HEURISTICS FOR URBAN SOLID WASTE COLLECTION

K. Florios, C. Kiranoudis, G. Mavrotas
School of Chemical Engineering, National Technical University of Athens
Zografou Campus, Athens 15780, Greece

KEYWORDS
Solid waste management, Facilities organization and operational management, Management bodies (public and private sector), technological developments

ABSTRACT
The objective of this paper is to develop and test two heuristics for the collection of urban solid waste. The heuristics are a local search and a capacitated version of k-means, called ck-means. Comparison is made to MINLP modelling and well known heuristics as k-means. The main idea is to decompose urban solid waste collection in sequential location, clustering and scheduling phases. The main emphasis is given in the clustering phase. The overall approach is based on the green reverse logistics framework formulating a complex discreet optimization problem. The raw spatial data are the GIS coordinates of the city’s blocks and corners. Computer tests include the municipality of Athens and results for the clustering phase are in favour of the local search method.

ΕΥΡΕΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΓΙΑ ΤΗ ΣΥΛΛΟΓΗ ΑΣΤΙΚΩΝ ΣΤΕΡΕΩΝ ΑΠΟΒΛΗΤΩΝ

Κ. Φλώριος, Χ. Κυρανούδης, Γ. Μαυρωτάς
Σχολή Χημικών Μηχανικών, Εθνικό Μετσόβιο Πολυτεχνείο
Πολυτεχνειούπολη Ζωγράφου, Αθήνα 15780, Ελλάδα

ΠΕΡΙΛΗΨΗ
Ο στόχος της εργασίας είναι η ανάπτυξη και δοκιμή δύο ευρετικών αλγορίθμων για τη συλλογή αστικών στερεών αποβλήτων. Οι ευρετικοί είναι μια τοπική έρευνα και μια παράλλαγη του αλγορίθμου k-means, που ονομάστηκε ck-means. Γίνεται σύγκριση με το μοντέλο MINLP και με τον k-means. Η κεντρική ιδέα είναι να αναλυθεί η συλλογή των αστικών στερεών αποβλήτων σε σειριακές φάσεις χωροθέτησης, ομαδοποίησης και χρονικού προγραμματισμού. Έμφαση αποδίδεται στη φάση της ομαδοποίησης. Η συνολική προσέγγιση εντάσσεται στην πράσινη αντίστροφη εροδιαστική διαχείριση. Τα αρχικά χωρικά δεδομένα είναι οι συντεταγμένες σε ένα ΓΣΠ των τετραγώνων και γωνιών της πόλης. Τα υπολογιστικά τεστ περιλαμβάνουν το Δ. Αθηναίων και τα αποτελέσματα για τη φάση ομαδοποίησης είναι υπέρ της τοπικής έρευνας.

Heleco '05, ΤΕΕ, Αθήνα, 3-6 Φεβρουαρίου 2005 1
1. INTRODUCTION

Urban solid waste collection is a fundamental task for all municipalities, since it consumes a significant share of their annual budget mainly due to the cost of purchasing and maintaining the garbage trucks fleet and the high working costs of workers/drivers occupied in the cleaning administration of the municipality. Furthermore, the need for rational transportation systems in all cities is well known today, especially in metropolises, both from traffic jam and air pollution perspectives. Thus, for various economic, traffic and environmental reasons, the urban solid waste collection problem is an interesting field of research and optimization.

In this paper, urban solid waste collection is described by the following decomposition in three subsequent sub-problems: first a location problem, in which we roughly locate the trash bins throughout the city corners, second a clustering problem, in which we gather spatially adjacent bins in cleaning sectors as we state such collection areas, and third a scheduling phase in which the tasks of gathering the garbage from each collection area are scheduled to truck crews working a day’s work.

