2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology

An Empirical Study on a Locality Based Heuristic in Multi-Agent Constraint Satisfaction

Dev Minotra* Jiming Liu+*(corresponding)

*School of Computer Science, University of Windsor
+Department of Computer Science, Hong Kong Baptist University

Abstract— **Previous work on multi-agent oriented constraint-satisfaction [8] shows that ERA is competitively suitable in solving the n-queen problem, and is capable of providing approximate solutions to different types of CSPs in about two or three time steps. Challenges remain in developing more efficient ERA based systems for solving CSPs beyond just approximation. In this paper, a modified ERA algorithm, involving a locality based heuristic and a random tie-breaking method, is proposed and tested. Experimental results have shown significant performance improvements, indicating some of the key characteristics and strengths of ERA as a multi-agent solution search process.**

I. Introduction

Graph-Coloring Problems (GCPs) are constraint-satisfaction problems that involve assigning values to variables that are represented as nodes in a graph, $G(V, E)$, where V is the set of vertices that represent variables and E represents the set of edges. The variables are bounded by constraints and these constraints are represented by the edges in the graph where no two nodes connected by an edge can share the same color. GCPs are NP-complete [11].

ERA is an Autonomy Oriented Computing method [5] that is designed to solve constraint-satisfaction problems in a distributed environment [8], [6]. In ERA, the variables of a CSP are represented by agents. An agent can be assigned to a single variable or a group of variables. Each agent is composed of a set of reactive behaviors, and an ability to observe the states of its neighbors. The search process in ERA consists of the asynchronous movements of distributed agents within each clock cycle. These movements correspond to the intermediate variable assignments by the agents based on their local observation and evaluation of variable conflicts. An elaborate explanation of the algorithm can be found in [8]. Although ERA is multi-agent oriented, it can also be used as a variation of local-search to solve centralized CSPs [8]. The original ERA algorithm is capable of obtaining approximate (but not optimal) solutions to GCPs in only two to three time steps [8]. Here, one time step corresponds to one clock cycle, in which distributed agents can make asynchronous movements based on their evaluations and behaviors.

It should be pointed out that in our present work, the evaluations and asynchronous movements of agents are *simulated* using a sequential implementation, in which agents within each clock cycle will behave in an *independent* manner.

Autonomy Oriented Computing (AOC) is the area of study that explicitly explores the role of self-organization in multi-entity systems in an attempt to solve computational problems or modeling complex systems behavior [5]. ERA is the example representing how AOC based approaches formulate and tackle a CSP, and furthermore the heuristics being studied and presented in this paper illustrate how the local behaviors of entities impact the emergent global performance of CSP solving as a self-organizing process.

Specifically, the goal of this work is to provide and validate a modified version of ERA in solving GCPs. We will experimentally show how a degree heuristic integrated with other modifications can significantly reduce the number of time steps in the search process, and compare the performance with that of an agent-compromise method [9].

A. Related Work

Tang et al [9] have reported a GCP solving method that incorporates the behavior of compromise-making to address the local-minima problem in the search process of the original ERA method. Their agent-compromise method has led to a reduced number of agent movements. In our present work, we will adopt a different performance measurement, i.e., the number of time steps as opposed to the number of movements. This is because a time step can be composed of several movements, and these movements will take place in an asynchronous manner. When we consider the application of the ERA method in a distributed application environment such as a sensor network, it would be more relevant to evaluate the performance in terms of the total number of time steps required in solving a CSP.

Yokoo et al [15] have proposed another approach to distributed CSPs with the methods of asynchronous weak-commitment search and asynchronous backtracking. Their approach has been successful in solving CSPs in such distributed environments as telecommunication networks [3], [16]. The drawback of the asynchronous weak-commitment search is that it abandons partial solutions in its search process, when variable combinations extending the partial solution do not lead to a solution [8]. On the other hand, ERA allows variables to be modified simultaneously. Depending on the local behavior of individual agents, a partial solution keeps getting self-organized in a stable and spontaneous manner as to be illustrated in this paper. In addition, ERA differs

0-7695-3027-3/07 $25.00 © 2007 IEEE
DOI 10.1109/IAT.2007.83

from the asynchronous weak-commitment search in terms of completeness [8]. In ERA, completeness is traded for efficiency, whereas the asynchronous weak-commitment search is a complete method. Most importantly, the communications in ERA are not as explicit as those in the methods by Yokoo et al [8], making it better suitable for multiple light-weight agents/nodes (i.e., low in cost and resources) in highly distributed environments [5].

