A GRASP ALGORITHM TO SOLVE THE UNICOST SET COVERING PROBLEM

Joaquín Bautista, Jordi Pereira
Escola Tècnica Superior d'Enginyers Industrials de Barcelona. Universitat Politècnica de Catalunya.
Av. Diagonal 647, 7th floor, 08028 Barcelona, Spain
joaquin.bautista@upc.edu, jorge.pereira@upc.edu

Abstract – The set covering problem (SCP) is a well-known combinatorial optimization problem. We present a GRASP algorithm to Unicost Set Covering Problem, a special case of the Set Covering Problem where no distinction is made between covering sets. The most significant contribution of the algorithm is the incorporation of a local improvement procedure based on the heuristics to solve satisfiability problems (SAT). The quality of the proposed algorithm is tested on a set of reference instances, comparing the obtained results with those found in the literature. Our algorithm improves the best known solutions for many of these instances.

Keywords: Set Covering; Optimization; Constraint Satisfaction.

I. INTRODUCTION

The Set Covering Problem (SCP) is a well-known combinatorial optimization problem with numerous applications in such diverse fields as job assignment in manufacturing, selection of operators, simplification of Boolean expressions or service location, see [4] and [9] between others.

The SCP can be described as follows: given a set M, |M|=m and n subsets $S_j \subseteq M$, $j \in N=\{1,\ldots, n\}$ each with a non-negative cost c_j, the objective is to find a minimum cost family of subsets S_j such that each element $i \in M$ belongs to at least one subset of the family. An integer programming formulation of the SCP follows:

$$\text{MAX} \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{st:} \sum_{j \in N} a_{ij} x_j \geq 1 \qquad i \in M \tag{2}$$

$$x_j \in \{0,1\} \qquad j \in N \tag{3}$$

The variable x_j, equals 1 if the subset S_j is in the selected family, and 0 otherwise. The coefficients a_{ij} take the value 1 when element i belongs to S_j, and 0 otherwise. The matrix $A=(a_{ij})$, $i=1,\ldots,m$, $j=1,\ldots n$, is known as covering matrix. Each row, the constraints, is associated to an element, while each column, variable, is associated to a subset. We say that S_j covers i or that i is covered by S_j if $a_{ij}=1$.

A particular case of the problem occurs when the costs, c_j, associated to each subset, S_j, are equal. In this case, all costs may be considered equal to 1. This problem was introduced in [24], and is found under different names in the literature, such as Location Set Covering Problem (LSCP), Minimum Cardinality Set Covering Problem (MCSCP) or Unicost Set Covering Problem.

The SCP with arbitrary positive costs and the unicost SCP are NP-Hard, see [11], and are therefore considered difficult to solve to optimality. This paper presents a GRASP [21] algorithm to solve the unicost SCP. GRASP is a random iterative optimization procedure where each iteration is made up of a randomized greedy constructive step and a local search. During the constructive step diversity is added to the search while the local search step provides the required intensification method.

The motivation of this work was originated by the necessity to develop effective algorithms to locate curbside collection points for urban waste management in the metropolitan area of Barcelona, [4]. One of the problems faced was the location of collection points where citizens leave their waste in refuse bins. If we attempt to minimize the number of collection points while satisfying a service measure in terms of maximum allowable distance between citizens and their nearest collection point, the problem can be seen as a continuous set covering problem, and can be discretized into a unicost set covering problem. Afterwards the model and solutions procedures were integrated to a decision support system to aid planners in decision-making. The results provided by the proposed algorithm in this paper outperform the genetic algorithm and grasp algorithms presented in [4] and improves the best known solution for several instances in the literature.

