21

Using a Genetic Algorithm to Solve the Benders' Master Problem for Capacitated Plant Location

Ming-Che Lai[1] and Han-suk Sohn[2,*]
[1]*Yu Da University,*
[2]*New Mexico State University*
[1]*Taiwan*
[2]*USA*

1. Introduction

The capacitated plant location problem (CPL) consists of locating a set of potential plants with capacities, and assigning a set of customers to these plants. The objective is to minimize the total fixed and shipping costs while at the same time demand of all the customers can be satisfied without violating the capacity restrictions of the plants. The CPL is a well-known combinatorial optimization problem and a number of decision problems can be obtained as special cases of CPL. There are substantial numbers of heuristic solution algorithms proposed in the literature (See Rolland et al., 1996; Holmberg & Ling, 1997; Delmaire et al., 1999; Kratica et al., 2001; He et al., 2003; Uno et al., 2005). As well, exact solution methods have been studied by many authors. These include branch-and-bound procedures, typically with linear programming relaxation (Van Roy & Erlenkotter, 1982; Geoffrion & Graves, 1974) or Lagrangiran relaxation (Cortinhal & Captivo, 2003). Van Roy (1986) used the Cross decomposition which is a hybrid of primal and dual decomposition algorithm, and Geoffrion & Graves (1974) considered Benders' decomposition to solve CPL problem. Unlike many other mixed-integer linear programming applications, however, Benders decomposition algorithm was not successful in this problem domain because of the difficulty of solving the master system. In mixed-integer linear programming problems, where Benders' algorithm is most often applied, the master problem selects values for the integer variables (the more difficult decisions) and the subproblem is a linear programming problem which selects values for the continuous variables (the easier decisions). If the constraints are explicit only in the subproblem, then the master problem is free of explicit constraints, making it more amenable to solution by genetic algorithm (GA). The fitness function of the GA is, in this case, evaluated quickly and simply by evaluating a set of linear functions. In this chapter, therefore, we discuss about a hybrid algorithm (Lai et al., 2010) and its implementation to overcome the difficulty of Benders' decomposition. The hybrid algorithm is based on the solution framework of Benders' decomposition algorithm, together with the use of GA to effectively reduce the computational difficulty. The rest of

[*] Corresponding Author

this chapter is organized as follows. In section 2 the classical capacitated plant location problem is presented. The applications of Benders' decomposition and genetic algorithm are described in sections 3 and 4, respectively. In Section 5 the hybrid Benders/genetic algorithm to solve the addressed problem is illustrated. A numerical example is described in Section 6. Finally, some concluding remarks are presented in Section 7 followed by an acknowledgment and a list of references in Sections 8 and 9, respectively.

2. Problem formulation

The classical capacitated plant location problem with *n* potential plants and *m* customers can be formulated as a mixed integer program:

$$\text{CPL: Min} \sum_{i=1}^{m} F_i Y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij} \tag{1}$$

$$\text{Subject to } \sum_{i=1}^{m} X_{ij} \geq D_j, \quad j = 1, \ldots n \tag{2}$$

$$\sum_{j=1}^{n} X_{ij} \leq S_i Y_i, \quad i = 1, \ldots m \tag{3}$$

$$X_{ij} \geq 0 \text{ , } i = 1, \ldots m; \; j = 1, \ldots n \tag{4}$$

$$Y_i \in \{0,1\}, \quad i = 1, \ldots m \tag{5}$$

Here, *Y* is a vector of binary variables which selects the plants to be opened, while *X* is an array of continuous variables which indicate the shipments from the plants to the customers. *Fi* is the fixed cost of operating plant *i* and *Si* its capacity if it is opened. *Cij* is the shipping cost of all of customer *j*'s demand *Dj* from plant *i*. The first constraint ensures that all the demand of each customer must be satisfied. The second constraint ensures that the total demand supplied from each plant does not exceed its capacity. As well, it ensures that no customer can be supplied from a closed plant.

3. Benders' decomposition algorithm

Benders' decomposition algorithm was initially developed to solve mixed-integer linear programming problems (Benders, 1962), i.e., linear optimization problems which involve a mixture of either different types of variables or different types of functions. A successful implementation of the method to design a large-scale multi-commodity distribution system has been described in the paper of Geoffrion & Graves (1974). Since then, Benders' decomposition algorithm has been successfully applied in many other areas, for example, in vehicle assignment (Cordeau et al., 2000, 2001), cellular manufacturing system (Heragu, 1998), local access network design (Randazzo et al., 2001), spare capacity allocation (Kennington, 1999), multi-commodity multi-mode distribution planning, (Cakir, 2009), and generation expansion planning (Kim et al., 2011). Benders' algorithm projects the problem onto the *Y*-space by defining the function

$$v(Y) = \sum_{i=1}^{m} F_i Y_i + \text{Min} \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} X_{ij} \tag{6}$$

$$\text{Subject to } \sum_{i=1}^{m} X_{ij} \geq D_j, \quad j = 1,\ldots n \tag{7}$$

$$\sum_{j=1}^{n} X_{ij} \leq S_i Y_i, \quad i = 1,\ldots m \tag{8}$$

$$X_{ij} \geq 0 \text{ , } i = 1,\ldots m; \; j = 1,\ldots n \tag{9}$$

and restating the problem (CPL) as

$$\underset{Y\in\{0,1\}^m}{\text{Min}} \; v(Y) \tag{10}$$

We will refer to the evaluation of *v(Y)* as the (primal) subproblem, a transportation LP whose dual LP problem is

$$v(Y) = \sum_{i=1}^{m} F_i Y_i + \text{Max} \sum_{i=1}^{m} S_i Y_i U_i + \sum_{j=1}^{n} D_j V_j \tag{11}$$

$$\text{Subject to } -U_i + V_j \leq C_{ij} \quad \text{for } i = 1,\ldots m; j = 1,\ldots n \tag{12}$$

$$U_i \geq 0, \quad i = 1,\ldots m; \quad V_j \geq 0, \quad j = 1,\ldots n \tag{13}$$

If $\psi = \{(\hat{U}^k, \hat{V}^k), k=1,\ldots,K\}$ is the set of basic feasible solutions to the dual subproblem, then in principle *v(Y)* could be evaluated by a complete enumeration of the *K* basic feasible solutions. (The motivation for using the dual problem is, of course, that ψ is independent of *Y*.) That is,

$$v(Y) = \sum_{i=1}^{m} F_i Y_i + \underset{k=1,2,\ldots K}{\text{Max}} \left\{ \sum_{i=1}^{m} S_i \hat{U}_i^k Y_i + \sum_{j=1}^{n} D_j \hat{V}_j^k \right\} = \underset{k=1,2,\ldots K}{\text{Max}} \left\{ \alpha^k Y + \beta^k \right\} \tag{14}$$

where $\alpha_i^k \equiv F_i + S_i \hat{U}_i^k, \beta^k \equiv \sum_{j=1}^{n} D_j \hat{V}_j^k$.

