Int J Wireless Inf Networks (2017) 24:404–412
DOI 10.1007/s10776-017-0347-2

Application of Asynchronous Weak Commitment Search in Self Organized Cognitive Radio Networks

Shabnam Sodagari[1]

Received: 20 August 2016 / Accepted: 28 February 2017 / Published online: 6 March 2017

Abstract This article presents a distributed solution to autonomous quality of service provision. This method paves the way for decentralized and autonomous quality of service (QoS) provision in capillary networks that reach end nodes at Internet of Things, when, due to any reason, central management is either unavailable or not efficient. As case studies, cognitive spatial reuse time division access and code division multiple access communication networks are investigated. Based on asynchronous weak commitment search the task of QoS provision is distributed among different network nodes. This application of artificial intelligence in wireless and mobile communications can be used in home automation and networking, and vehicular technology. Extensions of this approach can be used in self organizing networks, specifically for machine to machine communications.

Keywords Quality of service · Internet of Things · Autonomous networks · Distributed constraint satisfaction · Cognitive radio · Asynchronous weak commitment search · Self-organizing networks (SONs)

1 Introduction

Cognitive radio (CR) entered the lexicon of wireless communication to enable dynamic spectrum access to increase spectral efficiency. In a cognitive radio network (CRN) the spectrum license holder is called a primary user (PU) and other devices that try to dynamically access unused resources of PU, without affecting its performance, are called secondary users or CRs. The terms secondary and CR are used interchangeably.

In underlay CRN scheme CRs simultaneously use the bands with primary, conditioned on avoiding interference to PUs.

To provide quality of service (QoS) in CRNs the constraint of avoiding interference with legacy license holders, i.e., PUs is inevitable. There are several drawbacks associated with centralized control of CRNs for QoS provision. A central management entity might not always be feasible, especially in CRNs, where the topology of the network and spectrum usage patterns are varying. Also, when the central management entity fails, the whole network experiences failure. Once a major link failure or disaster happens, the network should go to autonomous mode. Above reasons are convincing to migrate toward a distributed and autonomous management of QoS in CRNs. The distributed approach has further advantage of breaking down the load of coordination among all nodes. Here, inspired by asynchronous weak commitment search (AWCS) [1], a method for autonomous network management and recovery is put forward. This method distributes the task of providing QoS among different network nodes. Table 1 contains the notation and abbreviations used throughout this paper.

Example 1 To elucidate the many potential applications of distributed constraint satisfaction algorithms to realize self-organizing wireless communication systems, a simplified example is depicted in Figs. 1, 2, 3, 4, 5 and 6. The method used in this example facilitates automatic self-configuration of parameters, such as antenna tilt and power in cellular networks. It can replace tedious manual adjustments for SON dynamic radio configuration (DRC).

✉ Shabnam Sodagari
shabnam@csulb.edu

[1] California State University Long Beach, Long Beach, CA, USA

Table 1 Abbreviations

CR	Cognitive radio
CRN	Cognitive radio network
PU	Primary user
SU	Secondary user
AWCS	Asynchronous weak commitment search
SON	Self organizing networks
SCMA	Sparse code multiple access
STDMA	Spatial reuse time division multiple access
CDMA	Code division multiple access
SINR	Signal to interference plus noise ratio
CSP	Constraint satisfaction problem

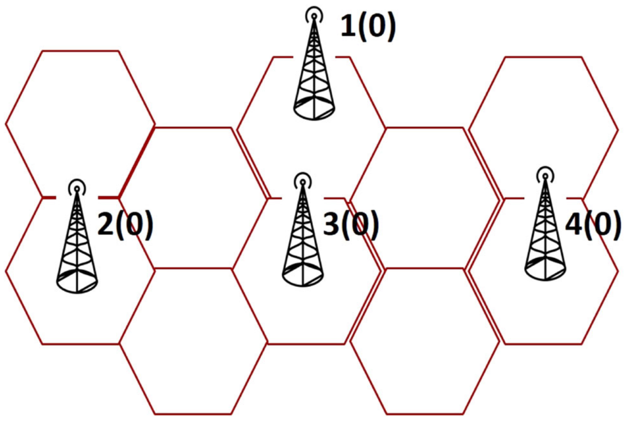

Fig. 1 Initial setup for example 1

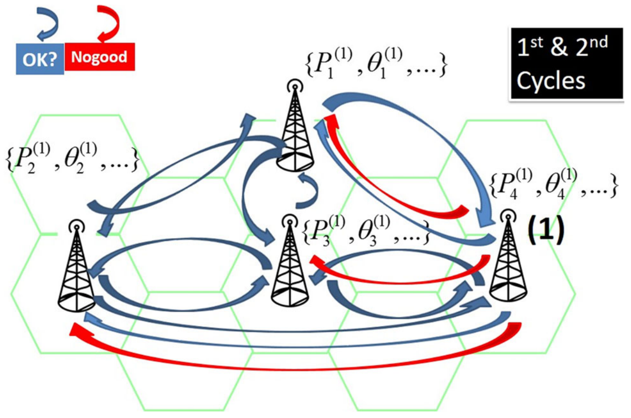
OK?
Nogood
1st & 2nd Cycles
$\{P_1^{(1)},\theta_1^{(1)},\ldots\}$
$\{P_2^{(1)},\theta_2^{(1)},\ldots\}$
$\{P_3^{(1)},\theta_3^{(1)},\ldots\}$
$\{P_4^{(1)},\theta_4^{(1)},\ldots\}$
(1)