Indeed, urban or municipal solid waste collection is a demanding fleet management problem. Vehicle routing techniques might be extremely difficult to work in such a scale. Even if they did the cost of buying such a specialized software could be extremely high for smaller municipalities. The features of urban solid waste collection as a green reverse logistics problem are the following. First, it is a reverse distribution or collection problem which is usually vast in real world scale involving, for instance, as many as 10,000 garbage bins and 130 trucks for the municipality of Athens. Because of the dense manner in which the garbage bins are distributed in the city, there are no remote customers or bins to be serviced. Thus, we are happy to know just the area which is assigned to a garbage truck and not the full route bin to bin. Second, the road network is sometimes undigitized mainly for province cities of developed countries and even metropolises of under-developed countries. Another clue is that the bin locations in the city are typically invariant with time for a long time unlike customers in corporate distribution systems. Due to the previously described characteristics, the aforementioned decomposition of the collection in three subsequent phases, emphasizing in the clustering phase, seems to provide a good working tool for various collection cases through the world.

The urban solid waste collection problem (USWCP) is thoughtfully stripped down to a capacitated set partitioning problem (CSPP) for the clustering phase. The latter problem is a specific case of the well known set partitioning problem in which integer capacities are added on the set points. The set partitioning problem is known to be NP-hard and so must be it’s capacitated cousin. Many discreet optimization problems like the ones we are discussing in this paper are notorious for their exponential complexity by mathematical programming techniques like the brouhch and bound method. This fame is experimentally checked and confirmed for the CSPP. The MINLP model of this problem is formulated and optimized in GAMS software.

In the rest of the paper, the text consists of the following paragraphs: in paragraph 2 the MINLP formulation of the capacitated set partitioning problem is presented. Paragraph 3 presents the integrated local search approach for all three stages of the initial urban solid waste collection problem, namely location, clustering and scheduling. Paragraph 4 presents the k-means approach for clustering or partitioning and in paragraph 5 the ck-means variant is discussed. Computational tests and results are discussed in paragraph 6, emphasizing in the Athenian Case Study. Finally, conclusions are drawn in paragraph 7.
2. THE MINLP FORMULATION OF THE CSPP

The Mixed Integer Non Linear Programming (MINLP) formulation of the Capacitated Set Partitioning Problem addressed in the clustering phase of the urban solid waste collection is as follows:

\[
\min z = SSSD = \sum_{j}^{c} \sum_{i}^{p} \left( \sum_{k}^{2} (t_{jk} - a_{ik})^2 \right) \cdot y_{ij}
\]

(CSPP)

subject to

\[
\sum_{i} w_{ij} y_{ij} \leq W_c, \forall j = 1,\ldots,c
\]

(1)

\[
\sum_{j} y_{ij} = 1, \forall i = 1,\ldots,p
\]

(2)

\[
\sum_{i} z_{ijk} = \sum_{i} a_{ik} y_{ij} = 0, \forall j = 1,\ldots,c \land \forall k = 1,2
\]

(3)

\[
z_{ijk} - My_{ij} \leq 0, \forall i = 1,\ldots,p \land \forall j = 1,\ldots,c \land \forall k = 1,2
\]

(4)

\[-t_{jk} + z_{ijk} \leq 0, \forall i = 1,\ldots,p \land \forall j = 1,\ldots,c \land \forall k = 1,2
\]

(5)

\[t_{jk} - z_{ijk} + My_{ij} \leq M, \forall i = 1,\ldots,p \land \forall j = 1,\ldots,c \land \forall k = 1,2
\]

(6)

The decision variables are the binary \( y_{ij} \) variables as well as the continuous \( t_{jk}, z_{ijk} \) variables. The objective to be minimized is \( z \). The binary variable \( y_{ij} \) equals 1 if point \( i \) belongs in group \( j \) else equals 0 if point \( i \) doesn’t belong in cluster \( j \). The continuous variable \( t_{jk} \) is the \( k \)-coordinate of the centroid of cluster \( j \), analytically the \( x \)-coordinate if \( k \) equals 1 and the \( y \)-coordinate if \( k \) equals 2. Finally the continuous variable \( z_{ijk} \) is an auxiliary variable derived from the linearization of the model according to the mathematical programming literature [1].

The integer parameters \( w_{i} \) are the capacities of points \( i=1,\ldots,p \). The integer parameter \( W_c \) is the maximum allowable capacity of each cluster of points. The continuous parameters \( a_{ik} \) are the \( k \)-coordinates of \( i=1,\ldots,p \) points of the CSPP. It is considered that there are \( p \) points and \( c \) groups or clusters \((c<p)\). The goal is to group all \( p \) points in \( c \) groups, ensuring that the capacities \( w_{i} \) within the same group points do not cumulatively exceed the parameter \( W_c \) for all groups. The optimal partition of the capacitated set is the one that, while satisfying the capacity constraint, minimizes the objective function Sum of Sums of Squared Distances (SSSD) of the partition.