B. Locality Based Heuristics

Agents in ERA are composed of predefined primitive behaviors that cause their movements at each time step of the algorithm execution. Primitive behaviors, such as the least-move, better-move, and random-move, have been defined by Liu et al [8]. An important property of these reactive rules is that they do not involve global behavior. Local behavior is conducive to the success of a system in a distributed environment. The agents are collectively coupled with an environment that records violation update values to provide agents with conflict information in an implicit way.

The locality based heuristic proposed here involves a degree-based approach, where an agent uses the degree of connectivity of its neighbors as a factor in measuring violation values in the environment update function. This heuristic is combined with another modification in tie-breaking. In local search approaches, different tie-breaking methods could often lead to different search performances [16]. In this work, we will experimentally show that random tie-breaking improves the performance of ERA.

C. Organization of the Paper

The next section, Section II, will discuss the nature of the solution search in ERA and introduce the concept of the retention ratio. Section III will provide details about the least-move behavior and include definitions of new modifications being proposed. In Section IV, the degree based heuristic is covered in detail. Section V provides an overview of the modified specifications. Section VI reports the experiments conducted and their results. Section VII discusses the experimental results. Final conclusions and directions for future work are provided in Section VIII.

II. The nature of the solution search in ERA

A. Synergy in the ERA system

An important characteristic that any multi-agent solution search process should exhibit is the ability to allow multiple agents to make decisions on the basis of their individual local behaviors, simultaneously. The idea of reactive movements is central to Autonomy Oriented Computing, where agents have the ability to self-organize, leading to a solution of a computational problem where every agent makes a decision at each time step based on the state of its local environment and its reactive rules. The effect of the decision will impact the state of the environment at the next time step. This means that if an agent chooses to move to a 'minimal-position' in the state of the environment, it does so with the anticipation that such a move would lead to a positive result in the next time step, despite dynamic changes in the environment that take place due to other agents that also asynchronously move with very similar motivations.

The asynchronous movements of agents in the environment can be either chaotic or made in such a way that allows stable and spontaneous changes in the system that lead to a global solution convergence. The original ERA algorithm allows such a synergetic and stable solution convergence for the n-queen problem [8]. However, in solving GCPs, this may not be always the case. From the experiments to be reported in this paper, the original ERA has good solution-search characteristics on simple GCPs, however, for more complex and large-scale problems, it is not effective.

B. The Retention Ratio

Instantaneous *retention ratio* at time step t is defined as the ratio of the number of agents that reach a minimal position at time step t after performing a least-move at $t-1$ to the number of agents that have made a least-move at $t-1$. Here, 'minimal' is with respect to the state of the environment after all agents have moved and new environment updates have been made. It may not be necessary that an agent arrives at a minimal position after performing the least move. This is because all agents move asynchronously, and change the environment accordingly at each time step. Therefore, it would be useful to observe the plot of instantaneous retention ratio changes; different trend patterns may indicate different characteristics of the algorithm. We will refer to retention-ratio at time step t as γ_t in this paper. In other words,

$$\gamma_t = \frac{\text{Number of agents at minimum-positions at } t}{\text{Number of agents making a least-move at } t-1} \quad (1)$$

If γ_t is consistently above a certain level (e.g., .8) throughout the search, we say that the system has exhibited *parallelizability*. A multi-agent search process where agents work on local reactive rules is parallelizable when agents can make decisions simultaneously and also accomplish individual motives. In general, for all successful ERA search processes, it has been observed that γ_t increases over time at a comparatively higher rate in the first few time steps.