Fig. 2 First cycle of self-organized parameter configuration using AWCS in example 1

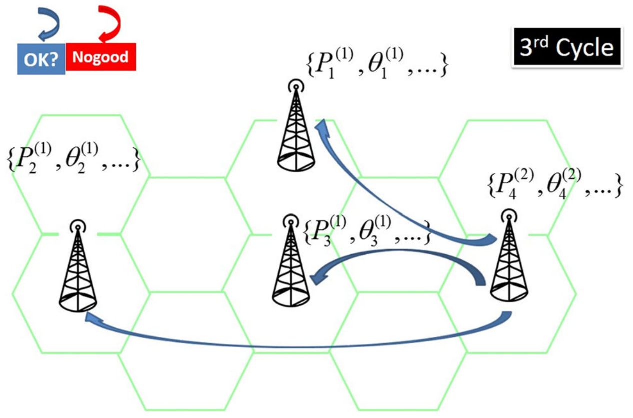
OK?
Nogood
3rd Cycle
$\{P_1^{(1)},\theta_1^{(1)},\ldots\}$
$\{P_2^{(1)},\theta_2^{(1)},\ldots\}$
$\{P_3^{(1)},\theta_3^{(1)},\ldots\}$
$\{P_4^{(2)},\theta_4^{(2)},\ldots\}$

Fig. 3 Second cycle of self-organized parameter configuration using AWCS in example 1

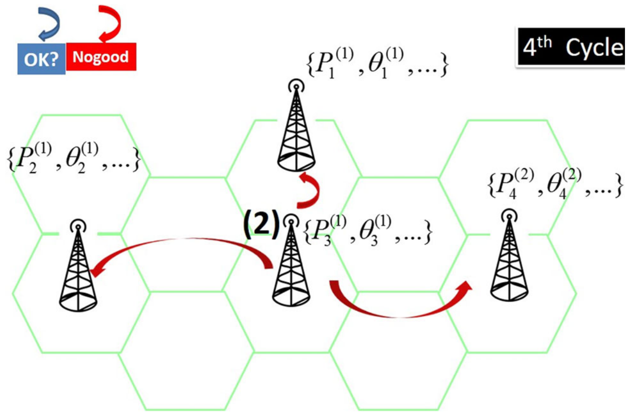
OK?
Nogood
4th Cycle
$\{P_1^{(1)},\theta_1^{(1)},\ldots\}$
$\{P_2^{(1)},\theta_2^{(1)},\ldots\}$
(2)
$\{P_3^{(1)},\theta_3^{(1)},\ldots\}$
$\{P_4^{(2)},\theta_4^{(2)},\ldots\}$

Fig. 4 Third cycle of self-organized parameter configuration using AWCS in example 1

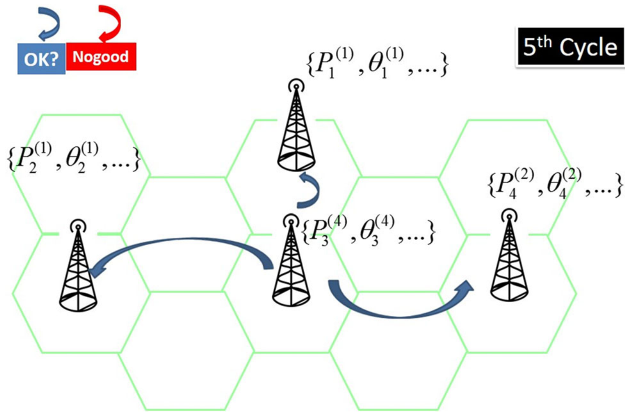
OK?
Nogood
5th Cycle
$\{P_1^{(1)},\theta_1^{(1)},\ldots\}$
$\{P_2^{(1)},\theta_2^{(1)},\ldots\}$
$\{P_3^{(4)},\theta_3^{(4)},\ldots\}$
$\{P_4^{(2)},\theta_4^{(2)},\ldots\}$

Fig. 5 Fourth cycle of self-organized parameter configuration using AWCS in example 1

Sets of variables and domains for each base station include power $p \in [P_{\min}, P_{\max}]$ and antenna tilt $\theta \in [\theta_{min}, \theta_{\max}]$, as in Fig. 1. Values in parentheses are priority values in Figs. 1, 2, 3, 4, 5 and 6. Since initially all priorities are equal, the priority of CRs can be determined by a conventional order of base stations, e.g., their number. Nodes exchange assignments using *ok?* messages. At the start, each base station selects its initial variables, and sends them with *ok?* to other base stations. Figure 1 shows the number of each base station beside its priority value, which is initially 0. In Figs. 2, 3, 4, 5 and 6, number of each base