The main difference between our algorithm and previous approaches is the novel use of a local improvement search based on constraint satisfiability problems. The local search procedure allows a better exploration of the solution space and provides a new tool to tackle difficult unicost SCP problems where a local search can be of great importance. To the best of our knowledge, no local search has been proposed for the unicost SCP other than testing which subsets are redundant for a solution. There are few studies involving the application of local search techniques to the unicost SCP problem due to the difficulty to define good neighborhoods leading to fast improvements, the difficulty to reach better solutions

keeping feasibility during the search and the efficiency of random sampling techniques. Our procedure searches broad areas with identical objective value. This is achieved by losing the feasibility of the solution. Feasibility can be recovered at any given time while keeping the tentative objective value of the unfeasible solution. This encourages to keep searching high quality solutions even when the search is working with an unfeasible solution. The results found in this papers were also presented in [5].

The novel contribution of this work is the combination of the algorithms presented in [5] into the decision support system presented in [4],

After reviewing the available literature about the set covering problem in section II, we go on to show the existing relation between this problem and constraint satisfiability problems, in section III. Section IV is devoted to the proposed procedure while section V shows the computational experience conducted to test the implemented algorithm using instances from the literature [7]. We finally present the integration of the proposed methods in a Decision Support System and the conclusions of this work in sections VI and VII.

II. LITERATURE REVIEW

The literature covers several exact and heuristic approaches to solve set covering problems. We highlight the first successful approaches from Fisher and Kedia, based on a dual heuristic and able to solve instances with up to 200 constraints and 2000 variables [10]. More recently, Balas and Carrera have proposed an algorithm based on lagrangean relaxations and subgradient optimization, which clearly outperforms the previous approaches [2].

The heuristic approaches can be divided in two main categories. The first one exploits problem characteristics and specific features of each instance. Examples are lagrangean relaxation-based procedures, subgradient optimization methods, [6] and the relaxed dual model exploitation, [9]. The second category includes local search procedures and the adaptation of metaheuristics to the set covering problem, such as genetic algorithms, [8], and ant algorithms, [17], as well as specifically tailored local search procedures, [26]. The first category has been used in many real life applications; some with structured data [9], but the quality of metaheuristic approaches using some features from the first category of heuristics, and the late appearance of a highly effective local search procedure make this category a competitive approach.

Due to the unicost version specific characteristics, specific procedures are required. Between the procedures developed for the unicost case, we highlight the adaptation of general heuristics have been adapted to the unicost case. See, for instance, Almiñana and Pastor adaptation, [1], as well as the work of Grossman and Wool, [13].

Almiñana and Pastor heuristic is based on lagrangean relaxations and the surrogate problem solution. The heuristic is tested solving 60 newly generated random instances and five literature-based instances, and thus it is quite difficult to test new procedures against this algorithm.

Grossman and Wool, [13], compared several heuristics for the unicost set covering problem. The work presents a comprehensive comparison of the results offered by a neuronal network, and several heuristics that appeared between 1974 and 1993 for both the unicost and non-unicost problem. The presented heuristics range from greedy ones to rounding algorithms and they are tested using the full set of SCP instances coming from the OR-Library, indicating that the greedy heuristic and two its variants are the best suited to solve the problem.

In a recent paper Lan et al [16] describe a Meta-RaPS implementation for both the unicost and general set covering problem. Their procedure uses an innovative local search procedure which performs equally well for both categories of problems.

III. THE SET COVERING PROBLEM AS A CONSTRAINT SATISFIABILITY PROBLEM

A. Transformation of the problem into a of maximum constraint satisfiability problem

The relation between the Set Covering Problem and the Maximum Satisfiability Problem is well-known, as noted in Garey and Johnson, [11], and Papadimitrious, [18], where the relationship between both problems is put forward. To the best of our knowledge this relationship has not been used to develop algorithms to solve the unicost case, even if at least one paper has used arbitrary cost set covering instances to evaluate the quality of the maximum satisfiability algorithms, [25].

The constraint satisfiability problem consists of assigning a true or false value to a set of literals in such a way that it satisfies a clause set in normal disjunctive form. When the goal consists in searching a solution satisfying all clauses, we have a feasibility problem known as SAT, but if the goal is to satisfy the maximum number of clauses, we have an optimization problem known as MAX SAT.

