,1737,1508,
529, 598, 222, 814,1094, 510, 235,1335, 820, 892, 100, 626, 541, 219, 524,1897,
90, 410, 952, 605, 238, 706, 570, 622, 503,1581, 257,1985, 396, 309,1453,1039,
2043,1972,1744, 514, 661,1025,1227, 617, 90,1525, 835,1114, 263, 770, 700, 400,
740,2049, 311, 630,1087, 630, 459, 924, 405, 739, 360,1749, 115,2055, 428, 492,
1568,1256,2166,2026,1796, 303, 853, 663, 632, 999, 572, 972, 225, 763, 908, 451,
767, 293,1240, 726, 420,1111, 862, 617, 443,1374, 586,1299, 887,1070,1633,1057,
547, 999, 252,1483,1681,1489,1326, 236, 610,1156, 557, 642, 879,1000,1467, 558,
1178, 780, 831,1038, 879,1726, 700,1023,1082,1631, 488,1579, 586,1320, 982,1463,
796, 371, 802, 949,1021, 826,1508, 550, 546, 983, 397, 821, 411,1023, 180, 651,
478,1438, 476, 485,1333, 235, 525, 827,1022, 123, 973,1155, 715,1475, 902, 273,
953, 882,1550,1467,1240, 898, 396,1479, 745,1105, 240, 831, 645, 442, 723,1983,
316, 623,1152, 543, 470, 939, 482, 669, 443,1690, 205,1969, 510, 455,1492,1238,
2091,1938,1709, 354, 813,1163, 676,1264,1473, 839,1326, 847, 801,1254, 976,1643,
1157,1169, 983,1905, 878,1836, 346,1590,1286,1621,1034, 689, 503, 995,1376,1239,
1828, 674,1183, 725,1399, 549,1004, 869,1427, 818, 882,1716, 214, 902,1222,1210,
390,1184,1225, 949,1239,1210, 660, 863,1207,1446,1197, 969,1042, 741, 865, 821,
644, 790, 388,1374, 803, 484, 968,1056, 665, 318,1420, 794,1341,1017,1137,1836,
1056, 679,1200, 189,1645,1891,1704,1403, 442, 699, 453, 290, 483,1809, 107, 384,
1024, 511, 251, 712, 646, 525, 585,1499, 330,1885, 495, 231,1356, 999,1949,1872,
1644, 567, 591, 950, 410, 690,2147, 594, 590, 326,1191, 499, 504, 838,1098, 758,
1794, 703,2439, 414, 751,1837,1011,2374,2455,2237, 928, 921, 624, 325,1356, 480,
369,1241, 413, 473, 680,1097, 166,1038,1049, 781,1497, 905, 238, 925, 702,1506,
1506,1287, 998, 216, 479,1941, 188, 350, 736, 792, 161, 547, 632, 745, 552,1607,
375,2115, 296, 392,1547, 959,2121,2114,1890, 641, 676,1480, 435, 129, 949, 708,
325, 355,1081, 492,1007,1137, 779,1759, 774, 291,1148, 516,1688,1785,1573,1038,
231,1829,1603,2339,1524,1780,1673,2421,1315,2394, 357,2136, 825,2237,1589, 579,
1204, 347, 959, 940,2336,1266, 320, 919, 605, 154, 623, 652, 580, 582,1508, 344,
1950, 429, 242,1402, 949,1986,1943,1717, 603, 582, 872, 699, 197, 358, 957, 529,
881,1263, 660,1849, 645, 240,1250, 631,1802,1867,1650, 923, 341,1511, 815, 669,
1092,1397,1019,1982,1010,2708, 695,1061,2089,1148,2594,2734,2520,1212,1176, 697,
1051,1018, 290, 985,1280, 743,1427, 996, 466, 987,1110,1584,1395,1166, 861, 626,
469, 761, 607, 685,1446, 472,1969, 457, 254,1393, 823,1963,1975,1752, 739, 515,
1171, 847,1089,1316, 919,2063, 776, 598,1434, 507,1926,2101,1898,1187, 548,1144,
83,2145, 317,2445, 426, 875,1972,1584,2571,2408,2179, 194,1231,1094,1036, 836,
1371,1008, 354, 833, 828,1429,1369,1146,1021, 352,2083, 259,2412, 345, 811,1925,
1507,2523,2380,2151, 220,1183,1828,1005,1903,1272, 504, 649, 653,1114,1019,2044,