The function *v(Y)* may be approximated by the underestimate

$$\underline{v}_{\mathrm{T}}(Y) \equiv \underset{k=1,2,\ldots T}{\text{Max}} \left\{ \alpha^k Y + \beta^k \right\} \tag{15}$$

where *T*≤*K*. Benders' decomposition alternates between a master problem

$$\underset{Y\in\{0,1\}}{\text{Min}} \; \underline{v}^T(Y) \tag{16}$$

which selects a trial Y^k, and the subproblem, which evaluates $v(Y^k)$ and computes a new linear support $\alpha^k Y+\beta^k$ using the dual solution of the transportation subproblem. The major effort required by Benders' algorithm is the repeated solution of the master problem, or its mixed-integer LP equivalent,

$$\text{Min } Z \tag{17}$$

$$\text{Subject to } Z \geq \alpha^k Y + \beta^k, \quad k = 1,\ldots T \tag{18}$$

$$Y_i \in \{0,1\} \tag{19}$$

One approach to avoiding some of this effort is by suboptimizing the master problem, i.e., finding a feasible solution of the linear system

$$\hat{Z} > \alpha^k Y + \beta^k, \quad k = 1,\ldots T \tag{18}$$

$$Y_i \in \{0,1\}, \quad i = 1,\ldots m \tag{19}$$

i.e., Y such that $\underline{v}^T(Y) < \hat{Z}$, where $\hat{Z}$ is the value of the incumbent at the current iteration, i.e., the least upper bound provided by the subproblems. (By using implicit enumeration to suboptimize the master problem, and restarting the enumeration when solving the following master problem, this modification of Benders' algorithm allows a single search of the enumeration tree, interrupted repeatedly to solve subproblems.) For more information on the problem and the application of Benders' algorithm for its solution, refer to Salkin et al. (1989).

4. Genetic algorithm

Genetic algorithm (GA) has been effective and has been employed for solving a variety of difficult optimization problems. Much of the basic ground work in implementing and adapting GAs has been developed by Holland (1992). Since then, a large number of papers have appeared in the literature, proposing variations to the basic algorithm or describing different applications. In many cases, the GA can produce excellent solutions in a reasonable amount of time. For certain cases, however, the GA can fail to perform for a variety of reasons. Liepins & Hilliard (1989) have pointed out three of these reasons: (1) choice of a representation that is not consistent with the crossover operator; (2) failure to represent problem-specific information such as constraints; and (3) convergence to local optima (premature convergence). The first reason for failure, a representation inconsistent with the crossover operator, is most easily illustrated by an example of the traveling salesman problem, in which the crossover operator simply fails to preserve the feasible permutation in most cases. The second reason for failure is the inability to represent problem specific information such as constraints in an optimization problem. In general, for constrained problems, there is no guarantee that feasibility will be preserved by crossover or mutation, or even that a randomly-generated initial population is feasible. A broad range of approaches have been used in the literature to remedy this situation. However, there is no single mechanism that has performed consistently well in handling constrained problems with genetic algorithms (Reeves, 1997). The most direct solution is simply to ignore this problem. If an infeasible solution is encountered, it may be assigned a very low fitness value to increase the chance that it will "die off" soon. But sometimes, infeasible solutions are close to the optimum by any reasonable distance measure. Another direct solution is to modify the objective function by incorporating a penalty function which reduces the fitness by an amount which varies as the degree of infeasibility. Unfortunately, not all penalty functions work equally well, and care must be exercised in their choice (Liepins & Hillard, 1989). If the penalty is too small, many infeasible solutions are allowed to enter the population pool; if it is too large, the search is confined to a very small portion of the search space. Another increasingly popular technique for coping with infeasibility is the use of repair algorithms. These heuristic algorithms accept infeasible solutions but repair them in order to make them feasible before inserting them into the population. We can find various repair algorithms in the context of the traveling salesman problem in the literature (Goldberg & Lingle, 1985; Oliver et al., 1987; Chatterjee et al., 1996). Several practical questions arise, such as whether it should be the original offspring or the repaired version that should be used in the next generaion, and whether the entire randomness should be sacrificed because of the adoption of the repair methods. The third reason for failure is convergence to local optima (premature convergence). This condition occurs when most strings in the population have similar allele values. In this case, applying crossover to similar strings results in another similar string, and no new areas of the search space are explored (Levine, 1997). Many improvements to the genetic algorithms help to avoid premature convergence, such as thorough randomization of initial populations, multiple restart of problems, and appropriate parameter settings, i.e., carefully adjustment of the mutation rate and a suitable population size.

