GRASP WITH PATH-RELINKING FOR THE COOPERATIVE COMMUNICATION PROBLEM ON AD-HOC NETWORKS

C.W. COMMANDER*, P. FESTA†, C.A.S. OLIVEIRA‡, P.M. PARDALOS§, M.G.C. RESENDE¶, AND M. TSITSELIS‖

Abstract. Ad-hoc networks are a new paradigm for communications systems in which wireless nodes can freely connect to each other without the need of a pre-specified structure. Difficult combinatorial optimization problems are associated with the design and operation of these networks. In this paper, we consider the problem of maximizing the connection time between a set of mobile agents in an ad-hoc network. Given a network and a set of wireless agents with starting nodes and target nodes, the objective is to find a set of trajectories for the agents that maximizes connectivity during their operation. This problem, referred to as the COOPERATIVE COMMUNICATION ON AD-HOC NETWORKS is known to be NP-hard. We look for heuristic algorithms that are able to efficiently compute high quality solutions to instances of large size. We propose the use of a greedy randomized adaptive search procedure (GRASP) to compute solutions for this problem. The GRASP is enhanced by applying a path-relinking intensification procedure. Extensive experimental results are presented, demonstrating that the proposed strategy provides near optimal solutions for the 900 instances tested.

Key words. Cooperative communication, ad hoc networks, metaheuristics, GRASP, path-relinking.

AMS subject classifications. 68T20, 90B18, 90B06.

1. Introduction.

Cooperative control has received a significant amount of attention by researchers [6, 7, 21, 25]. In many situations, multiple "agents" collaborate to achieve a shared goal. Cooperation between the agents is important to improve the efficiency and effectiveness by which their goal is reached. In wireless networks, groups of agents are often employed to perform a number of cooperative tasks, including the synchronization of information among a set of users and the accomplishment of missions in remote areas. In such situations, it is useful to maintain collaboration among the agents performing the cooperative tasks to maximize the probability of success.

There are many applications of this described system. These include situations where communication in a region is required, but no topologically fixed transmission system exists. Specific examples include emergency/rescue operations, disaster relief, and battlefield operations [27]. In each of these examples the goals and objectives are fixed in advance and communication is important for the attainment of these goals. The current technologies used in these types of applications allow improved communication systems that rely on ad-hoc wireless protocols. However, it is computationally difficult to decide how to maintain communication for the maximum possible time when faced with the inherent restrictions of wireless systems.

Ad-hoc networks are composed of a set of wireless units that can communicate directly, without the use of a pre-established server infrastructure. In this respect, ad-hoc systems are

* Air Force Research Laboratory, Munitions Directorate, and Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, FL USA (clayton.commander@eglin.af.mil)

† Dept. of Mathematics and Applications, University of Napoli FEDERICO II, Napoli, Italy (paola.festa@unina.it).

‡ Dept. of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK USA. (coliv@okstate.edu).

§ Center for Applied Optimization, Dept. of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA. (pardalos@ufl.edu).

¶ Internet and Network Systems Research Center, AT&T Labs Research, Florham Park, NJ 07932 USA. (mgcr@research.att.com).

‖ Dept. of Mathematics and Applications, University of Napoli FEDERICO II, Napoli, Italy (marco.tsitselis@gmail.com).

fundamentally different from traditional cellular systems, where each user has an assigned base station which connects it to the wired telephony system. In an ad-hoc network, each client has the capacity of accessing network nodes that are within its reach. This connectivity model allows for the existence of networks without a predefined topology, reaching a different state every time a node changes its position. Due to this inherent variability, ad-hoc networks present serious challenges for the design of efficient protocols. The optimization of activities in such networks is subject to the lack of global information. Furthermore, optimal solutions may be short-lived, due to the dynamics of the agents' positions and connectivity status.

In this paper, we study a problem involving the coordination of wireless users involved in a mission of tasks that requires each user to go from an initial location to a target location. The problem consists in maximizing the amount of connectivity among the set of users, subject to constraints on the maximum distance traveled by each user, as well as restrictions on what types of movements can be performed. This problem is referred to as the COOPERATIVE COMMUNICATION PROBLEM ON MOBILE AD-HOC NETWORKS (CCPM) [13, 27].

The CCPM is a path-planning problem and has numerous military applications. For example, suppose a force is planning a reconnaissance mission over a battle space and will be using a set of so-called unmanned aerial vehicles (UAVs) for this task. It may also be required that the UAVs maintain connectivity with each other in order to share information that is being gathered. Before the mission begins, the mission planner could solve an instance of the CCPM using the knowledge of the battle space to determine a set of optimal paths for the UAVs. In the CCPM, a set of paths is said to be optimal if the communication between all pairs of agents is maximized, subject to the agents arriving at their respective destinations within the specified time horizon. After briefly reviewing some related work from the literature, we will provide a mathematical model of the CCPM and describe an effective algorithm for solving large instances.

1.1. Related work. Ad-hoc networks represent an extremely active area of research [30]. Several problems related to routing, power control, and accurate position update have been studied in the last few years [26]. In terms of routing, one of the main problems in ad-hoc networks is the computation of a network backbone. The objective is to find a subset of nodes with a small number of elements that can be used to send routing information. Such a structure is useful to simplify the management tasks required by a routing protocol. The backbone computation problem can be modeled as a CONNECTED DOMINATING SET (CDS) problem. Here, the objective is to find a set of minimum size forming a connected backbone, with the additional property that each network client can directly reach this set. The CDS problem, which can be modeled using unit graphs, has several approximation algorithms [4, 5, 9], all of which are based on approximation properties of the MAXIMUM INDEPENDENT SET problem on planar unit graphs [3]. The use of optimization techniques to maximize connectivity in ad-hoc systems was studied by Oliveira and Pardalos in [27]. Until now, only local search heuristics have been applied to the CCPM [13].

Given the difficulty of solving the CCPM exactly, we are interested in studying the application of metaheuristic methods for the problem. In particular, we propose a greedy randomized adaptive search procedure (GRASP) for the CCPM. In addition, path-relinking [20] is applied to increase the effectiveness of GRASP. We show that the resulting algorithm is able to efficiently provide high quality solutions for large scale instances of the problem.

1.2. Problem definition. An ad-hoc network is composed of a set of autonomous clients that can connect to each other using their own wireless capabilities. This includes using scarce resources such as computational processing and battery power. We model this situation using a special type of graph called a unit graph [10]. A unit graph is a planar graph $G = (V, E)$ with associated positions for each node $v \in V$. In a unit graph an edge occurs between nodes

$v, w \in V$ if $dist(v, w) \leq 1$, where $dist : V \times V \rightarrow \mathbb{R}$ is the Euclidean distance. Unit graphs occur as a natural model in ad-hoc networks and are used in this paper to represent the set of configurations of nodes that share connections.