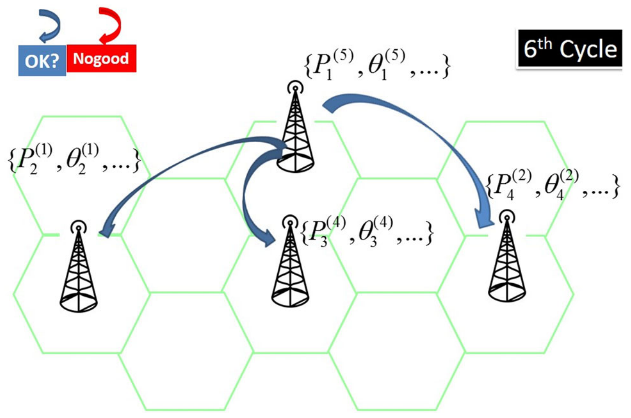
OK?
Nogood
6th Cycle
$\{P_1^{(5)}, \theta_1^{(5)}, \ldots\}$
$\{P_2^{(1)}, \theta_2^{(1)}, \ldots\}$
$\{P_3^{(4)}, \theta_3^{(4)}, \ldots\}$
$\{P_4^{(2)}, \theta_4^{(2)}, \ldots\}$

Fig. 6 Solution obtained at fifth cycle of self-organized parameter configuration using AWCS in example 1

station is omitted and only priority of each base station is shown in parentheses. In Fig. 2, all base stations have sent their initial values along with *ok?* messages to each other. Assume base station 4 finds constraint violation with base station 2, e.g., its antenna tilt and that of base station 2 interfere. It sends *nogood* messages to other base stations and increments its priority value by 1. In the third cycle, shown in Fig. 3, base station 4 chooses a value minimizing constraint violations, which only conflicts with base station 3. Base station 4 sends its value along with *ok?* messages to other agents. In the fourth cycle, shown in Fig. 4, constraint of base station 3 is violated. Therefore, base station 3 increments its priority to 2 and sends *nogood* messages. A value minimizing constraint is a tilt angle that is not in the *nogood* list received so far and is also among possible options for the base station.

In the fifth cycle, as in Fig. 5, base station 3 selects a value that minimizes constraint violations, but still violating base station 1, and sends it with *ok?* messages to other base stations. In the sixth cycle, depicted in Fig. 6, base station 1 changes its value and after sending *ok?* messages, other base stations find it not violating and a solution is obtained.

The goal is to derive a solution that satisfies all QoS constraints in PU and CRs coexistence in the shortest possible time. To this end, CRs use AWCS, which is an efficient method to solve distributed constraint satisfaction problems in comparison with other similar methods in that it offers shorter convergence time and less message exchange overhead [2].

The method of this paper is advantageous over distributed control using game theory, especially in large scale systems. Using game theory approach with local, scalable, and budget-balanced utility functions results in extra computational complexity [3]. This stems from the requirement to compute Shapley value to find a pure Nash equilibrium. As a result, computational burden can become intractable in such approach.

The contributions of this article can be summarized as follows:

– A multi-stage protocol for decentralized QoS provision in cognitive radio networks, which utilizes AWCS at one of the stages.
– Bypassing the need to estimate mutual interference channel gains.
– Controlled messaging overhead.
– Discussing effects of delay in passing messages on the performance.

In Sect. 2 models and constraints for QoS provision in cognitive cellular networks are presented. Section 3 contains details of decentralized QoS protocol. Simulation results are presented in Sects. 4 and 5 concludes.

2 System Model and Problem Statement

To show applications of AWCS algorithm in providing distributed QoS in cognitive radio networks the following subsections consider scenarios related to cognitive underlay spatial reuse time division multiple access (STDMA) networks and cognitive underlay code division multiple access (CDMA) networks. These system models involve optimal scheduling, power and rate allocations to satisfy required QoS constraints of each CR, while avoiding violation of constraints of other network nodes.

2.1 QoS in Cognitive STDMA Networks

In a cognitive underlay STDMA network a set of CR links (a transmitter receiver pair) coexists with a set of PU links [4]. Each frame contains a scheduling period for CR links and a transmission period. The CR transmitter receiver pairs communicate in an ad-hoc manner. CRs use spatial resources to transmit at the same time at their scheduled time slots. One QoS requirement for each CR's traffic involves a minimum number of time-slots within a frame. CRs should avoid interference to PUs, because transmission slots of CRs are underlaid with the transmissions from PUs. Using AWCS signaling messages during the scheduling period CRs find optimal schedules and transmission powers to be used during the transmission period.

Another QoS constraint is the signal to interference plus noise (SINR) level at the receiver, which must be above a certain threshold for the data to be transmitted successfully over a link.