Most researchers agree that, to guarantee success of an application of genetic algorithms, the representation system is of crucial importance. The difference between a successful application and an unsuccessful one often lies in the encoding. Kershenbaum (1997) pointed out that an ideal encoding would have the following properties: (a) It should be able to represent all feasible solutions; (b) It should be able to represent only feasible solutions. (An encoding that represents fewer infeasible solutions is generally better than one that represents a large number of infeasible solutions. The larger the number of representable infeasible solutions, the more likely it is that crossover and mutation will produce infeasible offspring, and the less effective the GA will become.); (c) All (feasible) solutions should have an equal probability of being represented; (d) It should represent useful schemata using a small number of genes that are close to one other in the chromosome. (It is generally very difficult to create an encoding with this property a priori, since we do not know in advance what the useful schemata are. It is, however, possible to recognize the presence of short, compact schemata in solutions with high fitness and thus to validate the encoding after the fact. This is important for recognizing successful GA applications.); and (e) The encoding itself should possess locality, in the sense that small changes to the chromosome make small changes in the solution. Kershenbaum also pointed out taht although some of these properties conflict (often making tradeoffs), to the extent taht those properties can be achieved, the genetic algorithms are likely to work well. In this section, we focus on the design of the GA approach for the master problem of CPL problem. More discussion of some of these as well as definitions and some of the basic GA terminology that is used in this section can be found in Goldberg (1989) and Davis (1991). The implementation of GA is a step-by-step procedure:

4.1 Initialization

Initialization is to generate an initial population. The population size and length of "chromosome" depends on the users' choice and other requirements of the specific problem. To start, we usually have a totally random population. Each random string (or "chromosome") of the population, representing a possible solution for the problem, is then evaluated using an objective function. The selection of this objective function is important because it practically encompasses all the knowledge of the problem to be solved. The user is supposed to choose the proper combination of desirable attributes that could be best fit to his purposes. In CPL problem, the variable Y is a vector of binary integers. It is easily to be coded as a string of binary bit with the position #*i* corresponding to the plant #*i*. For example, Y = (0 1 1 0 1 0 0) means that plants #1, 4, 6 and 7 are not open and plants 2, 3 and 5 are open. In our GA, a population size of 50 was used and the fitness function is evaluated quickly and simply by evaluating a set of linear functions, i.e., $\underline{v}_T(Y) \equiv \underset{k=1,2,\ldots T}{\text{Max}} \{\alpha^k Y + \beta^k\}$.

4.2 Selection

Selection (called "reproduction" by Goldberg) starts with the current population. Selection is applied to create an intermediate population or mating pool. All the chromosomes in the mating pool are waiting for other operations such as crossover and/or mutation to create the next population. In the canonical genetic algorithm, selection is made according to the fitness. The fitness could be determined by many ways. For example, the fitness could be assigned according to probability of a string in the current population (Goldberg, 1989), a string's rank in the population (Baker, 1985; Whitley, 1989), or simply by its performance of scores. In our GA, the latter case is used, i.e., a string with an average score is given one mating; a string scoring one standard deviation above the average is given two matings; and a string scoring one standard deviation below the average is given no mating (Michalewicz, 1998).

4.3 Crossover and mutation

We use a standard single-point crossover method. The duplicated strings in the mating pool are randomly paired off to produce two offspring per mating. The crossover location of the strings is generally chosen at random but not necessary always the case. For example, the distribution for selection the crossover point of the GenJam system, an interactive genetic algorithm jazz improviser, which was developed by Dannenberg for the Carnegie Mellon MIDI Toolkit, is biased toward the center of the chromosome to promote diversity in the population. If a crossover point is too near one end of the chromosome or the other, the resulting children are more likely to resemble their parents. This will lead the GenJam system to repeat itself when two nearly identical phrases happen to be played close to one another in the same solo and it does not seem desirable for GenJam to perform in that way. The role of mutation is to guarantee the diversity of the population. In most case, mutation alters one or more genes (positions in a chromosome) with a probability equal to the mutation rate. Typically, but not always, mutation will flip a single bit. In fact, GenJam's mutation operators, on the other hand, are more complex than flipping a bit. They adopt several standard melodic development techniques, such as transposition, retrograde, rotation, inversion, sorting, and retrograde-inversion. Because these operators are all musically meaningful, they operate at the event level rather than on individual bits (Biles, 2001).

4.4 Replacement

After the process of selection, crossover, and mutation, the current population is replaced by the new population. Those successful individuals of the each generation are more likely to survive in the next generation and those unsuccessful individuals are less likely to survive. In our GA, we use the incremental replacement method (See Beasley et al., 1993), i.e., only the new individuals whose fitness values are better than those of the current will be replaced. Thus, the individuals with the best fitness are always in the population.

4.5 Termination

In general, a genetic algorithm is terminated after a specified number of generations or when fitness values have converged. Our GA terminates when there has been no improvement in the best solution found for 100 iterations.

5. Hybrid Benders/Genetic algorithm

The basic idea of Benders' partitioning algorithm for mixed-integer linear problems is to decompose the original problem into a pure integer master problem and one or more subproblems in the continuous variables, and then to iterate between these two problems. If the objective function value of the optimal solution to the master problem is equal to that of the subproblem, then the algorithm terminates with the optimal solution of the original mixed-integer problem. Otherwise, we add constraints, termed Benders' cuts, one at a time to the master problem, and solve it repeatedly until the termination criteria are met. A major difficulty with this decomposition lies in the solution of the master problem, which is a "hard" problem, costly to compute.

For the addressed CPL problem, however, the constraints are explicit only in the subproblem and the master problem is free of explicit constraints. Thus, the master problem is more amenable to solution by GA.

Lai et al. (2010) introduced a hybrid Benders/Genetic algorithm which is a variation of Benders' algorithm that uses a genetic algorithm to obtain "good" subproblem solutions to the master problem. Lai and Sohn (2011) conducted a study applying the hybrid Benders/Genetic algorithm to the vehicle routing problem. Below is a detailed description of the hybrid algorithm and it is illlustrated in Fig. 1 as well.

Step 1. Initialization. We initialize the iteration counter k to zero, select initial trial values for the vector of binary variables Y which selects the plants to be opened.

Step 2. Primal Subsystem. We evaluate the value of $v(Y)$ by solving a tranportation linear programming problem whose fesible region is independent of Y.

Step 3. Generation of Benders' Cut. We compute a new linear support using the dual solution of the transportation subproblem and increment k by 1.

Step 4. Primal Master system by GA. A trial location paln Y is to be computed by implementing a GA whose solution delivers both a feasible investment plan and a lower bound to the minimal cost for the equivalent program.