Let $G = (V, E)$ be a graph representing the set of valid positions for network clients. Suppose this graph has the property that each node is connected only to nodes that can be reached in one unit of time. Therefore, the graph can be used to represent all possible trajectories of a node. Each such trajectory is a path $\mathcal{P} = \{v_1, \ldots, v_k\}$ where $v_1 \in V$ is the starting node and $v_k \in V$ is the destination node. We also consider a set U of wireless units, a set of initial positions $S \subseteq V$, such that $|S| = |U|$, and a set of destinations $D \subseteq V$, with $|D| = |U|$. We assume that, to perform its task, each wireless unit $u_i \in U$ starts from a position $s_i \in S$, and ends at position $d_i \in D$. We are given a time limit T such that all units must reach their destinations by time T.

The trajectory of users in the system occurs as follows. For $v \in V$, let $\eta(v) \subseteq V$ be the set of all nodes in the neighborhood of v, i.e. the set of nodes $w \in V$ such that $(v, w) \in E$. Let $p_t : U \rightarrow V$ be a function returning the position of a wireless unit at time t. Then, at each time step t, a wireless unit $u \in U$ can stay in its previous position p_{t-1} or move to one of its neighboring nodes $v \in \eta(p_{t-1})$. More formally, at time step t, position $p_t(u) \in p_{t-1}(u) \cup \eta(p_{t-1})$. Let $\{\mathcal{P}_i\}_{i=1}^{|U|}$ be the set of paths representing the trajectories of the wireless units in U (obviously, the first node of $\mathcal{P}_i$ is s_i and its last node is d_i). Let L_i, for $i \in \{1, \ldots, |V|\}$, be a threshold on the total cost of path $\mathcal{P}_i$. Define $\omega : V \times V \rightarrow \mathbb{R}$ to be the weight of the edge between two nodes. Then, for each path $\mathcal{P}_i = \{v_1, \ldots, v_{n_i}\}$, we require that

$$\sum_{i=2}^{n_i} \omega(v_{i-1}, v_i) \leq L_i. \tag{1.1}$$

With this, we can define the COOPERATIVE COMMUNICATION PROBLEM ON MOBILE AD-HOC NETWORKS (CCPM) as follows. Given a network $G = (V, E)$, a set U of wireless agents, a set $S \subseteq V$ of starting nodes, a set $D \subseteq V$ of destination nodes, a maximum travel time T, and distance thresholds L_i, for $i \in \{1, \ldots, |V|\}$, a feasible solution for the CCPM consists of a set of positions $p_t(u)$, for $t \in \{1, \ldots, T\}$ and $u \in U$, such that the initial position satisfies $p_1(u) = s(u)$ for $u \in U$, the final position is $p_T(u) = d(u)$, a set positions given by $p_t(u) \in p_{t-1}(u) \cup \eta(p_{t-1}),\ \forall\, t \in \{2, \ldots, T-1\}$, and the inequalities in (1.1) are satisfied. The objective is to maximize the connectivity among the users in U over all time steps, which is measured by

$$\max \sum_{t=1}^{T} \sum_{\substack{u,v \in U \\ u \neq v}} c(p_t(u), p_t(v)),$$

where $c : V \times V \rightarrow \{0, 1\}$ is a function returning 1 if and only if $dist(p(u), p(v)) \leq 1$. The reader is referred to the paper by Oliveira and Pardalos [27] for additional integer programming formulations in which other objectives are considered and discussed. We finish this section by providing two results related to the computational complexity of the problem.

THEOREM 1. *Finding an optimal solution for an instance of the* COOPERATIVE COMMUNICATION PROBLEM ON MOBILE AD-HOC NETWORKS *is NP-hard.* This result, due to Oliveira and Pardalos [27], follows by a reduction from MAXIMUM 3-SAT [18]. We now extend this result in the following theorem.

THEOREM 2. *Consider an instance of the* CCPM, *with T as the time-horizon. Finding an optimal solution at each time-step $t \in [1, T]$ is NP-hard.*

```
procedure GRASP(MaxIter, RandomSeed)
1   X* ← ∅
2   for i = 1 to MaxIter do
3     X ← ConstructionSolution(G, g, X)
4     X ← LocalSearch(X, MaxIterLS)
5     if f(X) ≥ f(X*) then
6       X* ← X
7     end
8   end
9   return X*
end procedure GRASP
```

Fig. 2.1: GRASP for maximization.

Proof. We will show this result by reducing CLIQUE to CCPM at an arbitrary time-step. Recall that the CLIQUE problem is as follows. Given a graph $G = (V, E)$ and an integer $J \leq |V|$, does G contain a clique, or complete subgraph, of size J or more [18]?

Consider an instance of CCPM at any time step t. An optimal solution is one in which all the agents are pairwise connected. Thus, for n agents the number of connections in an optimal solution is $n(n-1)/2$. Notice that this is equivalent to finding clique on n nodes of the graph. Therefore, given an instance of CLIQUE, by letting $J = n$, we have the result. Thus there is a bijection between optimal configurations of agents and cliques in the graph. □

COROLLARY 1. *For any instance of* CCPM*, an upper bound on the optimal solution is given by*

$$T \cdot \frac{u(u-1)}{2}, \tag{1.2}$$

where T is the time horizon and $u = |U|$ is the number of agents.

Proof. This proof follows directly from Theorem 2. If all u agents communicate at a given time, then they form a clique on u nodes. The clique will contain $u(u-1)/2$ vertices representing the communication links. If the agents maintain the clique formation over all time steps, then the number of communication connections will be

$$T \cdot \frac{u(u-1)}{2} \tag{1.3}$$

and the lemma is proved. □

2. GRASP for CCPM. A greedy randomized adaptive search procedure (GRASP) [14] is a multi-start metaheuristic that has been used widely to provide solutions for several difficult combinatorial optimization problems [17], including SATISFIABILITY [29], JOB SHOP SCHEDULING [2], VEHICLE ROUTING [8], and QUADRATIC ASSIGNMENT [24, 28].