Within a time-slot a subset of the CR links can simultaneously transmit data, i.e., have transmission powers greater than zero, to spatially reuse the frequencies, provided that they meet the interference limit constraints of PUs and the minimum SINR requirements of CR links. CRs should meet their traffic demands and at the same time minimize their transmission length in terms of number of time slots within a frame. In addition to CR minimum SINR constraints and SINR constraints of PUs, power budget of CRs is also limited [4].

2.2 QoS for Cognitive CDMA Cellular Networks

SCMA or sparse code multiple access is a multiple access scheme that combines QAM symbol mapping and spreading of CDMA [5]. Due to resemblance of SCMA and CDMA in this aspect, it is noteworthy to consider QoS in CDMA networks. Here, providing QoS is considered in terms of rate and power allocation for cognitive CDMA cellular networks [6], where PUs and CRs can transmit simultaneously in a shared frequency band. PUs in a cellular wireless network communicate with their corresponding base stations [7]. Rates and power values selected by CRs should satisfy QoS requirements in terms of SINR and minimum data rates, while at the same time the interference to PUs must be kept below a certain threshold. In addition to power levels, SINR in CDMA is affected by processing gain or bandwidth divided by rate. It is assumed that each user has equal bandwidth and therefore, processing gain depends only on each user's rate.

In contrast with [6], where it is assumed that a central controller in the CR network performs the joint admission control, and rate/power allocation for CRs, here, a decentralized method using distributed constraint satisfaction is deployed, which does not need knowledge of channel gains among PUs and CRs and peer CRs. The goal is to maximize the proportionally fair data rates of CRs [8]. The problem involves finding fair resource allocations for CRs subject to interference constraints introduced to PUs in a cognitive cellular CDMA network [6].

To keep interference to PUs below the desired threshold, the PU informs CRs if their selected transmit powers are above the threshold. Accordingly, CRs decrease their power to the next lower quantized level until the interference constraint of PU is met. Nevertheless, if through some mechanisms, channel gains or their statistics can be estimated by CR nodes (using e.g., pilot signals and channel reciprocity), the algorithm would converge faster to a solution because of the insight into selection of initial values.

3 QoS Using Asynchronous Weak Commitment Search for Distributed Constraint Satisfaction

A constraint satisfaction problem (CSP) consists of n variables $x_1, x_2, \ldots, x_n$, whose values are taken from finite, discrete domains $D_1, D_2, \ldots, D_n$, respectively, and a set of constraints on their values [1]. Solving a CSP is equivalent to finding an assignment of values to all variables such that all constraints are satisfied.

In distributed CSP variables and constraints are distributed among autonomous agents. Agents that are related by constraints are called neighbors and communicate by sending messages. The random delay in delivering a message is finite. Furthermore, for the transmission between any two agents, messages are received in the order in which they were sent.

Each agent records its own *agent–view* and *nogood* [9]. Here, agents are associated to different CR nodes. The *agent–view* of a CRi is the set of values (e.g., transmission power) of other CRs communicated to CRi. A *nogood* is a conflicting arrangement of parameter values of CRs. It is used as a constraint. Since *agent–view* is a record of parameter values from the viewpoint of a CR, a *nogood* is also a subset of agent-view. If a *nogood* exists, it means the agent cannot find a value from the domain of its variable to be consistent with the *nogood*. When the *agent–view* of a node contains a *nogood*, the values of other agents must be changed. When a CR receives new variable assignments initiated by other CRs, it updates its *agent–view* accordingly. When a CR cannot take any value consistent with its *agent–view*, because of the original constraints or because of received *nogoods*, new *nogoods* are generated as inconsistent subsets of the *agent–view*, and sent to other CRs. The process terminates when a solution has been found, or when the empty *nogood* is generated, which means the problem has no solution [9]. Empty *nogood* is generated when a CR receives *nogood* messages for all possible values of its parameter. In this case, the problem has no solution because any choice leads to a constraint violation. When a CR finds empty *nogood*, it will announce to other CRs that the problem has no solution and the protocol terminates. AWCS is an efficient algorithm for solving distributed constraint satisfaction problems involving multiple agents. AWCS algorithm uses the two types of *ok?* and *nogood* messages, with the same significance. When an agent receives an *ok?* message, it updates its *agent–view* list and checks if its constraints are violated. If no *nogood* value of higher priority agents is violated, no action is required. If there are a few higher priority *nogood* values that have inconsistent values and these values could be eliminated by changing the variable assignment, the CR will change this value and will send the *ok?* message [1, 9]. *Nogood* learning can improve the performance of AWCS algorithm, in terms of required cycles to solve the problem [10].

AWCS algorithm uses the min-conflict heuristic as a value ordering heuristic. In min-conflict heuristic when selecting a variable value, if there exist multiple values consistent with the *agent–view*, i.e., those that satisfy the constraints with variables of higher priority CRs, the agent prefers the value that minimizes the number of constraint violations with variables of lower priority agents. Using this method without any CR having exact information on the partial solution, CRs can operate concurrently and asynchronously.