A simple scheme to transform the set covering problem into a constraint satisfiability problem follows:

- Create a set W of n literals. Each literal w_j is associated to one subset $Sj \subseteq M$, $j=1,\ldots,n$. A literal w_j takes value "true", or "false", if the variable x_j associated to the set S_j is 1, or 0 respectively
- Define a set of unitary clauses Z corresponding to the complement of each literal of W. $Z=\{z_j=\neg w_j : w_j \in W\}$
- Define a set of clauses Y, made up of m clauses, each one associated with an element of the covering problem. Each clause is formed by the disjunction of W literals whose original subsets are said to cover the element in the covering problem.

The resulting instance has n literals and $n+m$ clauses, the goal being to satisfy the maximum number of clauses of the set $Y \cup Z$.

An example of the transformation procedure is given below.

Let the following example be the mathematical formulation of a covering problem with four variables and three constraints:

$$[\text{MIN}]\ x_1+x_2+x_3+x_4 \quad (4)$$

$$\text{st: } x_1+x_2=1 \quad (5)$$

$$x_2+x_3=1 \quad (6)$$

$$x_3+x_4=1 \quad (7)$$

The set W of literals belonging to the associated satisfiability problem is $W=\{w_1; w_2; w_3; w_4\}$. The set of clauses Z would be formed by four clauses $Z=\{\neg w_1, \neg w_2; \neg w_3; \neg w_4\}$ and the set of clauses Y would be formed by three clauses $Y=\{w_1 \vee w_2 ; w_2 \vee w_3; w_3 \vee w_4\}$. An optimum solution to the MAX SAT problem is w_1=false, w_2=true, w_3=true, w_4=false, which satisfies all the clauses of set Y, and two of the clauses of set Z. The associated covering problem solution is $x_2= x_3=1$, with a value of 2, and obviously, it is the optimal solution for the SCP instance.

It is straightforward to transform a solution for the associated MAX SAT instance back to a valid SCP solution. For each unsatisfied clause Y, choose a literal from set W to take value "true". This change will keep or improve the solution to the MAXSAT instance, as it increases the number of unsatisfied clauses of set Y by one and reduces by one or more the set of unsatisfied Z clauses. When all Z clauses are satisfied, let be Ws the set of literals with value "true" from W. W_s can be mapped to a solution of the associated SCP, with value $|W_s|$.

B. Procedures to solve SAT binary satisfiability problems

The most successful heuristics for SAT problems are based on the iterative application of two different phases. The first phase consists of a constructive procedure allowing an initial solution to be obtained using a lot-drawing procedure. The proposed procedures for this first phase range from GRASP procedures, mostly found in Weighted Satisfiability, [19] [20], to random assignments, see [14], mostly found in SAT.

Once obtained a solution to the problem, the second phase, known as local search, seeks solutions of higher quality until it reaches a limited number of attempts or it does not find any better solution. The local search takes the initial solution and searches its neighborhood, choosing one of them to become the new current solution. This search is conducted through a neighborhood relation specifying the possible solutions that can be reached in one step of the local search for each solution. Most of the conducted research is associated to this phase, and a review of different procedures and a quantitative evaluation of their performance is to be seen in [15].Both phases are repeated during a specified number of iterations or until a stopping criterion is met.

Among the different procedures, we highlight the GSAT, [22], and WALKSAT [23]. Both procedures have reported good results for several feasibility problems sets. They may be seen as descent algorithms with some diversification to avoid getting trapped in local optima. Starting from a random assignment to each literal, they try to increment the number of satisfied clauses until a feasible assignment is found or a maximum number of iterations is reached.

GSAT counts on a neighborhood relation based on FLIP exchanges. A FLIP exchange consists in negating the assignment of a single literal. During each iteration of the GSAT local search, all possible FLIP exchanges for the current solution are tested and the best one is implemented, chosen randomly in case of a tie.