GRASP is a two-phase procedure which generates solutions through the controlled use of random sampling, greedy selection, and local search. For a given problem Π, let F be the set of feasible solutions for Π. Each solution $X \in F$ is composed of k discrete components $a_1, \ldots, a_k$. GRASP constructs a sequence $\{X_1, X_2, \ldots\}$ of solutions for Π, such that each $X_i \in F$. The algorithm returns the best solution found after all iterations. The GRASP procedure is described in the pseudo-code shown in Figure 2.1. The *construction phase* receives as parameters an instance G of the problem and a ranking function $g : A(X) \rightarrow \mathbb{R}$,

where $A(X)$ is the domain of feasible components $a_1, \ldots, a_k$ for a partial solution X. The construction phase begins with an empty partial solution X. Assuming that $|A(X)| = k$, the algorithm creates a so-called *Restricted Candidate List* (RCL) containing the α % best ranked components in $A(X)$. The parameter $\alpha \in [0, 1] \cdot 100$ can be fixed by the user or chosen randomly. An element $x \in$ RCL is uniformly chosen and the current partial solution is augmented to include x. This procedure is repeated until the solution is feasible, i.e., until $X \in F$.

The *intensification phase* consists of a local search which could include gradient descent methods, swap based methods, and even the use of other metaheuristics. Our local search consists of an implementation of a hill-climbing procedure. Given a solution $X \in F$, let $N(X)$ be the set of solutions that can found from X by changing one of the components $a \in X$. Then, $N(X)$ is called the neighborhood of X. The improvement algorithm consists in finding, at each step, the element X^* such that

$$X^* = \arg \max_{X' \in N(X)} f(X'),$$

where $f : F \to \mathbb{R}$ is the objective function of the problem. At the end of each step we make $X^* \leftarrow X$ if $f(X) > f(X^*)$. The algorithm will eventually achieve a local optimum, in which case the solution X^* is such that $f(X^*) \geq f(X')$ for all $X' \in N(X^*)$. X^* is returned as the best solution from the iteration and the best solution from all iterations is returned as the overall GRASP solution.

We discuss in this section how the above algorithm can be specialized to solve the CCPM. In the following subsection, we describe an algorithm for the GRASP construction phase that provides initial solutions for instances of the CCPM problem. Then, in Subsection 2.2 we provide a local search algorithm for the improvement phase.

2.1. Construction Phase. The first task in a GRASP algorithm is to build good feasible solutions in terms of a given objective function. To do this, we need to specify the set A, the greedy function g, and, for $X \in F$, the neighborhood $N(X)$. The components of each solution X are feasible moves of a member of the ad-hoc network from a node v to a node $w \in N(v) \cup \{v\}$. For an agent $u_i \in U$ located at node v in the graph, $\mathcal{P}_i(v)$ represents a shortest path from the current node v to the destination for agent u_i, namely node d_i.

The complete solution is constructed according to the following procedure outlined in the pseudo-code shown in Figure 2.2. In the pseudo-code, a_h refers to the current location of an agent. First, the solution which is initially empty is augmented to include the starting locations for all agents. Then, the time variable t is initialized to 1, and in line 6 an agent $u_i \in U$ is selected at random and routed from its along a shortest path $\mathcal{P}_i(s_i)$ from its source node s_i to its destination node d_i. If the total distance of $\mathcal{P}_i(s_i)$ is greater than L_i, then the instance is clearly infeasible and the algorithm ends. Otherwise, the procedure continues and the remaining agents are scheduled in the loop beginning at line 8. The procedure considers each feasible move (q, w, u) before scheduling an agent. A feasible move connects the final node q of a sub-path P_u, for $u \in U$, to another node w, such that the shortest path from w to d_u has distance at most $L_u - \sum_{e \in P_u} dist(e)$. The set of all feasible moves in a solution is defined as $A(X)$.

The loop in lines 12–14 ensures that a node currently at its destination remains there. Likewise, the loop in lines 15–17 schedules an agent h on a shortest path $\mathcal{P}_h(a_h)$ from its current position a_h to d_h if the maximum allowed travel time for agent h is equal to the $|\mathcal{P}_h(a_h)|$. In lines 19–21, the set $L \subseteq A(X)$ is formed and consists of all feasible moves for agents not yet scheduled. Then, in line 25, the greedy function g returns for each move $k \in L$ the number of additional connections created by that move. As described above, the

```
procedure ConstructionSolution(G, g, X)
1   X ← ∅
2   for i = 1 to k do
3     X ← X ∪ {s_i}
4   end
5   t ← 1
6   Select randomly u_i ∈ U and route u_i on shortest-path P_i(s_i)
7   X ← X ∪ {P_i(s_i)}
8   while t < T and ∃ u_i ∈ U \ X do
9     L ← ∅
10    RCL ← ∅
11    for u_h ∈ U \ X do
12      if a_h = d_h do
13        X ← X ∪ {d_h}
14      end
15      if dist(a_h, d_h) = L_h − Σ_{e∈P_h} dist(e) do
16        Route u_h on shortest path P_h(a_h)
17        X ← X ∪ {P_h(a_h)}
18      end
19      while ∃ (l_q, l_w, u_h) do
20        L ← {(l_q, l_w, u_h)}
21      end
22    end
23    α ← RealRandomNumber(0, 1)
24    δ ← (MaxContribute − (α ∗ (MaxContribute − MinContribute)))
25    for all (l_i, l_j, u_e) such that g((l_i, l_j, u_e)) ≥ δ do
26      RCL ← RCL ∪ {(l_i, l_j, u_e)}
27    end
28    (l_i, l_j, u_e) ← Get randomly from RCL
29    Add (l_i, l_j, u_e) into the path of agent u_e in the solution
30    t ← t + 1
31  end
32  return(X)
end procedure ConstructionSolution
```

Fig. 2.2: Greedy randomized constructor for CCPM.

construction procedure will rank the elements of L according to g. The α % best-ranked elements are then added to the RCL and in lines 28–29, a move is selected at random and added to the solution. This is repeated until a complete feasible solution for the problem is obtained.

2.2. Improvement phase. In the local search phase, GRASP attempts to improve the solution built in the construction phase. As mentioned above, we use a hill-climbing procedure where the objective is to improve the solution as much as possible until a local optimal solution is found as described in the pseudo-code provided in Figure 2.3.