The objective \( z \) or SSSD is well known in the literature as the squared distance criterion for partitioning [2]. As the SSSD gets smaller, partition becomes better increasing its two main quality features: cohesion within groups and isolation among groups. Constraint (1) expresses the cumulative capacity constraint for each group. Constraint (2) ensures that every point \( i \) belongs to exactly one and only group \( j \). Constraint (3) is the linearized expression of the centroid computation of a group of points. The interested reader may find more about the linearization that has been implemented at this point in the literature [1]. Finally, constraints (4), (5), and (6) linearize the nonlinear expression substituted by its linear version (3). Roughly speaking, the auxiliary variables \( z_{ijk} \) substitute the product \( ty \) appearing in the initial version of (3). The model of CSPP is optimized in GAMS with the DICOPT/(CONOPT,EXPRESS) solver in a P4 Windows PC.
3. GREEDY ALGORITHM AND LOCAL SEARCH

The local search is developed in three stages and is a refined version of an algorithm previously published [3]. Every stage directly corresponds to one of the three sequential phases of location, clustering and scheduling, as stated previously. The first stage implements a greedy algorithm for the bin location sub-problem and can be seen as the application’s preprocessor. The second stage implements a greedy algorithm and a local search procedure for the collection points set partitioning or clustering sub-problem and can be seen as the application’s main processor. Finally, the third stage implements an open-shop-like list heuristic for the scheduling sub-problem and can be thought as the application’s postprocessor. Analytically, all three stages have as following:

3.1. Greedy location: Preprocessor

The greedy location algorithm finds the city block that has the maximum garbage bin demand, then assigns the whole demand in the nearest corner and finally deletes the city block from those still waiting to be covered. Thus, the greedy function is

\[ g_b(s_c) = \min_{s_c \in C_b} \text{dist}^2(b, s_c) \]  

(7)

Where \( b \) is the city block, \( C_b \) is the set of corners in acceptable distance from \( b \) and \( s_c \in C_b \)

![Figure 1. Greedy location of bin demand for block b in corner c according to Eq. (7).](image)

3.2 Greedy and local search clustering: Main Processor

The clustering stage consists of two sub-stages, namely:

3.2.1 Greedy clustering: Main Processor Initialization

The greedy clustering algorithm works as follows: first, the algorithm initializes from scratch every empty corner group by adding to it the first corner found with the largest bin number and subsequently adds to the current group the corner that minimizes the following spatial greedy function, until no other corner can be added to the current group with respect to the cumulative capacity constraint. The spatial greedy function is:

\[ g_{cl}^{(n+1)}(s_c) = \min_{s_c \in C_f} \{g_{cl}^{(n)} + n \cdot \text{dist}^2(x, \hat{x}) + \text{dist}^2(s_c, \hat{x})\} \]  

(8)

Where \( cl \) is the current cluster or group of corners, \( C_f \) is the set of free corners not yet clustered in groups that satisfy the constraint (1), if added, \( n \) and \( x \) are the corners number and the centroid vector of the current group and \( n+1 \) as well as \( \hat{x} \) are the same variables for the modified current group, after a new single corner has been clustered into it.

The initial condition for the recursive formula (8) is: \( g_{cl}^{(1)}(s_c) = 0 \), for each \( cl \) starting from scratch.
3.2.2 Local search clustering implementing swaps: Main Processor Optimization

After the construction of the feasible greedy partition, the local search procedure via swap moves for the clustering sub-problem, or the CSSP of paragraph 2, is implemented. A swap move is a pairwise interchange or 2-exchange move between two different clusters of the partition with respect to two distinct corners, one belonging to each cluster of the partition. The swap is implemented with exhaustive search within the whole neighborhood of the current partition via the swap transformation. The best feasible neighboring partition is found applying first a feasibility and then an optimality check. Analytically, the feasibility check is double according to Eq. (9):

$$ f^i = l_{\text{max}} + n^i - n^{\text{noti}} - l^i \geq 0 $$

Where $f^i$ is a partial feasibility index for group $i=1$ or $2$, $l_{\text{max}}$ is the maximum bin load per group analogous to the $W_c$ parameter of paragraph 2, $n^i$ is the bin number added to cluster $i=1$ or $2$, $n^{\text{noti}}$ is the bin number subtracted from cluster $i$ and $l^i$ is the current bin load of cluster $i$. The symbol noti equals 2 if $i$ equals 1, and equals 1 if $i$ equals 2, thus noti equals 3-$i$. The general feasibility check is: $f^1 \geq 0$ AND $f^2 \geq 0$.