WALKSAT adds a mechanism to the GSAT procedure which allows movements that worsen the quality of the incumbent solution, as in Tabu search [12] though in a much simpler format. The mechanism consists of including a certain level of random search. With high probability each iteration will use the GSAT rule. If not, a random literal is chosen and flipped.

To avoid stagnation in locally optimal solutions, new random assignments are constructed and used as a diversification mechanism.

The present study implements the WALKSAT local improvement within a GRASP schema, substituting the random generation of initial solutions by a generation phase that builds initial solutions sequentially, taking into account information about the most promising literals to appear in the final solution.

IV. A GRASP ALGORITHM FOR THE PROBLEM

A. The GRASP metaheuristic

The GRASP metaheuristic [21] is a random iterative optimization procedure. This metaheuristic has been used to solve diverse problems of optimization, including scheduling, route design, logic, location, graphs, assignment, manufacturing, transport, and telecommunications problems, among others.

Each iteration in the metaheuristic is made up of two phases: a constructive and a local search phase. During the constructive phase, the algorithm uses a randomized greedy heuristic to obtain an initial solution to the problem. This is based on modified greedy procedures, where the greedy rule is substituted by a random selection among a limited list of candidates showing the best values for the greedy selection rule.

On the other hand, the local search phase permits exploration of the generated solution neighborhood in an attempt to find higher quality solutions. After local search, the best solution found during this phase is compared to the best known solution, and becomes substitutes it if the objective value is better than the

previously known. Once a stopping criterion is met, the best solution obtained during the procedure is returned.

In order to solve an optimization problem by means of a GRASP procedure, it is necessary to define at least the following elements integrated in the heuristics:

- the randomized constructive procedure used during this procedure
- the neighborhood of the solution and the procedure to investigate it
- the stopping criterion, usually associated to a maximum number of iterations.

One of the major advantages of the GRASP metaheuristic is how easy this general scheme may be adapted to the solution of particular problems. GRASP requires few parameters, basically the stopping criterion, associated to the maximum number of iterations, and a rule to construct the restricted candidate list during the constructive phase.

B. Constructive phase

The constructive phase starts with an unfeasible trivial solution w_j=false, $\forall j$=1,..,n. At each iteration of the construction phase, a literal is selected and its value is flipped to true until a feasible solution regarding each clause from Y is obtained.

The selection of the next literal to flip is limited to a restricted candidate list (RCL) which consists of the most promising literals according to a desirable function. The desirability of each literal equals the number of additional satisfied clauses from Y obtained by a true assignment to the literal. The restricted candidate list is made up of literals whose desirability is higher than a threshold value L. This threshold is based on the desirability of the best candidate and deterioration parameter $\alpha \in [0,1]$. When α=0, the constructive algorithm corresponds to a totally random algorithm; when α=1, the algorithm corresponds to a deterministic constructive algorithm with random resolution of ties.

Once the restricted candidate list is available, a candidate is then randomly chosen from the restricted list.

This constructive procedure is based on a greedy SCP heuristic. At every iteration the procedure chooses a literal associated to a variable which is not yet present in the solution. Literals are evaluated according to the number of unsatisfied Y clauses where the literal is present, equivalent to the number of additional constraints satisfied by the related variable in the set covering problem. The procedure differs from a greedy procedure because the literal included in the solution is chosen between the restricted candidate list RCL with equal probabilities, and not between those featuring the best local value for the SCP heuristic.

C. Improvement phase

The solutions generated by the constructive procedure are not necessarily optimal, even with respect to simple neighborhoods. Therefore, a local search phase usually improves the solutions provided by the constructive phase. The expected effectiveness of the local search procedure depends on a variety of aspects, such as the structure of the neighborhood definition, the neighborhood search techniques, the evaluation of the neighbors cost function, the explored neighborhood, and the initial solution.