The local search receives the construction phase solution X and a parameter `MaxIterLS` as input. In each iteration, the neighborhood $N(X)$ of X is explored in search of a solution X' such that $f(X') > f(X)$. In order to explore $N(X)$, a perturbation function is defined as follows. In the loop in lines 5–21, agents are re-routed using a greedy method similar to

```
procedure LocalSearch(X, MaxIterLS)
1   X' ← ∅
2   t ← 1
3   LastImprove ← 1
4   i ← 2
5   iter ← 0
6   while i ≠ LastImprove and iter < MaxIterLS do
7      Remove current path from s_i to d_i for agent u_i
8      while a_i ≠ d_i do
9         if dist(a_i, d_i) = L_i − Σ_{e∈P_i} dist(e) then
10           Route user u_i using its shortest path P_i(a_i)
11        else
12           BestMove ← {(a_i, l_w, u_i) | g((a_i, l_w, u_i)) > g((a_i, l_j, u_i)), ∀ (a_i, l_j, u_i)}
13        end
14        Add BestMove into the new path for u_i in solution X'
15        t ← t + 1
16     end
17     if f(X') > f(X) then
18        X ← X'
19        LastImprove ← i
20     end
21     i ← i + 1 mod k
22     iter ← iter + 1
23  end
24  return(X')
end procedure LocalSearch
```

Fig. 2.3: Local search for CCPM.

that of the construction phase. In line 6, the current construction phase path for agent u_i is removed from the solution. Then each feasible move is considered and the move which adds the greatest increase to the objective function, $BestMove$, is added to the new path for agent u_i. This is repeated for all agents until a new feasible solution $X' \in N(X) \subseteq A(X)$ is created. If $f(X') > f(X)$, then in line 17, X' is set as the new current solution. The process returns to line 5 and repeats until no agent can be re-routed according to this greedy method and improve the current solution or until some maximum number of iterations MaxIterLS are completed.

2.3. Complexity of the heuristic. The following theorems address the computational complexity of the proposed algorithm.

THEOREM 3. *The construction phase finds a feasible solution for the* CCPM *in* $\mathcal{O}(Tmu^2)$ *time, where* T *is the time horizon,* $u = |U|$*, and* $m = |E|$*.*

Proof. Notice that the **while** loop in lines 8–31 will require $T(|U| - 1)$ iterations to complete. Likewise, the loop in lines 11–22 requires $|U|$ iterations. Within the loop, the most time consuming step is the construction of the shortest path. However, this can be done using a breadth-first search in $\mathcal{O}(m)$ time [1]. Thus, we have the result. □

THEOREM 4. *The time complexity of the local search phase is* $\mathcal{O}(kTu^2m)$*, where* T *is the time horizon,* $u = |U|$*,* $m = |E|$*, and* $k =$ MaxIterLS*.*

Proof. The proof is similar to Theorem 3. Notice that the **while** loop in lines 5–22 perform local improvements according the greedy re-routing scheme. Again, the most time

```
procedure PathRelinking(x_s, E)
1    x_g ← randSelect(y ∈ E : Δ(x_s, y) > δ)
2    f* ← max{f(x_s), f(x_g)}
3    x* ← arg max{f(x_s), f(x_g)}
4    x ← x_s
5    while Δ(x_s, x_g) ≠ ∅ do
6       m* ← arg max{f(x ⊕ m) : m ∈ Δ(x, x_g)}
7       Δ(x ⊕ m*, x_g) ← Δ(x, x_g) \ {m*}
8       x ← x ⊕ m*
9       if f(x) > f* then
10         f* ← f(x)
11         x* ← x
12      end
13   end
14   return x*
end procedure PathRelinking
```

Fig. 3.1: A path-relinking subroutine adapted from [31].

consuming step is the construction of a shortest path which can be accomplished in $\mathcal{O}(m)$ time. Each improvement can require up to k iterations of the loop, and we have the proof. □

COROLLARY 2. *The overall time complexity of the proposed GRASP is* $\mathcal{O}(lTu^2m(k+1))$*, where where* T *is the time horizon,* $u = |U|$*,* $m = |E(G)|$*,* $k =$ `MaxIterLS`*, and* $l =$ `MaxIter` *is the overall number of GRASP iterations.*

Proof. The proof is immediate from Theorem 3 and Theorem 4. □

3. Path-relinking. First introduced by Glover in [19], path-relinking (PR) was used as an enhancement for tabu search heuristics. PR was first combined with GRASP by Laguna and Martí [23]. When applied to GRASP, path-relinking introduces a memory to the heuristic which usually results in improvements in solution quality. This is because in the standard GRASP framework, the multi-start nature of the heuristic does not include any long-term memory mechanism for saving traits of good solutions generated by the algorithm. Path-relinking allows GRASP to remember these traits and favor them in successive iterations. GRASP with path-relinking has been successfully applied to problems such as MAXIMUM CUT [16], QUADRATIC ASSIGNMENT [28], TDMA MESSAGE SCHEDULING [12], and originally for LINE CROSSING MINIMIZATION [23]. For a survey of GRASP with path-relinking, the reader is referred to [31].

Path-relinking works by maintaining a set of elite solutions $\mathcal{E}$, known as *guides* and examines point-to-point trajectories between a guiding solution and an incumbent solution in search of an optimum. Pseudo-code for a generic path-relinking procedure is provided in Figure 3.1. To perform path-relinking, we begin with a guiding solution $x_g \in \mathcal{E}$, and an initial starting solution x_s. The guiding solution x_g is selected at random from the pool of elite solutions $\mathcal{E}$, so long as the symmetric difference $\Delta(x_s, x_g)$ between the two solutions x_s and x_g is sufficiently large. The symmetric difference is defined as the set of pairwise exchanges needed to transform x_s in to x_g. Recall that all solutions in $\mathcal{E}$ are local optima, and we are trying to discover solutions which are not located in the neighborhoods of x_s or x_g. Therefore this constraint prevents us from applying path-relinking to solutions which are too similar to each other, and would not likely yield an improved solution [15].

At each step, the procedure examines all moves $m \in \Delta(x, x_g)$, and greedily selects the

move which results in the maximum increase in the objective of the current solution. This occurs in line 6 of the pseudo-code in which the move m^* is selected as the move which maximizes $f(x \oplus m)$, where $x \oplus m$ is the solution which results from incorporating m in to x. In line 7, the symmetric difference is updated, and if necessary the best solution is updated in lines 9-12. The procedure ends when $\Delta(x, x_g) = \emptyset$, i.e. when $x = x_g$ [31].