III. Heuristic additions and Least-move variants

A. Operational Modes for the Least-Move

The heuristics to be added to ERA is embedded into the least-move behavior. In order to describe the changes made, this paper makes use of several modes to define the variations of the least-move. As details on the least-move behavior are significant to this paper, it will be described in this section along with the specifics of new modifications. The least-move has been defined by Liu et al [8]. In a few words, it can be described as the move in which an agent a_i at time t moves to a position of the minimal-violation in its environment.

Briefly, it can be said that a least-move is conservative, if an agent would not move to a new minimal position at time $t+1$ when it is already at a minimal position at time t. The

LMODE can be described as the mode that acts as a tie-breaker when there are multiple minimal positions to choose from. On LMODE activation, a minimal position is chosen at random. When LMODE is not used, the default minimum position is the one with the smallest position-index.

In this paper, the least-move is classified into 4 different forms. The two modes LMODE and CONMODE direct the behavior of the least-move being implemented. The activation of CONMODE would implement the *conservative least-move*. When CONMODE is deactivated, the least-move is *non-conservative*. According to the description of the least-move provided by Tang et al [9], the past implementation of least-moves are of the conservative type. Hence, the introduction of the conservative least-move is not a novel addition. However, it is important to discuss the significance of this simple mode setting that when changed it can have a drastic impact on the performance of the solution search. Our experimental results show that with CONMODE, ERA would perform much better than when CONMODE is not used.

Suppose that all lattice points for $agent_r$ are indexed from 0 to n. The corresponding violations for these lattice points are v_0, v_1, v_2....v_n. The set of all indices of the minimal-positions at time t will be M_t, where M_t = $\{m_0, m_1, m_2,....,m_k\}$, where $m_i \leq m_j$ for every $i<j$. We let new_pos_t denote the position of the agent at time t, and new_pos_{t-1} at time $t-1$.

When the set of all possible values that CONMODE can take is $\{1,0\}$ and that LMODE can take is $\{1,0\}$, the definition of CONMODE and LMODE for any $agent_r$ is stated as follows:

$$new\_pos_t = \begin{cases} new\_pos_{t-1} & \textbf{if CONMODE = 1 and } new\_pos_{t-1} \in M_t. \\ m_0 & \textbf{if CONMODE = 1, and LMODE = 0. and } new\_pos_{t-1} \notin M_t \\ random(M_t) & \textbf{if CONMODE = 1 and, } \text{LMODE =1. } new\_pos_{t-1} \notin M_t \\ m_0 & \textbf{if CONMODE = 0 and, } \text{LMODE = 0.} \\ random(M_t) & \textbf{if CONMODE = 0, and LMODE = 1.} \end{cases}$$

where $random(A)$ is a function that randomly returns an element from set A.

The original version of ERA can be classified as the specification that uses CONMODE and does not use the LMODE based on the statements given in [9].

B. The Degree-Based Heuristic

The degree-based heuristic is a strategy used in a CSP solution-search process where the ordering of variable assignments is priority based [10]. Applying the degree-based heuristic to a (GCP) involves a priority ordering based on the degree of the node connectivity. In our present work, the degree-based heuristic is incorporated into the violation-updating step of the ERA algorithm. Violation updating is central to the functioning of ERA where an implicit inter-agent communication is achieved; this can readily be noted in the following outline of ERA:

```
1  procedure ERA
2  begin
3  Initialize the system
4   while(true) do
5    for r = 1 to agent_number do
6      Select behavior for agent_r
7      Move agent_r to a new position if needed
8    end for
9   if(Current state satisfies constraints)
10  Print all variable values and exit
11  Update violations
12  time step++
13  end while
14 end
```

There are two ways that we can update violation values [8]. One involves updating an entire matrix containing all conflict values at the end of a time step, and the other involves updating a violation-array of an agent before it would decide its move. The violation-array would contain violations of all prospective moves. Both are equivalent.

Let a lattice-position refer to a possible assignment to a variable, or assignments to a small group of variables that belong to one agent. An agent's environment consists of all such possible lattice positions. In the original ERA, the lattice violations were calculated purely on the basis of the number of variable conflicts for an assignment, where every variable conflict would add one unit of violation.