The procedure uses a WALKSAT improvement procedure used in this paper, [23]. The procedure applies a descent procedure with probability p, choosing the best neighbouring solution from the initial reachable solution applying the best available flip exchange. Otherwise the procedure applies an escape from local optimums movement which consists of randomly choosing the literal to flip and applying this exchange.

While the constructive phase uses a procedure in consonance with the procedures to solve the set covering problem, the local search phase is completely based on the constraint satisfiability problem. During local search phases, the solution may therefore be no longer feasible for the original covering problem, as it may not fulfill constraint-associated clauses. In exchange, the procedure allows the exploration of feasible solutions areas unreachable without an unfeasibility phase. In the case of the procedure returning an unfeasible solution, Algorithm 1 may be applied to achieve a feasible solution.

V. COMPUTATIONAL RESULTS

The algorithm was programmed in C and compiled using GCC 3.2, with the –O3 optimization flag. All the runs were carried out on a Pentium 4 computer at 1800MHz with 512Mb RAM under the Linux operating system. The test were obtained from the instance sets for the general set covering problem available from the OR-Library [7]; as in previous studies devoted to the unicost problem, the costs associated to the variables appearing in each instance were ignored.

The control parameters of the GRASP heuristic and WALKSAT local search were fixed to number_iterations=500, α=0.9, MAXFLIPS=10*$|W|$ and p=0.75 for the computational experience.

Obviously the number of iterations, equal to the number of solutions generated during the construction phase, and number of flips, equal to the number of different neighbourhoods explored during each local search phase, is related to the computation time, although an increase does not necessarily improve the quality of the obtained solution.

In order to evaluate the quality of the solution obtained by the grasp heuristic presented here, we compare the best performing heuristic from the Grossman and Wool experiment, R-Gr, to compare its performance with the constructive phase of the GRASP heuristic and the total GRASP heuristic. R-Gr is a randomized greedy algorithm. In each iteration of the construction phase the variable that appears in the largest number of unsatisfied inequalities is picked. Ties are broken at random. A post optimization procedure is also used for each generated

solution, based on a redundancy elimination procedure, [13]. Grossman and Wool iterated the basic algorithm for 100 runs as ties are very common and they vary greatly the performance of the algorithm.

Table I- Results reported by the proposed algorithms for each instance present in the literature.

Inst.	R-Gr 105 it.	GRASP 105 it.	GRASP +SAT
4.1	39	39	**38**
4.2	**37**	**37**	**37**
4.3	39	39	**38**
4.4	40	40	**39**
4.5	39	39	**38**
4.6	**38**	**38**	**38**
4.7	39	**38**	**38**
4.8	**38**	**38**	**38**
4.9	39	39	**38**
4.10	40	39	**38**
5.1	**35**	**35**	**35**
5.2	**34**	**34**	**34**
5.3	**35**	**35**	**35**
5.4	35	**34**	**34**
5.5	35	35	**34**
5.6	35	35	**34**
5.7	35	**34**	**34**
5.8	36	**35**	**35**
5.9	**36**	**36**	**36**
5.10	**35**	**35**	**35**
6.1	22	**21**	**21**
6.2	21	21	**20**
6.3	**21**	**21**	**21**
6.4	**22**	**21**	**21**
6.5	**21**	**21**	**21**
A.1	**39**	**39**	**39**
A.2	40	**39**	**39**
A.3	40	**39**	**39**
A.4	**38**	**38**	**38**
A.5	**39**	**39**	**39**
B.1	**22**	**22**	**22**
B.2	**22**	**22**	**22**
B.3	**22**	**22**	**22**
B.4	23	**22**	**22**
B.5	23	**22**	**22**
C.2	**44**	**44**	**44**
C.3	**44**	**44**	**44**
C.4	**44**	**44**	**44**
C.5	**44**	**44**	**44**
C.2	**44**	**44**	**44**

The R-Gr algorithm can be seen as a special GRASP algorithm with α=1 and redundancy elimination procedure as the local search phase. We implemented the redundancy elimination procedure and iterated the algorithm for 100000 iterations. Even if the running times were still far smaller than the proposed algorithm, we stopped the search, as no improvement for any instance was found during the last 10000 iterations.