4a. Initialization. We initialize the variable Y as a string of binary bit with the position #i corresponding to the plant #i. We generate initial population and their fitness function are evluated as well.

4b. Genetic Operations. We perform a standard single-point crossover approach. The mutation operation to guarantee the diversity of the population is performed as well. The current population is replaced by the new population through the incremental replacement method.

4c. Termination. We terminate the GA if no improvement within 100 iterations.

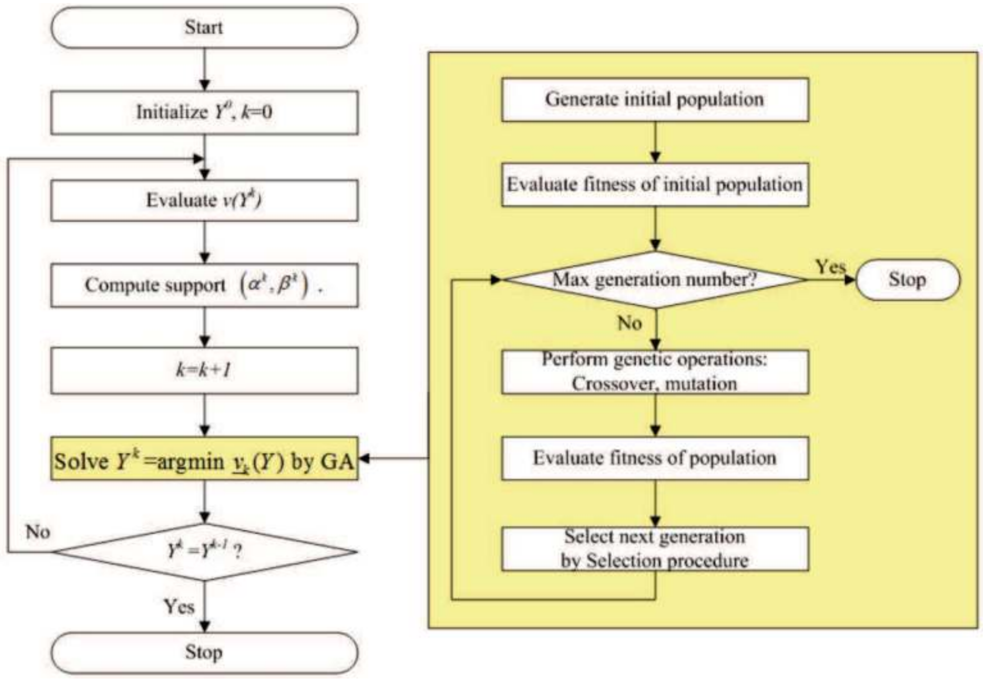

Start
Initialize Y^0, k=0
Evaluate $v(Y^k)$
Compute support (α^k, β^k)
$k=k+1$
Solve Y^k=argmin $\underline{v}_k(Y)$ by GA
No
$Y^k = Y^{k-1}$?
Yes
Stop
Generate initial population
Evaluate fitness of initial population
Max generation number?
Yes
Stop
No
Perform genetic operations: Crossover, mutation
Evaluate fitness of population
Select next generation by Selection procedure

Fig. 1. Flowchart of the Hybrid Benders/Genetic Algorithm

This hybrid algorithm would avoid other traditional search methods, i.e., branch-and-bound, which were used in the master problem. It will search the solution space in parallel fashion and take advantage of the "easy" evaluation of the fitness function.

6. Example

To illustrate the hybrid algorithm discussed in the earlier section, we use a randomly-generated problem with 20 plant sites and 50 customers. Fifty points in a square area were randomly generated, and the first 20 of these points were designated as both demand points and potential plant sites (see Fig. 2).

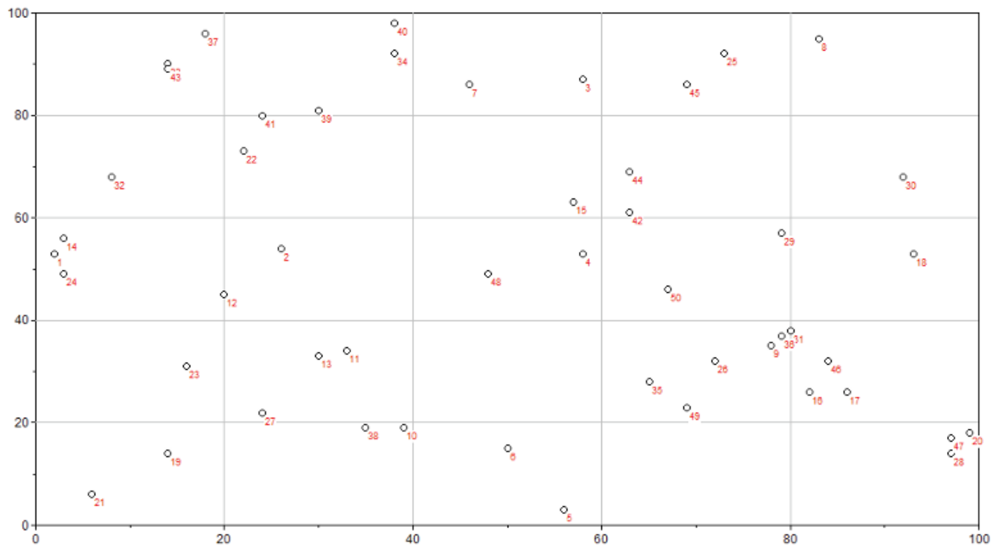


Fig. 2. Fifty Randomly Generated Points

The transportation cost between two points is proportional to the Euclidean distance between them. Three variations of Benders' algorithm were applied to this plant location problem: (1) Optimization of master problem using implicit enumeration (BD-Opt); (2) Suboptimization of master problem using implicit enumeration (BD-Subopt); and (3) Suboptimization of master problem using a genetic algorithm (Hybrid BD/GA). In each case, the problem was not solved to completion, but was terminated after solving 50 subproblems.