Path-relinking can be applied to a pure GRASP in a straightforward manner, which can be visualized in the pseudo-code of Figure 3.2. First, the set of elite solutions $\mathcal{E}$ is initialized to the empty set in line 2 and is built by including the solutions from the first `MaxElite` iterations. After a standard GRASP iteration of greedy randomized construction and local search produces a local optimal solution X, the `PathRelinking` procedure is called on line 7. For the CCPM, the elements in the symmetric difference are the agent paths which differ between the initial and guiding solutions. The value of m^* from Figure 3.1 is the path for an agent in the symmetric difference which results in the maximum increase in the total number of communications between the agents. In line 8, a function `UpdateElite` is called in which the elite pool is possibly updated. The solution returned from path-relinking is included in the elite pool if it is better than the best solution in $\mathcal{E}$ or if it worse than the best but better than the worst and is sufficiently different from all elite solutions [31]. Finally, the optimal solution is updated in lines 12 to 14 if necessary.

```
procedure GRASP+PR(MaxIter, RandomSeed)
1    X* ← ∅
2    E ← ∅
3    for i = 1 to MaxIter do
4       X ← ConstructionSolution(G, g, X)
5       X ← LocalSearch(X, MaxIterLS)
6       if |E| = MaxElite then
7          X ← PathRelinking(X, E)
8          UpdateElite(X, E)
9       else
10         E ← E ∪ {X}
11      end
12      if f(X) ≥ f(X*) then
13         X* ← X
14      end
15   end
16   return X*
end procedure GRASP+PR
```

Fig. 3.2: GRASP with path-relinking for maximization [15].

4. Computational experiments. The proposed procedure was implemented in the `C` programming language and compiled using the Microsoft® Visual C++ 6.0. It was tested on a PC equipped with a 1800MHz Intel® Pentium® 4 processor and 256 megabytes of RAM operating under the Microsoft® Windows® 2000 Professional environment.

Both the pure GRASP and the GRASP with path-relinking were tested on a set of 60 random unit graphs with varying densities 20 each having 50, 75, and 100 nodes. The radius of communication varies from 1 to 5 units (miles) in unit increments. We tested each case with three sets of mobile agents to achieve better comparisons and model real-world scenarios. Thus, in total 900 test cases were examined. The graphs were created by a generator used by

Table 4.1: Three instances with different sets of agents on 50 node graphs are given. The value in the `UBound` column was found using Corollary 1.

Instance: 50r30i1 Nodes: 50 Agents: 10 MaxTime: 10 Source: [6 10 10 3 5 7 4 2 10 6] Destination: [49 47 44 48 46 40 48 42 47 47]			
Radius	GRASP	GRASP+PR	UBound
1	291	303	450
2	365	373	450
3	412	423	450
4	443	443	450
5	449	449	450

(a)

Instance: 50r30i1 Nodes: 50 Agents: 15 MaxTime: 10 Source: [10 9 8 9 6 8 1 7 2 5 5 5 2 1 1] Destination: [49 47 44 48 46 40 48 42 47 47]			
Radius	GRASP	GRASP+PR	UBound
1	756	757	1050
2	881	909	1050
3	963	972	1050
4	1029	1029	1050
5	1050	1050	1050

(b)

Instance: 50r40i4 Nodes: 50 Agents: 25 MaxTime: 10 Source: [8 9 8 4 5 4 4 6 2 7 4 2 1 7 5 8 9 3 8 11 8 7 5 6 8] Destination: [49 48 44 48 46 42 49 40 48 49 45 46 49 45 48 44 42 41 48 43 40 49 45 49 43]			
Radius	GRASP	GRASP+PR	UBound
1	2613	2653	3000
2	2896	2918	3000
3	3000	3000	3000
4	3000	3000	3000
5	3000	3000	3000

(c)

Butenko et al. [11] on the TDMA MESSAGE SCHEDULING PROBLEM.

Since any instance of the CCPM is composed of several parameters, i.e. the number of mobile agents, their respective source and destination nodes, the radius of communication, and the maximum time horizon, each of which impacts the optimal solution for the instance, we will provide our numerical results in several sets of tables. First, we report solutions for several representative instances and provide all input parameters in order to establish an inference base for the overall experiment. Then we will summarize the overall results by providing the average solutions for each problem set[1].

In Table 4.1, we report solutions for three different instances on 50 node graphs. The `Source` vector and `Destination` vector provide the respective (s_i, d_i) pair for each agent respectively. The specific values of s_i were randomly selected from the first 20% of the nodes of the graph. Likewise, the d_i values were chosen randomly from the last 20% of nodes. This method of selection is preferred over a completely randomized design because in real-world situations such as a combat scenario, the available entry and exit points from a battle space

[1]Complete results for all experiments are at `http://www.research.att.com/~mgcr/doc/gccpmdata.pdf`.

Table 4.2: Three instances with different sets of agents on 75 node graphs are given. The value in the UBound column was found using Corollary 1.

Instance: 75r30i2 Nodes: 75 Agents: 10 MaxTime: 15 Source: [7 6 13 3 10 13 15 6 6 2] Destination: [68 68 71 73 68 68 73 70 74 62]			
Radius	GRASP	GRASP+PR	UBound
1	571	575	675
2	614	621	675
3	658	658	675
4	670	670	675
5	675	675	675

(a)

Instance: 75r40i4 Nodes: 75 Agents: 20 MaxTime: 15 Source: [11 5 7 15 3 8 9 4 6 13 14 3 8 3 10 14 11 15 9 3] Destination: [63 66 69 61 62 68 62 67 68 66 62 60 61 66 63 73 72 64 71 71]			
Radius	GRASP	GRASP+PR	UBound
1	2535	2554	2850
2	2746	2758	2850
3	2842	2842	2850
4	2850	2850	2850
5	2850	2850	2850

(b)

Instance: 75r30i1 Nodes: 75 Agents: 30 MaxTime: 15 Source: [14 15 2 8 10 4 13 3 4 5 12 6 4 9 2 3 15 8 5 12 9 3 7 7 11 3 4 1 15 4] Destination: [68 69 63 62 65 72 62 62 67 71 65 69 63 64 60 64 60 60 66 64 74 63 73 64 64 63 65 65 60 63]			
Radius	GRASP	GRASP+PR	UBound
1	4721	4870	6525
2	6002	6012	6525
3	6265	6285	6525
4	6497	6497	6525
5	6525	6525	6525

(c)

are likely to be limited. However, using a random selection from the available subset of nodes allows for more thorough testing and helps avoid unintentional biases.

The column MaxTime is the maximum time horizon T. Recall that all agents must reach their destination node by this time. The GRASP column provides the solution from GRASP after 1000 iterations and UBound is the upper bound on the solution value and was calculated by the equation in Corollary 1. Notice that as the radius value increases, the number of connections between the agents tends to converge to the value of the upper bound. Recall that the upper bound value from Corollary 1 is not an upper bound on the optimal solution for the given graph per se; it is an upper bound on the solution for the given time horizon and number of agents. Thus, the more dense the graph, the tighter the bound.