For each CR, a non-negative integer value representing the priority order of the CR is defined. This is called the *priority value*. Any CR with a larger *priority value* has higher priority. In case the *priority values* of multiple CRs are the same, the order is determined by an agreed upon convention. For each CR the initial *priority value* is 0. If there exists no consistent value for CRi, the priority value of CRi is changed to $l + 1$, where l is the largest *priority value* among CRs [2]. When a CR makes a mistake in selecting a value for its variables, e.g., transmit power level, the priority of another CR becomes higher and consequently, the CR that made the bad decision will not commit to it, and the selected value is changed. This implies giving up the partial solution if there exists no consistent value with the partial solution and restarting the search process. This is obtained through dynamically changing the priority order.

Asynchronous backtracking is another distributed constraint satisfaction method. Table 2 compares AWCS with asynchronous backtracking [11]. As in Table 2, AWCS outperforms the other distributed constraint satisfaction solution for SON by converging faster and having lower signaling overhead.

3.1 QoS Solution for Cognitive STDMA Networks

As shown in Fig. 7, CRs first exchange their minimum required SINR levels to be able to find out, right from the beginning, if the problem has a solution. If the required SINR level of CRs does not result in mutual interference, no messages are exchanged. Otherwise, a one bit message is sent from the victim CR. This way, CRs find out which groups of them can use same timeslots for transmission. Then, each group of non-interfering CRs uses AWCS to find out what maximum possible power levels they can use, besides each other, while keeping it below the power budget of each CR and still causing no interference to other CRs. They start with selecting their maximum power budget and if a constraint violation occurs, they decrement their selected power to the next lower level. This process iterates by exchanging *ok?* and *nogood* messages until AWCS terminates with a solution. After adjusting power levels, interfering subsets of CRs should schedule transmission timeslots based on their traffic demands and interference. For this part of proposed protocol, CRs use a conventional order (e.g., using a number assigned to each CR) at the beginning of the first frame. Disjoint sets of interfering CRs, not allowed to use same time slots within a frame, form a partition. Each set is specified by its head, i.e., the CR in the set with lowest identification number. At each frame, sets update their order in a circular fashion. Within a single frame, the set with lowest head number uses its required time slots, then, the next set and so forth.

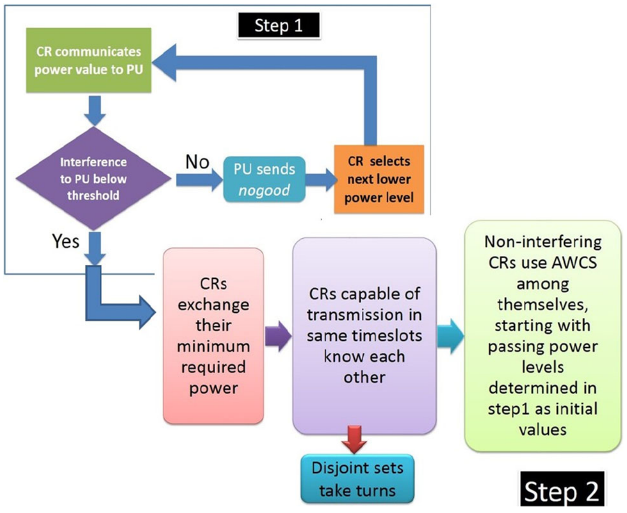
Step 1
CR communicates power value to PU
Interference to PU below threshold
No
PU sends nogood
CR selects next lower power level
Yes
CRs exchange their minimum required power
CRs capable of transmission in same timeslots know each other
Non-interfering CRs use AWCS among themselves, starting with passing power levels determined in step1 as initial values
Disjoint sets take turns
Step 2

Fig. 7 Steps of the proposed decentralized QoS protocol for cognitive STDMA networks

Consider, for instance, 7 CRs partitioned into 3 disjoint sets, based on interference, i.e., $\{CR1, CR2, CR4\}, \{CR3, CR6\}, \{CR5, CR7\}$. In other words, $CR5$ and $CR7$ are spatially apart so they can use the same frequencies for transmission in the same time slots. The heads are $CR1$, $CR3$ and $CR5$. In the first frame, the set containing $CR1$ is scheduled first to use its required time slots, based on its users' minimum traffic demands. In the next frame, the set containing $CR3$ uses timeslots. At last, $CR5$ and $CR7$ get to use time slots. In each frame, the priority of sets to use time slots is re-ordered in a round robin manner. The new order of heads is $CR5$, $CR1$, $CR3$. After the set containing the last head uses the frame, the set of $\{CR1, CR2, CR4\}$ is scheduled again.