First, an implicit enumeration algorithm was used to optimize Benders' master problem. Fig. 3 shows the values of the upper and lower bounds, i.e., the solutions of the subproblems and master problems, respectively. The incumbent solution, which was found at iteration #10, is shown in Fig. 4 and requires opening 11 plants with a total cost of 5398, of which 2619, or 48.5%, are fixed costs of the plants and the remaining costs are transportation costs. The greatest lower bound at this stage is 4325, so that the gap is approximately 19.9% when the algorithm was terminated.

Secondly, the algorithm was restarted and again 50 iterations were performed, but suboptimizing the master problem using implicit enumeration. Fig. 5 shows the progress of this case. Because the master problem was suboptimized, no lower bound is available. After 50 iterations, the incumbent solution shown in Fig 6, which requires opening seven plants, has a total cost of 5983, of which 1710, or approximately 28.6%, are fixed costs of the plants. It is important to note, of course, that although the quality of the incumbent solution is somewhat inferior to that found by optimizing the master problem, the computational effort is miniscule compared to that required when the master problem is optimized.

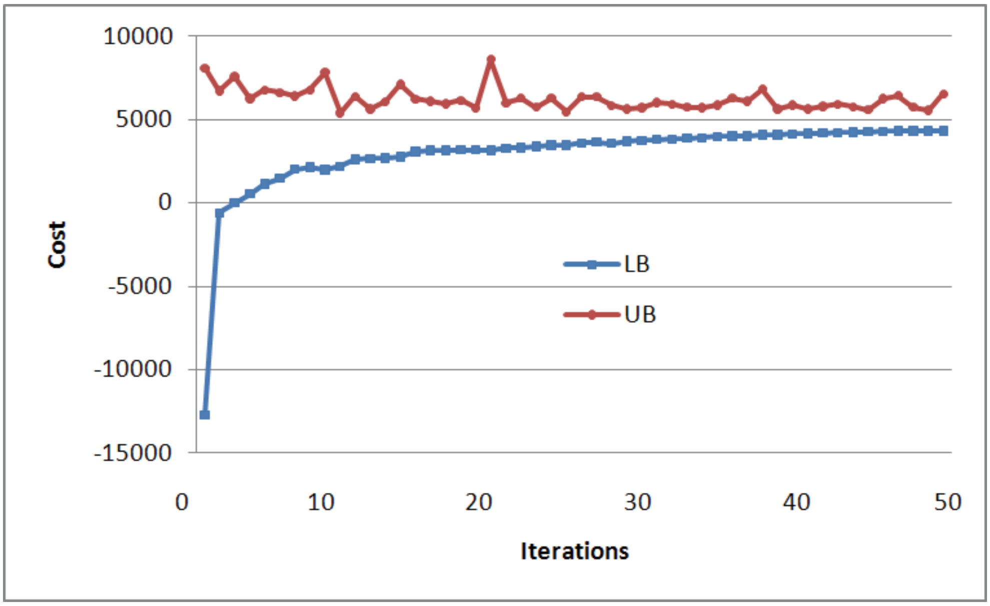


Fig. 3. Upper and lower bounds provided by Benders' algorithm (BD-Opt).

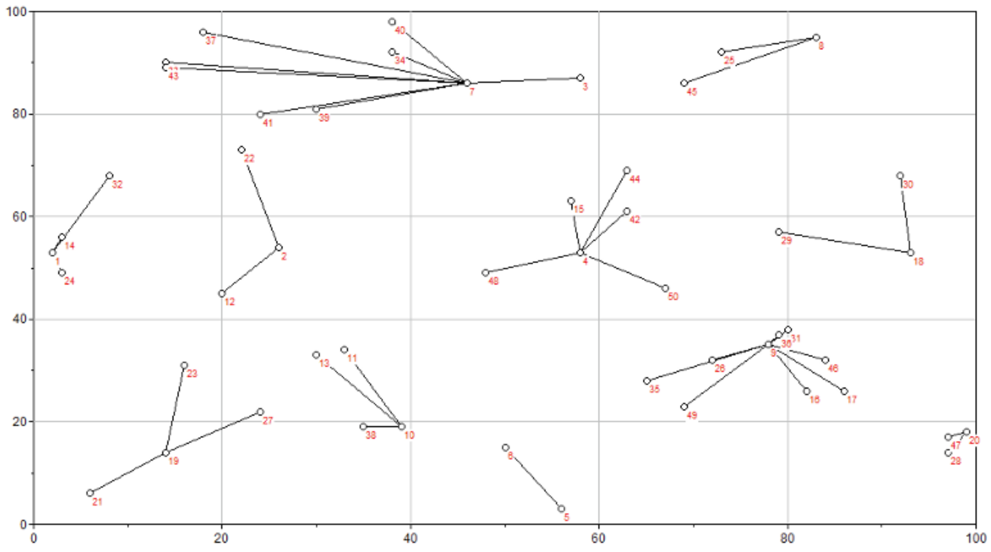


Fig. 4. Incumbent Solution Found by Benders' algorithm (BD-Opt).

Finally, the algorithm was again restarted, and 50 trial solutions were evaluated by the subproblems, this time using a genetic algorithm, so that the master problem is again suboptimized to generate the trial solutions. Each master problem was terminated after 40 trial solutions better than the incumbent have been found (or after a maximum of 100 generations) at which time all those solutions better than the incumbent were evaluated. (After each subproblem, the trial solutions are re-evaluated, using the updated master problem cost function, $\underline{v}^T(Y)$, and only those with cost less than the incumbent are evaluated by the subproblem.)