Table 4.2 presents the specific parameters and related solutions for three instances of the CCPM on 75 node graphs. On these networks, the number of agents varied from 10 to 30, and the maximum time horizon was 15. Again, we see that as the communication radius increases the solutions tend to the upper bound values. Similar results for three graphs having 100 nodes are provided in Table 4.3. For the 100 node instances, the number of agents varied from 15 to 35 and the maximum travel time was 20 units. The results for these instances also indicate that the heuristic is robust and able to provide excellent solutions for large instances.

Table 4.3: A 100 node instance with solutions with radius varying from 1 to 5 units. The value in UBound was found using Corollary 1.

Instance: 100r30i2 Nodes: 100 Agents: 15 MaxTime: 20
Source: [9 19 10 18 13 18 12 18 15 8 6 6 20 18 1]
Destination: [84 88 83 84 96 96 81 95 83 82 93 80 90 85 81]

Radius	GRASP	GRASP+PR	UBound
1	1819	1821	2100
2	1960	1974	2100
3	2065	2067	2100
4	2100	2100	2100
5	2100	2100	2100

(a)

Instance: 100r30i1 Nodes: 100 Agents: 25 MaxTime: 20
Source: [17 6 9 19 9 12 2 15 7 8 1 2 8 6 3 13 16 17 13 13 17 19 2 5 21]
Destination: [81 89 84 82 88 99 93 89 93 97 84 96 96 91 90 86 98 86 81 89 82 89 81 80 99]

Radius	GRASP	GRASP+PR	UBound
1	5183	5186	6000
2	5577	5647	6000
3	5898	5909	6000
4	5992	5992	6000
5	6000	6000	6000

(b)

Instance: 100r30i2 Nodes: 100 Agents: 35 MaxTime: 20
Source: [3 5 11 2 14 4 4 5 10 12 14 13 17 4 17 8 16 7 15 7 15 13 12 9 5 6 19 18 3 16 18 19 10 5 2]
Destination: [89 95 89 91 84 99 88 91 92 82 98 84 85 89 85 98 92 80 81 85 98 94 82 89 90 96 91 92 90 96 96 96 99 81 82]

Radius	GRASP	GRASP+PR	UBound
1	10222	10255	11900
2	11108	11224	11900
3	11660	11704	11900
4	11842	11845	11900
5	11900	11900	11900

(c)

Tables 4.4, 4.5, and 4.6 show the evolution of the average solution values as the communication range increases for the 50, 75, and 100 node graphs, respectively. Notice once more that as the communication range increases, the average solution converges to the value of the upper bound given by Corollary 1. In these tables we also report the average computing time required by both the pure GRASP and the GRASP+PR to find their best solutions within the specified number of iterations. For all of the experiments, the GRASP+PR found solutions at least as good as the pure GRASP, finding superior solutions for 45% of the instances tested.

In Figures 4.1, 4.2, and 4.3, we provide plots of the average objective function value versus communication range found using GRASP with path-relinking. The upper bound values for each case as computed by Corollary 1 are also plotted in the charts. These graphs indicate that on average, as the radius of communication increases, the objective function values tend to the upper bound values.

5. Concluding remarks. In this paper, we extended the work of Commander et al. [13] by providing a greedy randomized adaptive search procedure (GRASP) for the COOPERATIVE COMMUNICATION PROBLEM ON MOBILE AD-HOC NETWORKS. This is a new problem in cooperative control [21], which has many applications in path-planning for multiple

Table 4.4: Average solution values for GRASP and GRASP with path-relinking on 50 node graphs.

Nodes	Agents	Radius	GRASP	GRASP+PR	Bound
50	10	1	347	352.21	450
50	10	2	404.58	407.58	450
50	10	3	428.32	429.47	450
50	10	4	437.84	438.53	450
50	10	5	444.37	444.58	450
50	15	1	813.11	817.32	1050
50	15	2	937.74	945.47	1050
50	15	3	1001.11	1003.58	1050
50	15	4	1025.37	1026.21	1050
50	15	5	1037.16	1037.53	1050
50	25	1	2272.79	2315.58	3000
50	25	2	2686.26	2704.53	3000
50	25	3	2850.84	2861.95	3000
50	25	4	2924.05	2927.68	3000
50	25	5	2959	2959.26	3000
Average Comp Time (s)			2.89	4.29	–

Table 4.5: Average solution values for GRASP and GRASP with path-relinking on 75 node graphs.

Nodes	Agents	Radius	GRASP	GRASP+PR	Bound
75	10	1	574.95	577.42	675
75	10	2	629.42	631.37	675
75	10	3	653.53	654.63	675
75	10	4	665.42	665.89	675
75	10	5	669.47	669.84	675
75	20	1	2288	2319.63	2850
75	20	2	2639.37	2651.5	2850
75	20	3	2756.69	2762	2850
75	20	4	2805.53	2807.68	2850
75	20	5	2827.42	2828.42	2850
75	30	1	5349.84	5391.26	6525
75	30	2	6037.47	6064	6525
75	30	3	6310.90	6332.37	6525
75	30	4	6422.11	6430.80	6525
75	30	5	6472.42	6478.84	6525
Average Comp Time (s)			6.16	7.43	–

autonomous agents. Since this problem is NP-hard, the need for efficient heuristics which quickly provide high-quality solutions arises. The proposed GRASP is one such heuristic. Extensive computational experiments indicate that this method is superior to the shortest path protocol and one-pass local search heuristic presented in [13]. Furthermore, the method finds near optimal solutions for the 900 cases tested and converges to the derived upper bound as the communication range increases. Future research efforts include developing an algorithm to find optimal solutions for small CCPM instances such as a branch and cut or a column generation method [22]. This will help evaluate the performance of heuristics whose solutions

Table 4.6: Average solution values for GRASP and GRASP with path-relinking on 100 node graphs.

Nodes	Agents	Radius	GRASP	GRASP+PR	Bound
100	15	1	1838.25	1840.45	2100
100	15	2	1996.75	2003.15	2100
100	15	3	2061.9	2064.7	2100
100	15	4	2083.1	2084.4	2100
100	15	5	2093.95	2094.05	2100
100	25	1	4979.1	5019.2	6000
100	25	2	5655.3	5674.35	6000
100	25	3	5869.35	5876.9	6000
100	25	4	5940.65	5944.7	6000
100	25	5	5978.2	5979.2	6000
100	35	1	9947.45	9997.15	11900
100	35	2	11254.55	11280	11900
100	35	3	11636.85	11664.5	11900
100	35	4	11787.9	11793	11900
100	35	5	11859.1	11860.35	11900
Average Comp Time (s)			5.17	8.05	–

for certain instances are not verifiably optimal, i.e. the solutions that are not at the computed upper bound.