932,2165, 330, 559,1668,1291,2264,2138,1908, 268, 917,2377,1723, 636,1720, 534,
145, 290,2281,1531, 667,1829,1235,2410,2367,2139, 519, 972,1162, 792,1744,1724,
1500, 796, 361,1087, 600, 701, 550,1835, 917,1490,1787,1614,1553, 486, 678, 727,
2435,1461, 229,2238,1560,2010,1353,1157;

The way of defining c_{ij}:

```
nr=0;
for(i=1;i¡N;i++)
for(j=i;j¡N;j++){
c[i-1][j]=dane[nr];
nr++;}
for(i=0;i¡N;i++){
c[i][i]=100000;
for(j=0;j¡i;j++)
c[i][j]=c[j][i];}
```

References

[1] Yu.M. Ermoliev, A. V. Kryazhimskii and A. Ruszczyński *"Constraint Aggregation Principle in Convex Optimization"*, IIASA Working Paper WP-95-015, Laxenburg, February 1995.

[2] J. Buga, I. Nykowski *"Transportation problems in Linear Programming"*, PWN, Warszawa, 1974 (in Polish).

[3] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, Springer-Verlag, Berlin, 1993.

[4] K.C. Kiwiel, "Proximity control in bundle methods for convex nondifferentiable minimization", *Mathematical Programming* 46(1990) 105-122.

[5] C. Lemaréchal and R. Mifflin, eds., *Nonsmooth Optimization*, Pergamon Press, Oxford, 1978.

[6] *Using the CPLEX™ Callable Library and CPLEX™ Mixed Integer Library*, CPLEX Optimization, Incline Village 1993.

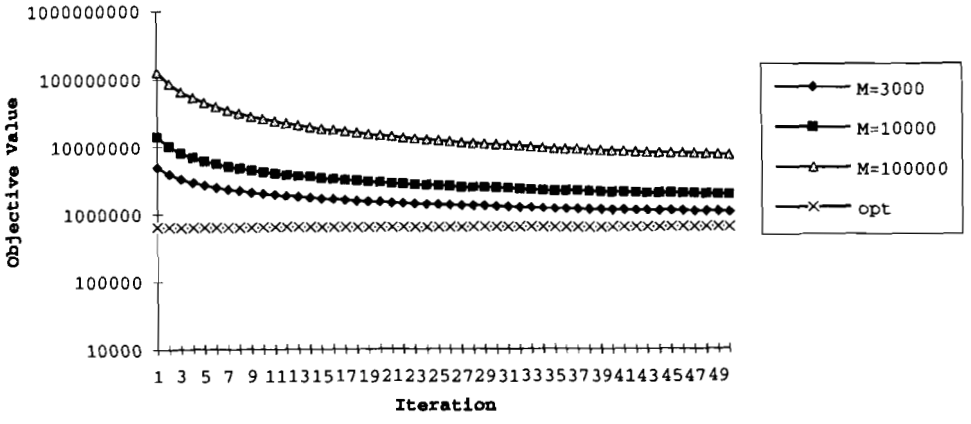

Figure 1: Objective value $f(w^k, v^k)$ as a function of the iteration number k for various upper bounds.