The unique optimality check is $dz > 0$ with

$$ dz = \sum_{\kappa=1}^{2} \frac{n_{\kappa}}{n_{\kappa} - 1} \cdot d^2(e_{\kappa}, x_{\kappa}) - \frac{n_{\kappa} - 1}{n_{\kappa}} \cdot d^2(e_{\text{noti}}, x_{\text{noti}}') > 0 $$

Where

Figure 2. Greedy clustering of corners according to Eq. (8).

Figure 3. Swap move layout between clusters clus1 and clus2 having centroids centre1 and centre2 examining corners corn1 and corn2.
c₁ and c₂ are the corners corn1 και corn2 respectively, x₁ and x₂ are the centroids centre1 and centre2 respectively, x₁´ and x₂´ are the modified centroids of clusters clus1 και clus2, if corners c₁ and c₂ are temporarily subtracted from their clusters clus1 and clus2 respectively. Finally n₁ and n₂ are the corner numbers of clusters clus1 and clus2 respectively. The dz is a spatial quantity or index that expresses the decrease of the SSSD or z metric of the partition. The neighboring partition to which the search moves is the one having the greatest positive decrease in the SSSD metric, dzmax. Thus, a steepest descent on the SSSD metric is imposed and the decrease per iteration i is dzmaxᵢ.

The symbol d in Eq. (10) stands for the euclidean distance symbol dist of Eq’s (7) and (8). The x’ variables are computed from the x variables by simple formulas. Equation (10) is a refined version of the optimality index dz having a sum of four terms instead of an earlier version of the same index as a sum of eight terms [3].

3. Open shop scheduling: Postprocessor

The grand total of timedistance traveled for every collection area or cluster is computed by the sum of partial timedistances according to Eq. (11):

\[ d_{cl}^{total} = d_{cl}^{1} + d_{cl}^{in} + d_{cl}^{2} + d_{cl} \]  

(11)

Where

\( d_{cl}^{total} \) is the grand total of timedistance traveled for the specific collection area or cluster
\( d_{cl}^{1} \) is the timedistance traveled straightforwardly from the parking to the cluster centroid d(p,x)
\( d_{cl}^{2} \) is the timedistance traveled straightforwardly from the cluster centroid to the landfill d(x,l)
\( d_{cl}^{in} \) is the internal timedistance or service time of the collection area or cluster
\( d_{cl} \) is the constant for all clusters timedistance traveled from the landfill to the parking d(l,p)

For simplicity, it is assumed that for every collection area serviced by a garbage truck, the truck returns to its parking after the disposal of the garbage at the landfill. The distances \( d_{cl}^{1}, d_{cl}^{2} \) and \( d_{cl} \) are Euclidean and equal to d(p,x), d(x,l) and d(l,p) respectively, where p is the unique garbage trucks fleet parking spot, x is the collection area or cluster centroid and l the unique landfill spot. For simplicity also, no transshipment stations have been examined at this stage.

The internal timedistance or service time of cluster cl, \( d_{cl}^{in} \), is defined as the sum of distances between each corner in the collection cluster and the cluster’s centroid and is equivalent to the sum of distances index (sd) of the area according to Eq. (12)

\[ d_{cl}^{in} = sd_{cl} = \sum_{j=1}^{n_{cl}} \| c_{j,cl} - x_{cl} \| \]  

(12)

Next, the open shop scheduling heuristic implements a max-min strategy in two succesive steps:

Step max: Find the active collection area having the largest timedistance grand total
Step min: Schedule this active collection area to the crew that has undertaken the smallest task load until this point of scheduling

The max–min steps are repeated until the number of active collection areas is zero. Collection areas are thought as active if they have not yet been scheduled in the step by step process of scheduling described above. The finding of the max element in the set of active areas can be simplified by the a priori sorting of the areas in decreasing order, a fact that results in positioning the max element of the list at the head of it.

