2012 International Conference on Information and Computer Networks (ICICN 2012)
IPCSIT vol. 27 (2012) © (2012) IACSIT Press, Singapore

A Shared Incumbent Environment for the Minimum Power Broadcasting Problem in Wireless Networks

N.E. Toklu, R. Montemanni[*], G. Di Caro and L.M. Gambardella

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Manno, Switzerland

Abstract. In this study, we consider the minimum power broadcasting problem in wireless actuator networks. We attack the problem with a method - the shared incumbent environment - that executes two algorithms in parallel: a mathematical programming approach and a simulated annealing approach. According to the shared incumbent environment paradigm, when an incumbent solution is found by one method, the other method is notified and profits from the information received. Experimental results show that the shared incumbent environment lead to results which are better than those of the two algorithms combined in it taken singularly.

Keywords: Shared incumbent environment, mixed integer linear programming, simulated annealing, matheuristics, minimum power broadcast, wireless networks.

1. Introduction

Wireless actuator networks establish communication by using devices called terminals that use omni-directional antennea to send and receive radio signals. The same data can be sent to multiple terminals at the same time, as long as they are within the coverage area of the sender terminal (this is called wireless multicast advantage [1]). Terminals which are outside the coverage area of the sender terminal can still receive the data with the help of intermediate terminals acting as routers [1, 2].

Among the applications of wireless networks, an interesting one is that consisting in commanding from remote the actuators, which are at locations difficult to be reached by people [3]. In such applications, the command is generated in a source terminal and sent to the wireless terminals attached to the actuators.

The Minimum Power Broadcast Problem (MPBP) is faced because of the fact that the terminals usually depend on small mobile batteries. This means, there is limited power available for the network. As the coverage area of a terminal increases, the power usage also increases. Therefore, finding a topology where the coverage areas are minimized would decrease the power usage and ensure a longer life span for the network. Thus, MPBP is the problem of finding a topology in which all terminals can receive data from the source terminal, with the total transmission power is minimized [1].

Many different approaches were previously proposed for solving MPBP. Among them are a polynomial-time heuristic called broadcast incremental power algorithm [1] and different approaches based on mixed integer linear programming (MILP) [4, 5]. A *matheuristic* [6] approach for MPBP was also presented, where the algorithm was a combination of linear programming and a quantum inspired evolutionary algorithm [2].

Recently, a matheuristic called shared incumbent environment, which executes a linear programming solver and an ant system meta-heuristic search in parallel, was proposed for sequential ordering problems [7]. According to the shared incumbent environment, the MILP and the meta-heuristic search inform each other

[*] Corresponding author. Tel.: +41 58 666 666 7; fax: +41 58 666 666 1.
E-mail address: roberto@idsia.ch.

when they improve the best solution retrieved since the beginning of the computation. The purpose of the shared incumbent environment is to quickly find solutions with the help of the meta-heuristic and inform the MILP solver, so that, knowing that better solutions already exist, the solver does not explore unpromising areas of the solution space and reaches to the useful solutions more quickly. On the other hand, new improving solutions retrieved by the MILP solver are used by the meta-heuristic to exit from situations of premature convergence/stagnation.

In this study, we aim to obtain improved lower and upper bound for large MPBP instances under limited computation times, by implementing a shared incumbent environment, where the solver works in parallel with a simulated annealing meta-heuristic algorithm.

This paper is organized as follows. In section 2 the problem definition is provided via a MILP model. Section 3 explains a simulated annealing approach for the problem. In section 4 introduces the details of the shared incumbent environment for this problem. Experimental results are shown in section 5. Finally, the conclusions are drawn in section 6.

2. A mixed integer linear programming model

The MPBP consists in setting the transmission power of each terminal to create such a topology that the source terminal will be able to transmit to all other terminals directly or with the help of other terminals acting as routers, with the total transmission power minimized.

To formulate the MPBP, let us have graph $G=(V,A)$, where V is the set of terminals and A is the set of arcs which represent connections between pairs of terminals. An arc consisting of the two terminals $i, j \in V$ is denoted as (i,j). The source terminal is represented by $s \in V$. The transmission power required for terminal i to include terminal j within its coverage area is denoted by r_{ij}. We also set the power requirement for terminal i to reach itself as $r_{ii} =0$. Now, let us have an array v_i for each terminal i. The array v^i stores the indices of all terminals, sorted in a non-decreasing order according to the power requirement they impose on terminal i. We call v^i_k as the k-th destination of terminal i.