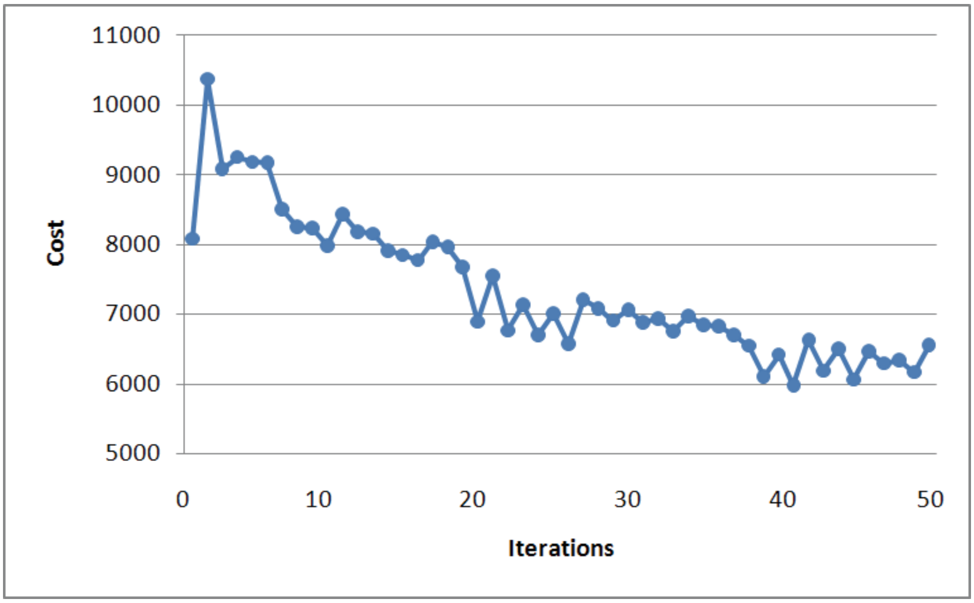


Fig. 5. Subproblem solutions of variation 2 of Benders' algorithm (BD-Subopt).

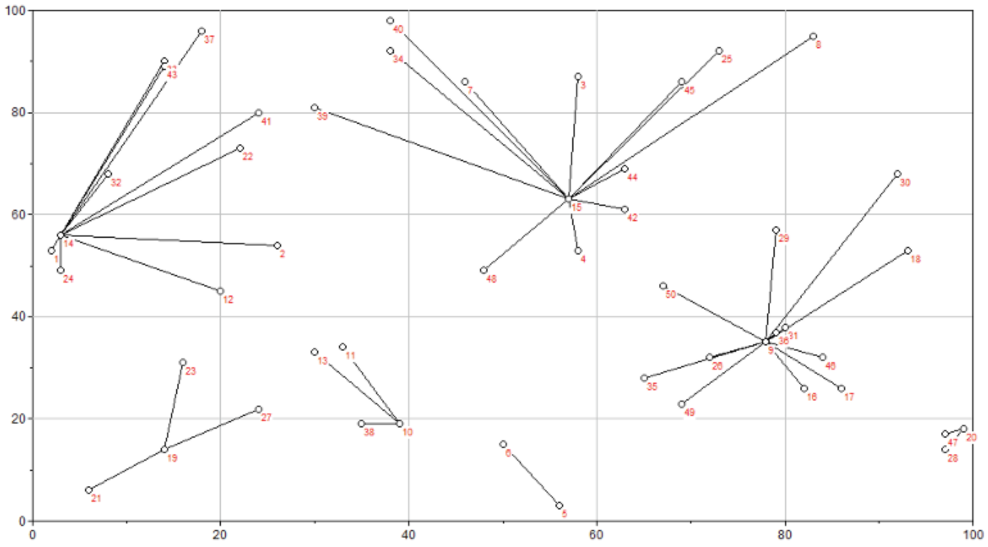


Fig. 6. Incumbent Solution Found by variation 2 of Benders' algorithm (BD-Subopt).

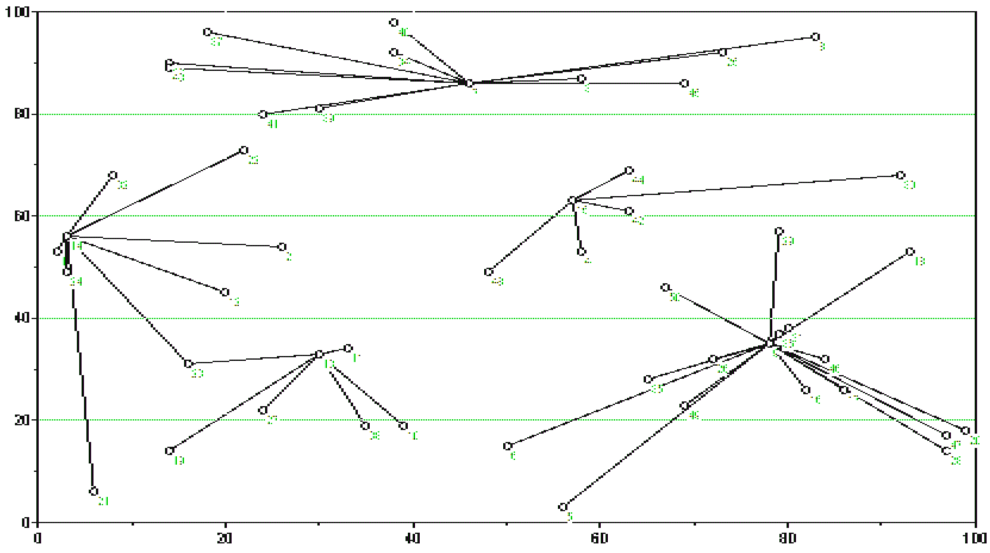


Fig. 7. Incumbent Solution by variation 3 of Benders' algorithm (Hybrid BD/GA) *trial 1*.