Table I and II compares the Random Greedy algorithm, column R-Gr., a GRASP with α=0.9 and local search based on redundancy elimination, column GRASP, and a GRASP with α=0.9 and local search based on WALKSAT, column GRASP+SAT, after 10^5 solutions have been constructed (500 solutions and MAXFLIPS=10*|W| for GRASP+SAT algorithm).

It can be seen that the GRASP algorithm with WALKSAT local search outperforms all other algorithms obtaining the best solutions for each instance and improving any other algorithm in 13 instances, but it shows a larger computational running time, as noted in Table III where the mean running time in seconds per algorithm and instance set is reported. Let us note that in most cases the algorithm does not require the given running to find the reported solutions. The results for the C and D sets were found within comparable running times to those reported by the R-Gr heuristic, and NRE and NRF instances were solved in a few seconds.

Table II- Results reported by the proposed algorithms for each instance present in the literature (cont.)

Inst.	R-Gr 105 it.	GRASP 105 it.	GRASP +SAT
D.1	26	**25**	**25**
D.2	**25**	**25**	**25**
D.3	26	**25**	**25**
D.4	26	**25**	**25**
D.5	26	**25**	**25**
E.1	**5**	**5**	**5**
E.2	**5**	**5**	**5**
E.3	**5**	**5**	**5**
E.4	**5**	**5**	**5**
E.5	**5**	**5**	**5**
NRE.1	**17**	**17**	**17**
NRE.2	**17**	**17**	**17**
NRE.3	**17**	**17**	**17**
NRE.4	**17**	**17**	**17**
NRE.5	**17**	**17**	**17**
NRF.1	**10**	**10**	**10**
NRF.2	11	**10**	**10**
NRF.3	11	**10**	**10**
NRF.4	11	**10**	**10**
NRF.5	11	**10**	**10**
CYC.6	**60**	61	**60**
CYC.7	156	155	**144**
CYC.8	378	377	**348**
CYC.9	894	888	**813**
CYC.10	2061	2063	**1918**
CYC.11	4688	4677	**4268**
CLR.10-4	**25**	26	**25**
CLR.11-4	25	24	**23**
CLR.12-4	**23**	25	**23**
CLR.13-4	26	26	**23**

The second best performing heuristic is the GRASP algorithm without WALKSAT search. This algorithm uses the redundancy test as the local search phase of the GRASP heuristic and obtains the best known solution for 51 out of 70 instances, with a minimal running time increase compared to the R-Gr heuristic, which is the fastest algorithm.

From the previous results, we conclude that the algorithm to apply depends on the available time to solve the instance. When the running time is not an issue, our WALKSAT algorithm can be applied as it is capable of improving the solution of several instances, while if the available time is an issue and a fast result must be provided, the GRASP approach with redundancy testing is preferable.

Let us note that the different parameters of the instances modify the quality of the solutions found from our proposal. When the density of the matrix A is relatively small, as instance sets 4, 5, A and C, the WALKSAT search usually obtains better solutions than the construction procedures with redundancy test. When the density of the matrix is high, the local search phase does not improve the quality of the solution, mostly because the quality of the solutions obtained by the constructive procedure plus redundancy testing are very near to the optimal or even optimal.

Table III- Running time in seconds per algorithm and instance set.