The problem can be expressed as a flow-based MILP formulation [2]. In the formulation, x_{ik} represents the binary decision whether terminal i should have its k-th destination as the farthest reaching point for its coverage area or not, and y_{ij} represents the flow in arc (i,j). The assumption behind the formulation is that an arborescence is contained in each feasible connected topology.

$$\text{minimize} \quad \sum_{i \in V} \sum_{j=1}^{|V|} r_{i,v_k^i} x_{ik} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{|V|} x_{ik} = 1 \qquad \forall i \in V \tag{2}$$

$$(|V|-1) \sum_{k=l\,:\,v_l^i=j}^{|V|} x_{ik} \geq y_{ij} \qquad \forall (i,j) \in A \tag{3}$$

$$\sum_{j \in V} y_{ji} - \sum_{j \in V} y_{ij} = \begin{cases} 1-|V| & \text{if } i = s \\ 1 & \text{otherwise} \end{cases} \qquad \forall i \in V \tag{4}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in V, \forall k \in \{1,2,\ldots,|V|\} \tag{5}$$

$$0 \leq y_{ij} \leq |V| \qquad \forall (i,j) \in A \tag{6}$$

According to the constraints (2), only one farthest reaching point can be selected for each terminal i. This means, one terminal can have only one coverage area. Constraints (3) connect x variables and y variables. According to (4), the network flow represented by y_{ij} must form an arborescence. The domains of x and y variables are specified in constraints (5) and (6), respectively.

3. A simulated annealing approach

We now introduce a simulated annealing approach which was proposed for finding suboptimal solutions for MPBP quickly [8]. Simulated annealing [9] (SA) is a meta-heuristic approach which simulates the process of annealing in metallurgy. This process involves heating, which triggers the atoms to change their initial positions and slow cooling, which slowly decreases the chances for the atoms to change to a worse configuration, turning the process into a controlled local search. Into the simulated annealing algorithm, the temperature is reflected as a variable which effects the chance of accepting a candidate solution vector with a worse quality than the current solution vector.

The simulated annealing approach we use here starts by applying the broadcast incremental power and the sweep algorithms [1]. The solution of the sweep becomes both the initial current solution and the initial best solution. Then the simulated annealing process starts with temperature $t=t_{init}$, where t_{init} is a parameter regulating the initial temperature. At each iteration, a candidate solution, which is obtained by modifying the current solution, is generated. The modification is done as follows: A terminal i is randomly selected and its coverage area is decreased in such a way that if terminal i reaches its k-th destination before the modification, it reaches its $(k-1)$-th destination after this step; The modified solution is checked. If the network is still connected, the modification is finished. Otherwise, the modification goes on by selecting a terminal $j \leq i$, increasing its coverage area and fixing the disconnectivity. There are two alternative behaviors for the selection of terminal j: with a probability p_r, terminal j is selected in such a way that fixing the disconnectivity will have the least addition of transmission power; with a probability $1-p_r$, terminal j is selected randomly.

If the candidate solution has a lesser cost (total transmission power), it is accepted, which the new current solution becomes the candidate solution. Otherwise, the acceptance occurs with probability calculated as $e^{(cost(candidate)-cost(current))/t}$. During the execution, the temperature t is decreased at every C_t iterations where the best solution was not improved. The decreasing of the temperature is done by multiplying it by a factor $\alpha < 1$. The process of modifying the current solution and probabilistically accepting the modified candidate as the new current solution is repeated until $t \geq t_{min}$, where *t_{min}* is the temperature threshold. As a result of previous experiments [8], the following parameter values were found to be effective for this algorithm $p_r = 0.2, t_{init} = 0.2, C_t = 30000, \alpha = 0.9, t_{min} = 0.1$.

4. A shared incumbent environment

A MILP solver represents the problem space as a tree, where each node is a solution subspace. When an integer variable is found to be fractional after a linear programming optimization procedure, the current node is divided into branches, where the fractional variable is forced to be rounded up and down to become an integer. Such expansion of the tree takes exponential time. To decrease the time requirement for solving the problem, the solver applies pruning: stopping the expansion of the nodes that are worst in quality than the current known best solution.