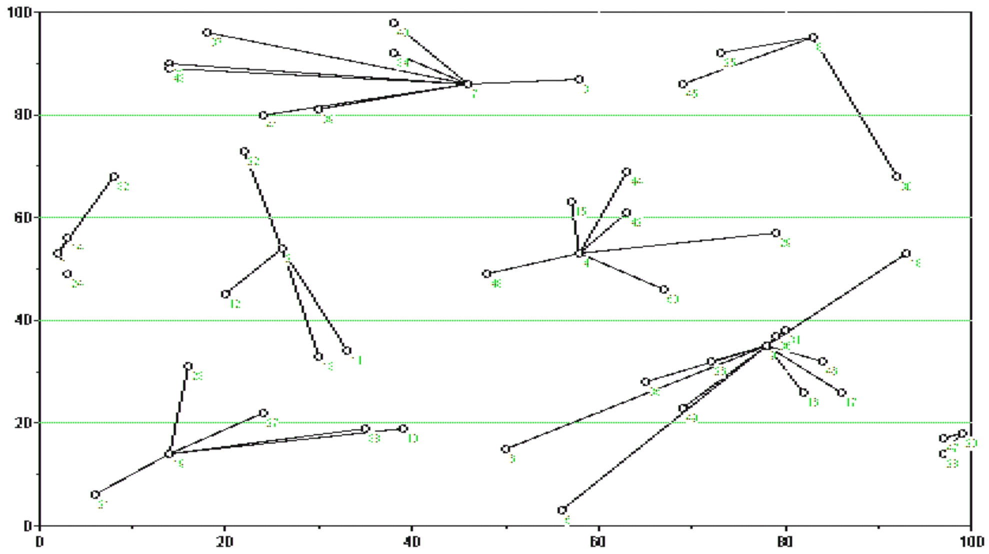


Fig. 8. Incumbent Solution by variation 3 of Benders' algorithm (Hybrid BD/GA) *trial 2*.

In this case, it happens that only 7 master problems were required to generate the 50 trial solutions. (A population size of 50 was used, with 75% probability of crossover and 1% probability of mutation.)

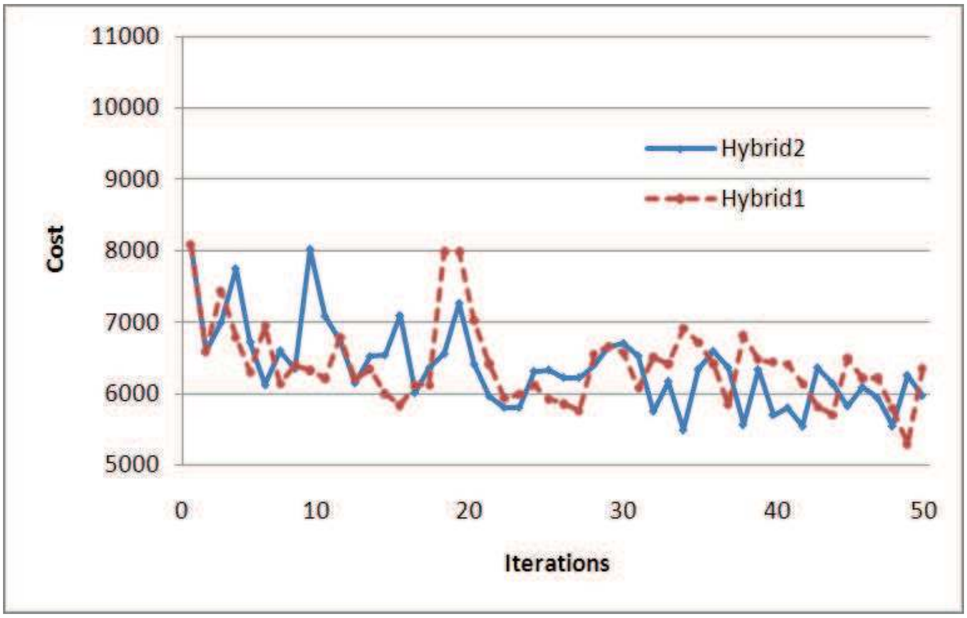


Fig. 9. Upper bounds provided by Benders' subproblems in variation 3 (Hybrid BD/GA).

The best of the 50 trial solutions was found at iteration 49, with a total cost of 5303, of which 988 (or approximately 18.6%) were fixed costs. Five plants were opened in this solution (see Fig. 7). Again, because the master problem is being suboptimized, no lower bound is available from the algorithm. Due to the random nature of the genetic algorithm, a second run of this variation was performed and found another incumbent solution (see Fig. 8). Fig. 9 shows the progress of two trials of the hybrid algorithm, i.e., the upper bounds provided by the subproblems.

Variation of Benders' algorithm	Incumbent total cost	Fixed costs	% fixed costs	# plants open
BD-Opt	5398	2619	48.5%	11
BD-Subopt	5983	1710	28.6%	7
Hybrid BD/GA, *trial 1*	5303	988	18.6%	5
Hybrid BD/GA, *trial 2*	5491	1856	33.8%	8

Table 1. Summary of results of variations of Benders' algorithm

As well, Table 1 summarizes the results obtained by these three variations of Benders' algorithm (terminated after 50 subproblems have been solved). Remarkably, in these results we observe no significant degradation of the quality of the solution when the master problem is suboptimized using a genetic algorithm, compared to optimizing the master problem and suboptimizing it by implicit enumeration.

7. Conclusion

In this chapter, we have demonstrated that Benders' decomposition algorithm for solving the capacitated plant location problem can be accelerated substantially when the master problem is solved heuristically. The hybrid Benders/GA algorithm is a variation of Benders' algorithm in which, instead of using a costly branch-and-bound method, a genetic algorithm is used to obtain "good" subproblem solutions to the master problem. The numerical example shows that the hybrid algorithm is effective to solve the capacitated plant location problem. The results imply that the hybrid algorithm is much more practical when only near-optimal solutions are required. Future work could extend the proposed algorithm to other location problems.

8. Acknowledgment

This work was partially supported by the US Department of Agriculture (USDA under grant #2011-38422-30803).

9. References

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 101-111, ISBN 0-8058-0426-9, Pittsburgh, PA, USA, July, 1985.

Beasley, D.; Bull, D. R. & Martin, R. R. (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, Vol. 15, No. 4, pp. 170-181.

Benders, J. F. (1962). Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numerische Mathematik*, Vol. 4, pp. 238-252.

Biles, J. A. (2001). GenJam: evolution of a jazz improviser. In: *Creative Evolutionary Systems (The Morgan Kaufmann Series in Artificial Intelligence)*, P. J. Bentley & D. W. Corne (Ed.), pp. 165-188, Morgan Kaufmann, ISBN 978-1558606739, SanFrancisco, CA, USA.

Cakir, O. (2009). Benders decomposition applied to multi-commodity, multi-mode distribution planning. *Expert Systems with Applications*, Vol. 36, pp. 8212-8217.