Heleco ’05, TEE, Αθήνα, 3-6 Φεβρουαρίου 2005
NG ALGORITHM K-MEANS

The partitioning algorithm k-means is undoubtedly the most famous partitioning algorithm [2]. The k-means algorithm is extremely efficient and effective. Roughly speaking, the algorithm starts from a random or a naïve partition of the points set and iteratively replaces, one at a time, points from their current group to the one that decreases the partitioning criterion SSSD the most. Convergence achieved when for p iterations, where p is the number of set points, the current partition of the set has not changed. The interested reader might find more about k-means in the classical monography of Spaeth [2], which deals exhaustively with clustering algorithms.

In our case, the partitioning algorithm of k-means has a fundamental weakness. It has been designed as an unconstrained optimization algorithm for the Set Partitioning Problem. Since the problem at hand is the Capacitated Set Partitioning Problem, as formulated in paragraph 2, k-means is bound to supply infeasible partitions with regard to the not modeled constraint (1). No matter how low the cost of the k-means partition for the CSPP is, it is still a highly infeasible solution to the problem. In real terms, this implies that the distribution of the collection areas will be probably much more economic, but lots of areas will definitely violate the capacity constraint (1), resulting in overloaded garbage trucks and severe traffic law violation.

5. THE PARTITIONING ALGORITHM CK-MEANS

The partitioning algorithm ck-means, which stands for capacitated k-means, is a custom variant of classic k-means tailored for the Capacitated Set Partitioning Problem. It is different from k-means in two points. First, each point i has an integer capacity \( w_i \) or \( \text{cap}(i) \). Respectively, each group j has an integer cumulative capacity \( \text{scap}(j) \). In standard k-means code, we intervene as follows: A feasibility index of the examined insertion of point i in the new group j, different from the current one for point i, is defined by the formula \( \text{df} = \text{scap}(j) - \text{cap}(i) \geq 0 \), where \( \text{scap} \) is the parameter \( W_c \) of Eq. (1). Thus, only the feasible insertions according to the previous formula are examined by the standard k-means optimality criterion. Second, ck-means is initialized from a feasible partition for the CSPPP unlike standard k-means. The initial feasible partition for ck-means is either a naïve feasible solution or a greedy feasible solution of Eq. (8).

Both algorithms, k-means and ck-means, are coded in Fortran 90/95, using Fortran 77 conventions, based on a Fortran 66 classical implementation of k-means by Spaeth [2].

Figure 4. Open shop scheduling for 4 collection clusters and 2 crews via the max-min algorithm described in the text.
6. COMPUTATIONAL TESTS AND RESULTS

The local search code is programmed in object oriented Visual Basic 6.0. The MINLP model, for the CSPP only, has been optimized in GAMS 21.2. Algorithms k-means and ek-means, as stated above, have been coded in Fortran 77. An insight at the local search code of paragraph 3, regarding all three stages of the USW collection problem, can be reached by looking at the manual of the application, in the corresponding MSc Thesis [4]. Computer tests involved seven random tests cases and a real world case study, specifically the municipality of Athens. The computer of the tests was a Pentium 4 PC 2.66GHz, 2×512 MB RAM running WindowsXP.

In all test cases, both random and real world, spatial data was the file of city blocks with fields x, y and bin demand for all city block centroids. Furthermore, spatial data included the file of city corners with fields x and y. All coordinates x and y of points, either block centroids or city corners, refered to the same GIS. Additional parameters were, for instance, the radius dts of Figure 1 for bin location, specifically 350m for Athens, the daily task limit of Figure 4, specifically 150km for Athens etc. Parking is set to be the point p(0,0) in random problems and p(472.646, 4204.299) in the Athenian problem. The landfill is set to be the point l(100,100) for random problems and l(476.516, 4219.277) for the Athenian Problem. Specifically for Athens, the municipal parking station of the garbage truck fleet is set to reside in 151, Iera Odos Str., Egaleo and the Athens Great Area Landfill is set to reside in Konstantinos Karamanlis Av., Ano Liosia.