The benefit of a shared incumbent environment is that it incorporates a meta-heuristic algorithm which finds useful heuristic solutions very quickly and informs the solver, so that the solver is aware of good feasible solutions in very early stages and applies pruning more efficiently. This leads the solver to promising solution subspaces more quickly. In addition to this, MILP solver can also improve the shared best solution and push the meta-heuristic out of premature convergence situations by informing it.

The implementation of this approach was done in a multi-threaded fashion: a MILP solver thread and a simulated annealing thread. The simulated annealing thread is run repeatedly until a given maximum computation time is reached, or the MILP thread has found a proven optimal solution. When the minimum temperature is reached, the simulated annealing restarts the search with the initial temperature and a different random seed.

5. Experimental results

To evaluate the performance of the shared incumbent environment, three alternative methods were considered for solving some random problem instances: *MILP*, a MILP solver, allowed to use two threads in parallel; *SA*, two simulated annealing threads working in parallel, sharing the best solutions; *SIE*, one thread dedicated to the MILP solver and once thread to the simulated annealing algorithm, working in parallel and sharing the best solutions.

The algorithms were all implemented in C++ and IBM ILOG CPLEX 12.3 (http://www.cplex.com) was used as the MILP solver, and a maximum computation time of 3000 seconds is allowed for each combination instance/method. All tests were run on a computer with Intel Core 2 Duo 2.66 GHz processor and 4 GB RAM.

Each problem instance was generated by laying terminals randomly on a two dimensional area. On the two dimensional area, d_{ij} being the euclidean distance between the terminals *i* and *j*, the power requirement value r_{ij} was calculated as d_{ij}^2, for each arc *(i,j)*. A random terminal was selected as the source *s*.

Let us start the evaluation by analysing the results for a single randomly generated instance with 180 terminals, which well represents the usual behavior of the methods considered. Figure 1 shows the upper and lower bounds obtained over time by *MILP*, *SA* and *SIE*. It can be seen that the upper bound (solution cost) found by *MILP* is over 20 000 000; *SA* was able to find a much more useful solution very quickly, but could not further improve the solution, being stuck already in the first minute over solution cost 310 000. *SIE* was finally able to find a feasible solution very quickly, like *SA*, but also over time it kept improving the solution and provided better results than *MILP* and *SA*, with a solution cost near 300 000 at the end. It is also interesting to compare the lower bounds provided by *MILP* and *SIE* over time. It can be seen that a lower bound was provided earlier by *SIE* and it kept increasing, which means the branch and bound implementation of the solver can approximate to desired solutions faster.

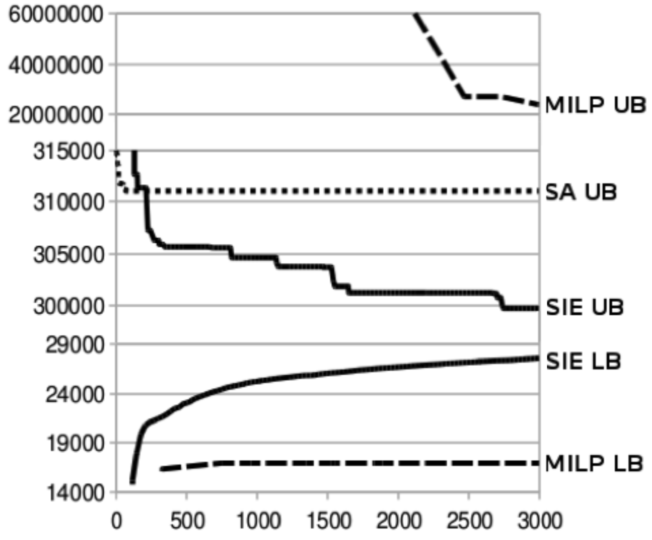

Figure 1. Upper bounds (solution costs, UB) and lower bounds (LB) found by MILP, SA and SIE on a 180-terminal example instance, drawn versus time in seconds. Note that the y-axis contains gaps to be able to cover all the progress made by the three methods