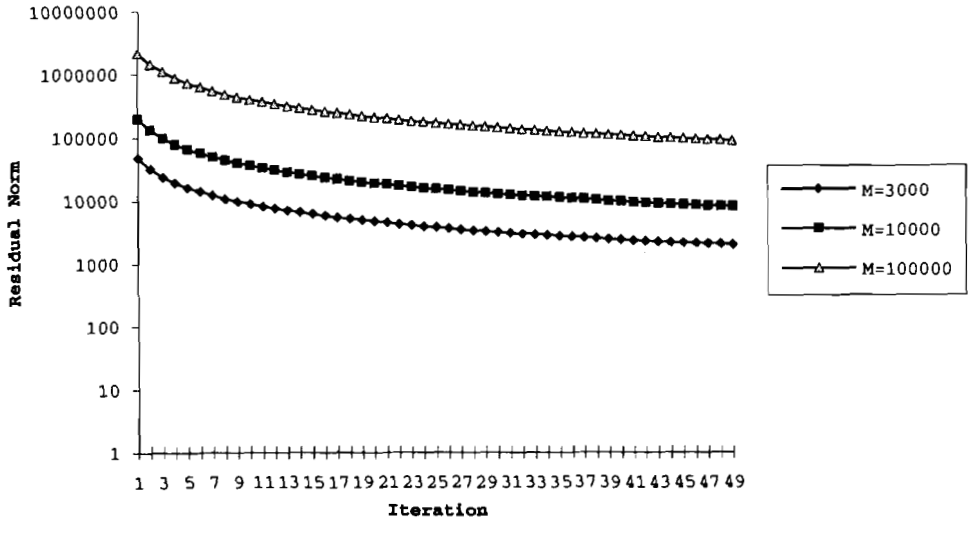

Figure 2: Norm of residual $\left(\sum_{i=1}^{N}\sum_{j=1}^{N}(h_{ij}^{+}(w^k, v^k))^2\right)^{1/2}$ as a function of the iteration number k for various upper bounds.

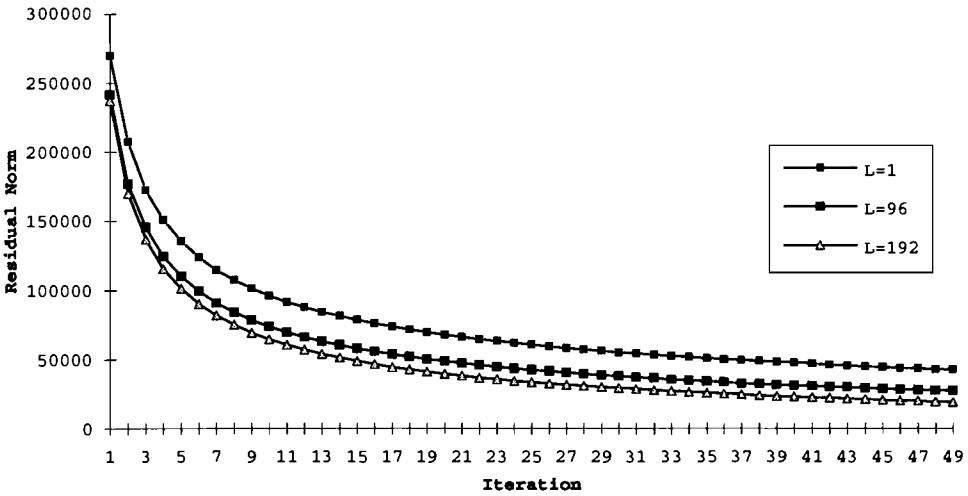

Figure 3: Norm of residual as a function of the iteration number k for various number of aggregates