Inst. Set	R-Gr 105 it.	GRASP 105 it.	GRASP +SAT
4	39	59	86
5	54	94	318
6	30	42	114
A	83	155	545
B	84	126	952
C	123	230	1067
D	136	198	2433
E	15	17	54
NRE	257	310	20374
NRF	418	450	41776
CYC	1872	2333	9111
CLR	186	184	1475

VI. APPLICATION OF THE ALGORITHMS TO DECISION SUPPORT SYSTEMS

The applicability of the proposed procedures to real-life circumstances is subject to their integration within decision support systems to aid planners in decision-making. The software application was named SIRUS [3] and it was designed to assist the decision-making, design and management steps related to municipal waste collection in an urban area. The system simplifies the periodical tasks of deciding on locations of collection points, refuse bin distribution and routing decisions.

SIRUS was designed to be integrated with a geographic information system. The geographical information system makes it possible to work with CAD-like geographic data and link this data with databases from the municipality's management systems, such as the census of inhabitants, taxation on economic activities and data relevant to traffic.

The decision maker starts by selecting the section of the city where the new collection system is to be established. The sections are usually determined by political concerns, different characteristics and are related to neighborhood divisions. The application obtains information about the corresponding streets, their sections, the population present in each street section, and the possibility of driving in with a collection truck. The user may then modify the attributes of the selection so as to define which street sections should be covered and which street sections are capable of containing a collection area.

The data is used to identify the vertices and edges of the graph, as well as the covering and population graph. Additionally, a covering distance should be specified by the user to represent the maximum allowable distance each citizen should travel to reach the nearest collection point. During a preprocess, the graph is discretized, the population that cannot be covered is identified and the aforementioned algorithms are executed. The results of the algorithm are shown graphically, see the colored circles in Figure 2, together with numeric data such as the distance separating the least favored citizen/s from the nearest collection point, and the identification of this/these group/s to indicate the precise position of the citizen are given to the decision makers. This data is available to facilitate greater detail to users and decision makers alike.

The advantages offered by the use of the system or similar ones are significant. This type of system facilitates planning and programming of the garbage collection service. Indeed, it is possible to plan with ease, for medium and long term periods, a selective collection service in a municipality or in one of its zones. To do this, it is sufficient to simulate future scenarios for which population evolution and information related to urban planning are the basic requirements. This means that it is possible to design the physical collection system in a new urban area without having to adjust the service by trial and error in a populated sector of the territory.

The application of this type of system leads to cost reductions and improvements in the service. The results offered by the system make it possible to reveal deficiencies in the quality of the service, such as very long distances covered by citizens and overloaded garbage accumulation areas. They also reveal wasteful elements such as an excessive number of containers or garbage accumulation areas which are very close. Results of this type, which are obtained interactively and quickly with each modification to the design of the physical system, may be used for taking corrective measures with the aim of improving quality and better use of resources

VII. CONCLUSIONS

Operations Management, Operations Research and Artificial Intelligence are capable of modeling and solving Selective Municipal Waste Management problems. One of the problems encountered in the design of selective municipal waste collection systems is treated as a unicost set covering problem and a novel procedure based on the combination of Operations Research and Artificial Intelligence methods is proposed. From an algorithmic point of view, the algorithm performs significantly well, improving several best known solutions.

It is important to point out that the proposed algorithm should be used in combination with a Decision Support System to obtain an application allowing a methodological approach to the design of municipal waste management systems and to increase the collection of recyclable products. A tool with these characteristics facilitates the calculations performed in the decision-making step for a given management plan, which might not be solved in absence of such a tool.

VIII. ACKNOWLEDGEMENTS

This work was partly funded by Grant BEC2003-03809 from the CYCIT and Grant TIC2002-10886E from the Spanish Ministry of Science and Technology. We also acknowledge the support given by the UPC Nissan Chair.