Chatterjee, S.; Carrera, C. and Lynch, L. (1996). Genetic algorithms and traveling salesman problems. *European journal of operational research,* Vol. 93, No. 3, pp. 490-510.

Cordeau, J.-F.; Soumis, F. & Desrosiers, J. (2000). A Benders decomposition approach for the locomotive and car assignment problem. *Transportation science*, Vol. 34, No. 2, pp. 133-149.

Cordeau, J.-F.; Soumis, F. & Desrosiers, J. (2001). Simultaneous assignment of locomotives and cars to passenger trains. *Operations research*, Vol. 49, No. 4, pp. 531-548.

Cortinhal, M. J. & Captivo, M. E. (2003). Upper and lower bounds for the single source capacitated location problem. *European journal of operational research*, Vol. 151, No. 2, pp. 333-351.

Davis, L. (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold Company, ISBN 978-0442001735, New York, USA.

Delmaire, H.; Di´az, J. A. & Ferna´ndez, E. (1999). Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR,* *Canadian Journal of Operational Research and Information Processing*, Vol. 37, pp. 194-225.

Geoffrion, A. M. & Graves, G.W. (1974). Multicommodity distribution system design by Benders decomposition. *Management science*, Vol. 20, No. 5, pp. 822-844.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, ISBN 978-0201157673, Boston, MA, USA.

Goldberg, D. E. & Lingle, R. J. (1985). Alleles, loci and the traveling salesman problem, *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 154-159, ISBN 0-8058-0426-9, Pittsburgh, PA, USA, July, 1985.

He, S.; Chaudhry, S. & Chaudhry, P. (2003). Solving a class of facility location problems using genetic algorithms. *Expert Systems*, Vol. 20, No. 2, pp. 86-91.

Heragu, S. S. & Chen, J. (1998). Optimal solution of cellular manufacturing system design: Benders' decomposition approach. *European journal of operational research*, Vol. 107, pp. 175-192.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, A Bradford Book, ISBN 978-0262581110, Cambridge, MA, USA

Holmberg, K.. & Ling, J. (1997). A Lagrangean heuristic for the facility location problem with staircase costs. *European journal of operational research*, Vol. 97, No. 1, pp. 63-74.

Kennington, J. L. & Whitler J. E. (1999). An efficient decomposition algorithm to optimize spare capacity in a telecommunications network. *INFORMS Journal on computing*, Vol. 11, No. 2, pp. 149-160.

Kershenbaum, A. (1997). When genetic algorithms work best. *INFORMS Journal on computing*, Vol. 9, No. 3, pp. 254-255.

Kim, H.; Sohn, H. & Bricker, D.L. (2011). Generation expansion planning using Benders' decomposition and generalized networks. *International Journal of Industrial Engineering*, Vol. 18, No. 1, pp. 25-39.

Kratica, J.; Tosic, D.; Filipovic, V. & Ljubic, I. (2001). Solving the simple plant location problem by genetic algorithm. *RAIRO Operations Research*, Vol. 35, pp. 127-142.

Levine, D. (1997). Genetic algorithms: a practitioner's view. *INFORMS Journal on computing*, Vol. 9, No. 3, pp. 256-259.

Lai, M.; Sohn, H.; Tseng, T. & Chiang, C. (2010). A Hybrid Algorithm for Capacitated Plant Location Problems. *Expert Systems with Applications*, Vol. 37, pp. 8599-8605.

Lai, M. & Sohn, H. A Hybrid Algorithm for Vehicle Routing Problems. (2011) working paper.

Liepins, G. E. & Hilliard, M. R. (1989). Genetic algorithms: foundations and applications. *Annals of operations research*, Vol. 21, pp. 31-58.

Michalewicz, Z. (1998). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, ISBN 978-3540606765, New York, USA

Oliver, I. M.; Smith, D. J. & Holland, J. R. (1987). A study of permutation crossover operators on the traveling salesman problem. *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 224-230, ISBN 0-8058-0158-8, Hillsdale, NJ, USA, July, 1985.

Randazzo, C.; Luna, H. & Mahey, P. (2001). Benders decomposition for local access network design with two technologies. *Discrete mathematics and theoretical computer science*, Vol. 4, pp. 235-246.

Reeves, R. C. (1997). Genetic algorithms for the operations researcher. *INFORMS Journal on computing*, Vol. 9, No. 3, pp. 231-250.

Rolland, E.; Schilling, D. A. & Current, J. R. (1996). An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research*, Vol. 96, pp. 329-342.

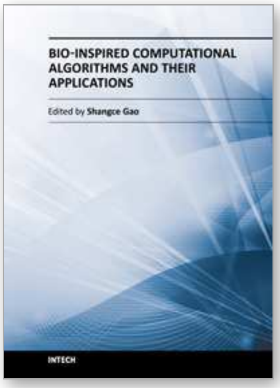
Salkin, H. M.; Mathur, K. & Haas, R. (1989). *Foundations of Integer Programming*, Appleton & Lange, ISBN 978-0130550392, New York, USA

Uno, T.; Hanaoka, S. & Sakawa, M. (2005). An application of genetic algorithm for multi-dimensional competitive facility location problem. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 3276-3280, ISBN 0-7803-9298-1, Big Island, Hawaii, USA, October 12, 2005

Van Roy, T. J. (1986). A cross decomposition algorithm for capacitated facility location. *Operations Research*, Vol. 34, pp. 145-163.

Van Roy, T. J. & Erlenkotter, D. (1982). Dual-based procedure for dynamic facility location. *Management Science*, Vol. 28, pp. 1091-1105.

Whitley, D. (1989). The GENITOR algorithm and selective pressure: Why rank-based allocation of reproductive trials is best. Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 116-121, ISBN 1-55860-066-3, Fairfax, Virginia, USA, June 1989.


BIO-INSPIRED COMPUTATIONAL ALGORITHMS AND THEIR APPLICATIONS
Edited by Shangce Gao
INTECH

Bio-Inspired Computational Algorithms and Their Applications
Edited by Dr. Shangce Gao

ISBN 978-953-51-0214-4
Hard cover, 420 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012