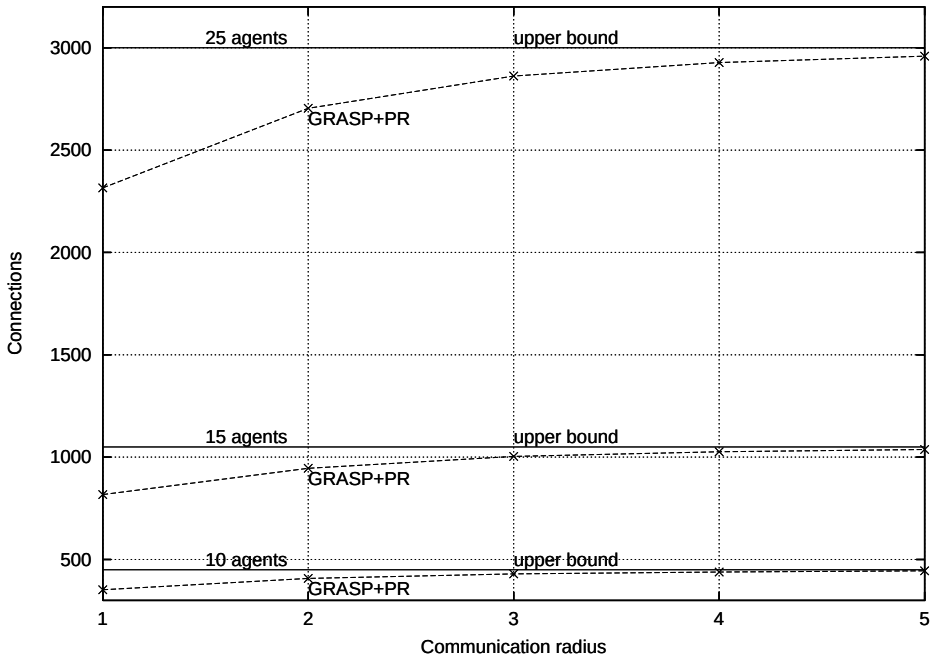

Fig. 4.1: Evolution of GRASP+PR solution values on 50 node graphs as the communication radius increases from 1 to 5 units.

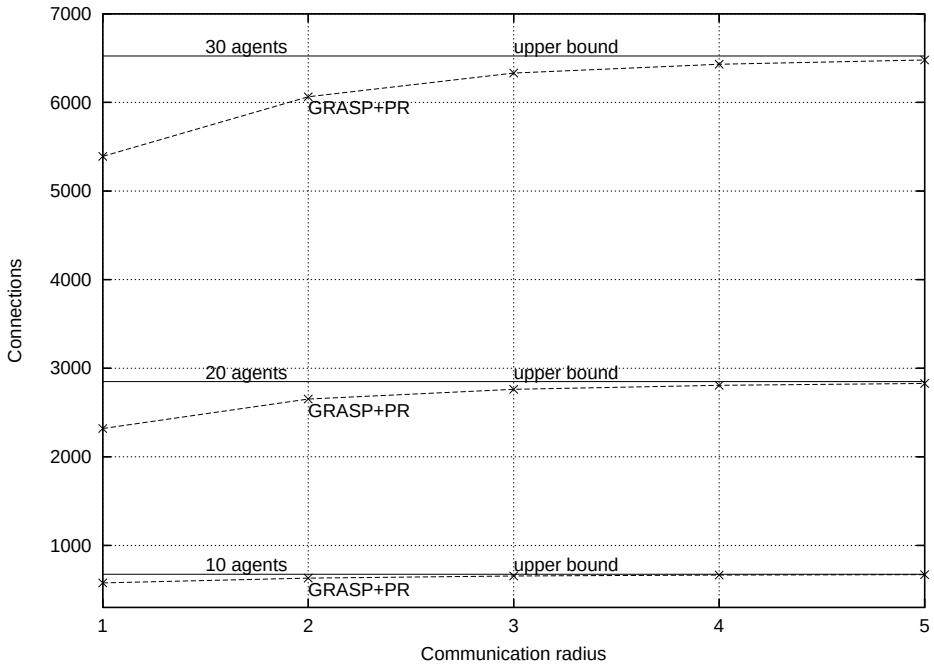

Fig. 4.2: Evolution of GRASP+PR solution values on 75 node graphs as the communication radius increases from 1 to 5 units.

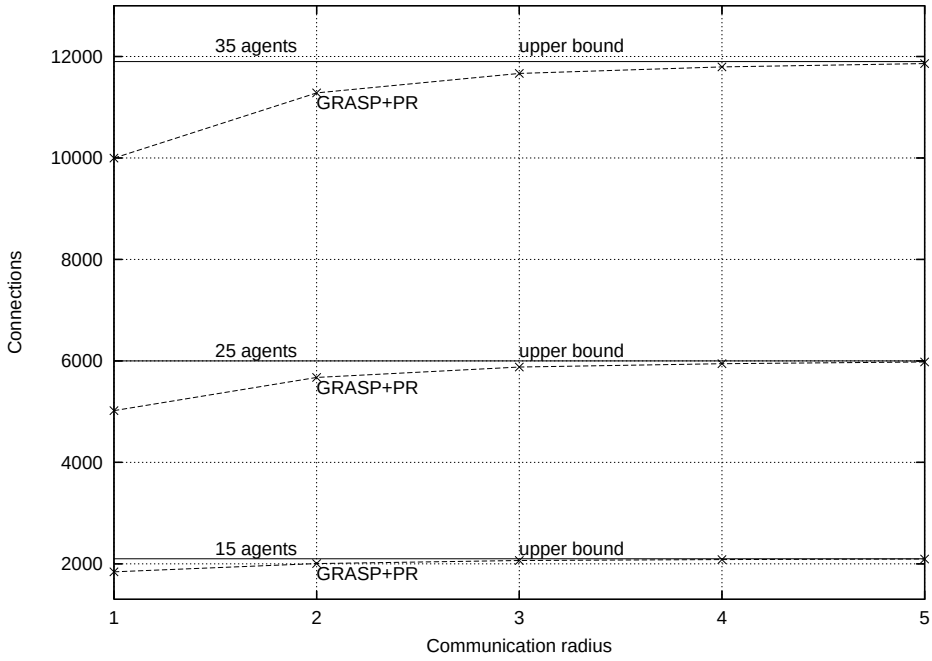

Fig. 4.3: Evolution of GRASP+PR solution values on 100 node graphs as the communication radius increases from 1 to 5 units.

REFERENCES

[1] R.K. AHUJA, T.L. MAGNANTI, AND J.B. ORLIN, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.