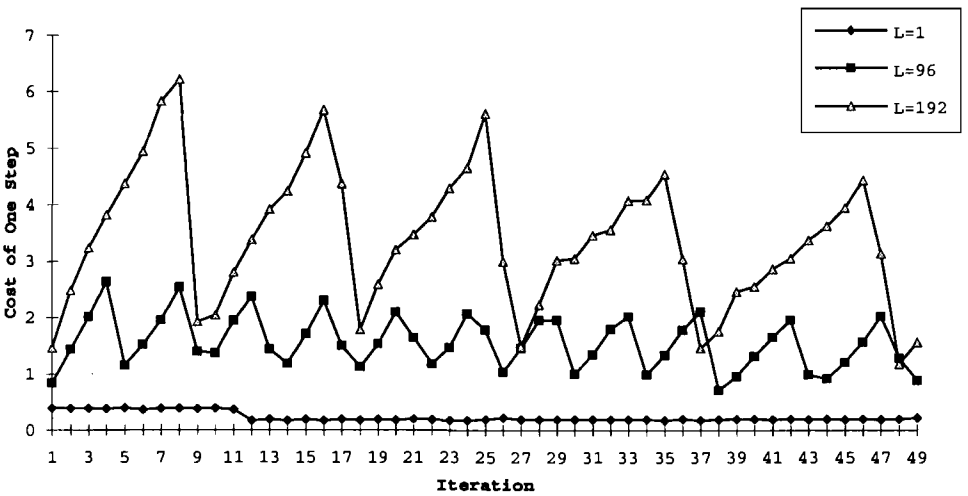

Figure 4: CPU time needed by one iteration of Algorithm A as a function of the iteration number k for various number of aggregates

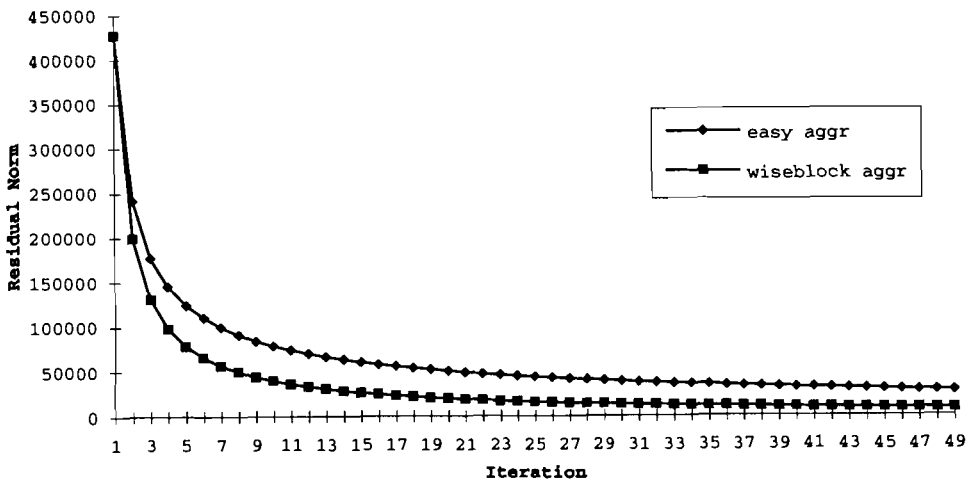

Figure 5: Norm of residual as a function of the iteration number k for various ways of aggregation

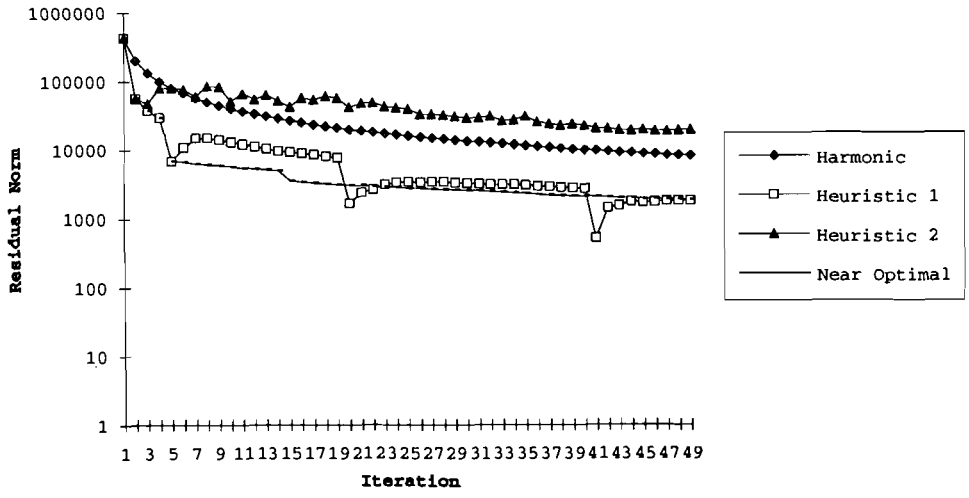

Figure 6: Norm of residual as a function of the iteration number k for various stepsize rules