Table 1 presents an overall description of all 8 USWC Problems in terms of data sets and main results. Tables 2 and 3 present results of various algorithms specifically for the CSP sub-Problem in terms of partitioning cost, thus effectiveness, and execution time, thus efficiency.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Blocks</th>
<th>Corners</th>
<th>Results</th>
<th>Active corners</th>
<th>Bins</th>
<th>Clusters</th>
<th>Crews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>100</td>
<td>36</td>
<td>108</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>200</td>
<td>76</td>
<td>220</td>
<td>12</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>400</td>
<td>147</td>
<td>454</td>
<td>23</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>800</td>
<td>305</td>
<td>924</td>
<td>47</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>800</td>
<td>1600</td>
<td>603</td>
<td>1812</td>
<td>91</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1600</td>
<td>3200</td>
<td>1188</td>
<td>3595</td>
<td>181</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3200</td>
<td>6400</td>
<td>2410</td>
<td>7181</td>
<td>360</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5524</td>
<td>9086</td>
<td>4475</td>
<td>10085</td>
<td>505</td>
<td>130</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 1.** Main overall descriptive statistics for all 8 Urban Solid Waste Collection Problems.

Some comments can be made regarding Tables 1, 2 and 3. We can comment more on the extreme cases of problems 1 and 8, the smallest and largest ones. In problem 1, as well as in problem 2 and partially in problem 3, the global optimization of the MINLP model, from paragraph 2, with GAMS was possible. The CSPP had for problem 1 a global minimum partitioning cost of 8320, which was computed in 4.5s. Local search, being an approximation heuristic method, yielded a local minimum partitioning cost of 10858 for problem 1, thus 30.5% worse than MINLP in GAMS. A 30% error for such an NP-hard problem as the CSPP is satisfactory.
### TABLE 2. Results of various algorithms in partitioning cost $z=$SSSD (note: M=big number)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>F/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>p36c6</td>
<td>p76c6</td>
<td>p147c23</td>
<td>p305c47</td>
<td>p603c91</td>
<td>p1188c181</td>
<td>p2410c360</td>
<td>athens</td>
<td>F/I</td>
</tr>
<tr>
<td>Execution</td>
<td>ls</td>
<td>gams</td>
<td>kms</td>
<td>greedy</td>
<td>ckmeans</td>
<td>ckmeans</td>
<td>gams</td>
<td>kms</td>
<td>greedy</td>
</tr>
<tr>
<td>time (s)</td>
<td>10858</td>
<td>8320</td>
<td>8160</td>
<td>18157</td>
<td>17103</td>
<td>11425</td>
<td>8320</td>
<td>9308</td>
<td>11425</td>
</tr>
<tr>
<td></td>
<td>16541</td>
<td>9688</td>
<td>9308</td>
<td>43011</td>
<td>34567</td>
<td>13240</td>
<td>9688</td>
<td>9103</td>
<td>13240</td>
</tr>
<tr>
<td></td>
<td>17962</td>
<td>9103</td>
<td>9103</td>
<td>37394</td>
<td>167841</td>
<td>25774</td>
<td>9103</td>
<td>8961</td>
<td>25774</td>
</tr>
<tr>
<td></td>
<td>17962</td>
<td>8961</td>
<td>8961</td>
<td>37862</td>
<td>387199</td>
<td>25774</td>
<td>8961</td>
<td>10091</td>
<td>387199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>47670</td>
<td>42681</td>
<td>42681</td>
<td>n/a</td>
<td>n/a</td>
<td>42681</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>42681</td>
<td>34411</td>
<td>34411</td>
<td>n/a</td>
<td>n/a</td>
<td>34411</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>54441</td>
<td>42223</td>
<td>42223</td>
<td>n/a</td>
<td>n/a</td>
<td>42223</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>42223</td>
<td>194.08</td>
<td>194.08</td>
<td>n/a</td>
<td>n/a</td>
<td>194.08</td>
</tr>
</tbody>
</table>