Table 2 Main differences of distributed constraint satisfaction algorithms

	Signaling	Priority order	*ok?* messages sent	Convergence
AWCS	Lower	Dynamic	To lower and higher priority CRs	Faster
Asynchronous backtracking	Higher	Static	To lower priority CRs	Slower

3.2 Decentralized QoS Provision in Cognitive CDMA Networks with Unequal Rates

So far, it was shown how AWCS can be used for optimal power allocation among CRs in CDMA networks with equal rates. A modified version of AWCS with optimizing two or more local variables, e.g., rate and power, can also be used [2]. AWCS for multiple local variables [2] originates from AWCS for one local variable, but a CR sequentially performs the computation for each variable, and communicates with other CRs only when it can find a local solution that satisfies all local constraints. A bad local solution can be modified without forcing other CRs to exhaustively search their local solutions, and the number of interactions among CRs can be decreased. Every CR changes the values of its local variables in order. It selects a variable that has the highest priority among variables that are violating constraints with higher priority variables, and modifies its value so that constraints with higher priority variables are satisfied. When all local variables satisfy constraints with higher priority variables, the CR sends changes to other CRs.

CRs choose initial values by starting from the maximum allowed power levels. Each CR communicates these initial values via *ok?* messages. After that, CRs wait for and respond to messages they receive. Any CR can handle multiple messages concurrently.

By sending messages to other CRs only when a CR finds a consistent local solution, the number of messages exchanged among CRs decreases. In the first phase of proposed protocol, CRs send their values to PUs. PUs, on the other hand, check if the value selection of CRs violates their QoS requirements by exceeding the interference threshold. If not, they do not send any messages; however, if they find the value selection of CRs to be violating, they send a one bit message to the corresponding CR. The CR then decreases its selected value to next quantized lower level and iterates the process until the constraints of PUs are met. CRs then enter the second phase to satisfy the constraints among themselves by using AWCS algorithm. Figure 8 shows the proposed decentralized QoS provisioning scheme for cognitive CDMA networks.

3.3 Effects of Delay in Message Exchange

In practice, messages do not arrive instantaneously but are delayed due to network properties [9]. Delays can vary on account of different network factors, such as hardware and topology. Effects of message delays on distributed constraint satisfaction algorithms have been measured using controlled simulation environments that apply randomly generated delays [12]. Main results are described with respect to CRNs applications.

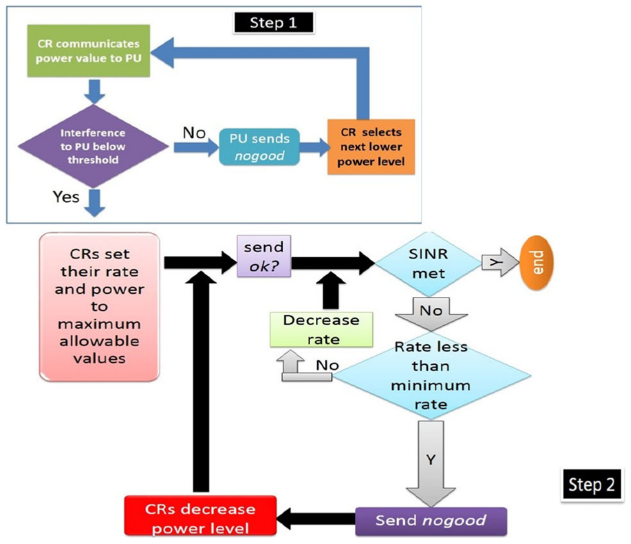
Step 1
CR communicates power value to PU
Interference to PU below threshold
No
PU sends nogood
CR selects next lower power level
Yes
CRs set their rate and power to maximum allowable values
send ok?
SINR met
end
Decrease rate
No
No
Rate less than minimum rate
Y
Step 2
CRs decrease power level
Send nogood

Fig. 8 Proposed decentralized QoS protocol for cognitive CDMA networks

For asynchronous algorithms, such as AWCS, two messages between the same pair of agents must arrive in the same order they were sent. AWCS is a single process algorithm in that all variables have exactly one assignment at each instant of algorithm run [12]. AWCS reads multiple messages at each step, based on their instantaneous arrival. Agents perform assignments asynchronously, and when the updating message is randomly delayed, some of their computation can be irrelevant due to inconsistent *agent–views* [13]. This is a consequence of the fact that with random message delays, agents might respond to a single message, instead of all messages sent in the previous cycle, and since messages in AWCS are sometimes conflicting, agents perform more unnecessary computations when responding to fewer messages in each cycle. Hence, the improvement that results from reading all incoming messages in each step [11], which is intrinsic to AWCS, is no longer useful in the case of random message delays. In sum, there exists a tradeoff between selecting either a non-concurrent algorithm, such as AWCS, or a concurrent algorithm. When delays happen, concurrent algorithms perform better, however, concurrent algorithms require each CR to be equipped with more processing power to handle concurrent computations, in terms of breaking the variable search space into multiple sub-spaces and keeping track of multiple assignments to a variable all at the same time.

4 Numerical Observations

Many problems related to SONs, e.g., tilt adjustment of neighboring antennas, power allocation of neighboring nodes, *etc* can be cast to distributed constraint satisfaction models. AWCS is an efficient method of distributed constraint satisfaction that outperforms several other graph coloring methods, in terms of e.g., delay and number of messages.