In our present approach, if the degree of connectivity of the conflicting-variable is x, $x \times K$ unit violations are added. The modified violation-updating procedure can be described as follows:

```
1  procedure Update violations
2    begin
3     for r = 1 to agent_number do
4      for j = 1 to C do
5       for i = 1 to agent_number do
6         if(link(r,i)= TRUE AND (j = value(i)) AND (r!=i))
7          violation_number_r[j] := violation_number_r[j]
8                + {K × connectivity(i)}
9    end
```

Note: $violation\_number_r[j]$ is the environment update array for $agent_r$ providing violation values on all lattice points

where $agent_r$ can move. *connectivity*(A) is a function that returns the degree of connectivity of $agent_A$, if $agent_r$ is connected to it. link(a,b) returns 1 if an edge connects a and b.

C. New additions and specifications on ERA

As will be reported below, our current empirical studies have shown that the following setting will yield a significant improvement on the applicability of ERA on GCPs:

- The degree based heuristic applied to the environment update function (D = 1).
- Setting LMODE to 1 based on the description provided (L = 1).
- Setting CONMODE to 1 based on the description provided (C = 1).

For the sake of description, we will use the letters of D, L, and C to refer to the above three conditions, respectively. Hence, when all three modes are applied, DLC = 111.

IV. Experiments and Evaluations

In this section, we will present several experimental studies that evaluate the performance of the modified ERA method incorporating the proposed locality based heuristic.

A. Determining the $least_p$ value

Experiment 1: The goal of this experiment is to determine the best $least_p$ value (i.e., the probability by which an agent would make a least-move) for the following validation experiments. The $least_p$ value plays an important role in that a change in this value creates the maximum impact on the search performance, specially when measured in terms of the total number of time steps. Note that we will keep other parameters constant throughout the tests. Specifically, the priority mode is set to F2BLR as defined and used in the previous studies. The ratio of the $least_p$ to $random_p$ is fixed to 2*V, where V is the number of vertices in the graph. The $better_p$ is fixed to .8. The number of variables assigned per agent is 1 and the maximum of 7,000 time steps has been fixed on all runs. The graph problems are taken from [1].

With the proposed modifications in the ERA algorithm, there will be three conditions that need to be studied. Considering all the possible settings for the three conditions, there will be eight possible cases to be tested. The tests will be conducted using benchmark GCP instances. With respect to each of the benchmark GCPs as listed in Table I, its best $least_p$ value will be determined over 50 test runs.

Note that in our experiments, performance evaluations will be made in terms of the number of time steps an approach would take to find a solution. As mentioned earlier, a time step corresponds to a cycle in which all agents independently decide whether or not (and/or how) to make their moves.

Results: The best $least_p$ values found in different modes for the corresponding graph-coloring instances are given in Table I. A detailed breakdown for the anna.col problem instance is provided in Table II. Figure 1 shows the time steps for each individual value of $least_p$ tested on the three modes, i.e., DLC = 001, DLC = 000, and DLC = 111.

TABLE I
Determination of the best $least_p$ in all modes

GCP Instance	LMODE	DC=00	DC=01	DC=10	DC=11
miles500.col	0	.4, .3	.8, .9	.3, .2	.7, .6
	1	HFR	.9, .8	HFR	.7, .5
miles250.col	0	.7, .4	.9, .8	.6, .4	.8, .5
	1	HFR	.9, .8	HFR	.7,.8
miles1000.col	0	HFR	HFR	.3	.7, .6
	1	HFR	.9, .8	HFR	.8, .7
miles1500.col	0	HFR	.7, .6	HFR	.5, .4
	1	HFR	.9, .7	HFR	.7, .6
anna.col	0	.3,.2	.9, .7	.5, .4	.6, .7
	1	HFR	.9, .8	HFR	.7, .8
jean.col	0	.5, .3	.8, .9	.4, .3	.6, .5
	1	HFR	.9, .8	HFR	.7, .6
queen5_5.col	0	.9, .7	.8, .6	.6, .8	.7, .6
	1	HFR	.9, .8	HFR	.7, .8
queen7_7.col	0	.7, .6	HFR	.6, .8	.9, .8
	1	HFR	.7, .5	HFR	.9, .8
myciel6.col	0	.6, .8	.8	.7	.80
	1	HFR	.9, .8	HFR	.80
myciel7.col	0	.7, .8	.7	.6	.9, .8
	1	HFR	.90	HFR	.7, .8

The best $least_p$ values have been shown for each of the GCP instances in different mode combinations. The two of the best cases are included in the table, with the better one shown first. In some cases only one is shown. Here, DC stands for Degree Based Heuristic and CONMODE flags. HFR stands for High Failure rate.