### TABLE 3. Results of various algorithm in execution time (s) (note: t=tiny time, I=infeasible)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>F/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>ls</td>
<td>gams</td>
<td>kms</td>
<td>greedy</td>
<td>ckmeans</td>
<td>ckmeans</td>
<td>gams</td>
<td>kms</td>
<td>greedy</td>
</tr>
<tr>
<td>Execution</td>
<td>p36c6</td>
<td>p76c6</td>
<td>p147c23</td>
<td>p305c47</td>
<td>p603c91</td>
<td>p1188c181</td>
<td>p2410c360</td>
<td>athens</td>
<td>F/I</td>
</tr>
<tr>
<td>time (s)</td>
<td>4.5s</td>
<td>245s</td>
<td>&gt;20h</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>&lt;1s</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td></td>
<td>2.5s</td>
<td>2.5s</td>
<td>&gt;20h</td>
<td>0.078s</td>
<td>0.3s</td>
<td>1.6s</td>
<td>13.1s</td>
<td>t</td>
<td>&lt;1s</td>
</tr>
<tr>
<td></td>
<td>3.5min</td>
<td>3.5min</td>
<td>&gt;20h</td>
<td>0.078s</td>
<td>0.3s</td>
<td>1.6s</td>
<td>13.1s</td>
<td>t</td>
<td>&lt;1s</td>
</tr>
<tr>
<td></td>
<td>35min</td>
<td>35min</td>
<td>&gt;20h</td>
<td>0.078s</td>
<td>0.3s</td>
<td>1.6s</td>
<td>13.1s</td>
<td>t</td>
<td>&lt;1s</td>
</tr>
<tr>
<td></td>
<td>9.7h</td>
<td>9.7h</td>
<td>&gt;20h</td>
<td>0.078s</td>
<td>0.3s</td>
<td>1.6s</td>
<td>13.1s</td>
<td>t</td>
<td>&lt;1s</td>
</tr>
<tr>
<td></td>
<td>128h</td>
<td>128h</td>
<td>&gt;20h</td>
<td>0.078s</td>
<td>0.3s</td>
<td>1.6s</td>
<td>13.1s</td>
<td>t</td>
<td>&lt;1s</td>
</tr>
</tbody>
</table>

Figure 5. Local search partition for problem 1 with SSSD equal to 10858.

The contribution of the local search is clear in problems 3 to 8. As the scale of the CSPP increases, GAMS can provide no solution at all even for problem 3. Indeed, it takes more than 20h for GAMS to compute any solution, while local search finds its own in 3s. Also, it is clearly confirmed, yet not

---

1. The $z =$ SSSD objective is dimensionless in problems 1 to 7 and [z]=km$^2$ in problem 8.
2. Initialized from the naïve feasible solution.
3. Initialized from the greedy feasible solution of Eq. (8).
illustrated directly in the tables above, that the k-means algorithm yields highly economic but also highly infeasible partitions in the CSPP. For instance, in problem 1 there existed a cluster in the k-means solution with 25 bins, while the upper limit was only 20 per cluster. Ck-means, though hopeful, was a disappointment. It easily fell in the trap of premature saturation in the larger problems, where a heuristic is mostly needed. In summary, the insertion move neighborhood structure of k-means implemented in ck-means as well, was the cause for the saturation in the CSPP context. This trap was skillfully overcome by the symmetric neighborhood structure of swap moves local search. The main default of the local search was slow convergence in the extremely large data sets, a fact that is partially explained by the Visual Basic implementation. Finally, Figures 5 and 6 present the local search partitions for problems 1 and 8, respectively, in two equivalent manners, thus plotting clusters as corner sets in Figure 5 or clusters as centroid bubbles in Figure 6. The obvious reason is economy of space in the latter case.

7. CONCLUSIONS

The urban solid waste collection problem can be analyzed in three subsequent location, clustering and scheduling phases. This paper emphasizes in the clustering phase which is modeled as a capacitated set partitioning problem. The CSPP is solved with four methods: local search, MINLP modeling, k-means and ck-means. Results are strongly in favour of the local search, which compromises effectiveness and efficiency aspects.

8. ACKNOWLEDGEMENTS

One of the authors, Mr Florios, would like to thank the Hellenic State Scholarships Foundation (I.K.Y.) for its financial support under the PhD Scholarship Contract No 4160.

9. REFERENCES


Figure 6. Local search partition for problem 8 with SSSD equal to 95.61km².