Another distributed graph coloring scheme, i.e., asynchronous backtracking is different from AWCS in various aspects. In asynchronous backtracking, priority order of nodes is fixed, whereas in AWCS there is weak commitment to priority of a node with a conflicting parameter. In AWCS, when a node finds violation to its constraints, it increases its priority value, whereas in asynchronous backtracking, priorities do not change from initial values.

Analysis of different distributed constraint satisfaction schemes [11, 14] show that time needed to converge to final solution is considerably shorter for AWCS than for other distributed SON algorithms, such as asynchronous backtracking. Also, number of cycles to converge to the final solution in AWCS is tremendously fewer than that of asynchronous backtracking, especially when the number of nodes grows. For instance, asynchronous backtracking does not always converge even after more than 1000 cycles for 90 nodes (Fig. 9). Figure 10 shows probability distributions of length of time to converge to final solution for AWCS and asynchronous backtracking. Using AWCS final solution is obtained with probability of almost 1 (certain event) by 200 time units, whereas this figure is 1000 time units for the other distributed constraint satisfaction method, i.e., asynchronous backtracking. In terms of signaling overhead, average number of messages passed among nodes to obtain final solution in AWCS is lower than the other SON solution, or asynchronous backtracking, by a ratio of $<1\%$. Furthermore, AWCS is more reliable to yield a solution within a time limit in that its time length shows less deviation from the average time to converge to a solution. On top of this, as in Fig. 11, in the lengthiest case, which happens with a very low probability, total signaling overhead of AWCS is still better than average signaling overhead for asynchronous backtracking.

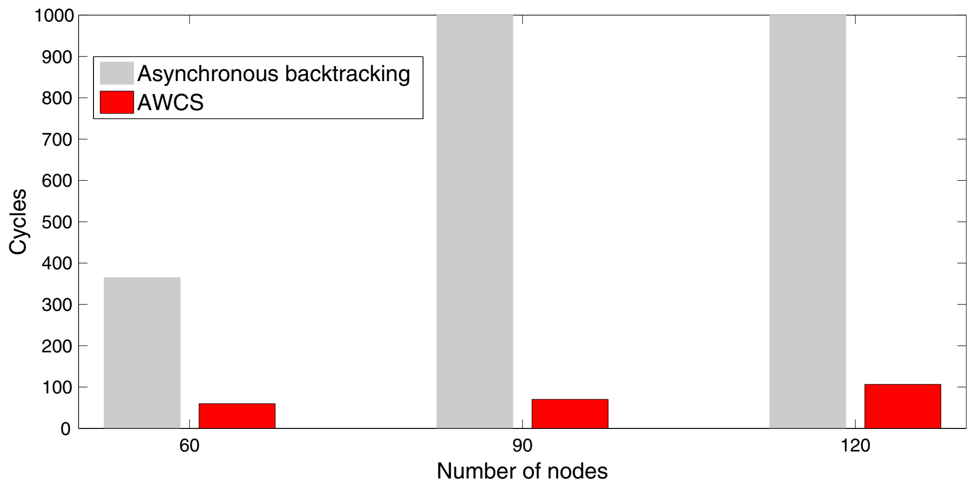

Fig. 9 Number of cycles until convergence to final solution for AWCS and asynchronous backtracking

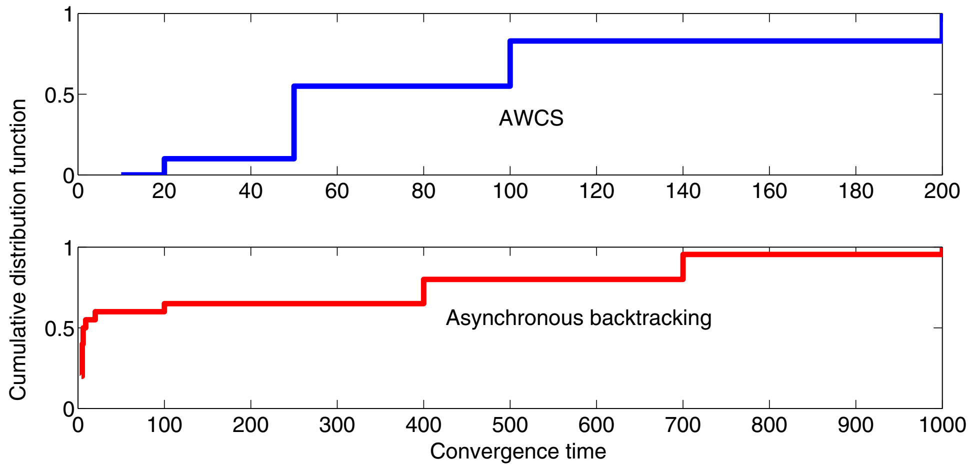

Fig. 10 Approximate cumulative distribution function of convergence time for AWCS and asynchronous backtracking