B. Performance Validation

Experiment 2: The goal of this experiment is to determine the performance in various test modes, e.g., LMODE = 1 and DC = 11, based on the best $least_p$ value obtained. Each mode will be tested on the benchmark graph-coloring problems, and 100 test runs will be performed in each case, unless specified in the table. The results obtained for the case of LMODE = 1 and DC = 11 ($least_p$ = .850) will be regarded as the performance evaluation without parameter adjustment. All other parameters are kept constant throughout all cases of the graph-coloring instances; we will use the same parameters as in the previous experiment. The performance has been rated on average time steps over all test runs.

Results: The experimental results are presented in two tables, where Table III provides the results in all modes where the degree-based heuristic is not applied, and Table IV provides the corresponding results with the degree-based heuristic applied. From the results, it can be inferred that there is a significant improvement in performance when the LMODE,

TABLE II

AVERAGE TIME STEPS AGAINST $least_p$ VALUE

$least_p$	DLC = 000 time steps	DLC = 001 time steps	DLC = 111 time steps
.1	182.0	1222.54	81.48
.2	74.8	898.76	37.66
.3	56.82	693.66	20.62
.4	103.54	757.68	19.14
.5	104.6	683.5	13.38
.6	80.0	595.92	12.74
.7	152.12	385.78	10.6
.8	140.18	416.46	12.34
.9	337.26	330.68	12.46

The evaluations have been conducted for the anna.col graph-coloring problem instance. The number of time steps for obtaining a solution has been recorded over different values of $least_p$. The data is obtained over 50 runs.

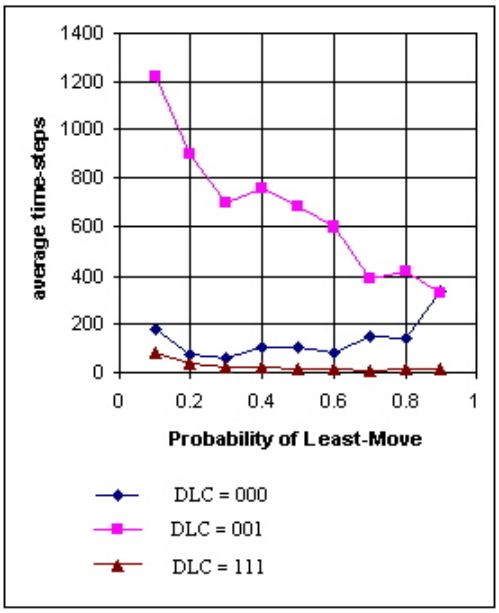

Fig. 1. Average time steps against $least_p$ value. The number of time steps for obtaining a solution has been calculated over different values of $least_p$. The data is obtained over 50 runs on the anna.col graph-coloring instance [1]. Here, letter D stands for Degree Based Heuristic, L is LMODE, and C is CONMODE.

CONMODE, and the proposed degree-based heuristic are applied.

V. DISCUSSION

Parallelizability is a characteristic that allows a large number of reactive agents to take part in the solution-search process. It is also an indicator that an approximate solution can be reached within a short period of time where the number of agents that make movements is very high in the beginning of the search-process.