REFERENCES

[1] Almiñana, M., Pastor, J. T., An adaptation of SH heuristic to the location set covering problem, Eur J Oper Res 1997, 100(3):586-593

[2] Balas, E., Carrera, M. C., A dynamic subgradient-based branch-and-bound procedure for set covering. Oper Res 1996, 44(6): 875-890

[3] Bautista J. Proyecto integral de gestión de residuos urbanos en el municipio de Sant Boi de Llobregat. Barcelona, CPDA, 2001

[4] Bautista, J., Pereira, J. Modeling the problem of locating collection areas for urban waste management. An application to the metropolitan area of Barcelona. OMEGA 2006, 34(6):617-629

[5] Bautista, J., Pereira, J. A GRASP algorithm to solve the unicost set covering problem. Comput. Oper. Res., In press. doi:10.1016/ j.cor.2005.11.026

[6] Beasley, J.E., A Lagrangean heuristic for set covering problems. Nav Res Log 1990, 37(1):151-164

[7] Beasley, J.E., OR-Library: distributing test problems by electronic mail, J Oper Res Soc 1990, 41(11):1069-1072

[8] Beasley JE, Chu P.C., A genetic algorithm for the set covering problem. Eur J Oper Res 1996, 94(2):392-404

[9] Caprara, A., Fischetti, M., Toth, P., A heuristic method for the set covering problem, Oper Res, 1999, 47(5):730-743

[10]Fisher, M.L., Kedia, P., Optimal solutions of set covering/partitioning problems using dual heuristics. Manage Sci 1990, 36(6):674-688

[11]Garey, M. R., Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Company, 1979

[12]Glover, F., Laguna, M., Tabu Search, Kluwer Academics Publishing 1997

[13]Grossman, T, Wool, A., Computational experience with approximation algorithms for the set covering problem. Eur J Oper Res, 1997, 101(1):81-92

[14]Gu, J., Efficient Local Search for Very Large-Scale Satisfiability Problems, ACM SIGART Bulletin 1992(3):8-12

[15]Hoos, H.H., Stutzle, T., Local search algorithms for SAT: An empirical evaluation, J Autom Reasoning 2000, 24(4):421-481

[16]Lan, G., DePuy, G.W., Whitehouse, G.E., An effective and simple heuristic for the set covering problem, Eur. J. Oper. Res., in press. doi:10.1016/j.ejor.2005.09.028

[17]Lessing, L., Dumitrescu, I., Stützle, T., A comparison between ACO algorithms for the Set Covering Problem, Lecture Notes in Computer Science 2004, 3172:1-12

[18]Papadimitrious, C.H., Computational Complexity, Addison-Wesley Pub Co, 1993

[19]Resende, M.G.C., Feo, T.A. A GRASP for Satisfiability, in Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Johnson, D.S., Trick, M.A., editors., DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1996, 26:499-520

[20]Resende, M.G.C., Pitsoulis, L.S., Pardalos, P.M., Approximate solution of weighted MAX-SAT problems using GRASP, in Du, D., Gu, J., Pardalos, P.M., editors, DIMACS Series on Discrete Mathematics and Theoretical Computer Science Satisfiability problem: Theory and Applications, American Mathematical Society, 1997, 35:393-405

[21]Resende, M. G. C., Ribeiro, C. C., Greedy randomized adaptive search procedures, in Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, 2003, p. 219-249

[22]Selman, B., Levesque, H., Mitchell, D., A new method for solving hard satisfiability problems. In Proceedings of the 10th National Conference on Artificial Intelligence, AAAI Press / The MIT Press, Menlo Park, CA, USA, 1992. p 440-446

[23]Selman, B., Levesque, H., Mitchell, D., A new method for solving hard satisfiability problems. In Proceedings of the 10th National Conference on Artificial Intelligence, AAAI Press / The MIT Press, Menlo Park, CA, USA, 1992. p 440-446

[24]Toregas, C., Swain, R., ReVelle, C., and Bergman, L., The Location of Emergency Service Facilities, Oper Res 1971, 19:1363-1373

[25]Yagiura, M., Ibaraki, T., Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: Experimental evaluation, J Heuristics 2001, 7(5):423-442

[26]Yagiura, M., Kishida, M., Ibaraki, T. , `A 3Flip Neighborhood Local Search for the Set Covering Problem, ' Technical Report #2004-001, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 2004