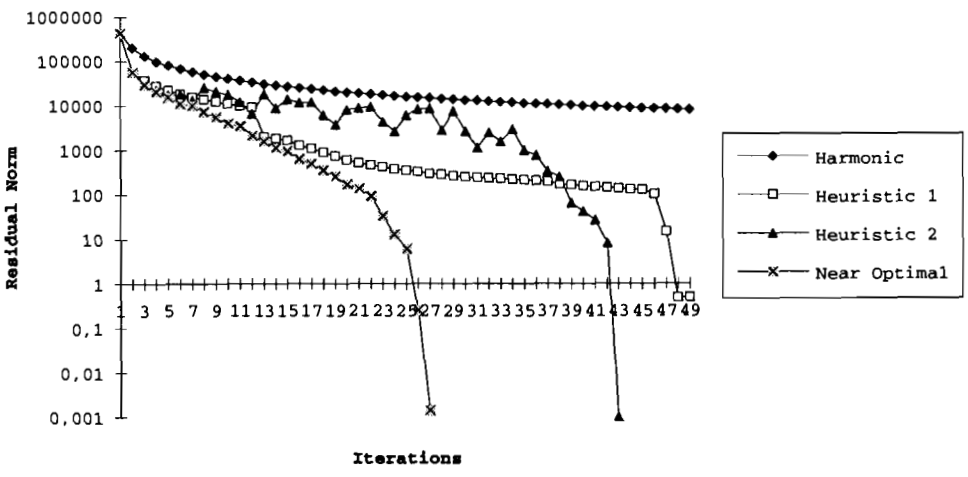

Figure 7: Norm of residual as a function of the iteration number k for kept in subproblems all previous active aggregates and various stepsize rules

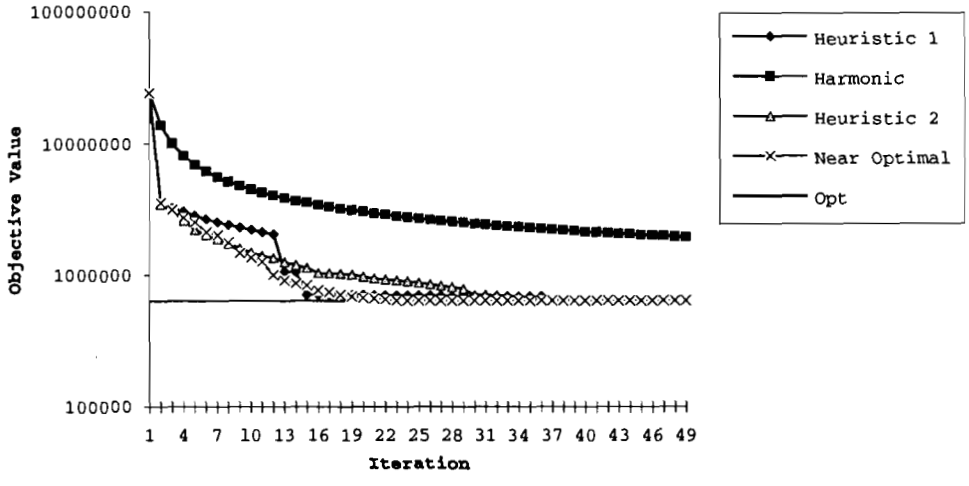

Figure 8: Objective value $f(w^k, v^k)$ as a function of the iteration number k for kept in subproblems all previous active aggregates and various stepsize rules