TABLE III

PERFORMANCE WITHOUT APPLYING THE DEGREE-BASED HEURISTIC

GCP Case		LC=00	LC=01	LC=10	LC=11
miles500	τ	1028.72	864.25	HFR	224
	σ	986.12	806.39	-	148.8
miles250	τ	128.13	314.12	HFR	145.81
	σ	123.49	323.9	-	160.75
miles1000	τ	HFR	HFR	HFR	2137
	S%				95%
	σ	-	-	-	1791.77
miles1500	τ	HFR	879.02	HFR	**293.66**
	σ	-	859.9	-	206.4
anna	τ	74.9	265.77	HFR	219.81
	σ	104.88	266.1	-	216.8
jean	τ	28.1	61.06	HFR	19.98
	σ	23.42	77.54	-	23.12
queen5_5	τ	42.08	57.1	HFR	59.69
	σ	33.06	95.92	-	69.18
queen7_7	τ	**1822.72**	HFR	HFR	4906.71
	S%				46%
	σ	1483.14	-	HFR	2501.84
myciel6	τ	14.34	29.66	64.44	22.72
	σ	8.76	40.7	51.77	30.9
myciel7	τ	96.97	73.04	81.16	87.84
	σ	143.6	125.59	64.19	147.09

τ refers to time steps. LC stands for LMODE and CONMODE flags. HFR stands for High Failure rate. σ indicates standard deviation for the data-set. The success percentage has been indicated for instances, where 100% was not observed. S% is used to indicate success%. The 'LC = 01' shows the performance of the original ERA.

An important relationship has been observed between $least_p$ and the number of time steps. Table II illustrates this on the anna.col problem [1]. A comparison between the original ERA, the proposed modification, and the setting of DLC = 000 is shown. The anna.col problem is comparatively simple and shows a minimum standard deviation on average value calculations, hence providing a reasonable, level-playing field for comparison.

It can be observed that when CONMODE is applied, a higher $least_p$ results in a faster solution convergence, indicating that a large number of agents can make decisions simultaneously. When CONMODE is absent, there is no characteristic relationship that can be observed between $least_p$ and time steps. When we look at DLC = 111, we observe a parallelizability characteristic.

Hence, an increase in the number of *active-agents* results to a corresponding reduction in time steps. This provides evidence that the strength of ERA lies in its ability to solve a CSP in a distributed fashion, by involving several reactive agents at one time.

The degree-based heuristic allows agent-violations to be considered on the basis of variable connectivity where the heuristic involves avoiding the problem of nodes with high connectivity failing to coordinate with the ones with low

TABLE IV

PERFORMANCE ON APPLYING DEGREE-BASED HEURISTIC

GCP Case		LC=00	LC=01	LC=10	LC=11	LC=11*
miles500	τ	93.52	**67.38**	HFR	**61.55**	**76.4**
	σ	84.01	80.70 -	64.68	101.1	
miles250	τ	38.62	40.54	HFR	**35.27**	**39.14**
	σ	34.74	48.18	-	50.23	42.8
miles1000	τ	4307.3	437.44	HFR	**103.31**	**115.41**
	S%	80%				
	σ	2266.2	445.02	-	75.95	76.48
miles1500	τ	HFR	**418.73**	HFR	**375.48**	**400.36**
	σ	-	433.49	-	441.616	400.50
anna	τ	14.97	**10.51**	HFR	**11.87**	**12.07**
	σ	5.703	3.856	-	4.48	3.08
jean	τ	18.54	**9.86**	HFR	**8.69**	**11.69**
	σ	9.71	4.12	-	3.6	5.74
queen5_5	τ	**29.84**	**30.23**	HFR	**26.62**	**36.38**
	σ	20.72	21.93	-	22.02	25.26
queen7_7	τ	3269.1	**1984.53**	HFR	**1781.54**	**2161.2**
	S%	98%			97%	97%
	σ	2279.27	1818.54	-	1686.72	1817.08
myciel6	τ	13.87	**9.10**	67.21	**10.69**	**10.69**
	σ	8.30	4.08	64.45	6.81	6.81
myciel7	τ	16.44	16.83	97.78	**14.57**	**15.2**
	σ	18.54	19.92	66.9	7.61	13.79

τ is used to indicate time steps. LC stands for LMODE and CONMODE flags. HFR stands for High Failure rate. σ indicates standard deviation for the data-set. The success percentage has been indicated for instances where 100% was not observed. S% is used to indicate success%. The 'LC = 11' shows the performance of the proposed modification. The column named LC = 11* is applied on $least_p$ = .850 over all cases to demonstrate performance on consistent values.

connectivity due to synchronous and continuous changes in variable assignments. This is well suited to a distributed, parallelizable system.