To make a general evaluation of *SIE*, 10 instances for each number of terminals were generated, number of terminals being 130, 140, 145, 150, 160, 170, 180, 190 and 200. Then, each instance was solved by the approaches *MILP*, *SA* and *SIE*. The results are shown in table 1. In the table, each row represents the average of 10 instances (and over how many of these 10 instances *SIE* was able to improve is given in parentheses), the column group "Improvement over MILP" represents the improvements made by *SIE* in comparison to *MILP*, and the column group "Improvement over SA" represents the improvements made by *SIE* in comparison to *SA*. Improvements for the upper bounds are calculated as *1-a/b*, where *a* is the upper bound of the solution of *SIE* and *b* is the upper bound of the solution of the other approach. Improvements for the lower bounds are calculated as *a/b-1*, where *a* is the lower bound of the solution of *SIE* and *b* is the lower bound of the solution of the other approach. The row "overall" shows the average and total number of instances where *SIE* was able to improve calculated over all instances.

Table 1. Results generated by each approach

Number of terminals	Improvement over MILP (%)		Improvement over SA (%)
	Lower bound	Upper bound	Upper bound
130	-79.23 (0/10)	-6.65 (0/10)	3.32 (10/10)
140	-78.02 (0/10)	-0.86 (1/10)	11.43 (10/10)
145	4.77 (6/10)	68.77 (7/10)	7.34 (9/10)
150	35.35 (7/10)	80.12 (9/10)	9.86 (10/10)
160	25.56 (7/10)	63.95 (9/10)	14.75 (10/10)
170	72.29 (9/10)	89.20 (9/10)	26.50 (10/10)
180	36.95 (9/10)	89.40 (9/10)	2.89 (10/10)
190	24.95 (9/10)	89.72 (9/10)	10.85 (10/10)
200	21.92 (9/10)	89.67 (9/10)	14.90 (9/10)
Overall	7.17 (56/90)	62.59 (62/90)	11.32 (88/90)

In table 1, it can be seen that, for both upper bounds (i.e. solution costs) and lower bounds, the shared incumbent environment *SIE* provides considerable improvements compared to *MILP*, except for instances with the number of terminals 130 and 140. This shows that the shared incumbent environment is effective on large instances to provide non-trivial lower and upper bounds. On top of that, the shared incumbent environment also provides improvements over the heuristic solution costs of the simulated annealing approach most of the time, which shows that the shared incumbent environment also results in a competitive heuristic method.

6. Conclusions

A shared incumbent environment, which executes a mathematical programming solver and a meta-heuristic search in parallel and makes them share their best solutions, was considered for the minimum power broadcasting problem in wireless actuator networks. Experimental results show that the shared incumbent environment finds better solutions in limited time for the minimum power broadcasting problem. Therefore, this approach can be seen as an effective heuristic for the large instances of this problem, and at the same time as a tool to provide better lower bounds than a standard MILP solver.

7. Acknowledgements

N.E. Toklu was supported by the Swiss National Science Foundation through project 200021-119720.

8. References

[1] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *Mobile networks and applications* 2002, 7(6): 481–492.

[2] T.S. Rappaport. *Wireless communications: principles and practices*. Prentice Hall, Englewood Cliffs, 1996.

[3] R. Montemanni, and P. Mahdabi. A linear programming-based evolutionary algorithm for the mini- mum power broadcast problem in wireless sensor network. *Journal of Mathematical Modelling and Algorithms* 2011, 2(10): 145–162.

[4] J. Leino. Optimal multicast routing in ad hoc networks. Tech. report, Helsinki University of Technology, 2002.

[5] S. Guo, and S. Yang. A constraint formulation for minimum energy multicast routing in wireless multi-hop networks. *Wireless Networks Journal* 2006, 12: 23–32.

[6] V. Maniezzo, T. Stuetzle, and S. Voß, eds. *Hybridizing Metaheuristics and Mathematical Programming*. Springer, 2009.

[7] M. Mojana, R. Montemanni, G. Di Caro and L. M. Gambardella. An algorithm combining linear programming and an ant system for the sequential ordering problem. *Proceedings of ATAI 2011*. 2011, pp. 80–85.

[8] R. Montemanni, L.M. Gambardella, and A.K. Das. The minimum power broadcast problem in wireless networks: a simulated annealing approach. *Proceedings of IEEE WCNC*. 2005.

[9] S. Kirkpatrick, C.G. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science* 1983, 220(4598): 671–680.