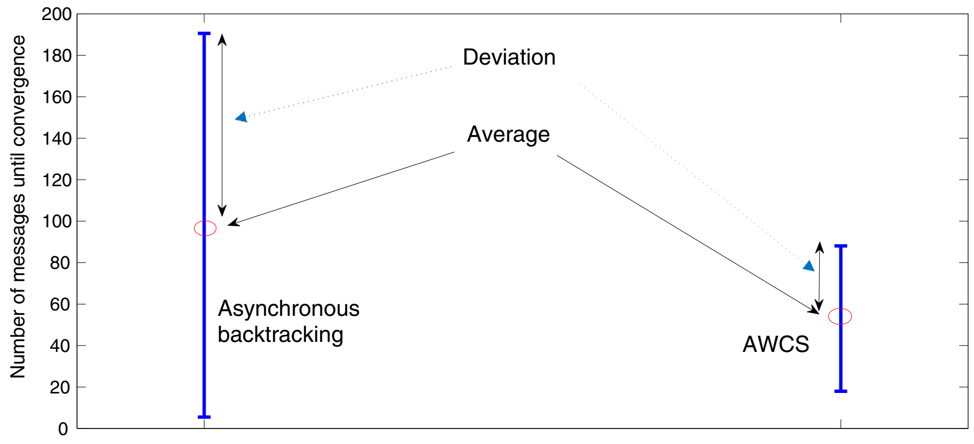

Fig. 11 Comparison of average and standard deviation of number of messages for AWCS and asynchronous backtracking

5 Conclusion

A lightweight decentralized protocol for robust scheduling and power control in ad-hoc cognitive STDMA and CDMA networks is developed. This method, which is based on exchanging local messages, removes the need for knowledge of channel gains. In STDMA networks this scheme has two stages. As a first step, CRs interact with primary, in an iterative manner, to make sure they do not impose interference in underlay PU and SU coexistence. Then, CRs use AWCS among themselves to reach optimal resource allocations. For cognitive CDMA networks it is shown how AWCS can be used for optimal power allocation meeting QoS needs of nodes, considering their constraints on each other. AWCS with multiple local variables is applied to assign optimal rate and powers to CR nodes, without a central management.

For cognitive STDMA networks, where multiple CRs can use same time slots and frequencies, provided they are spatially apart to avoid interference, this protocol includes a third step for scheduling, in addition to the first and second steps envisioned for cognitive CDMA networks. To this end, disjoint sets of interfering CRs are identified with their CR head and use inter-frame circular round robin ordering.

Numerical observations corroborate benefits of this protocol for practical applications in terms of lower algorithm cycles, meeting QoS constraints of nodes, and overcoming the need to know mutual interference channel gains. This method is applicable in situations where centralized management and control is not functional, for various reasons, such as cost and remote access.

This method of autonomous network management, or its modified variants, can be applied to SONs or for vehicular communications, and also capillary networks that realize Internet of Things.

References

1. M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, “The distributed constraint satisfaction problem: formalization and algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, pp. 673–685, September–October 1998.
2. M. Yokoo, *Distributed constraint satisfaction: foundations of cooperation in multi-agent systems.* Springer series on agent technology, Berlin, New York: Springer, 2000.
3. J. R. Marden and A. Wierman, “Overcoming limitations of game-theoretic distributed control,” *IEEE transactions on automatic control*, vol. 58, pp. 1402–1415, June 2013.
4. P. Phunchongharn and E. Hossain, “Distributed robust scheduling and power control for cognitive spatial-reuse TDMA networks,” *IEEE Journal on Selected Areas in Communications*, vol. 30, pp. 1934–1946, 2012.
5. A. Bayesteh, H. Nikopour, M. Taherzadeh, H. Baligh, and J. Ma, “Low complexity techniques for SCMA detection,” in *2015 GLOBECOM Workshops*, pp. 1–6, 2015.
6. D. I. Kim, L. B. Le, and E. Hossain, “Joint rate and power allocation for cognitive radios in dynamic spectrum access environment,” *IEEE Transactions on Wireless Communications*, vol. 7, pp. 5517–5527, December 2008.
7. L.-C. Wang and A. Chen, “On the coexistence of infrastructure-based and ad hoc connections for a cognitive radio system,” in *1st International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom)*, pp. 1–5, June 2006.
8. F. P. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: shadowing prices, proportional fairness, and stability,” *Journal of Operations Research*, vol. 49, pp. 237–252, March 1998.
9. I. Muscalagiu, J. M. Vidal, V. Cretu, P. H. Emil, and M. Panoiu, “The effects of agent synchronization in asynchronous search algorithms,” in *Proceedings of the 1st KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pp. 1–5, May–June 2007.
10. K. Hirayama and H. Yokoo, “The effect of nogood learning in distributed constraint satisfaction,” in *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, pp. 169–177, 2000.
11. M. Yokoo, “Algorithms for distributed constraint satisfaction problems: A review,” *Autonomous Agents and Multi-Agent Systems*, no. 3, pp. 198–212, 2000.