Figure 2 shows the comparison of the γ_t against time steps characteristic for three different cases in miles1000.col [1]. The γ_t variation indicates the stability of the search process. From the plot, we can see that when CONMODE is used, the system is more stable, allowing a more synergetic coordination, as compared to when it is absent. The former case also demonstrates a faster rate of increase in γ_t during the first few time steps.

Implementing LMODE allows the search process to explore different variable-value combinations under the condition that CONMODE is also used. This enables the agents to make a wider search of the problem space, by making agents choose minimal positions on a random tie-breaking behavior rather than on a deterministic rule. This behavior can deal with the local minima trap problem better. Other approaches to avoiding this problem have been reported in [4], [9], [12].

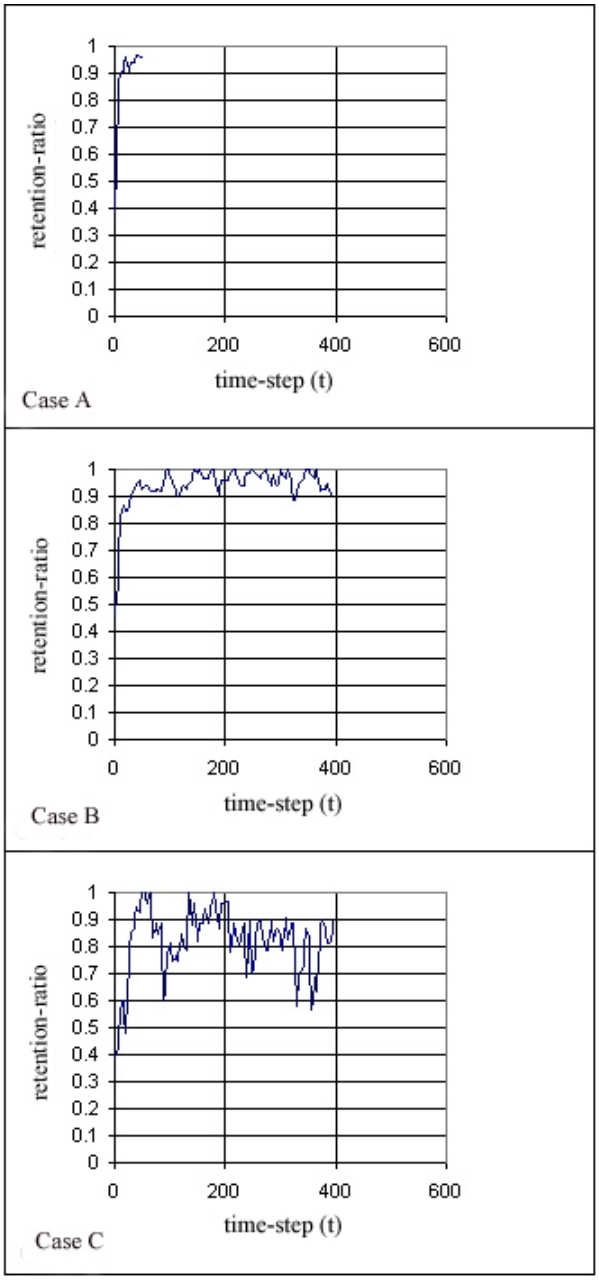

Fig. 2. A comparison of the γ_t against time steps characteristic for three different cases in the miles1000.col [1] problem. LCD corresponds to LMODE, CONMODE, and Degree Heuristic flags, respectively. In the diagram, Case A is LCD = 111, Case B is LCD = 010, and Case C is LCD = 000. LCD = 111 is the configuration for the proposed modification. LCD = 010 is the configuration for the original version of ERA [9].

VI. CONCLUSIONS AND FUTURE WORK

From our experimental studies, we can conclude that the proposed heuristic additions into ERA bring in significant

performance improvements in terms of reduced time steps. The reason attributes to a degree-based rule that fits well into the asynchronous nature of ERA, and a random tie-breaking method that permits a wider span of the search space to be explored.