[2] R. AIEX, S. BINATO, AND M.G.C. RESENDE, *Parallel GRASP with path-relinking for job shop scheduling*, Parallel Computing, 29 (2003), pp. 393–430.

[3] B.S. BAKER, *Approximation algorithms for np-complete problems on planar graphs*, Journal of the ACM, 41 (1994), pp. 153–180.

[4] S. BUTENKO, X. CHENG, D.-Z. DU, AND P. M. PARDALOS, *On the construction of virtual backbone for ad hoc wireless network*, in Cooperative Control: Models, Applications and Algorithms, S. Butenko, R. Murphey, and P. M. Pardalos, eds., Kluwer Academic Publishers, 2002, pp. 43–54.

[5] S.I. BUTENKO, X. CHENG, C.A.S. OLIVEIRA, AND P.M. PARDALOS, *A new algorithm for connected dominating sets in ad hoc networks*, in Recent Developments in Cooperative Control and Optimization, S. Butenko, R. Murphey, and P. Pardalos, eds., Kluwer Academic Publishers, 2003, pp. 61–73.

[6] S.I. BUTENKO, R.A. MURPHEY, AND P.M. PARDALOS, eds., *Cooperative Control: Models, Applications, and Algorithms*, Springer, 2003.

[7] ———, eds., *Recent Developments in Cooperative Control and Optimization*, Springer, 2004.

[8] W. CHAOVALITWONGSE, D. KIM, AND P.M. PARDALOS, *GRASP with a new local search scheme for vehicle routing problems with time windows*, Journal of Combinatorial Optimization, 7 (2003), pp. 179–207.

[9] X. CHENG, X. HUANG, D. LI, W. WU, AND D.Z. DU, *A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks*, Networks, 42 (2003), pp. 202–208.

[10] B.N. CLARK, C.J. COLBOURN, AND D.S. JOHNSON, *Unit disk graphs*, Discrete Mathematics, 86 (1990), pp. 165–177.

[11] C.W. COMMANDER, S.I. BUTENKO, AND P.M. PARDALOS, *On the performance of heuristics for broadcast scheduling*, in Theory and Algorithms for Cooperative Systems, D. Grundel, R. Murphey, and P. Pardalos, eds., World Scientific, 2004, pp. 63–80.

[12] C.W. COMMANDER, S.I. BUTENKO, P.M. PARDALOS, AND C.A.S. OLIVEIRA, *Reactive grasp with path relinking for the broadcast scheduling problem*, in Proceedings of the 40th Annual International Telemetry Conference, 2004, pp. 792–800.

[13] C.W. COMMANDER, C.A.S. OLIVEIRA, P.M. PARDALOS, AND M.G.C. RESENDE, *A one-pass heuristic for cooperative communication in mobile ad hoc networks*, in Advances in Cooperative Control and Optimization, D.A. Grundel, R.A. Murphey, P.M. Prokopyev, and P.M. Pardalos, eds., World Scientific, 2006.

[14] T.A. FEO AND M.G.C. RESENDE, *Greedy randomized adaptive search procedures*, Journal of Global Optimization, 6 (1995), pp. 109–133.

[15] P. FESTA, P.M. PARDALOS, L.S. PITSOULIS, AND M.G.C. RESENDE, *GRASP with path-relinking for the weighted MAXSAT problem*, ACM Journal of Experimental Algorithmics, accepted (2006).

[16] P. FESTA, P.M. PARDALOS, M.G.C. RESENDE, AND C.C. RIBEIRO, *Randomized heuristics for the* MAX-CUT *problem*, Optimization Methods and Software, 7 (2002), pp. 1033–1058.

[17] P. FESTA AND M.G.C. RESENDE, *GRASP: An annotated bibliography*, in Essays and surveys in metaheuristics, C. Ribeiro and P.Hansen, eds., Kluwer Academic Publishers, 2002, pp. 325–367.

[18] M.R. GAREY AND D.S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[19] F. GLOVER, *Tabu search and adaptive memory programming – advances, applications, and challenges*, in Interfaces in Computer Science and Operations Research, R.S. Barr, R.V. Helgason, and J.L. Kenngington, eds., Kluwer Academic Publishers, 1996, pp. 1–75.

[20] F. GLOVER, M. LAGUNA, AND R. MARTÍ, *Fundamentals of scatter search and path-relinking*, Control Cybernetics, 39 (2000), pp. 653–â684.

[21] D.A. GRUNDEL, R.A. MURPHEY, AND P.M. PARDALOS, eds., *Theory and Algorithms for Cooperative Systems*, World Scientific, 2004.

[22] R. HORST, P.M. PARDALOS, AND N.V. THOAI, *Introduction to Global Optimization*, vol. 3 of Nonconvex Optimization and its Applications, Kluwer Academic Publishers, 1995.

[23] M. LAGUNA AND R. MARTÍ, *Grasp with path relinking for 2-layer straight line crossing minimization*, INFORMS Journal on Computing, 11 (1999), pp. 44–52.

[24] Y. LI, P.M. PARDALOS, AND M.G.C. RESENDE, *A greedy randomized adaptive search procedure for the quadratic assignment problem*, in Quadratic Assignment and Related Problems, P.M. Pardalos and H. Wolkowicz, eds., vol. 16 of DIMACS Series on DIscrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1994, pp. 237–261.

[25] R.A. MURPHEY AND P.M. PARDALOS, eds., *Cooperative Control and Optimization*, Springer, 2002.

[26] C.A.S. OLIVEIRA AND P.M. PARDALOS, *Ad hoc networks: Optimization problems and solution methods*, in Combinatorial Optimization in Communication Networks, D.-Z. Du, M. Cheng, and Y. Li, eds., Kluwer, 2005.

[27] ———, *An optimization approach for cooperative communication in ad hoc networks*. Submitted for publication, 2005.

[28] C.A.S. Oliveira, P.M. Pardalos, and M.G.C. Resende, *GRASP with path-relinking for the QAP*, in 5th Metaheuristics International Conference, 2003, pp. 57.1–57.6.

[29] M.G.C. Resende and T.A. Feo, *A GRASP for satisfiability*, in Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenges, D.S. Johnson and M.A. Trick, eds., vol. 26, American Mathematical Society, 1996, pp. 499–520.

[30] M.G.C. Resende and P.M. Pardalos, *Handbook of Optimization in Telecommunications*, Springer, 2006.

[31] M.G.C. Resende and C.C. Ribeiro, *GRASP and path-relinking: Recent advances and applications*, in Metaheuristics: Progress as Real Problem Solvers, T. Ibaraki, K. Nonobe, and M. Yagiura, eds., Springer, 2005, pp. 29–63.