Our further research will involve generalizing the algorithm to accept problems in a wider domain that goes beyond the specific set of problems considered here. The following issues will be dealt with:

- Empirical studies on how ERA performs across the **phase-transition** [14] need to be conducted.
- The capability of ERA to tackle **over-constrained** problems will be investigated. The deadlock phenomenon and the inability of the ERA algorithm to perform in over-constrained situations has been highlighted in [18].
- The algorithm should be modified so that **small-world structures** in GCPs can be addressed [7][5]. This could potentially make the algorithm adaptable to any graph structures.
- Constraint optimization methods are important in multi-agent coordination [13] and the performance of ERA in this aspect is yet to be studied.

ACKNOWLEDGEMENT

The authors would like to thank the IAT'07 reviewers for their comments and suggestions on the earlier version of this paper. They appreciate receiving the inputs and help from Dr. Jianguo Lu on the presentation. The research work reported here has been supported in part by an NSERC Discovery Grant.

REFERENCES

[1] http://mat.gsia.cmu.edu/color/color.html.

[2] Muhammad Arshad and Marius C. Silaghi. Distributed simulated annnealing and comparison to DSA. 2003.

[3] Makoto Yokoo. Edmund Durfee. Toru Ishida. and Kazuhiro Kuwabara. Constraint satisfaction: The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Trans. Knowledge and Data Engineering*, 10:673–685, 1998.

[4] Jiming Liu. XiaoLong Jin. and Yi Tang. Multi-agent collaborative service and distributed problem solving. *Cognitive System Research. Special Issue on Intelligent Agents and Data Mining for Cognitive Systems*, 5:191 – 206, 2004.

[5] Jiming Liu. XiaoLong Jin. and Kwok Ching Tsui. *Autonomy Oriented Computing, From Problem Solving to Complex Systems Modeling*. Kluwer Academic Publishers, 2004.

[6] XiaoLong Jin. and Jiming Liu. Multiagent SAT (MASSAT): Autonomous pattern search in constrained domains. In *Proceedings of Intelligent Data Engineering and Automated Learning - IDEAL 2002: Third International Conference, Manchester, UK, August 12-14,2002.*, pages 318–328, 2002.

[7] Xiaolong Jin. and Jiming Liu. Efficiency of emergent constraint satisfaction in small-world and random agent networks. In *Proceedings of IAT, 2003*, volume 0, pages 304 – 310, 2003.

[8] Jiming Liu. Han Jing. and Yi Tang. Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, 136:101–144, 2001.

[9] Yi Tang. Jiming Liu. and XiaoLong Jin. Agent compromises in distributed problems solving. *Intelligent Data Engineering and Automated Learning*, 2690:35–42, 2003.

[10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 1995.

[11] Ellis Horowitz. Satraj Sahni. and Sanguthekar Rajasekaran. *Fundamentals of Computer Algorithms*. Galgotia, 1991.

[12] Yi Shang. and Benjamin Wah. A discrete lagrangian-based global-search method for solving satisfiability problems. *Journal of Global Optimization*, 12:61–99, 1998.

[13] Pragnesh Jay Modi. Wei-Min Shen. Milind Tambe. and Makoto Yokoo. An asynchronous complete method for distributed constraint optimization. pages 161 – 168, 2003.

[14] David Clark. Jeremy Frank. Ian Gent. Ewan MacIntyre. Neven Tomov. and Toby Walsh. Local search and number of solutions. *Lecture Notes in Computer Science*, pages 119 – 133, 1996.

[15] Makoto Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *CP '95: Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pages 88–102, London, UK, 1995.

[16] Makoto Yokoo. *Distributed constraint satisfaction: foundations of cooperation in multi-agent systems*. Springer-Verlag, London, UK, 2001.

[17] Weixiong Zhang and Lars Wittenberg. Distributed breakout revisited. In *In Proceedings of the National Conference in Artificial Intelligence*, number 18, pages 352–358, 2002.

[18] Hui Zou and Berthe Choueiry. Multi-agent based search versus local search and backtrack search for solving tight csps: A practical case study. *Working Notes of the Workshop on Stochastic Search Algorithms, IJCAI*, pages 17–24, 2003.